

UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos



Techniques and Tools for Bitcoin Cybercrime Intelligence

DOCTORAL THESIS

Submitted for the degree of Doctor by:

Gibran Alberto Gómez Montes

MSc in Software and Systems

Madrid, 2024



UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos

Doctoral Degree in Software, Systems and Computing

Techniques and Tools for Bitcoin Cybercrime Intelligence

DOCTORAL THESIS

Submitted for the degree of Doctor by:

Gibran Alberto Gómez Montes

MSc in Software and Systems

Under the supervision of:

Dr. Juan Caballero

Madrid, 2024

Title: Techniques and Tools for Bitcoin Cybercrime Intelligence

Author: Gibran Alberto Gómez Montes

Doctoral Programme: PhD Programme in Software, Systems and Computing

Thesis Supervision:

Dr. Juan Caballero, Research Professor at IMDEA Software Institute (Supervisor)

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

This thesis has been partially supported by the Spanish Government MCIN/AEI/10.13039/501100011033/ through grants RTI2018-102043-B-I00 (SCUM), TED2021-132464B-I00 (PRODIGY), PID2022-142290OB-I00 (ESPADA), and PRE2019-088472. Those projects are co-funded by European Union ESF, EIE, and NextGeneration funds.

A mis maestras y maestros.

Acknowledgements

Antes que nada, y sobre todo, me gustaría agradecer a todas y cada una de las personas que me han acompañado en este viaje.

Primero, a aquellas que con su ejemplo me han enseñado a trabajar duro y a cosechar resultados increíbles, a aquellas que con paciencia me han enseñado a ordenar mis pensamientos y han resuelto mis dudas, y a aquellas que me han enseñado el incommensurable valor del trabajo en equipo. A Juan, Bernhard, Dario, Pedro, Platon, Srdjan, Richard, Irfan, Silvia, Daniel, Dimitris, Kevin, Arturo, Stephan, Rainer, Pietro, Stefan, y demás gente de Norton, CSH, TU Wien, NEC Labs, y por supuesto, al resto del IMDEA gang B-)

Segundo, a quienes amorosamente me han brindado soporte y han amablemente cuidado de mi bienestar mental y físico durante estos duros, modestos, hermosos, inciertos, cambiantes, divertidos, estresantes, pandémicos, desenfrenados, pero sobre todo felices, cinco años. A mi Madre y a mi Padre, a mis abuelas y abuelos, a mi hermano (y descendencia), a mis dos familias, la de México y la de Pucela, y sobre todo, a Laura, por haberme compartido parte de su vida.

Y tercero, a quienes me han compartido un poco de su valioso y escaso tiempo para ser vilmente derrochado en unos tragos coquetos, un par de caladas, una noche de verano, o un paseo por el bosque, con el ánimo de aproximar mejor el valor de la vida en un experimento de campo. A todas mis amigas y amigos, que gracias a la vida son muchxs y variadx, y no cabrían en una sola página.

A quienes partieron antes. A quienes aún están. Y a quienes nunca se van a ir.

Punto y aparte, me gustaría aprovechar el tema que atañe este manuscrito para agradecer a dos personas en especial.

A Teresita, doña Tere, abuelita, *mama go'la*, o simplemente Abu. Maestra de profesión, educadora de vocación, evangelizadora nata de conocimientos varios y ejemplo de fortaleza inagotable, quien forjó a su stirpe desde la noble práctica del auto-conocimiento. *Xosh qle'no*.

A mi tío Luisma, quien armado de una caja de cartón, papel, lápiz, y un corazón enorme, enseñó a sus sobrinos imberbes cómo se suponía que una computadora debía funcionar, despertando en mí una profunda curiosidad por entender esa caja mágica una mañana de sábado cualquiera.

Abstract

Bitcoin is a decentralized network with pseudo-anonymous and immutable transactions, thus very attractive for cybercrime. For instance, it is abused to receive payments from victims and by malware developers for building their Command and Control (C&C) infrastructure. This dissertation presents novel techniques and tools for addressing three problems related to Bitcoin cybercrime: cybercrime attribution, estimating cybercriminal revenue, and automating the identification and classification of cryptocurrency addresses.

First, we present *back-and-forth exploration*, a novel Bitcoin transaction tracing technique to identify cybercrime financial relationships. It provides two key contributions. First, it explores both forward and backward, enabling the discovery of relationships that cannot be found by only exploring forward. Second, it prevents exploration explosion by combining a tag database with a novel machine learning classifier that identifies exchange addresses. We implement back-and-forth exploration in *WatchYourBack*, an open-source Bitcoin analysis platform. We evaluate WatchYourBack on 30 malware families. We find new attribution points missed by forward-only explorations, uncover a wealth of services used by the malware, and identify previously unknown relationships between cybercriminals.

Second, we perform the first systematic comparison of Bitcoin revenue estimation methodologies. We implement 15 estimation methodologies in WatchYourBack and use them to quantify the impact of the methodology steps. We show that some estimation methodologies introduce massive overestimation, and propose two more accurate revenue estimations. We collect 30,424 cybercrime payment addresses and use them to compare the financial impact of 6 abuse types (e.g., ransomware, sextortion) and 141 cybercriminal groups. For the first time, we quantify the impact of address coverage on estimations by proposing two techniques to achieve high coverage on the payment addresses of the DeadBolt ransomware. Our expanded coverage enables us to estimate DeadBolt’s revenue at \$2.47M, an estimation 39 times higher than the one resulting from using two Internet scan engines for discovering DeadBolt’s addresses.

Third, we propose techniques for address identification and classification. For address identification we develop *GoodFATR*, a platform for collecting threat reports, extracting indicators using multiple tools, and comparing their accuracy using a novel majority voting methodology that needs no ground truth. We use our methodology to evaluate *IOCSEARCHER*, GoodFATR’s IOC extraction tool, against 7 other tools. The evaluation shows that IOCSEARCHER is most accurate for extracting 11 of 13 indicator types. For address classification, we design a novel unsupervised classifier that takes as input an abuse report written by a victim and leverages a large language model (LLM) to interpret the text, filter the report if it is spam, and assign it one of 19 abuse types otherwise. We collect 290K cryptocurrency abuse reports from two popular abuse reporting services, build ground truth for 20K reports, and use them to evaluate different designs for our classifier, achieving a precision of 0.92, a recall of 0.87, and an F1 score of 0.89. We analyze the challenge of pollution in abuse reporting services showing that spam may flood unattended services, that a handful of incorrectly reported benign addresses receive 60% of the deposited funds, and that abuse types reported by users are untrustworthy. Finally, we quantify that revenue estimated from blockchain transactions can be 29 times higher than losses reported by victims.

Resumen

Bitcoin es una red descentralizada con transacciones pseudoanónimas e inmutables, muy atractiva para la ciberdelincuencia. Por ejemplo, los atacantes la usan para recibir pagos de víctimas y para construir infraestructura de Comando y Control (C&C). Esta tesis presenta técnicas y herramientas novedosas para abordar tres problemas sobre la ciberdelincuencia en Bitcoin: atribución, estimación de los ingresos de los ciberdelincuentes y automatización de la identificación y clasificación de direcciones.

Primero, presentamos la exploración *back-and-forth*, una novedosa técnica de rastreo de transacciones en Bitcoin que ofrece dos contribuciones clave. Primero, explora hacia delante y hacia atrás, permitiendo descubrir relaciones que no pueden encontrarse explorando sólo hacia delante. Segundo, evita la explosión de la exploración combinando una base de datos de etiquetas con un novedoso clasificador que identifica direcciones de exchanges de criptomonedas. Implementamos la exploración *back-and-forth* en *WatchYourBack*, una plataforma para el análisis de Bitcoin y la evaluamos en 30 familias de malware. Encontramos puntos de atribución omitidos por exploraciones solo hacia delante, decenas de servicios utilizados por el malware y relaciones entre ciberdelincuentes.

Segundo, realizamos la primera comparación sistemática de metodologías de estimación de ingresos en Bitcoin. Implementamos 15 metodologías en *WatchYourBack* y las utilizamos para cuantificar el impacto de los diversos pasos metodológicos. Demostramos que algunas metodologías introducen una sobreestimación masiva y proponemos dos que producen estimaciones más precisas. Recopilamos 30.424 direcciones cibercriminales y las utilizamos para comparar el impacto financiero de 6 tipos de abusos (e.g., ransomware, sextortion) y 141 grupos criminales. Por primera vez, cuantificamos el impacto de la cobertura de direcciones en la estimación, proponiendo dos técnicas para expandir la cobertura del ransomware *DeadBolt*. Nuestra cobertura ampliada nos permite estimar los ingresos de *DeadBolt* en USD \$2,47M, 39 veces más que la resultante al utilizar dos servicios de escaneado de Internet para descubrir direcciones de *DeadBolt*.

En tercer lugar, proponemos técnicas de identificación y clasificación de direcciones. Para la tarea de identificación desarrollamos *GoodFATR*, una plataforma para recopilar informes sobre amenazas, extraer indicadores utilizando múltiples herramientas y comparar su precisión mediante una novedosa metodología de votación por mayoría que no necesita ground truth. Utilizamos nuestra metodología para evaluar *IOCSEARCHER*, la herramienta de extracción de IOCs de *GoodFATR*, frente a otras 7 herramientas, mostrando que *IOCSEARCHER* es la más precisa en 11 de 13 tipos de indicadores. Para la tarea de clasificación, diseñamos un novedoso clasificador no supervisado que, dado una denuncia de abuso escrita por una víctima, usa un modelo de lenguaje de gran tamaño (LLM) para interpretar el texto, filtrar la denuncia si es spam, y asignarla en caso contrario uno de 19 tipos de abuso. Recopilamos 290K informes de abuso de dos servicios, construimos ground truth para 20K informes y los utilizamos para evaluar diferentes diseños para nuestro clasificador basado en LLM, logrando una precisión de 0,92, una recuperación de 0,87 y una puntuación F1 de 0,89. Analizamos la polución en los servicios de denuncia de abusos mostrando que el spam puede inundar servicios desatendidos, que un puñado de direcciones benignas incorrectamente denunciadas reciben 60% de los fondos

depositados, y que el tipo de abuso reportado por las víctimas no es confiable. Por último, cuantificamos que las estimaciones a partir de transacciones blockchain pueden ser 29 veces superiores que las pérdidas denunciadas por las víctimas.

Table of Contents

Acknowledgements	v
Abstract	vi
Resumen	vii
List of Figures	xi
List of Tables	xiv
Abbreviations and acronyms	xviii
1 Introduction	1
1.1 Thesis Contributions	7
1.2 Thesis Organization	9
2 Background and Related Work	11
2.1 Bitcoin and Cybercrime.	11
2.2 On-chain Bitcoin Analysis.	14
2.3 Off-chain Bitcoin Analysis.	18
3 Identifying Cybercrime Financial Relationships	23
3.1 Introduction	23
3.2 Back-and-Forth Exploration	25
3.3 Tagging	27
3.4 Exchange Detector	30
3.5 Back-and-Forth Exploration Evaluation	33
3.5.1 Evaluating Address Discovery	36
3.5.2 Exchange Classifier Impact	39
3.6 Relationships	42
3.6.1 Limited Ground Truth Validation	45
3.7 Discussion	45
3.8 Conclusion	47
4 The Impact of Methodology and Coverage in Revenue Estimations	49
4.1 Introduction	49
4.2 Bitcoin Revenue Estimation	51
4.2.1 Seeds	52
4.2.2 Expansions	53
4.2.3 Filtering	56

4.2.4	Conversion Rates	58
4.3	Methodology Impact	58
4.3.1	Methodologies and Datasets	58
4.3.2	Impact on CryptoLocker	60
4.3.3	Impact on the Whole Dataset	62
4.3.4	Cybercrime Estimations	63
4.3.5	Multi-Input Evasion	65
4.4	Coverage Impact	65
4.4.1	DeadBolt	66
4.4.2	DeadBolt Datasets	67
4.4.3	Increasing the Coverage	68
4.4.4	Coverage Comparison	70
4.5	How To Estimate	71
4.6	Further Discussion	73
4.7	Conclusion	74
5	A Comparison Methodology for Indicator Extraction Tools	75
5.1	Introduction	75
5.2	Overview	77
5.3	Threat Report Collection	79
5.3.1	Threat Report Sources	80
5.3.2	Downloader	82
5.4	Indicator Extraction: A Comparative Study	82
5.4.1	Text Extraction	83
5.4.2	Indicator Extraction	84
5.4.3	Filtering.	89
5.5	Evaluation	90
5.6	Evaluation of IOC Extraction Tools	92
5.7	Case Study: Blockchain Address Tagging	100
5.8	Discussion	101
5.9	Conclusions	102
6	Automatic Classification of Cryptocurrency Abuse Reports	105
6.1	Introduction	105
6.2	Abuse Reporting Services	107
6.3	Abuse Type Taxonomy and Ground Truth	109
6.3.1	Taxonomy	109
6.3.2	Taxonomy and Ground Truth Construction	111
6.4	Data Pollution in Abuse Reporting Services	113
6.4.1	Spam Reports	113
6.4.2	Accuracy of User-Reported Abuse Types	114
6.4.3	Benign Addresses Reported	116
6.5	Abuse Report Classification	117
6.5.1	LLM-Based Unsupervised Classifier	117
6.5.2	Supervised Classifier	120

6.5.3	Scam Tracker Classification	121
6.5.4	Updating the Taxonomy and Classifier	122
6.5.5	Spam Detection	123
6.6	Abuse Type Revenue Estimation	124
6.6.1	Victim-Reported Financial Losses	124
6.6.2	Estimation from Address Deposits	124
6.7	Discussion	126
6.8	Conclusion	128
7	Conclusions & Future Work	129
7.1	Conclusions	129
7.2	Future Work	131
	Bibliography	133
	Appendix	145

List of Figures

2.1	Example transaction graph from two malicious seeds in gray. Forward transaction tracing would discover the right subgraph, while back-and-forth exploration would discover the whole graph including a new malicious address.	17
3.1	Back-and-forth exploration architecture.	26
3.2	Exchange classifier ROC curve.	32
3.3	False positive example.	35
3.4	Relation detection impact when introducing exchange classifier errors. Introducing C-FPs produces R-FNs (green line). Introducing C-FNs produce R-FPs (red line).	39
3.5	Phorpiex-tldr – SaveYourself relations.	43
3.6	Mekotio-MrPr0gr4mmer relations.	44
4.1	Two approaches to collect payments from two seeds S_1, S_2 (in gray) and two unknown payment address P_1, P_2 into a collector address C . On the left, MI clustering discovers the unknown addresses P_1, P_2 , while on the right it does not.	55
4.2	Cumulative distribution of the proportion of withdrawals with one input address (1-to-n) over all the withdrawals across the 141 groups for both withdrawals of seeds (orange line) and withdrawals of their MI clusters (blue line).	66
4.3	DeadBolt key release transactions. First, the victim deposits the ransom of 0.03 BTC or 0.05 BTC to the payment address. Then, the cybercriminals release the decryption key to the blockchain through a deposit to the payment address.	67
4.4	Overlap in DeadBolt datasets. Left: Bitcoin addresses. Right: IP addresses.	68
4.5	DeadBolt’s revenue using full coverage. On the left is the daily BTC revenue. On the right, the cumulative revenue over time using full coverage (blue line) and only scan engines (orange line).	71
5.1	The architecture of GoodFATR.	78
5.2	Distribution of IOC classes grouped per source.	91
5.3	Eta-square values resulting from pairwise comparison of tools.	98
6.1	Our taxonomy. White nodes come from the initial taxonomy. Light gray solid nodes were identified by the clustering. Dark gray dashed nodes were not originally part of the taxonomy but were added during our evaluation.	109

6.2	Volume of submitted reports and spam reports in BitcoinAbuse (log-scale) per month. Spam first appears in 2021 and dominates all reports since June 2022.	113
6.3	Heat-map capturing the fraction of the 43K evaluation reports assigned the BA type on the y-axis and tagged with the taxonomy abuse type on the x-axis. White means the ratio is zero, light orange is a small ratio, and dark red is a high ratio.	114
6.4	LLM prompt. Macros in angle brackets are replaced with abuse type definitions and the description text.	117
6.5	LLM prompt used in the baseline. Macros in angle brackets are replaced with a list of abuse types and the description text. Abuse type definitions are not included.	118
6.6	Confusion matrix of the top-to-bottom classification using gpt-4 on the Scam Tracker dataset of 199 samples.	122

List of Tables

2.1	Summary of prior academic works on threat report collection and IOC extraction.	20
3.1	Top-20 tag source URLs.	28
3.2	Tag database summary.	29
3.3	Dataset of 108,530 addresses used for training and testing exchange classifier.	30
3.4	Classifier features extracted per address.	31
3.5	Classifier results using Stratified 5-fold Cross Validation on 80% training data, and on 20% of unseen testing data.	32
3.6	Back-and-forth explorations for the 30 malware families. In parenthesis, forward-only exploration results for comparison: fraction of addresses and transactions and total number of relations present in the forward-only graph.	33
3.7	Clipper (top) and ransomware (bottom) profits measured on seeds.	34
3.8	Cerber signaling addresses. Exploring from 4 seeds, 3 previously unknown signaling addresses were discovered.	36
3.9	Pony signaling addresses. Exploring from 4 seeds, 8 previously unknown signaling addresses were discovered.	37
3.10	Glupteba signaling addresses. Exploring from 3 seeds, 2 previously unknown addresses were discovered.	38
3.11	Total runtime and graph size, across the 23 families where at least one seed is not an online wallet, when running the exploration with and without the exchange classifier. To handle explosion when not using the classifier, 7 explorations were stopped when reaching 20K addresses.	40
3.12	Summary of relationships found. Tags with an asterisk at the end were found by multi-input clustering on the seeds, the rest were found by the exploration.	41
4.1	Related work on estimating cybercrime Bitcoin financial impact. A tick (✓) indicates a property is implemented by the work, a cross (✗) no support, a half-filled circle (◐) partial support, and a dash (-) that the paper does not specify it.	51
4.2	Previous estimations on CryptoLocker.	60
4.3	CryptoLocker estimations using different methodologies starting from the two seeds used in [116, 151]. A dash indicates the estimation did not finish in 5 days.	60
4.4	CryptoLocker estimations using different methodologies starting from the four seeds used in [45]. A dash indicates the estimation did not finish in 5 days.	61

4.5	Cybercrime type estimates on April 12, 2023 using the seeds from public datasets and a <i>DD-OW+MI-DC</i> estimation.	62
4.6	Top-20 groups by USD financial impact on April 12, 2023 using a <i>DD-OW+MI-DC</i> estimation. Ransomware families are prefixed by R:, clipper families by C:, and Ponzi schemes by P:. The last row captures total revenue of all 141 labeled groups.	64
4.7	DeadBolt datasets summary.	67
4.8	DeadBolt estimations from different vantage points.	70
5.1	Sources used for collecting threat reports.	80
5.2	Comparison of indicator extraction tools.	82
5.3	Indicators extracted by different tools.	87
5.4	Data collection summary.	90
5.5	Top-10 origins that contribute with the highest number of IOCs, grouped per source. In brackets we report the percentage of IOCs associated to each origin.	92
5.6	Precision (P), recall (R), and F1 score achieved by each indicator extraction tool on the comparative evaluation over 4,420 documents. Only indicator types extracted by at least two tools are included.	95
5.7	Top-10 indicator types extracted by all tools.	96
5.8	Post-hoc t-tests between groups summary.	97
5.9	Indicator extraction runtime and average by indicator type (in seconds).	98
5.10	Social indicators validation.	99
5.11	Accuracy of filtering approaches on a manually-built ground truth.	99
6.1	Abuse report datasets used.	108
6.2	Number of descriptions for each abuse type in ground truth datasets.	111
6.3	LLM-based classifier results on 1,049 BitcoinAbuse descriptions. Our best design in the bottom row more than doubles the F1 score compared to the baseline in the top row.	120
6.4	Accuracy for the supervised classification models.	121
6.5	Classification results on Scam Tracker ground truth dataset with 199 labeled descriptions.	121
6.6	Scam Tracker reports classified into each abuse type, the subset reporting a financial loss, the total loss in reports of the type, and the average and median loss per report.	123
6.7	Revenue estimation for 1,960 addresses tagged using the classification of 5K BitcoinAbuse descriptions.	125
6.8	For a set of 71 Scam Tracker reports mentioning 96 Bitcoin addresses, comparison of victim-reported losses (Rep. Loss) with revenue computed from the address deposits (BTC, USD).	126
1	Mapping between short name and full address.	146
2	Summary of forward-only explorations for the 30 malware families.	147
3	Summary of relationships found with forward-only explorations. Tags with an asterisk at the end were found by multi-input clustering on the seeds, the rest were found by the exploration.	148

4	Operations in the Ransomwhere [30], Ponzi [9], and Clipper [73] datasets, ordered by USD revenue using the <i>DD-OW+MI-DC</i> estimation. The last row captures total revenue of all 141 operations.	151
5	List of definitions for the LLM	153

Abbreviations and acronyms

APT	Advanced Persistent Threat
AML	Anti-Money Laundering
BBB	Better Business Bureau
BTC	Bitcoin (currency)
C&C	Command and Control Server
CTI	Cyber Threat Intelligence
DC	Double Counting
EUR	Euro (currency)
ICO	Initial Coin Offering
IOC	Indicator of Compromise
KYC	Know Your Customer
LEA	Law Enforcement Agency
LLM	Large Language Model
MI	Multi-input
ML	Machine Learning
NLP	Natural Language Processing
P2P	Peer-to-peer
RaaS	Ransomware as a Service
RSS	Really Simple Syndication
UTXO	Unspent Transaction Output

Chapter 1

Introduction

Bitcoin is the original implementation of the *blockchain* technology proposed by Satoshi Nakamoto in 2008 [137], a public, distributed, and decentralized mechanism that enables, among other things, pseudo-anonymous electronic payments. Bitcoin’s infrastructure consists of a permissionless peer-to-peer (P2P) network that allows users to directly coordinate to build the blockchain, an append-only data structure that stores user *transactions*. Bitcoin enables payments of the digital currency *bitcoin* (BTC) through transactions, simultaneously allowing users to store arbitrary information in the blockchain in small amounts. It implements public-key cryptography to produce and secure transactions, which removes the need for central authorities to authenticate users and makes transactions pseudo-anonymous. To receive payments, users generate Bitcoin *addresses* by encoding the public key of a cryptographic key pair into a randomized string. To send bitcoins, the sender uses the receiver’s address to produce a transaction that locks the coins using its public key, so only the receiver’s private key can spend the locked coins. To secure payments, a set of users called *miners* verify all transactions broadcast to the network before writing them in the blockchain. Miners guarantee, for instance, that only the corresponding private key of an address can spend its coins or that users spend their coins only once. Because mining consumes computer processing power, miners receive an amount of BTC in exchange for their work.

In summary, Bitcoin is a public, distributed, decentralized network with pseudo-anonymous and immutable transactions. Furthermore, its legal status has remained largely unregulated in many countries. These properties have made Bitcoin highly attractive for cybercrime. Cybercriminal threat groups exploit Bitcoin’s pseudo-anonymity, privacy-enhancing techniques, and services with shallow or no regulations (e.g., cryptocurrency exchanges) to acquire and manage illicit wealth while remaining undetected. For instance, Bitcoin is the preferred payment channel for darknet markets where users can obtain diverse illegal goods and services [38, 175, 195, 113, 112, 147], for massive extortion and scam campaigns [9, 152, 143, 222, 10, 142, 115, 15], for highly disrupting ransomware attacks targeting governments, health institutions, and major corporations [196, 103, 116, 28, 90, 45, 157, 151, 131, 79, 146, 101], and for several forms of money laundering that support cybercrime [134, 221, 210, 54, 62]. Another common type of abuse in Bitcoin is thefts from users and services, which in some cases has resulted in millionaire losses [130, 57, 104].

Bitcoin has also attracted the attention of malware developers, who write malware designed to steal coins or computer processing power. Besides ransomware, examples include clippers [177, 105, 34, 73], which make users send BTCs to the attacker’s address by taking control of the clipboard of the operating system; information stealers [177, 105, 156], which, among other sensitive data, steal private keys to extract BTCs from their victims; cryptojackers [89, 155, 107, 95], which silently mine coins using the victim’s resources, and a combination of them [177, 105]. Malware developers also abuse the Bitcoin blockchain to build resilient communication mechanisms for hiding and transmitting information. For instance, some botnet masters use Bitcoin transactions to signal the location of the Command and Control (C&C) server their bots should connect to [157, 199, 180, 136].

Although Bitcoin is decentralized, centralized services like cryptocurrency exchanges play a critical role in the ecosystem. Services facilitate transacting BTCs for fiat currencies like euros (EUR) or US dollars (USD) by providing users with cryptocurrency addresses managed by the services but assigned to their users to receive BTC deposits. Services lower the technical skills users need to adopt Bitcoin and thus have become the preferred entry and exit point to the Bitcoin economy [130, 83]. For this reason, governments worldwide have started regulating services like exchanges and gambling sites, establishing Know-Your-Customer (KYC) and anti-money laundering (AML) policies [120, 42, 44] that force the services to identify their users. Although Bitcoin is pseudo-anonymous, it does not obfuscate transactions by default like other blockchains (e.g., Monero [162]). Instead, Bitcoin transactions describe financial relationships among pseudo-anonymous entities. By tracing BTC flows and attributing some of the payment addresses, e.g., by leveraging information from public sources to link addresses with their owner [13], it is possible to learn which cybercrime groups interact with others, how often they transact, how many BTCs they send or receive, and the services they use or abuse to buy and cash them out. Finding relations between cybercrime and services is crucial for attribution when the services implement KYC/AML policies, as it enables two intervention points. First, the service controlling the addresses receiving the illicit funds can uncover the identity of the misbehaving users. Second, the service can freeze the transactions of those users, opening the possibility of recovering the funds if the incident response is fast. Besides tracing BTC flows, identifying the addresses owned by cybercrime groups enables the estimation of their profits by analyzing their deposits. This information is vital for law enforcement agencies (LEAs) to estimate the financial impact of different abuse types (e.g., ransomware, sextortion) and cybercriminal groups (e.g., Conti, Glupteba) and thus prioritize the most dangerous ones to be dealt with first.

This dissertation presents novel techniques and tools for extracting intelligence from the Bitcoin blockchain to analyze cybercrime activities. Our techniques leverage on-chain data extracted directly from the blockchain and off-chain data collected and curated from public sources. First, we propose a novel methodology for finding financial relationships between cybercrime groups and services by tracing the funds obtained from victims or used to enable malicious activities. In particular, we propose a transaction tracing technique that finds relationships that previous techniques miss by design, allowing the identification of previously unknown attribution points for cybercrime operations and new addresses related to cybercrime, which gives LEAs better chances for attribution. Second, we perform the first systematic comparison between revenue estimation methodologies. We analyze common causes of cybercrime revenue

overestimating and underestimating and propose strategies to avoid them. We additionally measure the impact of the lack of coverage on cybercrime-controlled addresses, a prevalent problem when estimating cybercrime revenue. Accurately estimating the revenue is vital for LEAs to optimize resources assigned to fight cybercrime, i.e., to avoid underestimating serious threats and overestimating those with lower risk. Third, we propose techniques for automating the generation of attribution data from public sources. We divide this task into two: address identification and address classification. For address identification, we develop a platform to monitor cybercrime activities reported in 6 CTI sources. It allows us to identify cryptocurrency addresses related to cybercrime and build attribution data on demand by analyzing the sources where they appear. For address classification, we leverage abuse reporting services where users report abusive cryptocurrency addresses. We propose an abuse report classifier to assign one of 19 abuse types to abuse reports. Then, we classify addresses through a majority vote on the addresses' reports. We show that our classifier accurately filters spam reports that pollute abuse reporting services. Finally, we show that revenue estimations from victim-reported losses highly underestimate cybercriminal revenue.

Bitcoin tracing. Tracing how money flows between cybercriminals and third-party entities, like cryptocurrency exchanges, is fundamental for attribution, i.e., to find services with KYC requirements where cybercriminals buy BTCs or exchange BTCs for fiat currencies (e.g., USD). Tracing can also discover previously unknown financial relationships between cybercrime groups and boost the detection of other malicious actors. Previous works use *forward transaction tracing* to analyze Bitcoin addresses related to cybercrime [130, 45, 90, 151, 152, 113]. Forward tracing observes the withdrawals of a set of initial addresses, called *seeds*, to find the receiver addresses. This process may be repeated transitively with the addresses found in previous iterations until reaching some stop condition, effectively following (forward) the money flow. The entities and relations found are usually output in a graph when the exploration stops. A principal limitation of forward tracing is that money flows in other directions (e.g., deposits to the explored addresses) are omitted, which limits the discovery of new relations. For example, botnet masters should buy BTCs to pay for the signaling of the C&C server. Yet, the signaling may spend all the funds, rendering it impossible to find attribution points by exploring forward. However, exploring backward may reveal the services used to replenish BTCs. While forward exploring services used or abused for money laundering, such as mixers, is useless due to the techniques these services use to obfuscate money flows, backward exploring them may still identify their clients. Besides, current forward-only explorations often consist of just one iteration (i.e., the exploration stops after analyzing all the withdrawals of the seeds), mainly due to limitations in detecting when the coins change owner, a problem usually addressed with *tags*. Tags are pieces of information that link cryptocurrency addresses to an owner and abuse type, which are scarce. Errors in detecting *change of ownership* introduce noise into the exploration and may provoke an explosion in the number of addresses and transactions analyzed. For instance, exploring addresses owned by services will bring tons of other (unrelated) addresses into the exploration, which belong to the service and its users.

In Chapter 3, we present *back-and-forth exploration*, a novel automated Bitcoin transaction tracing technique to identify cybercrime financial relationships forward and backward, and

WatchYourBack, a blockchain analysis platform implementing it. Given seed addresses belonging to a cybercrime campaign, WatchYourBack outputs a transaction graph and identifies paths corresponding to relationships between the campaign under study, external services, and other cybercrime campaigns. Back-and-forth exploration provides two key contributions. First, it explores both forward and backward, enabling the discovery of relations that cannot be found by only exploring forward (e.g., deposits from clients of a mixer). Second, it prevents explosion by combining a tag database, built using publicly available information from diverse sources, with a machine learning (ML) classifier that identifies untagged addresses belonging to exchanges, the most common services in the Bitcoin ecosystem. In addition, we build *operation oracles*, classifiers that capture the behavior of malicious signaling addresses, a vital component in the infrastructure of some cybercrime groups under study. Oracles detect previously unknown addresses showing the signaling behavior, so we use them to assess the relations found during the back-and-forth explorations.

We evaluate back-and-forth exploration on 30 malware families. We build oracles for 4 families using Bitcoin for signaling C&C servers. We use them to demonstrate that back-and-forth exploration identifies 13 signaling addresses missed by prior work, 8 of which are completely unreachable by forward-only explorations. Our approach uncovers a wealth of services used by the malware, including 44 exchanges, 11 gambling sites, 5 payment service providers, 4 underground markets, 4 mining pools, and 2 mixers. In 4 families, the relations include new attribution points missed by forward-only explorations. It also identifies relationships between the malware families and other cybercrime campaigns, highlighting how some malware operators participate in diverse cybercriminal activities.

Bitcoin revenue estimations. Multiple works have leveraged the Bitcoin ledger to analyze addresses controlled by cybercriminal groups. Transactions sent to such addresses are public and transparent, so it is possible to inspect them to estimate the revenue they obtain from their victims, along with other interesting information like the periods in which the groups were active or the number of victims with losses in BTC. Estimating the revenue is necessary for measuring and comparing the impact of different abuse types and cybercrime groups to decide which ones are the most severe and must be dealt with first. Underestimating cybercrime revenue may make dangerous operations fly under the radar, leading them to continue unfettered. Overestimating revenue could cause law enforcement to assign them unnecessary resources, which are frequently scarce. Interestingly, previous estimations focusing on the same target (e.g., a ransomware family) largely disagree due to using different methodologies, seed addresses, and activity periods. These factors complicate understanding the impact of the techniques involved in each estimation methodology. Furthermore, estimation methodologies are widely assumed to underestimate the revenue due to the lack of coverage on the target’s payment addresses, but the magnitude of this impact remains unknown. On the other hand, no previous work has systematically studied if these methodologies introduce overestimation and, if so, if it is possible to develop techniques to improve the results.

In Chapter 4, we perform the first systematic comparison of cybercrime Bitcoin revenue estimation methodologies. During this process, we show how self-deposits, a consequence of how Bitcoin’s transaction model handles money change, inflate estimations and are a significant source of overestimation. Then, we propose a general technique for detecting and

filtering out *double-counting* (DC) Bitcoin deposits. To enable our work, we implement an estimation module on our blockchain analysis platform WatchYourBack to replicate different estimation methodologies proposed in the past, with and without applying our general filtering technique. Then, we use our platform to quantify the impact of the diverse methodology steps in a controlled setting. By fixing the seed addresses, the tag database, and the block height, we compare 15 methodologies to isolate their impact on the estimation. We show that not all estimation methodologies underestimate the revenue, as some may introduce massive overestimation. We propose two estimation methodologies from this analysis that produce more accurate revenue estimations.

We collect 30,424 payment addresses and, using the two proposed methodologies, we compare the financial impact of 6 abuse types (ransomware, clippers, sextortion, Ponzi schemes, giveaway scams, exchange scams) and 141 cybercriminal groups. While analyzing the seeds, we observed that the popular multi-input clustering [6, 174, 130], the most used address expansion technique, fails to discover addresses for 40% of groups, highlighting the need for new address expansion techniques. To conclude this Chapter, we quantify, for the first time, the impact of the lack of address coverage on the estimation. For this, we propose two techniques to achieve high coverage, possibly nearly complete, on the payment addresses of the DeadBolt server ransomware. The first technique leverages relationships between DeadBolt addresses with different roles extracted from their transactions. The second one leverages an operation oracle to scan all the transactions in the blockchain for over 15 months, searching for DeadBolt’s signaling transactions. Our expanded coverage enables us to estimate DeadBolt’s revenue at \$2.47M, an estimation 39 times higher than the one resulting from using two popular Internet scan engines for discovering DeadBolt’s payment addresses.

Cryptocurrency address identification. Due to the abuse of cryptocurrencies by cybercrime, cryptocurrency addresses frequently appear in threat reports published by cyber-security analysts along with other indicators of compromise (IOCs), i.e., forensic artifacts found in compromised hosts or networks, such as domain names, URLs, and file hashes. For instance, it is possible to find cybercrime-related cryptocurrency addresses in threat reports that dissect clippers [67, 105, 177] and ransomware [211, 1]. Threat reports may also mention and identify addresses that belong to third-party services (like cryptocurrency exchanges) abused by cybercriminals to cash out cryptocurrencies for fiat money and addresses that belong to attacked services [57]. Besides, services may announce addresses they own [13]. Identifying addresses controlled by services is vital for the attribution of cybercrime. In general, open-source and commercial IOC feeds allow collecting IOCs. It is also possible to extract them from a myriad of unstructured threat reports written in natural language and distributed using several sources, such as blogs and social media. Although multiple indicator extraction tools using regular expressions to identify IOCs in threat reports exist, to the best of our knowledge, no previous work has systematically compared their accuracy. In addition, the false positives most tools may produce during the extraction are hard to identify due to the lack of validation, and most of the existing tools do not support the extraction of cryptocurrency addresses.

In Chapter 5, we present GoodFATR, an automated platform for collecting threat reports from diverse sources, extracting IOCs from the collected reports using multiple tools, and comparing

their accuracy. GoodFATR supports 6 threat report sources: Really Simple Syndication [219] (RSS), Twitter, Telegram, Malpedia [158], APTnotes [8], and ChainSmith [228]. GoodFATR continuously monitors the sources, downloads new threat reports, extracts 41 indicator types from the collected reports, and filters non-malicious indicators to output the IOCs. To improve the accuracy of IOC extraction, we additionally present IOCSEARCHER, GoodFATR’s IOC extraction tool. IOCSEARCHER supports 9 cryptocurrencies, including Bitcoin. It applies a match-and-validate approach, reducing the number of false positives due to indicators matching more than one regular expression, a prevalent problem when collecting cryptocurrency addresses and hashes like MD5 [171]. We propose a novel majority voting methodology for evaluating IOC extraction that does not require a manually-built ground truth. Instead, it leverages agreement on the artifacts extracted from a high volume of threat reports. We implement our approach in GoodFATR to evaluate IOCSEARCHER against other popular open-source indicator extraction tools.

We run GoodFATR over 15 months to collect 472,891 reports from the 6 sources, extracting 978,151 indicators from the reports and identifying 618,217 IOCs. We apply GoodFATR to compare the accuracy of 7 popular open-source tools against IOCSEARCHER. The evaluation shows that IOCSEARCHER is the more accurate IOC extraction tool when extracting 11 of 13 indicator types supported by multiple tools. Finally, we show how GoodFATR can assist in increasing the coverage of the tag database proposed in Chapter 3. For this, we collected all the identified cryptocurrency addresses along with their sources, e.g., a Bitcoin address and the URL of the blog post where it appears, to produce tags for them using tuples $\langle \textit{cryptocurrency}, \textit{address}, \textit{source} \rangle$. Such tagged addresses can be classified by analysts on demand, for instance, if they appear during explorations. We generate tags for 854 Bitcoin addresses identified in the wild, increasing the tag database of WatchYourBack by 12.6 addresses per week, on average.

Cryptocurrency address classification. IOC extraction tools rarely give the extracted IOCs an abuse type, i.e., a category characterizing the abuse where the IOC appears, such as sextortion, ransomware, or investment scam. Previous works have applied natural language processing (NLP) techniques for IOC extraction [117, 228, 94, 226], producing dedicated pipelines for extracting keywords from the input text, e.g., the IOCs and additional keywords. These approaches can classify IOCs by leveraging such keywords [226], e.g., giving the extracted IOCs a category related to their domain (IoT, finance, education). However, to our knowledge, previous works do not focus on giving an abuse type to the IOCs found. Besides, NLP tools build dedicated pipelines for a specific task (e.g., IOC extraction) that are challenging to adapt for other tasks (e.g., classification). To conclude this dissertation, we focus on techniques for giving an abuse type to cryptocurrency addresses by leveraging abuse reports written by victims. For this, we analyze abuse reporting services, online platforms designed to collect reports written by victims of different types of cryptocurrency abuse. Accurate classification of the submitted reports is fundamental for analyzing the prevalence and financial impact of the abuse types. Current classification approaches in these services are problematic because they require the reporter to select the abuse type from a list, assuming the reporter has the necessary experience for the classification, which we show is frequently not the case, or are manually classified by analysts, which does not scale.

In Chapter 6, we present a novel abuse report classifier that interprets cryptocurrency abuse reports written in natural language to identify the abuse type they describe, accompanied by a novel taxonomy of 19 prevalent abuse types. Our classifier detects if the input reports are related to cryptocurrency abuse, filters out reports not describing abuse, and selects an abuse type for the abuse reports. To build the taxonomy, we identify the most prevalent abuse types in our abuse reporting services and the hierarchical relationships between them, i.e., some abuse types are a subtype of others (e.g., *investment scam* is a subtype of *scam*). We write natural language definitions that describe each abuse type and build a classifier for processing natural language text using supervised and unsupervised ML techniques. We first build ground truth data to train a supervised classifier based on transfer learning as a baseline. Then, we compare it with an unsupervised approach that leverages a large language model (LLM) to interpret the textual description written by the reporter to assign it an abuse type in our taxonomy. In contrast to supervised classifiers, our LLM-based classifier predicts based on natural language definitions. It does not need training, which removes the need for ground truth data and avoids data overfitting.

To enable our study, we collect 290K cryptocurrency abuse reports from two popular abuse reporting services: BitcoinAbuse [15] and BBB Scam Tracker [184]. We build ground truth for 20K of those reports and use it to evaluate three designs for our LLM-based classifier and four LLMs, plus the supervised ML classifier used as a baseline. Our results show that, although the baseline model achieves an almost perfect score during its training, its performance dramatically drops when classifying samples from a different reporting service in an out-of-distribution experiment, achieving an F1 score of 0.55. In contrast, the LLM-based classifier achieves a precision of 0.92, a recall of 0.87, and an F1 score of 0.89 on the same data. We apply our abuse report classifier to examine pollution in abuse reporting services. We show that spam may flood unattended services, that a handful of benign addresses incorrectly reported receive 60% of the funds deposited to all reported addresses, and that abuse types reported by users are untrustworthy. We show how our classifier can identify valid reports, filter spam reports, and classify valid reports into 19 abuse types. We finally show that revenue estimations produced from blockchain transactions can be 29 times higher than those from victim-reported losses, which highly underestimate cybercriminal revenue.

1.1 Thesis Contributions

In this dissertation, we present the following contributions.

- We propose *back-and-forth exploration*, a novel automated Bitcoin transaction tracing technique to identify cybercrime financial relationships backward and forward, and WatchYourBack, a platform that implements it. Given a set of seeds, back-and-forth exploration produces a transaction graph that captures relations between entities that can serve as evidence of the relations identified. We apply back-and-forth exploration on 30 malware families and find relations in 93%. For 13% of the families, back-and-forth exploration identifies attribution points missed by forward-only explorations.
- We propose *operation oracles*, binary classifiers that identify malicious addresses using sequences of distinctive transactions. We show the utility of operation oracles by

applying them in two different scenarios: we use them during the explorations to identify addresses of four malware families signaling C&C servers to their botnets using Bitcoin, finding 13 addresses missed by previous works [157, 199, 93]; we also use them to scan the blockchain in a brute-force fashion to identify victim payments to the DeadBolt ransomware, uncovering previously unknown DeadBolt’s signaling and payment addresses.

- We quantify for the first time the impact of the lack of address coverage when estimating cybercriminal revenue. We use our DeadBolt operation oracle to show that the estimation with full coverage could be up to 39 times larger than another one produced with the partial coverage achieved by two popular Internet scanners [32, 190].
- We perform the first systematic analysis of cybercrime bitcoin revenue estimations and propose a technique to filter out self-deposits to Bitcoin addresses. We implement 15 different estimation methodologies and use them to show that our DC filtering reduces overestimation due to self-deposits and that some methodologies introduce massive overestimation. We finally select two estimation methodologies that produce the most accurate results, both with and without address expansion.
- We propose a novel majority-vote methodology for evaluating the accuracy of indicator extraction tools without the need for manually built ground truth and implement it in GoodFATR, an automated platform for threat report collection and IOC extraction. We use GoodFATR to collect 472,891 reports over 15 months and apply our majority-vote methodology for evaluating IOCSEARCHER, GoodFATR’s indicator extraction tool, against 7 popular tools. The results show that IOCSEARCHER is the most accurate in 84% of the IOC types evaluated.
- We propose a novel approach for classifying abuse reports and a taxonomy of 19 abuse types. We implement our approach in an abuse report classifier that selects the most fitting abuse type in our taxonomy for the reports. Our abuse report classifier achieves a precision of 0.92, a recall of 0.87, and an F1 score of 0.89 in an out-of-distribution experiment, showing that it performs well with diverse data. We demonstrate our abuse report classifier in two applications: providing financial loss statistics for fine-grained abuse types and generating new tagged addresses for WatchYourBack’s tag database.
- We openly release artifacts produced during our research:
 - WatchYourBack, our cryptocurrency analysis platform for tracing and revenue estimation [215]. It includes the tag database built from open sources, the ML classifier for identifying addresses that belong to exchanges, and four operation oracles for detecting addresses that belong to five malware families.
 - IOCSEARCHER, our IOC extraction tool [121] for extracting and validating 41 IOCs, including Bitcoin and other cryptocurrency addresses.

1.2 Thesis Organization

The remainder of this dissertation is as follows. We present the techniques used by previous works for Bitcoin transaction tracing, revenue estimation, IOC extraction and collection, and classification and analysis of cryptocurrency-related abuse reports in Chapter 2. Chapters 3 to 6 detail each of the thesis contributions. We introduce our techniques for exploring the Bitcoin blockchain to find relationships between real-world actors in Chapter 3. We present our systematic methodology comparison for cybercrime Bitcoin revenue estimation and the proposed techniques to get (possibly-)complete coverage of the addresses controlled by the DeadBolt ransomware family in Chapter 4. We present our threat intelligence approach for IOC extraction, collection, filtering, and validation, and our majority vote methodology for comparing the accuracy of IOC extraction tools in Chapter 5. We propose a novel approach for automatically classifying abuse reports related to cryptocurrencies and apply it to analyze pollution in abuse reporting services in Chapter 6. Finally, in Chapter 7, we present our conclusions and possible lines for future work.

Chapter 2

Background and Related Work

Almost since its conception, Bitcoin has suffered from abuse by cybercrime. However, due to its open, immutable, and public nature, it has also served to study such abuse to get valuable insights into cybercrime operations. Previous works have proposed different techniques to analyze Bitcoin cybercrime abuse. Commonly, such techniques leverage data extracted from the blockchain, sometimes called on-chain data, enriched by information found in public sources, referred to as off-chain data. In this dissertation, we call on-chain data to all information that can be extracted or inferred directly from the blockchain, like Bitcoin addresses, their transactions, and the clusters to which the addresses belong. We call off-chain data to all other data extracted from the rest of the world (e.g., the Internet, companies, users), useful to identify the ownership of the addresses in the Bitcoin ecosystem (e.g., services, abusers, and victims), which usually take the form of a tag database.

We first review different contexts of abuse in Bitcoin by cybercrime in Section 2.1. We review techniques largely based on Bitcoin on-chain data analysis, like clustering heuristics, forward transaction tracing, and techniques for estimating cybercrime revenue in Section 2.2. Finally, in Section 2.3, we review previous works that apply threat intelligence for collecting diverse indicators of compromise from cyber threat intelligence reports, like Bitcoin addresses used by cybercrime and other off-chain data. Additionally, we review previous works that have studied diverse forms of cryptocurrency abuse thanks to the information offered by abuse reporting services like BitcoinAbuse [15] and BBB Scam Tracker [184].

2.1 Bitcoin and Cybercrime.

Bitcoin has suffered from different types of abuse over time, mainly, but not only, as a cryptocurrency, e.g., to send and receive pseudo-anonymous payments. One of the earliest and probably the most notorious example of this is *The SilkRoad* [38], an online marketplace that also abused the Tor network [207] to remain untraced for two years. There, users could obtain diverse illegal goods and services, including illegal drugs, false identification documents, weapons, malware, illegal and violent sexual content, hacking services, hacked social media accounts, contract killings, and others. SilkRoad’s benefits in commissions worth hundreds of millions of dollars at the time it was alive (roughly 633k BTC [175]). Several other illegal

marketplaces that abuse Bitcoin and Tor in the same way have emerged since and some of them are still operative [187, 113, 112, 147].

Bitcoin is also the preferred payment channel of ransomware gangs, another notorious category of cybercrime [116, 45, 90, 151, 28, 79, 146, 131]. Ransomware is a type of malware that encrypts a set of critical files on a system to demand a ransom to decrypt them. The first massive and international ransomware attack, and thus probably the most widely known, took place in 2017, when *WannaCry* affected more than 230k computers in more than 150 countries and caused more than \$4 billion USD in damages. In recent years, most notorious ransomware families have targeted critical infrastructure of public and private sectors [102, 167, 168, 200, 77, 1], including health institutions, governments, and other major corporations. In 2020, *Ryuk* attacked several health services during the Covid-19 crisis; *Conti* attacked Ireland's health service in 2021; and *REvil* the medical insurer Medibank in 2022, additionally stealing and exposing sensitive personal medical records of its users. In 2021, *DarkSide* attacked and shut down the operations of Colonial Pipeline, a company responsible for nearly half the fuel supply for the East Coast of USA, causing a state of emergency in 18 states. On top of this, during the last years ransomware gangs have developed a double-extortion scheme in which victims are requested to make two payments, a first one to get a decryption key to recover their data, and second one to avoid leaking any exfiltrated data [146]. According to a private blockchain analytics firm, ransomware have extorted almost 2 billion USD from victims in just three years [203]. Although ransomware gangs have abused different platforms (like PayPal or Amazon gift cards) and cryptocurrencies (like Monero or ZCash) to receive the ransom, Bitcoin is still their preferred payment channel.

Another nowadays common form of online extortion where victims are coerced to deliver funds to the attackers is known as sextortion [152], an attack distributed by email in which the sender tries to convince as much victims as possible to be in possession of a footage that shows the victim in solitary sexual activity while visiting adult web sites, thus asking for a payment in Bitcoins for not releasing the footage to all the victim's relatives and friends [123]. Attackers have used data leaked in the past to perform targeted attacks, like user accounts and passwords, to try to convince the victims they first compromised their systems and then recorded the footage, and also have used spam platforms to massively reach potential victims with more generic emails. There exist variants of this attack also received by email in which the attacker uses different threats to try to convince the victim to pay, from social and psychological harm to physical harm, e.g., damaging the victim's reputation or attacking the victim with acid. The payment to stop the threat is commonly asked in Bitcoins [15].

Another category of cybercrime with both an important number of victims as well as a huge financial impact is composed by third-party service thefts. Unfortunately, we can find several examples of thefts at scale committed by external agents who exploit security vulnerabilities of third-party services. The largest thefts involve mainly cryptocurrency exchanges [57], which translates into several cryptocurrency users affected at the same time, sometimes thousands of them. A notorious example is the massive theft of 647k BTC, valued at nearly half a billion dollars, from one of the most popular cryptocurrency exchanges back in 2014, *Mt. Gox*, which ultimately led to the platform's collapse.

A different attack for stealing BTCs from users at scale involve the use of malware, like clip-

pers [138, 177]. Clippers are a set of malware families that monitor and replace cryptocurrency addresses in the clipboard of the operating system (OS) they infect. When a cryptocurrency user sends funds, the destination address (e.g., Bitcoin) is usually copied to the clipboard to then be pasted into some platform or software to proceed with the payment. Clippers replace the destination address copied by the user with one of their own as fast as it is detected on the clipboard, so if the user does not double-check the address being pasted (which would belong to the clipper), the funds will be sent to the clipper’s owner. Although clippers are very simple to implement in terms of source code, they are capable of stealing thousands of dollars [78] due they infect users at scale. Besides, clippers have been observed attached as modules of other malware families, which indicates that this type of attack will be more common in the future. Other malware families able to snatch BTCs include data stealers [105, 199], which focus on stealing credentials and cryptocurrency wallets stored in the system, to later access and steal the funds; and banking trojans [34], specialized in deceiving users when they interact with their online banking or cryptocurrency platform to steal their credentials or manipulate their transactions.

The use of deception as an alternative to coercion is also widespread, and several strategies to deceive users have been studied in the past. In a typical scam, victims are convinced to willingly transfer their own funds to the attackers for either an economical compensation (e.g., a reward or high profits) or to support a cause. To success, attackers use different tactics to make their victims trust them. For instance, in an investment scam, the perpetrators establish fraudulent investment platforms that promise high returns to deceive their victims [9, 10]. Commonly, the victims are requested to deposit commissions or fees in order to obtain the rewards or withdraw their money, but the perpetrators never return the original amount deposited by the victims, nor any profits. Scammers may impersonate celebrities, famous cryptocurrency supporters or experts, in order to promote fake rewards, as in giveaway scams, a common scam tactic where attackers offer a reward to users if they deposit to specific addresses. This attacks usually leverage phishing web sites and hacked social media accounts [213, 115]. Sometimes, attackers may impersonate official institutions to receive donations in name of others or simply use fake causes. Scammers may even pretend to be interested in establishing a sentimental relation with their victims, to request them money for supporting their travels or other expenses through cryptocurrency deposits, a common strategy in romance scams. Attackers also may set diverse fake platforms that offer false services to deceive their victims, impersonating for instance cryptocurrency exchanges [222].

Another type of scam that may affect users at scale is called *rug pull*, an strategy in which the attackers deploy some platform and deceive users into using it to manage in this way the funds of thousands of users (e.g., a service where users can keep their money or invest it, like a cryptocurrency-exchange). At some point, the attackers stop the platform and take all the funds. One of the most popular cases of what is believed to be a rug pull is QuadrigaCX [124], a Canadian cryptocurrency exchange that defrauded \$215M to roughly 76k investors.

Besides to stealing coins, attackers can also steal user’s computational power using cryptojackers [89, 86, 201], a type of malware that installs a miner software for exploiting the system’s local resources to produce benefits for the attackers [95, 180, 119]. Other forms of cryptojacking include installing javascript code in web sites to make users mine cryptocurrencies

while browsing the Internet [41], and exploit IoT devices, like routers, to build crypto-mining botnets [12].

Money laundering is another common form of Bitcoin abuse. Commonly, after receiving illegal coins, cybercriminals may *launder* the coins by abusing mixing techniques or mixing services, presumably before cashing out using other services. Actually, in the last years, the law enforcement has taken down several services related to money laundering [210, 54, 62]

A different type of abuse of the Bitcoin infrastructure is the one developed by malware authors, who have used it as a resilient mechanism for data propagation. The general idea is to store small pieces of data into the blockchain by crafting specific transactions that encode critical information, like IP addresses or domain names, wait for the transaction(s) to be included into the blockchain, and finally use some mechanism to retrieve blockchain information from the Internet. This technique is relatively simple to implement, for instance using stand alone tools, like a Bitcoin client, or by exploiting a combination of third-party services, like cryptocurrency exchanges (e.g., for receiving the transactions) and blockchain explorers, public web services that parse and publish into the Internet all transactions from specific blockchains like Bitcoin, e.g., blockchain.info [23]. Malware authors have implemented *Comand and Control (C&C) domain signaling* for botnets using this technique. They first store the next location of the C&C server into the blockchain, and then the bots can retrieve the location under demand by querying blockchain explorers, a mechanism that does not introduce new dependencies for the bots. C&C domain signaling has been observed implemented in the wild by diverse malware families, like cerber (ransomware) [157], pony (infostealer) [58, 156], skidmap (cryptojacker) [95], or glupteba (dropper) [82, 87, 136]. Cerber’s bots monitor withdrawal transactions of the signaling address being used. The next C&C’s domain name is formed by the first six characters of the last address receiving funds from the active signaling address, commonly complemented with the *.top* top-level domain. Pony and Skidmap use a similar technique based in a different encoding algorithm, which consists in monitoring the deposits to a signaling address. The amounts received by this address in two consecutive deposits encode an IP address. The encoding algorithm of Glupteba is quite different. It consists in storing an encrypted string into a transaction (using a special OP_RETURN Bitcoin address), which could be decrypted by Glupteba’s bots using a hard-coded decryption key embedded into the binaries. A different application of this data propagation mechanism is the one used by the DeadBolt ransomware [1, 55, 227] to automatically deliver decryption keys to victims. After a victim’s payment to a payout address, DeadBolt authors send a second mini-payment to the same address (only known by the victim) with the decryption key. For this, DeadBolt crafts a transaction that embeds an string (the decryption key) into an OP_RETURN address.

2.2 On-chain Bitcoin Analysis.

Due to the public nature of Bitcoin, previous works have proposed software tools for parsing and extracting information from it [22, 99, 84]. There are two main elements in the blockchain that enable on-chain analysis: addresses and transactions.

Bitcoin Address Ownership. The notion of *ownership* of a Bitcoin address is associated to the user who possesses its corresponding cryptographic key pair and can thus authorize a transfer of coins from that address. Addresses are easy to generate, and are costless, so a user may own multiple addresses. For example, a (privacy-aware) user may create a new address for every payment he receives. However, the current implementation of the Bitcoin protocol, based in UTXOs, mandates certain usage patterns that inherently leak information that can be used to undermine the privacy of the users by means of heuristics, which allow to group addresses based on their transactions.

Multi-input clustering. The most extended technique to identify Bitcoin addresses belonging to the same owner is called multi-input (MI) clustering [137, 6, 174, 130]. It assumes that input addresses to the same transaction have the same owner because their private keys are used together to sign the transaction. This heuristic follows from the fact that a user may not have enough coins in a single address for paying a certain value and thus need to accumulate coins from several of her own addresses to make the transaction valid. Clusters can be created transitively. If a transaction has addresses A and B as inputs, and another transaction has B and C as inputs, then A , B and C are all clustered together and have the same owner. One exception are CoinJoin [128] transactions where a group of users create a single transaction that simultaneously spends all their inputs into a (shuffled) list of outputs. For this reason, before computing MI clustering, we apply proposed heuristics to identify CoinJoin transactions [76, 99]. There exist other address clustering heuristics such as shadow address [6], change address [130], and its variants (peeling chain, power of ten, address reuse) [99]. However, those techniques rely on usage patterns that change over time and can lead to high false positives. We study the differences between blockchain clustering heuristics and their impact on estimations more deeply in Chapter 4.

Tagging. While MI clustering identifies addresses belonging to the same owner, it does not determine who the owner is. For this, commercial services (e.g., [36, 59]) provide paid access to tag databases. A tag is a tuple that associates an address with its owner or usage, commonly presented in the form of a label. Tags are scarce but important for detecting change of ownership and for the attribution task, so we build a tag database from public sources, and propose techniques to automatically increment it. We review the literature related to tags and the techniques for generating tag databases in Section 2.3.

Exchange address classifier. Previous works have applied machine learning and feature extraction techniques to try to characterize different benign and malicious entities in the Bitcoin ecosystem, proposing features focused on detecting mainly two type of entities: third-party services [225, 81, 118, 98] and malicious actors [225, 9, 81]. However, the data used for training and testing (e.g., the addresses with their labels), as well as the resulting models, are not made public, which difficult the implementation of the proposed approaches. Thus, we propose a set of 41 features useful to build a binary machine learning classifier from our tag database that is specialized in identifying addresses belonging to exchanges, even if the specific exchange the address belongs to was never tagged and is therefore not part of the training dataset. In contrast to previous works, we use our exchange classifier while exploring

malicious campaigns. Our exchange classifier prevents graph explosion due to limited tagging coverage, but can only determine a given address belongs to an exchange, not to which specific exchange. Our exchange classifier and the proposed features are detailed in Section 3.4.

Operation oracles. Previous works build functions that given a ransomware payment address distinguish victim payments by examining if its deposit amounts are within payment ranges used by the family [116, 45, 151, 90]. We expand this concept by proposing *operation oracles*, binary classifiers that identify specific types of malicious addresses (performing distinctive transactions) by applying rules to the sequence of transactions involving the address. We build operation oracles, detailed in Section 3.5 and Section 4.4, for five malware families: four that leverage the Bitcoin blockchain to signal C&C endpoints, and one ransomware family that uses Bitcoin transactions to deliver the decryption key to its victims. Our operation oracles consider transaction sequences, e.g., that address A sends a payment to address B and soon after address B returns the same amount, minus the transaction fee, back to A. Considering multiple transactions and dependencies between them makes the classifier more distinctive than considering only transaction values, reducing false positives (FPs).

Bitcoin Transaction Tracing. Forward transaction tracing focuses on the outgoing transactions performed by the seeds and transitively by their destination addresses [90, 151, 152, 113]. A main limitation of forward transaction tracing is that it only explores forward, but it does not explore backward. In particular, in forward transaction tracing, if an address is being traced, then the outputs of all transactions where that address is used as input will also be traced. But, other inputs in those transactions will not be traced. In addition, deposit transactions into a traced address, where the traced address appears as output, but no input is currently traced, are not traced either. Due to this limitation, forward transaction tracing cannot discover relations only reachable by exploring backward. For example, forward transaction tracing starting from addresses belonging to a mixer makes the exploration explode due to the mixing behavior, while exploring backward can reveal the clients using the mixer. In another example, exploring backward from a C&C signaling address can reveal the exchange used to fund the signaling. But, forward transaction tracing may not reveal such attribution point as the funds are consumed by the signaling.

To address this limitation we propose a novel *back-and-forth exploration*, which traces forward and backward. In a bit more detail, beginning from a seed, our back-and-forth exploration retrieves deposit addresses of incoming transactions, withdrawal addresses of incoming transactions, deposit addresses of outgoing transactions, and withdrawal addresses of outgoing transactions.

We illustrate the differences between forward transaction tracing and back-and-forth exploration using the example transaction graph in Figure 2.1. In the graph, circle nodes correspond to addresses and square nodes to transactions. Nodes filled in black are those belonging to a cluster tagged as exchange. Edges capture coins flowing from an input address to a transaction, or from a transaction to an output address. The exploration starts from the two seeds in gray. The graph is split in two. The right side corresponds to the graph that forward transaction tracing would produce and the left side is the additional subgraph that back-and-forth exploration would additionally discover, including a new malware address.

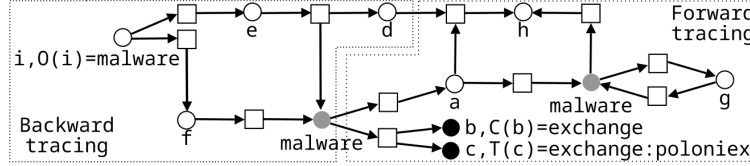


Figure 2.1: Example transaction graph from two malicious seeds in gray. Forward transaction tracing would discover the right subgraph, while back-and-forth exploration would discover the whole graph including a new malicious address.

There are two points where both explorations differ. First, back-and-forth exploration traces deposit addresses to the leftmost seed, including addresses f , e , and d into the graph. Second, once address a has been discovered through a forward step from the leftmost seed, the transaction where a is an input and h is an output will be traced. At that point, back-and-forth exploration would include address d in the tracing while forward transaction tracing would not.

One risk of backward exploration is that any entity can deposit BTCs into any address. If a benign entity sends funds to a malicious address, the backward exploration may consider the sender address part of the malicious campaign. Since addresses in ransom notes and scam emails receive victim's payments and are typically seeds for our back-and-forth exploration, our platform has an option to prevent exploration backward from the seeds. Note that even if this option is used, back-and-forth exploration still explores backward at intermediate nodes. For example, in Figure 2.1, even if backward exploration on the seeds is disabled, the final graph would be the same because the transaction between a and h would bring d into the graph, and successive backward steps would then include e , i , and f . There are two other cases where backward exploration can be problematic. One case are *forced address reuse* [64] (or *dust*) attacks, where an entity performs micro-payments to addresses it does not own hoping to incentivize the receivers to move those small amounts in later transactions [90]. In this work, we use some filtering heuristics to identify and remove forced address reuse transactions. The other case is researchers trying to hijack the C&C infrastructure of a malware by sending payments to the signaling addresses [199]. In such cases, researchers are likely to specify their transactions to avoid being incorrectly linked to the malicious campaign, allowing their filtering.

Change of ownership. A fundamental challenge when tracing a malicious campaign is to detect change of ownership, i.e., addresses that belong to different entities. A critical change of ownership is when BTCs flow into exchanges, e.g., cybercriminals cashing out their profits. Failure to detect an exchange address would make the graph explode by introducing thousands (or even millions) of transactions from other clients of the exchange. Detecting change of ownership is also fundamental to analyze relationships between the seed owners and external entities. In this case, failure to detect an exchange address would make it look like there exist relations between unrelated cybercriminal campaigns that simply use the same exchange. For example, in Figure 2.1 address b is identified as an exchange by our ML classifier, while c is identified as belonging to the *poloniex* exchange through a previously available tag. Failure to detect these addresses would have made the graph explode.

Revenue Estimation. In addition to finding financial relationships through transaction tracing, several works have also studied the economics of diverse abuse types by estimating their revenue from victim payments. The estimation process is relatively simple. It consists in analyzing the deposits received by the seeds, to determine the amount of coins flowing to them. However, there exist several factors that may influence the results, like the initial set of seeds or the period of time under study, the conversion ratios (i.e., to USD) or the conversion date used, the heuristics used for address expansion, for which different variants and implementations exist, or filters to account for a specific type of payment (e.g., ransomware payments). Surprisingly, although the blockchain is immutable, previous works that have focused in studying the same campaign (e.g., CryptoLocker [196, 116, 90, 45, 151]) often disagree in the estimation. Besides, most previous works have focused on one type of cybercrime, like ransomware [196, 116, 90, 45, 151, 146] hidden marketplaces [175, 195], scams [9, 222, 10, 115], or sextortion campaigns [152], making it challenging to compare different abuse types. To our knowledge, no previous work have systematically studied how these different factors influence the estimation. In Chapter 4, we survey previous works that estimate cybercrime campaigns and the different methodologies they apply to get insights about how methodologies may introduce noise to the estimation. From this analysis we derive two methodologies for improving revenue estimations, which better avoid both overestimation and underestimation, and we use them to compare different types of abuse types.

2.3 Off-chain Bitcoin Analysis.

Tags are one of the most effective ways for attributing addresses, so previous works have proposed techniques for building tag databases and techniques to expand them using on-chain information (e.g., clustering heuristics). Although tags are scarce, to our knowledge, no previous work has focused on automatically produce tags for previously unknown addresses to increment them. In the present work, we make the observation that Bitcoin addresses can be threatened as IOCs, so threat intelligence platforms can be useful for automatically collect and extract addresses in the wild. Extracting IOCs from natural language documents comprises two problems: threat report collection and IOC extraction. Most of the existing open source tools that focus on extracting IOCs from a given threat report use regular expressions (e.g., [172, 26, 173, 191, 96, 141]), and do not address the problem of threat report collection. On the other hand, prior academic works have proposed threat intelligence platforms to address both [228, 117, 226, 158], but none is focused on collecting and extracting cryptocurrency-related IOCs. In Chapter 5 we summarize the most popular open-source IOC extraction tools and detail how they work in Section 5.4. Then, we propose a threat intelligence platform that supports cryptocurrency address collection, which we use to increment our tag database.

Besides identifying new addresses in the wild, a different challenge is to give them a category, for instance by analyzing the context where the address was found. Related to this task, previous works have applied NLP techniques to identify and classify different types of IOCs [117, 228, 94, 226]. However, the categories used are not normalized, but depend on the context itself. Moreover, most of the IOC extraction tools they use do not support cryptocurrency addresses. In Chapter 6 we collect data from online platforms where users report cryptocurrency-related abuses and leverage an LLM to automatically classify user reports. Combining the techniques

proposed in both Chapters allows to automatically increment the tag database. In the rest of this section we will introduce concepts related to address tagging, as well as automated IOC collection and extraction. Then, we will introduce concepts related to automatically classifying addresses using their context, by leveraging natural language processing techniques.

Tagging. The most reliable way, although the slowest and most expensive, to get attribution tags is by directly interacting with an entity, e.g., by registering into services like exchanges or online shops to directly transact with them to obtain some of their Bitcoin addresses [130], scraping hidden marketplaces’ websites [38, 195], extracting addresses embedded into malware samples [157, 199], or executing malware samples in a sandbox to obtain fresh ransom notes [90]. On the other hand, it is also possible to collect attribution data to generate tags from online sources, like forums [19, 17], blogs [70, 1, 71, 54, 13], threat reports [177, 95, 67], previous scientific literature [9, 223, 222, 115], abuse reporting services [15, 184, 49], and other specialized websites and platforms [212, 84, 75]. Additionally, addresses related to cybercrime can be found on the wild. For instance, ransom notes produced by ransomware contain payout addresses [196, 103, 116, 90, 45, 151, 131], as well as some extortion campaigns [123]. Other methodologies used to obtain cybercrime-related attribution data include working with data providers [152], deploying crowd-sourcing platforms [146, 30], or monitoring websites offering fraudulent promotions and the addresses they use [115].

Different Bitcoin analysis platforms that implement tag databases have been proposed in the past [22, 99, 84]. Although an intrinsic limitation of tag databases is that they only include a small fraction of Bitcoin addresses, having thus a low coverage, such platforms do not focus on techniques for automatically increase the coverage of tag databases, which requires addressing two main challenges: the identification of new addresses and their classification (e.g., giving them a category). Additionally, tag databases are commonly private and difficult to access. Thus, we leverage public resources to build our own tag database and study different techniques to increasing its coverage without manual intervention. In Chapter 5 we present a threat intelligence platform for automatically identifying Bitcoin addresses and other IOCs on the wild, which allows to increase the coverage of tag databases using unclassified tags. In Chapter 6 we explore NLP techniques and LLMs for the classification task. For this, we collect abuse reports from victims, filter reports that does not correspond to an abuse report, extract a category for the text of the report, and finally use them to tag reported addresses.

Double ownership. Online wallets are special Bitcoin addresses that are managed and owned by third-party services, but at the same time are assigned in exclusivity to their users for transacting within their platforms. So, in a sense, online wallets have more than one owner and it is fundamental to handle *double ownership* for avoiding certain types of conflicts when tagging addresses. In the double ownership scheme, a first user (i.e., the *owner*) possesses the cryptographic key pair corresponding to the address and accepts transaction requests (possibly authenticated with e.g., username and password) from a second user (i.e., the *beneficiary*) who is thereby the one governing the funds received into the address. However, every deposit received into the online wallet will be managed by the owner, who will decide where the coins are transferred and when. For instance, a common practice among exchanges and other services is to centralize any coins deposited to online wallets they manage into *hot wallets*,

Work	Year	Collection Sources				Extraction	
		Blogs - Crawlers	Blogs - RSS	Telegram	Twitter	Regexp	NLP
Liao et al. [117]	2016	✓	✗	✗	✗	✓	✓
TTPDrill [94]	2017	✗	✗	✗	✗	✓	✓
Malpedia [158]	2017	✓	✗	✗	✗	✓	✗
ChainSmith [228]	2018	✓	✗	✗	✗	✓	✓
IOCMiner [139]	2019	✗	✗	✗	✓	✓	✗
TIMiner [226]	2020	✓	✗	✗	✗	✓	✓
Alves et al. [4]	2020	✗	✗	✗	✓	✓	✗
Twiti [188]	2021	✗	✗	✗	✓	✓	✓
GoodFATR	2022	✗	✓	✓	✓	✓	✗

Table 2.1: Summary of prior academic works on threat report collection and IOC extraction.

special wallets owned and controlled by the service, with no beneficiary, used to manage funds of several users. Meanwhile, the user will maintain a virtual balance into the platform to use it at demand. Note that the owner does not share the key pair with the beneficiary, so the beneficiary cannot produce transactions on its own. Handling double ownership is fundamental when collecting tagged addresses from multiple public sources. For example, the same address may be tagged in one source as belonging to an exchange and in another source as being the replacement address for a clipper malware. Without taking double ownership into account that would create a conflict in assigning ownership to the address. Instead, the address should be tagged as belonging to an exchange (i.e., the owner) that has assigned that address to one of its users (i.e., the beneficiary) who happens to be the clipper’s operator. Our tag database construction is detailed in Section 6.6.2.

Threat Report Collection. We compare prior academic work that has collected threat reports in Table 2.1. As shown by the table, previous works collect reports from one source, either from cybersecurity blogs [228, 117, 226, 158] or from Twitter [139, 4, 188]. There is also one work that does not address threat report collection, focusing exclusively on IOC extraction [94]. In contrast, in Chapter 5, we propose a modular threat report collection platform to collect reports from multiple sources, namely cybersecurity blogs, Telegram, Twitter, and datasets provided by prior works such as ChainSmith [228], Malpedia [158], and APTnotes [8].

Previous work that collected threat reports from cybersecurity blogs built dedicated crawlers for each blog of interest (e.g., [228, 117, 226]). The advantage of dedicated crawlers is that they can be designed to collect the historical archive of reports available on a blog’s website. One disadvantage is that a crawler needs to be developed for each blog, which limits the number of monitored blogs. For example, Liao et al. [117] collected reports from 45 blogs, Zhu and Dumitras from 10 [228], and Zhao et al. [226] from 75. Moreover, as websites evolve, the crawlers may break and need to be updated. For example, ChainSmith originally collected reports from 10 blogs in 2018, but the supported blogs gradually decreased over time, possibly due to crawlers breaking due to website updates. In mid-2020 only two blogs were still being

crawled, and since mid-2021 ChainSmith does not collect any new blog posts.

Rather than building dedicated crawlers, we propose a platform for collecting threat reports from cybersecurity blogs that offer an RSS feed. The use of RSS feeds allows us to scale the collection to 300 blogs, 6 times larger than prior works. Furthermore, our RSS module still collects entries from the same set of blogs supported by ChainSmith, showing that those origins are still active. It is important to note that our RSS feed collection may also require updates (*e.g.*, if a feed’s XML URL changes), but updating a feed’s URL is significantly easier than building a new dedicated crawler. Furthermore, if a blog does not offer an RSS feed, it will still need a dedicated crawler.

IOC Extraction. All open-source IOC extraction tools and three academic works analyzed in Chapter 5 [158, 139, 4] extract IOCs from the collected reports using regular expressions. Some academic papers have instead proposed natural language processing (NLP) approaches to improve the extraction [117, 228, 94, 226]. However, NLP approaches also rely on regular expressions to extract indicators in their pipelines. Despite the popularity of regular expression-based IOC extraction, the large number of tools a user can choose for this task, and the impact of regular expression differences in the extraction results, we are not aware of any prior work that systematically analyzes which tool works for each indicator type. Such a comparison should include the regular expression used for the extraction, but also other important steps such as how to filter benign indicators that are not IOCs. The main difficulty for such evaluation is building large-scale datasets from real documents. To address this challenge, in Chapter 5 we propose a novel methodology to compare indicator extraction tools, which does not require a manually-built ground truth.

Also related are metrics to evaluate the quality of IOC feeds [114]. We apply such metrics to compare different sources and origins to identify the best and worst IOC contributors.

Cryptocurrency abuse. Previous works have analyzed a wealth of cryptocurrency abuse types including ransomware [196, 116, 45, 90, 151], sextortion [152], scams [9, 130, 222, 10, 115], clippers [73], cryptojacking [89, 201], hidden marketplaces [38, 113, 175], human trafficking [160], money laundering [134], and thefts [130]. These works analyze a single abuse type, whereas our work analyzes 19. Other works compare the revenue obtained by different abuse types [78, 176]. These works require addresses to be tagged with an abuse type, which does not happen in the abuse reporting services analyzed. To address this issue, we present a novel classification approach to assign abuse types to victim reports and then select a type for the address through a majority vote.

BitcoinAbuse. Previous works have analyzed BitcoinAbuse reports [27, 37, 176]. Those works analyze issues such as the presence of heavy reporters [37], the concentration of revenue on a few addresses [27], and the revenue per abuse type [176]. However, they did not consider the data pollution issues that we measured. In particular, they did not filter out spam reports, considered all reported addresses to be malicious (and thus their deposits to be cybercriminal revenue), and that the reporter-selected abuse types were correct. Our results cast a shadow over their conclusions.

Chapter 3

Identifying Cybercrime Financial Relationships

3.1 Introduction

Cryptocurrencies such as Bitcoin are attractive for cybercriminals due to their decentralized nature, (pseudo-)anonymity, irreversible transactions, and the ease to buy and sell them. However, some cryptocurrencies such as Bitcoin have a public transaction ledger, which has enabled the analysis of diverse cybercrime activities such as ransomware [157, 116, 45, 90, 151, 196], thefts [130], scams [152, 9, 130], human trafficking [160], hidden marketplaces [38, 113, 175], money laundering [134], and cryptojacking [201].

A key component of a Bitcoin cybercrime investigation is the analysis of how money flows, e.g., how cybercriminals finance their activities and where the profits of those activities go. Identifying the financial relationships (relations for short) between a cybercrime campaign and external services (e.g., exchanges, mixers, underground markets), or with other cybercriminal campaigns, is fundamental in attribution. For example, if a coin flow is identified between a malicious campaign of interest and a service (e.g., an exchange) with know-your-customer (KYC) requirements [74], the service can be leveraged as an attribution point by law enforcement to identify the perpetrators. Moreover, identifying relations between cybercrime campaigns, which were initially considered independent, sheds light into services supporting cybercriminal activities [39], and their operators, which become then targets for takedown [210, 62, 54]. Yet another application is identifying the malicious clients of a specific service, e.g., a mixer suspected of laundering money from illicit activities. This information has been fundamental in past takedown actions, by justifying to a judge that the service is a hotspot of malicious activity. For example, the arrest warrant for the *Bitcoin Fog* mixer operator was justified on its abuse for money laundering by multiple underground markets [11].

A common technique for analyzing Bitcoin abuse is *forward transaction tracing*, i.e., tracing the flow of BTCs starting from a given set of *seed addresses* known to belong to a cybercrime campaign [90, 151, 152, 113]. Examples of seed addresses are those in ransom notes (e.g., [90, 151]) and those in scam emails (e.g., [152]). But, forward transaction tracing has two

fundamental limitations. First, it only considers forward propagation of the funds from the seeds, i.e., it does not explore backward. For example, if the seed is used as input to a transaction with two inputs and one output, it traces the output, but it does not trace (backward) the other input, which we can also attribute to the campaign since multiple inputs of the same transaction are associated to the same actor [130]. Second, if the output address receives another deposit transaction (where the seed is not an input) that deposit transaction is ignored. Thus, forward-only exploration may often miss fundamental relations. For instance, malware that uses the Bitcoin blockchain for signaling C&C servers (e.g., [157, 199]) needs to seed the signaling addresses with funds. Failure to explore backward from the signaling addresses may miss the exchange used to seed the C&C signaling that can be leveraged as an attribution point. Furthermore, since the funds are consumed to pay the fees of the signaling transactions, there may not be a cash out that forward-only exploration can identify. Another situation is when examining the clients of a potentially malicious service, e.g., a mixer. Forward-only exploration from the mixer addresses is useless as the mixer’s goal is precisely to prevent forward tracing. However, backward exploration from the mixer’s addresses allows identifying the clients that leverage the mixer, which can be used to justify an arrest warrant [11]. Last but not least, existing approaches perform shallow explorations to prevent the transaction graph from exploding in size. We posit that the key challenge to prevent such explosion, and also for identifying relations, is to identify *change of ownership*, i.e., when BTCs change hands from the operators of the campaign under study to other entities. Prior works detect change of ownership using commercial databases of tagged addresses [36, 59]. However, such databases have limited coverage, which can make the graph explode. For example, failing to identify that the cybercriminals cashed out at an exchange would include a massive amount of transactions from other innocent clients of the exchange into the transaction graph.

To address these limitations, we contribute the following:

- We propose *back-and-forth exploration*, a novel automated transaction tracing technique to identify cybercrime financial relationships. Given a set of seed addresses belonging to a cybercrime campaign, it outputs a transaction graph and it extracts paths in the transaction graph that capture relations to other entities and campaigns. The transaction graph can be used as evidence of the relations identified, and can be shared with law enforcement and other analysts for independent validation, or with a judge to justify a warrant. Moreover, the relations could be publicly shared, preventing services (e.g., exchanges) that interact with the cybercrime campaigns from claiming that they were unaware of the abuse. Back-and-forth exploration provides two key contributions over current forward transaction tracing techniques. First, it explores both forward and backward, enabling the discovery of important relations missed otherwise. Second, it combines the use of a tagging database with a machine learning classifier to identify addresses belonging to exchanges. The classifier can detect previously unknown (i.e., untagged) exchange addresses, preventing graph explosion.
- We propose *operation oracles*, binary classifiers that can identify specific types of malicious addresses using sequences of distinctive transactions. We build operation oracles for four malware families that use the Bitcoin blockchain for signaling C&C servers.

We use them to identify signaling addresses in the transaction graphs, demonstrating that back-and-forth exploration can discover 13 signaling addresses missed by previous works [157, 199, 93] and that backward exploration is key for their discovery.

- We build a database of 230K tagged addresses from open sources and expand it into 116M tagged addresses through multi-input clustering. We show that tag databases need to handle *double ownership*, i.e., Bitcoin addresses with more than one owner. Double ownership is common in services offering online wallets, which create an address for a specific user. In that scenario the service owns the address (i.e., has the private key), but the address is handled (and thus indirectly owned) by the user for which it is created.
- We apply back-and-forth exploration on 30 malware families: 19 clippers that replace addresses in the OS clipboard of the infected host with an address controlled by the malware operators, 7 ransomware, and 4 families using Bitcoin for C&C. Back-and-forth exploration finds relations in 93% of the explorations. For 13% of the families, it identifies new attribution points, missed by forward-only explorations. It uncovers that 23 replacement addresses for 9 clippers are online wallets in exchanges (double ownership). This shows that clippers, a largely overlooked, but highly effective, malware class, often directly cash out the stolen funds without intermediate steps. It is surprising that the abusive nature of those addresses goes unnoticed for the exchanges. Overall, the most common relations are with 44 exchanges. Exchanges used by multiple families (e.g. *localbitcoins*, *coinpayments*, *coinbase*, *poloniex*, *cryptonator*, *btc-e*) are often not the largest ones, which could indicate a more lax approach to abuse detection. In fact, one of those exchanges has already been taken down [210]. Other identified relations are 2 mixers (*wasabi*, *bitcoinfog*), the latter being recently taken down for money laundering [54]; 11 gambling sites likely also used for money laundering; 5 payment services used for acquiring resources; 4 mining pools, and 4 underground markets. In 7 explorations, relations are found between a malware family and other cybercrime entities. We observe that some malware operators participate in a variety of cybercriminal activities. For example, the *MrPr0gr4mmer* operator sells spam tools, sends sextortion scam campaigns, and handles a clipper. We also identify clippers run by the same operators and observe malware operators using profits to upgrade their software arsenal (e.g., buying licenses for malware kits and spam tools) and for seeding new C&C signaling approaches. Finally, we perform a backward-only exploration on the *bitcoinfog* mixer identifying 15 underground markets, 130 other services, and 15 malicious operations that leveraged it as clients.
- We release the source code of our blockchain analysis platform, along with our ML classifier and tag database [73].

3.2 Back-and-Forth Exploration

Given seed Bitcoin addresses belonging to a cybercrime campaign, back-and-forth exploration outputs a transaction graph and identifies paths in the graph that capture relations to other entities and campaigns. As illustrated in Figure 2.1, nodes in the transaction graph are of

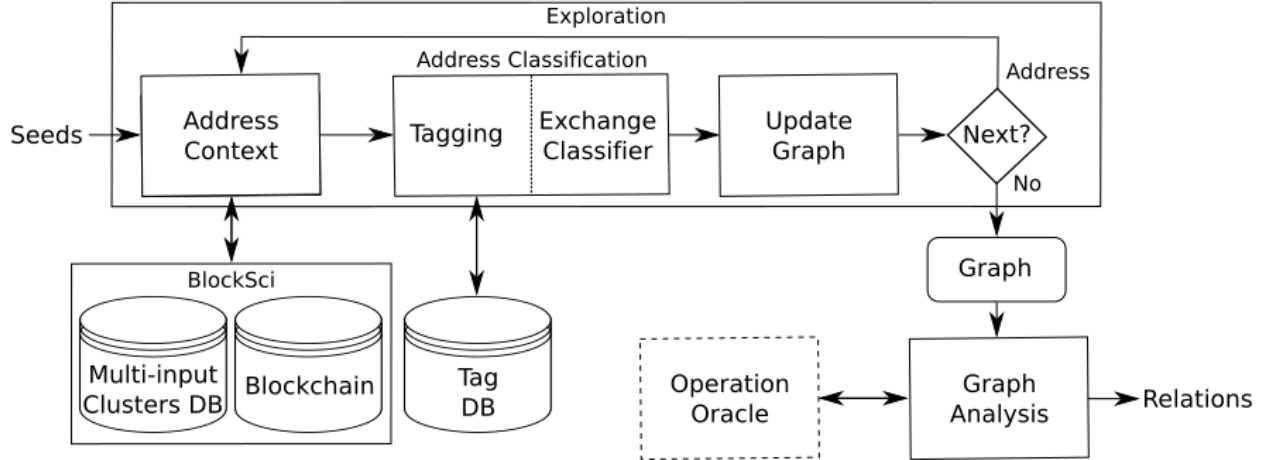


Figure 3.1: Back-and-forth exploration architecture.

two types: addresses and transactions. Edges are directed; they may connect an output slot in a transaction to the address where funds are received, or connect an address to the input slot in a transaction where funds are sent to. If the same address appears several times as input or output of the same transaction, a single edge connects address and transaction. Address nodes are reused if the same address appears in multiple transactions. Address nodes have attributes capturing their alphanumeric address string, the list of tags associated to the address (if any), the identifier of the MI cluster the address belongs to, and the address type (e.g., if detected by the exchange classifier or by the operation oracle). Transaction nodes have as attributes the transaction identifier and the total BTC amount transferred. Edges have as attributes the transaction ID, the slot number, and the BTC amount transferred in that slot.

The back-and-forth exploration architecture is shown in Figure 3.1. It performs an iterative process. At each iteration, it selects a Bitcoin address from the worklist and performs three steps. First, it obtains the address context by identifying the MI-cluster the address belongs to and extracting from the blockchain its deposit and withdrawal transactions. The blockchain data is processed using the BlockSci framework [99]. Before launching any exploration, BlockSci is used to parse the blockchain data up to the desired block height and to pre-compute the MI clustering results. Second, it classifies the address to check for change of ownership using the input tag database (detailed in Section 6.6.2) and the exchange classifier (detailed in Section 3.4). Finally, it updates the transaction graph with the obtained transactions and addresses; and adds to the worklist any new addresses that need to be explored. The process starts by placing the seeds in the worklist and finishes when the worklist is empty, or a configurable exploration limit is reached. At the end of this iterative process the transaction graph is output.

The graph analysis module takes as input the transaction graph and an optional operation oracle. It outputs relations between the campaign under study and other entities in the graph. First, it tags addresses satisfying the oracle, e.g., as a C&C signaling address. Then it identifies relations, which are paths between a seed and a tagged address that satisfy the following constraints. First, the found tag should differ from the seeds' tag. Second, the path

should be directed from the seed to the tagged address, or from the tagged address to the seed, i.e., it should not mix edges with opposite directions. This constraint makes sure that the relation captures money flow from the campaign under analysis to an external entity or vice-versa. Third, addresses identified by the exchange classifier, for which no tags were available, are excluded since it is not known to what specific exchanges they belong to.

Address context. At each iteration, the address being explored is used to query BlockSci to obtain the ID of the MI cluster to which the address belongs. If the cluster has already been examined, e.g., if detected as an exchange when examining a previous address in the cluster, its classification results are recovered. Then, BlockSci is queried again to obtain the list of all transactions involving the address. CoinJoin and forced address reuse transactions are filtered from the exploration. The former are identified using a set of previously proposed heuristics [76, 99] implemented by BlockSci. The latter are identified using our own heuristic, which looks for transactions that send the same amount to a large number of (at least 100) output addresses. Next, it obtains from BlockSci the list of addresses involved in the remaining transactions, both in input and output slots. Transactions and addresses are added to the transaction graph and passed to the address classification that determines which addresses should be explored in later iterations.

Address classification. Each address received from context extraction is examined for change of ownership. Addresses identified as belonging to other entities, either through tags or by the exchange classifier, will be labeled as such and will not be traced. The rest will be added to the worklist to be traced. First, the address is queried to the tag database. If tagged, and if its tag category identifies it as a service (e.g., exchange, tormarket, gambling, etc), the address is not added to the worklist. There is no need to keep exploring the address since that would explore the service, introducing many extraneous nodes in the transaction graph. Second, the ML exchange classifier, detailed in Section 3.4, is applied to the address. If detected, the address is not added to the worklist. As an optimization, the ID of the address' MI-cluster is cached, so that any other address in the same cluster can be directly marked as an exchange without going through the classification. Addresses added to the worklist are assigned a priority inversely proportional to the total input and output slots in their transactions. The address with highest priority is selected and another iteration starts. The intuition behind the priority is that addresses with many transactions, or with very large transactions, could belong to services not in the tag database or to false negatives of the exchange classifier. To avoid the exploration being stuck exploring those addresses, their priority is lowered so that they are explored at the end. The priority only affects the order of the exploration as long as the exploration runs to its end. However, if the user sets a maximum exploration limit, high volume addresses may not be explored before the limit is reached and some relations may be missed.

3.3 Tagging

Tag database construction comprises of two steps. First, we collect 230K addresses from public sources and tag them using their context. Then, we propagate the address tags to

Tags	Source
58606	https://blog.talosintelligence.com/2018/10/anatomy-of-sextortion-scam.html
57014	https://www.bitcoinabuse.com
37437	https://github.com/MatteoRomiti/Deep_Dive_BTC_Mining_Pools/blob/master/dataset/miners.json
36548	https://github.com/nopara73/WasabiVsSamourai/blob/master/WasabiVsSamourai/SamouraiCoinJoins.txt
10203	https://www.cs.princeton.edu/~yuxingh/ransomware-public-data/cerber-locky-addresses.csv.txt
9412	https://raw.githubusercontent.com/avast/ioc/master/Clipsa/appendix_files/btc_addresses_complete.txt
7291	https://zenodo.org/record/1238041
6594	https://bitcoin-otc.com/viewgpg.php
3907	https://github.com/MatteoRomiti/Sextortion_Spam_Bitcoin
356	https://www.walletexplorer.com
321	https://www.justice.gov/opa/press-release/file/1304296/download
312	https://raw.githubusercontent.com/VHRanger/tether/main/data/known_addresses.json
295	https://gist.github.com/MrChrisJ/4a959a51a0d2be356cc2e89566fc1d87
223	https://bitinfocharts.com
207	https://bitcointalk.org/index.php?topic=576337
121	https://chain.info/richlist
115	https://medium.com/meetbitfury/crystal-blockchain-analytics-investigation-of-the-zaif-exchange-hack-a3b4d1faed8f
115	https://explorer.crystalblockchain.com
92	https://wbtc.network/dashboard/audit
58	https://research.checkpoint.com/2019/in-the-footsteps-of-a-sextortion-campaign

Table 3.1: Top-20 tag source URLs.

their MI clusters obtaining 116M tagged addresses. After each step there is a disambiguation process to resolve any conflicts due to multiple tags assigned to the same address or cluster. We explore a variety of publicly available sources to tag Bitcoin addresses including databases of already tagged addresses, (e.g., WalletExplorer tags [212] available in GraphSense [84]), web pages where services disclose their addresses (e.g., the SatoshiDice gambling service [182]), forums where users rate each other (e.g., BitcoinTalk [19], Bitcoin-OTC [17]), crowd-sourced databases where users report abusive addresses (e.g., Bitcoin Abuse [15]), and security blog articles [29]. The different sources have little overlap and each source provides some addresses that do not appear in any other source. Thus, they all contribute to making the tag database as complete as possible. Some of these sources such as WalletExplorer and Bitcoin-OTC provide their own tagging database, which can be automatically imported into our platform. For others like BitcoinTalk, tags are produced by manually inspecting specific reports. For Bitcoin Abuse, the category is provided, but selecting the label requires manually inspecting the reports. Updating the tagging database needs to be done periodically and is the only step in our approach that requires human involvement. In future work, we plan to explore automating this step using NLP. Table 3.1 details the top 20 source URLs by number of tagged addresses.

Each database entry is a 6-tuple (*address*, *class*, *category*, *label*, *subtype*, *urls*). There are 28 categories, detailed in Table 3.2, which can be grouped into 5 classes: malware, abuse, services, individuals, and benign. Label captures the ownership, i.e., the specific malware family, service, or individual that owns the address. Subtype is an optional component that details the usage of the address, e.g., if it is the hot wallet of an exchange or a malware C&C signaling address. The final component is the list of source URLs that discuss the address. For example, address *1NDyJt* has a category *exchange*, label *binance* (the specific exchange), hot wallet as subtype, and a URL to the WalletExplorer webpage that describes it. Throughout this chapter, when we refer to a tag we mean its category and label, e.g., *exchange:binance*. We identify addresses by their first 6 characters, full addresses can be found in the repository [215].

Category	Class	Addresses		Multi-Input Clustering	
		All	wTXES	Clusters	Addresses
sextortion	Abuse	62,600	407	171	1,131
miscabuse	Abuse	57,031	12,018	7,568	8,656,754
mining	Service	37,454	37,453	29,526	594,609
mixuser	Individ.	36,548	36,548	7,308	36,860
ransomware	Malware	17,668	17,447	10,348	21,160
clipper	Malware	9,513	464	105	1,376
username	Individ.	6,618	5,399	5,110	153,826
exchange	Service	1,301	1,299	737	75,257,530
terrorism	Abuse	275	275	48	306
theft	Abuse	221	220	129	2,072,269
gambling	Service	182	182	106	8,800,225
onlinewallet	Service	140	129	37	4,612,732
defi	Service	93	17	17	15,945
scam	Abuse	72	61	32	314,537
mixer	Service	56	55	52	2,137,493
ponzi	Abuse	54	54	41	20,908
service	Service	50	50	43	1,293,660
malware	Malware	19	18	17	38
tormarket	Service	19	19	19	2,327,272
payment	Service	15	15	12	9,717,967
donation	Benign	14	12	8	217
state-sponsored	Abuse	14	14	8	66,855
drugtrafficking	Abuse	11	11	2	38
bankingtrojan	Malware	3	1	1	1
cryptojacking	Malware	3	3	2	25
miscbenign	Benign	3	3	1	2
webskimming	Malware	3	3	3	6
usgov	Benign	2	2	2	3
All		229,982	112,179	61,453	116,103,745

Table 3.2: Tag database summary.

We initially assign a tag to an address for each source that mentions it. Then, we select a unique address tag by applying the following criteria for resolving potential ambiguity in addresses with multiple tags. If an address has multiple tags with the same category and label, we favor the tag that provides more information, i.e., the one with a subtype, or the longest one otherwise. If an address has multiple tags with different category and label pairs, we favor tags not coming from Bitcoin Abuse because categories in that service are assigned by users with varying expertise and it is not rare that they are wrong. If an address has multiple tags of the mining category, we merge its labels to capture all the mining pools where the miner has participated. In any other cases, we manually resolve the inconsistency. If we cannot confidently assign a tag, we remove the address from the database. When applying this procedure to the 230K addresses, only 66 (0.03%) required manual inspection and we could disambiguate those, showing that this strategy is effective.

Next, we obtain the MI clusters for all the tagged addresses. If a single tagged address appears in the cluster, we assign that tag to all other addresses in the cluster. If multiple tagged addresses appear in the same cluster, we apply the following disambiguation process. If a cluster has only one service tag, we assume this is a case of double ownership where the owner of the address is the service, and the beneficiary is the owner of the other tags. If multiple service tags are present, we try to manually disambiguate them, e.g., checking if the two services are related to each other. For cases where no service tags are present, we compute the edit distance between their labels. If they are close, e.g., user:Metabank.ru and user:Metabank, we consider them aliases and merge them. Otherwise, we manually try to

Class	Source	StAddr.	Entities	MI Clust.	Addr.
positive	WalletExplorer	167	117	151	54,265
negative	Bitcoin-OTC	923	923	788	38,702
negative	SatoshiDice	40	1	13	1,154
negative	Ransomware	7,223	65	78	14,409

Table 3.3: Dataset of 108,530 addresses used for training and testing exchange classifier.

disambiguate them. If manual disambiguation fails, we do not expand the tags to the cluster. Applying MI clustering to the 230K tagged addresses produces 61K clusters comprising of 116M addresses. Of the 61K clusters, 817 (1.15%) have more than one tag, from which 368 have only miscabuse tags from Bitcoin Abuse and 286 can be automatically solved since they contain multiple tagged addresses of the same owner. Of the remaining 163 clusters, we could manually resolve conflicts in 91 corresponding to 54 services with double-ownership (e.g. exchanges) and 37 equivalent tags (e.g., usernames with two aliases). We could not resolve 72 clusters, for which we do not propagate tags. Of those, 66 were clusters without service tags (63 of them with multiple Bitcoin-OTC user tags) and 6 clusters with more than one service tag (4 of them with multiple mining tags). An interesting case is one of the clusters with multiple service tags. It is due to the special address 3J98t1, which corresponds to the empty string private key. Surprisingly, that address occasionally receives deposits possibly because it appears in many Bitcoin tutorials. Any Bitcoin user can spend funds from this address (since the private key is not really secret), and many users actually do, creating a MI cluster with 13M addresses. In summary, only 0.27% of the 61K clusters required manual inspection, and we could expand tags for 99.3% of all tagged clusters, showing that our strategy is effective.

3.4 Exchange Detector

We build a binary ML classifier to identify if an address belongs to an exchange in three steps. First, we build a dataset of 108K labeled addresses. Then, we propose 42 features and use them to train different models. Finally, we evaluate the accuracy of the models.

Dataset. To train and test the classifier we build a dataset of 108,530 labeled addresses, summarized in Table 3.3. Half of them (positive class) are addresses tagged as exchanges by WalletExplorer [212]. The other half (negative class) are non-exchange addresses belonging to individuals, ransomware, and gambling sites. Individuals correspond to Bitcoin-OTC [17] users with a registered Bitcoin address and GPG key, and having received ratings. To avoid abusive users we require that they have received less than 100 negative ratings and have an overall positive rating. Ransomware addresses come from [151] and gambling addresses belong to SatoshiDice [182]. We obtain the MI clusters of those starting addresses and select additional addresses from the clusters until the dataset is balanced. We use 80% of the dataset for training and the other 20% for testing.

Model training. We extract 42 features for an address, summarized in Table 3.4. Features can be grouped into three classes. The 29 transaction (TX) features capture properties of the transactions where the address appears such as number of (all/deposit/withdrawal/CoinJoin)

Feature	New	Type	Class	Description
type	✓	Cat.	AD	Address type
equiv_addrs	✓	Int.	AD	Number of equivalent addresses
lifetime	✗	Int.	TS	Total seconds from first TX to last
timespan_d	✓	Int.	TS	Seconds from first deposit TX to last
timespan_w	✓	Int.	TS	Seconds from first withdrawal TX to last
activity	✗	Int.	TS	Number of days with at least one TX
activity_d	✓	Int.	TS	Number of days with at least one deposit TX
activity_w	✓	Int.	TS	Number of days with at least one withdrawal TX
idle_time	✓	Int.	TS	Number of days without TXES
daily_d_rate	✓	Float	TS	Rate of deposits per day in its lifetime (tx/day)
daily_w_rate	✓	Float	TS	Rate of withdrawals per day in its lifetime (tx/day)
yearly_d_txes	✓	Float	TS	Average number of deposit TXES per year
yearly_w_txes	✓	Float	TS	Average number of withdrawal TXES per year
balance	✓	Int.	TX	Balance to date, in satoshis
deposited	✗	Int.	TX	Total satoshis deposited/received into address
withdrawn	✗	Int.	TX	Total satoshis withdrawn/sent from address
txes	✗	Int.	TX	Total number of transactions
txes_out	✗	Int.	TX	Number of TXES with this address as output
txes_in	✗	Int.	TX	Number of TXES with this address as input
addr_as_change	✓	Float	TX	Ratio of TXES address used as input and output
outputs	✓	Int.	TX	Total output slots in which the address appear
inputs	✓	Int.	TX	Total input slots in which the address appear
utxos	✓	Int.	TX	Number of unspent transaction outputs to date
tx_size_mean	✓	Float	TX	Average total size (in bytes) of all its TXES
tx_weight_mean	✓	Float	TX	Average weight of TXES
tx_fee_mean	✓	Float	TX	Average fee paid in TXES
ins_age_mean	✓	Float	TX	Average blocks between input and spent output
coinbase	✗	Int.	TX	Number of TXES that are coinbase
coinjoin	✓	Int.	TX	Number of TXES that are coinjoin
coinjoin_out	✓	Int.	TX	Coinjoin TXES with this address as output
coinjoin_in	✓	Int.	TX	Coinjoin TXES with this address as input
tx_ratio	✗	Float	TX	Number of input TXES divided by output TXES
outs_per_tx	✓	Float	TX	Average number of output slots per TX
ins_per_tx	✓	Float	TX	Average number of input slots per TX
outs_per_out	✓	Float	TX	Average number of output slots per output TX
ins_per_out	✓	Float	TX	Average number of input slots per output TX
outs_per_in	✗	Float	TX	Average number of output slots per input TX
ins_per_in	✗	Float	TX	Average number of input slots per input TX
profit_rate	✓	Float	TX	Total satoshis deposited over lifetime
expense_rate	✓	Float	TX	Total satoshis withdrawn over lifetime
d_per_tx	✓	Float	TX	Total deposited amount per TX (satoshis/tx)
w_per_tx	✓	Float	TX	Total withdrawn amount per TX (satoshis/tx)

Table 3.4: Classifier features extracted per address.

Classifier	Training			Testing		
	Prec.	Rec.	F1	Prec.	Rec.	F1
Random Forest	0.956	0.956	0.956	0.971	0.946	0.958
Gradient Boosting	0.949	0.948	0.948	0.965	0.935	0.950
Decision Tree	0.934	0.934	0.934	0.938	0.936	0.937

Table 3.5: Classifier results using Stratified 5-fold Cross Validation on 80% training data, and on 20% of unseen testing data.

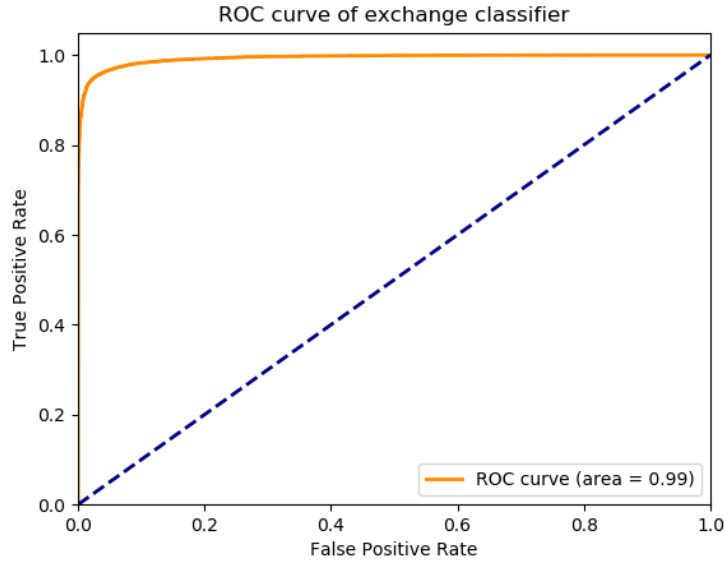


Figure 3.2: Exchange classifier ROC curve.

transactions, number of transactions where the address appears as input or output, and statistics on the transferred amounts and paid fees. The 11 block timestamp (TS) features capture time properties such as address lifetime, time differences between noteworthy events (e.g., first and last deposits), and statistics on the daily rate of transactions. Finally, two address (AD) features capture the address type (e.g., pubkeyhash, multisig) and the number of other addresses (e.g., multisig) built from the address. Of the 42 features, 74% (31) are novel and the rest have been used in previous works [9, 225, 81, 118, 98]. Features are normalized using z-score to have zero mean and unit standard deviation. To understand the importance of the different features, we perform a feature analysis using mutual information (MI) gain. The feature analysis shows that the most important features are those that capture the high volume of transactions over time by exchanges. The top 5 features and their MI gain are: yearly_d_txes (0.1626), yearly_w_txes (0.1601), activity_w (0.1475) txes_in (0.1463) and daily_w_rate (0.1437).

We train three models using the 42 features and 80% of the labeled dataset: Decision Tree, Random Forest, and Gradient Boosting. We focus on tree-based models because they are easy to interpret. The left side of Table 3.5 shows the precision, recall, and F1 score for each model, using a stratified 5-fold cross validation training. The results show that Random Forest achieves the best F1 score of 0.956.

Class	Operation	SDD	Seeds			Graph				Classification		Relat.	Time
			All	OW	Exp.	Comp.	Addr	Txes	Unexp.	Tagged	Exch.		
C&C	cerber	✓	4	0	4	1	608 (73.7%)	1,050 (86.0%)	1	12	394	1 (0)	1'26"
	glupteba	✓	3	0	3	3	195 (17.4%)	167 (24.6%)	5	14	83	0 (0)	1'02"
	pony	✓	4	0	4	2	1,154 (2.3%)	4,252 (36.4%)	72	379	188	1 (0)	5'57"
	skidmap	✓	1	1	1	1	132 (0.8%)	107 (11.2%)	5	37	7	5 (1)	4'04"
Clipper	aggah	✗	1	0	1	1	54 (13.0%)	44 (18.2%)	6	7	19	1 (1)	1'04"
	androidclipper	✗	1	1	0	1	1 (-)	0 (-)	0	1	0	1 (1)	-
	azorult	✗	1	1	0	1	1 (-)	0 (-)	0	1	0	1 (1)	-
	bitcoingrabber	✗	1	0	1	1	504 (18.7%)	650 (46.9%)	32	203	116	0 (5)	3'19"
	casbaneiro	✗	1	0	1	1	125 (47.2%)	162 (59.3%)	5	48	55	4 (4)	0'51"
	clipbanker	✗	1	1	0	1	1 (-)	0 (-)	0	1	0	1 (1)	-
	clipsa	✗	9,412	0	375	25	4,655 (14.6%)	5,756 (14.6%)	120	1,339	2,471	6 (6)	25'18"
	cliptomaner	✗	1	1	0	1	1 (-)	0 (-)	0	1	0	1 (1)	-
	cryptoshuffler	✗	1	0	1	1	3,253 (15.6%)	7,744 (30.1%)	220	966	1,279	11 (12)	42'00"
	masad	✗	2	0	2	1	3,882 (12.1%)	7,389 (20.0%)	182	1,366	1,404	11 (11)	28'40"
	mekotio	✗	3	0	2	1	2,180 (6.7%)	4,031 (9.6%)	71	600	1,125	6 (17)	21'08"
	mekotion40	✗	3	0	3	3	1,153 (12.6%)	1,152 (27.0%)	30	507	338	6 (8)	24'25"
	mispadu	✗	1	1	0	1	1 (-)	0 (-)	0	1	0	1 (1)	-
	mrpr0gr4mmer	✗	2	0	2	1	837 (68.9%)	1,704 (92.1%)	19	321	350	2 (2)	2'18"
	n40	✗	1	1	0	1	1 (-)	0 (-)	0	1	0	1 (1)	-
	phorpiex-tldr	✗	15	14	1	1	203 (5.4%)	107 (51.4%)	14	109	62	3 (3)	2'16"
	phorpiex-trik	✗	8	1	6	4	282 (36.5%)	539 (35.0%)	2	19	116	5 (8)	1'14"
	predatorhethief	✗	1	0	1	1	2,702 (15.2%)	5,039 (32.3%)	157	1,100	947	8 (9)	38'22"
	protonbot	✗	2	2	0	1	2 (-)	0 (-)	0	2	0	2 (2)	-
Ransomware	cryptorlocker2015	✗	1	0	1	1	336 (43.8%)	3,695 (57.8%)	31	103	125	11 (8)	122'02"
	cryptxxx	✗	1	0	1	1	4,015 (15.5%)	10,087 (18.2%)	43	1,607	984	13 (11)	31'29"
	dmalocker	✗	11	0	11	1	1,631 (29.6%)	4,855 (28.3%)	25	525	531	8 (6)	3'44"
	globeimposter	✗	3	0	1	1	13,821 (4.3%)	62,570 (8.4%)	2,871	3,614	4,737	9 (7)	112'42"
	locky	✗	7,076	0	7,074	1	9,546 (83.7%)	22,688 (51.5%)	89	7,303	715	15 (15)	22'59"
	samsam	✗	45	0	20	10	1,020 (29.7%)	1,108 (48.1%)	15	184	300	6 (6)	2'52"
	wannacry	✗	6	0	5	2	38 (78.9%)	531 (97.9%)	0	18	10	2 (2)	0'25"

Table 3.6: Back-and-forth explorations for the 30 malware families. In parenthesis, forward-only exploration results for comparison: fraction of addresses and transactions and total number of relations present in the forward-only graph.

Testing. We evaluate the models to make predictions on unseen data by using the 20% of the dataset reserved for testing. The results, on the right side of Table 3.5, show that the best model is again Random Forest with a F1 of 0.958, so we use that model throughout the explorations with the following hyper-parameters: 600 trees, maximum depth of 40, and prediction threshold of 0.5. Figure 3.2 shows the classifier’s ROC curve.

3.5 Back-and-Forth Exploration Evaluation

We evaluate our platform on 30 malware families: 4 families that use Bitcoin for C&C, 19 clippers that replace addresses in the OS clipboard of the infected host with an address controlled by the malware operators, and 7 ransomware. We compare back-and-forth exploration with a baseline forward-only exploration where we turn off the backward exploration step in our platform, but allow forward-only exploration to benefit from our exchange classifier and tag database. We use this baseline because prior works implement different versions of forward-only exploration that vary, among others, in the number of steps or addresses explored, whether MI clustering is used, and what tags or other heuristics are used to filter unrelated addresses. For each family, we run two explorations: back-and-forth and forward-only. Table 3.6 summarizes the back-and-forth explorations and the delta with the forward-only explorations (in parenthesis). Full forward-only results are in Table 2 in the Appendix.

For clipper and ransomware families back-and-forth exploration is configured to avoid exploring backward on the seeds since addresses depositing into the seeds belong to victims (SDD column). Note however, that back-and-forth exploration will still explore backward after the first forward step from the seeds. The seed information is broken into three columns. *All*

Family	Addresses		Funds Received		Dates	
	All	wTX	BTC	USD	Start	End
clipsa	9,412	375	17.9741	277,514	2018-08-21	2021-02-23
cliptomaner	1	1	16.1839	216,834	2019-11-17	2021-02-23
n40	1	1	27.4139	187,461	2017-05-31	2019-10-02
phorpiextldr	15	15	19.2527	175,473	2018-05-31	2021-02-21
mrpr0gr4mmer	3	3	7.3710	82,216	2020-01-04	2021-02-23
phorpiextrik	8	7	11.4699	66,482	2016-08-14	2021-02-06
masad	2	2	5.4072	46,850	2019-01-24	2021-02-20
clipbanker	1	1	9.2829	36,425	2016-04-22	2021-02-20
cryptoshuffler	1	1	23.5951	26,083	2016-09-27	2019-02-11
predatorthethief	1	1	2.3695	22,539	2019-08-09	2020-11-15
azorult	1	1	5.9413	21,872	2018-07-19	2019-03-24
mekotion40	3	3	1.4321	14,614	2018-01-29	2021-02-08
casbaneiro	1	1	1.3246	8,922	2019-03-07	2021-01-07
protonbot	2	2	0.9428	7,339	2018-04-17	2020-11-25
mispadu	1	1	0.4752	5,235	2019-09-28	2021-01-12
mekotio	3	2	0.3088	3,725	2020-06-25	2021-02-11
androidclipper	1	1	0.1287	942	2018-02-23	2018-12-06
aggah	1	1	0.0130	141	2020-05-18	2020-08-26
bitcoingrabber	1	1	0.0082	71	2019-06-12	2019-10-28
Total	9,459	420	150.8949	1,200,738	2016-04-22	2021-02-23
locky	7,076	7,074	16,353.7008	8,240,542	2016-01-14	2017-05-04
dmalocker	10	10	1,131.7652	1,072,152	2015-12-28	2017-09-21
samsam	45	20	435.6787	795,434	2016-01-13	2018-01-19
globeimposter	3	1	594.4195	260,862	2014-11-23	2017-12-27
wannacry	6	5	59.5967	118,460	2017-03-31	2021-02-05
cryptxxx	1	1	94.8582	60,604	2016-06-04	2016-08-23
cryptotorlocker2015	1	1	5.6998	1,268	2015-01-18	2015-02-14
Total	7,142	7,112	18,675.7190	10,549,322	2014-11-23	2021-02-05

Table 3.7: Clipper (top) and ransomware (bottom) profits measured on seeds.

is the total addresses in our tag database for the family. *OW* captures how many of those seeds are identified as belonging to online wallets. Those seeds are not explored forward as that would explore the transactions of the service holding the online wallet, but they can still be explored backward if they do not belong to a clipper or ransomware (i.e., skidmap). *Exp.* captures the number of addresses effectively explored, after removing addresses in online wallets and addresses without transactions. Some families like *clipsa* and *locky* have thousands of seeds. In the case of *clipsa*, when replacing the address in the clipboard, this clipper selects a visually-similar address from an address database, so that the replacement has a higher chance of not being noticed by the victim. Since the address database ships with the malware, researchers have published its contents [177]. However, only 375 (4%) of *clipsa* addresses have transactions and thus are effectively used in the exploration. Those 375 addresses collect nearly 18 BTC (\$277K). Other families like *n40* collect even larger BTC amounts (27.4 BTC, \$187K) using a single replacement address. Thus, despite their simplicity, clippers can produce significant gains for their operators. Table 3.7 shows the total amounts received by seed addresses.

An important observation is that 24 seeds are online wallets whose owner is an exchange and whose beneficiary is the malware operator (or a money mule). Nearly all of those (23/24) are replacement addresses from 9 clipper families, the other belonging to the *skidmap* crypto miner. Thus, replacement addresses hardcoded in clipper samples are often hosted in exchanges. It is surprising that the abusive nature of those addresses goes unnoticed for the exchanges, as they are often mentioned in public abuse reports by the victims, as well as in analyses by

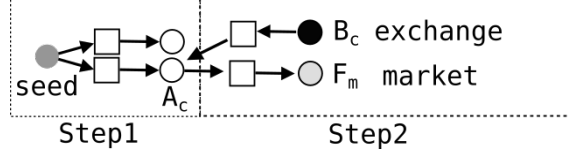


Figure 3.3: False positive example.

security vendors. Note that this does not happen for addresses in ransom notes, possibly because of ransomware’s higher visibility. Based on this observation, we discuss in Section 3.7 a defense that would examine public reports for malicious addresses, identify online wallets, and (publicly) report them to exchanges, so that it cannot be claimed that their abusive nature was not known.

The middle section of the table details the output graphs. For 7 clipper families all seeds are in online wallets. Thus, no exploration is performed for them and the graph contains only the seeds. For the remaining 23 families, 69% contain a single component. Each component contains at least one seed. Thus, multiple components indicate that some seeds were not connected to the rest of the seeds. The graphs for those 23 families contain an average of 2,274 addresses, with a maximum of 13.8K for *globeimposter*. Unexplored addresses are those not processed before the exploration limit was reached. The numbers in parenthesis show the fraction of addresses and transactions in the forward-only graph compared to the back-and-forth graph. On average, the back-and-forth graph is three times larger (3.6x addresses, 2.6x transactions) offering a more complete view of an operation. In Sections 3.5.1 and 3.6 we show that the back-and-forth graphs contain critical information not available in the forward-only graphs such as previously unknown C&C signaling addresses for Cerber and Pony, unreported relations between malware families (e.g. DMALocker and Slave), and new attribution points (Pony, Skidmap, MrPr0gr4mmer, Phorpiextldr).

The right section of the table shows the number of addresses tagged and those without a tag identified as exchanges by the ML classifier. It also shows the total number of relations found in the back-and-forth and forward-only (in parenthesis) explorations. Surprisingly, forward-only exploration finds more relations (150) than back-and-forth exploration (142). The additional relations are FPs introduced by the exchange classifier failing to identify low-volume exchange addresses. Figure 3.3 illustrates this. Step 1 in the exploration finds address A_c , belonging to MI cluster c , which has no tag and is not identified as an exchange. If back-and-forth exploration is used, at step 2 B_c and F_m are introduced. Address F_m has an underground market tag, while address B_c is classified as an exchange. Now, our platform marks A_c also as an exchange because it belongs to the same cluster c as exchange address B_c , and removes paths starting from A_c introduced in step 2. Thus, the back-and-forth exploration graph corresponds to the step 1 box and contains a relation with exchange A_c . However, if forward-only exploration is used, then step 2 only discovers the market address F_m . The exchange address B_c is not discovered because it would require exploring backward from A_c . Thus, the forward-only exploration includes the path from the seed to the market, which is a false relation introduced by the exchange address A_c . While this issue can affect both explorations, it affects more often forward-only exploration as the probability of finding addresses in the same exchange cluster is smaller.

Address	In	Step	Found	FQDNs	Signaling Period
17gd1m	[157]	0	Seed	254	2016-07-26 → 2017-06-13
1HTDy9	[157]	1	MI	128	2017-02-13 → 2017-07-23
1ML94w	[157]	1	Forw	34	2017-06-11 → 2017-09-16
1CpTCV	[157]	1	MI	9	2017-08-31 → 2017-09-22
1GcnsL	✗	1	MI	17	2017-07-27 → 2017-09-22
14ru2h	✗	2	Back	4	2017-01-21 → 2017-02-13
1DiRvu	✗	2	Back	4	2017-03-12 → 2017-04-21

Table 3.8: Cerber signaling addresses. Exploring from 4 seeds, 3 previously unknown signaling addresses were discovered.

The rightmost column captures the exploration runtime. It ranges from a few seconds up to 2 hours, with an average of 22 minutes per operation. Forward-only exploration is faster with an average runtime of 7 minutes due to the fewer transactions it analyzes. However, Sections 3.5.1 and 3.6 show forward-only exploration misses important relations that can only be identified by exploring backward.

3.5.1 Evaluating Address Discovery

Evaluating the accuracy of the produced graphs is challenging as we lack ground truth on the malicious operations. However, for the four families that use the Bitcoin blockchain for C&C we are able to build operation oracles that can identify previously unknown C&C signaling addresses in the output graph, demonstrating the benefits of back-and-forth exploration.

The use of the Bitcoin blockchain for C&C follows a similar pattern for the four families. Samples have a hardcoded signaling address. Upon infection, the sample queries a external Bitcoin monitoring service (e.g., [16]) to obtain the most recent transactions involving the signaling address. Pony and Skidmap use the same signaling method that encodes C&C IP addresses in the value of two consecutive deposits into the signaling address [199]. Cerber encodes C&C domains in the value of a single transaction from the signaling address to a temporary address that then returns the funds [157]. And, Glupteba stores an encrypted C&C domain in the blockchain using a transaction from the signaling address with an OP_RETURN output [87]. In all cases, the signaling transactions create a distinct pattern for which we build an operation oracle. The oracle determines if a given address is a signaling address for the family. An exploration is performed from seeds obtained from previous reports and the operation oracle is applied to each address in the graph to identify previously unknown signaling addresses.

Cerber. The Cerber ransomware was the first malware to use the Bitcoin blockchain for C&C between July 2016 and late 2017 when the operators were arrested [216]. The Cerber oracle, detailed in Algorithm 2 in the Appendix, captures that a signaling round creates cyclic transactions where a signaling address sends x BTCs to a temporary address (only used for one signaling round) that then, within minutes, returns $x - fee$ to the signaling address, where fee is the Bitcoin transaction fee (i.e., 0.001 BTC). The first six characters of the temporary address correspond to a DNS domain on a specific TLD hardcoded in the sample. Given an address, it iterates over its withdrawal transactions. For each transaction, it performs five checks. First, it checks the transaction has a single input and a single output address. Then

Address	In	Step	Found	IPs	Signaling Period
1BkeGq	[199]	0	Seed	237	2019-08-28 → 2021-01-13
1CeLgF	[199]	0	Seed	99	2019-08-27 → 2019-12-11
19hi8B	[199]	0	Seed	336	2019-12-10 → 2021-01-13
1N9ALZ	[156]	0	Seed	19	2019-10-03 → 2020-08-06
bc1qwl	✗	2	Forw	2	2019-07-24 → 2019-07-24
1NL8QT	✗	2	Forw	3	2020-03-03 → 2020-07-22
1GUghN	✗	2	Back	6	2020-03-03 → 2020-07-11
bc1qh9	✗	2	Back	24	2019-06-09 → 2019-08-26
bc1q0n	✗	2	Back	2	2019-12-09 → 2019-12-09
bc1qzs	✗	2	Back	15	2019-06-29 → 2019-07-21
35KxqL	✗	3	Back	2	2019-12-10 → 2019-12-10
3Dmcs6	✗	3	Back	1	2019-12-09 → 2019-12-09

Table 3.9: Pony signaling addresses. Exploring from 4 seeds, 8 previously unknown signaling addresses were discovered.

it verifies that the recipient is a temporary address, i.e., has a single deposit transaction and a single withdrawal transaction. Next, it validates that the recipient returns the received funds to the sender by checking that the withdrawal transaction has just one input and one output, that its output is the input address, and that it returns the value it received minus the transaction fee.

The Cerber exploration starts from four signaling addresses mentioned in [157] and identifies three previously unknown signaling addresses. All seven addresses are detailed in Table 3.8. Full addresses are provided in Table 1 in the Appendix. For each signaling address, the table shows whether it was known, the exploration step at which it was discovered, the method used to discover it, the number of domains it signaled, and the time period in which it signaled. The three signaling addresses discovered are not mentioned in [157], although they were active simultaneously with other seeds. One is identified by MI clustering, but identifying the other two required exploring backward. This highlights how back-and-forth exploration is instrumental to identify new addresses.

Pony. The Pony downloader and information stealer started using the Bitcoin blockchain for signaling C&C servers in August 2019 [199]. Given an input address, the oracle tries to find pairs of consecutive transactions that deposit into the address, whose values encode a public (i.e., not private or reserved) IP address using a known encoding [58]. Each transaction in a valid pair encodes two bytes of the IP address. The oracle iterates over all transactions of the input address. Each deposit transaction should have at most three inputs and at most two outputs. The oracle temporarily stores the value of the first transaction and validates the transaction values if the second transaction happens within an hour of the first one. Both transaction values (in satoshis) are passed to the *decode* function described in [58]. If a public IP address is decoded, it is stored. The current pair is reset if a non-public IP address is found, any check fails, or if a withdrawal transaction is encountered since the two deposit transactions should be consecutive. If at least half of all deposit transactions received by the input address signal a public IP address, the input address is considered a signaling address. This last check handles high volume addresses where a pair of consecutive deposits may decode into a public IP address by chance. It also accounts for signaling addresses that were used for other purposes before they started signaling. The pony oracle is described in Algorithm 3 in the Appendix.

Address	In	Step	Found	FQDNs	Signaling Period
15y7ds	[136]	0	Seed	8	2019-06-19 → 2020-05-13
1CgPCp	[136]	0	Seed	6	2020-04-08 → 2021-10-19
1CUhaT	[93]	0	Seed	6	2021-10-13 → 2021-12-29
34Rqyw	✗	1	Forw	1	2020-01-23 → 2020-01-23
3NhC1b	✗	2	Forw	1	2020-01-24 → 2020-01-24

Table 3.10: Glupteba signaling addresses. Exploring from 3 seeds, 2 previously unknown addresses were discovered.

An exploration is performed starting from four seeds obtained from two sources [199, 156]. The operation oracle identifies 8 previously unknown signaling addresses. The 12 signaling addresses are detailed in Table 3.9. Discovering 6 of the 8 new addresses requires to explore at least one backward step, which highlights the importance of back-and-forth exploration. Four of the new addresses are SegWit (starting prefix *bc1*). They seem to be used for testing before switching to the production non-SegWit addresses (starting prefix 1). For example, *bc1q0n* generates two IP addresses on December 9th, 2019. The next day, *19hi8B* starts signaling IP addresses and the last IP signaled by *bc1q0n* is the first IP signaled by *19hi8B*. The other SegWit addresses behave similarly. Using SegWit addresses for testing could be motivated by smaller transaction fees.

In addition to the new signaling addresses, backward exploration also finds that funds used by two signaling addresses come from the MINE exchange [133], thus providing a previously unknown attribution point. This illustrates how backward exploration can find important relations that forward transaction tracing cannot.

Encoding the C&C servers on deposit transactions has the downside that analysts can deposit funds into a signaling address to hijack the C&C communication. On August 14th, 2020, Taniguchi et al. [199] performed a takeover on Pony by signaling a server under their control to *1Bke4Gq*. However, they did not sustain the takeover, so the attackers regained control two days later.

Skidmap. Skidmap is a crypto mining malware targeting Linux hosts [95]. On February 2021, Akamai researchers identified that Skidmap had started using Pony’s signaling as a backup C&C mechanism, and mentioned one signaling address [180]. We reuse Pony’s oracle for Skidmap. That seed is an online wallet. Thus, forward tracing from the seed cannot be performed. But, the exploration can proceed backward from the seed. In this case the exploration does not find any new signaling address. However, in Section 3.6, we show how the backward exploration identifies that the funds used in the signaling come from a previous Skidmap campaign and that operators cashed-out part of the funds in three exchanges that can serve as attribution points, showing how back-and-forth exploration reveals relations forward transaction tracing could not.

Glupteba. Glupteba is a botnet that has been operating since at least 2011 [82]. In September 2019, it was reported to have added Bitcoin as a mechanism to distribute backup C&C domains [87]. The C&C signaling leverages Bitcoin transactions with an OP_RETURN output, which stores in the blockchain an AES-GCM-256 encrypted backup C&C domain. The Glupteba oracle, detailed in Algorithm 4 in the Appendix, takes as input the address to

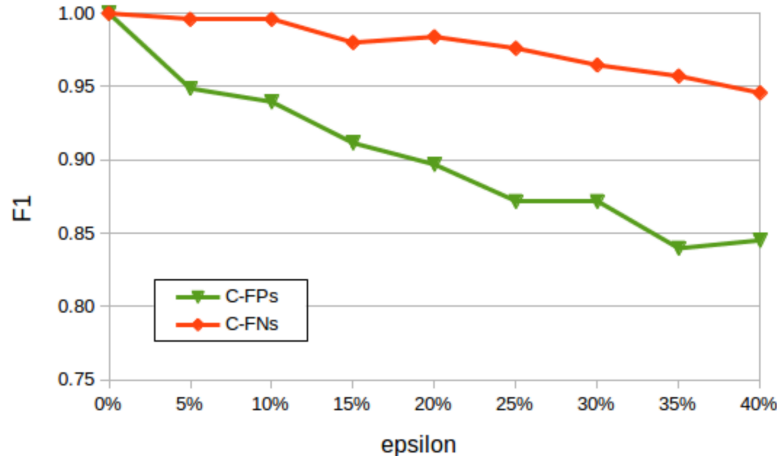


Figure 3.4: Relation detection impact when introducing exchange classifier errors. Introducing C-FPs produces R-FNs (green line). Introducing C-FNs produce R-FPs (red line).

validate and two decryption keys obtained from the malware binaries [136]. It first checks if the given address has a withdrawal transaction with an output with the field *data*, which indicates an OP_RETURN slot. If so, it checks that length of the data field is at least 56 hexadecimal characters (28 bytes). If so, it attempts to decrypt the data with the input keys, by using the first 12 bytes as the initialization vector, the last 16 bytes as the GCM tag, and the rest as the encrypted data. If the decryption succeeds, the decrypted plain text corresponds to the C&C domain.

An exploration is performed starting from three seeds obtained from prior reports [136, 93]. The operation oracle identifies 2 previously unknown signaling addresses in the graph. The 6 signaling addresses are detailed in Table 3.10. The two discovered addresses seem to have been used for testing. They signal the same domain in consecutive days and one day later that domain is signaled by address *15y7ds*. In December 2021, Google performed a take-down on the Glupteba botnet, although their note mentions that thanks to the Bitcoin C&C signaling the malware authors are likely to regain control of the botnet [93].

In summary, the results show back-and-forth exploration finds new C&C signaling addresses and important relations that forward transaction tracing could not. In particular, of the 13 new signaling addresses discovered, 8 are only discovered through backward exploration, and the 4 attribution points found in Skidmap and Pony are also only discovered through backward exploration.

3.5.2 Exchange Classifier Impact

This section evaluates the impact of the exchange classifier in the back-and-forth exploration. First, we measure how exchange classification errors impact the identification of end-to-end relations. Then, we perform an ablation study to measure the performance impact if the exchange classifier is not used. Both experiments are run on the 23 explorations with at least one seed that is not an online wallet.

	Runtime	Timeout	Addresses	TXES
With exchange classifier	499'37"	0	52,308	145,410
Without exchange classifier	2598'34"	7	243,019	787,397
Increase w/o exchange classifier	5.2x	7	4.6x	5.4x

Table 3.11: Total runtime and graph size, across the 23 families where at least one seed is not an online wallet, when running the exploration with and without the exchange classifier. To handle explosion when not using the classifier, 7 explorations were stopped when reaching 20K addresses.

Accuracy impact. A false positive by the exchange classifier (C-FP) might make the exploration discard valid paths and thus miss true relations, introducing end-to-end false negatives (R-FN). On the other hand, an exchange classifier false negative (C-FN) leads to larger graphs that may contain false relations introducing end-to-end false positives (R-FP). To measure the impact of exchange classification errors on the identification of end-to-end relations, we run the back-and-forth exploration multiple times on each family. Each exploration introduces errors to the exchange classifier results with an ϵ probability, and compares the relations found to the family’s baseline in Table 3.6. An R-FN is the absence of a relation in the baseline. An R-FP is a new relation not present in the baseline.

For ease of interpretation, we perform two experiments: introducing only C-FPs and introducing only C-FNs. For example, when introducing C-FPs, if the classifier says an address is not an exchange, with ϵ probability the decision is flipped to mark it as an exchange. We increase ϵ between 5% and 40% in steps of 5%. Figure 3.4 shows, for each experiment, the F1 score of the relation detection as a function of ϵ . The results show that the introduction of C-FPs creates a more pronounced decrease in the relation detection results. Mis-classifying a non-exchange address as an exchange (C-FP) stops the exploration too early, which causes relations to be missed (R-FNs). On the other hand, C-FNs introduce additional addresses in the graph, but since many of those new addresses are untagged, the number of R-FPs introduced is lower in comparison, and thus the F1 score reduces more slowly. These results indicate that favoring a high precision of the exchange classifier is beneficial to the overall relation detection. This is already true for our classifier as Table 3.5 shows that its precision is higher than its recall.

Ablation study. This experiment runs the back-and-forth exploration on each family without using the exchange classifier. This is equivalent to the classifier introducing 100% C-FNs. The removal of the classifier causes some explorations to explode, i.e., they fail to finish within 12 hours. To be able to complete the experiment in reasonable time, we stop an exploration when its graph reaches 20k addresses. Table 3.11 summarizes the impact of removing the exchange classifier in terms of runtime and graph size. When the exchange classifier is not used, the graphs are on average 4.6 times larger and the runtime 5.2 times larger. In 7 of the 23 explorations, the 20K address limit is reached. Thus, the performance impact for those 7 explorations would be larger in reality. These results show how the exchange classifier is crucial to avoid explosion by limiting the runtime and graph size. The removal of the exchange classifier also causes the R-FPs to increase, as illustrated in the previous experiment, but the 20K address limit prevents us from precisely measuring the F1 decrease.

Class	Operation	Relations Found
C&C	cerber	1 malware:cerber:ransomnote
	glupteba	0 -
	pony	1 exchange:mineexchange
	skidmap	5 cryptojacking:sysupdate, exchange:bitfinex, exchange:coinbase, exchange:coincola*, exchange:huobi
Clipper	aggah	1 exchange:binance
	androidclipper	1 exchange:luno.com*
	azorult	1 exchange:tradeogre*
	bitcoingrabber	0 -
	casbaneiro	4 exchange:bleutrade.com, exchange:localbitcoins, exchange:exmo, payment:coinpayments
	clipbanker	1 exchange:coinbase*
	clipsa	6 exchange:binance, exchange:bitfinex, exchange:changenow, exchange:localbitcoins, mixer:wasabi, payment:coinpayments
	cliptomaner	1 exchange:tradeogre*
	cryptoshuffler	11 exchange:bittrex, exchange:btc-e.com, exchange:btctrade.com, exchange:cryptopay.me, exchange:cubits.com, exchange:exmo, exchange:poloniex, exchange:spectrocoin, mining:btcc-pool, payment:bitpay.com, payment:coinpayments
	masad	11 exchange:any-cash, exchange:binance, exchange:bitzlato, exchange:coinbase, exchange:cryptonator, exchange:localbitcoins, exchange:whitebit, exchange:z-exchange.ru, gambling:1xbit, payment:coinpayments, payment:payeer
	mekotio	6 clipper:mekotion40*, exchange:hitbtc, exchange:mercadobitcoin, miscabuse:mrpr0gr4mmer, mixer:wasabi, payment:coinpayments
	mekotion40	6 clipper:mekotio*, exchange:dsx, exchange:mercadobitcoin, gambling:foxbit, gambling:yolo-group, mixer:wasabi
	mispadu	1 exchange:mercadobitcoin*
	mrpr0gr4mmer	2 exchange:binance, payment:coinpayments
	n40	1 exchange:mercadobitcoin*
	phorpiex-tldr	3 exchange:cryptonator*, gambling:duckdice.io, miscabuse:saveyourself
	phorpiex-trik	5 exchange:bitcoin.de, exchange:localbitcoins, exchange:cryptonator*, gambling:rollin.io, payment:webmoney
	predatorthethief	8 clipper:masad, exchange:binance, exchange:bitzlato, exchange:btcbank, exchange:cryptonator, exchange:totalcoin, exchange:whitebit, payment:coinpayments
	protonbot	2 exchange:cryptonator*, exchange:hitbtc*
Ransomware	cryptotorlocker2015	11 exchange:bitcoin.de, exchange:cex.io, gambling:betcoin-dice, gambling:fortunejack.com, gambling:luckybit, gambling:satoshibones, gambling:satoshidice.com, mining:cointerra, mining:deepbit, mining:eobot, tormarket:silkroad2market
	cryptxxx	13 exchange:bitbay.net, exchange:bitcoin.de, exchange:bitfinex, exchange:bitflyer, exchange:btc-e.com, exchange:coinbase, exchange:coinmate.io, exchange:coins.co.th, exchange:huobi, exchange:localbitcoins, exchange:okcoin.com, exchange:poloniex, exchange:xapo
	dmalocker	8 exchange:btc-e.com, exchange:localbitcoins, exchange:okex, exchange:poloniex, gambling:cloudbet.com, malware:slave, payment:bitpay.com, payment:coinpayments
	globeimposter	9 exchange:bitstamp, exchange:bittrex, exchange:btc-e.com, exchange:cex.io, exchange:localbitcoins, payment:bitpay.com, payment:easywallet, payment:webmoney, tormarket:evolutionmarket
	locky	15 exchange:bitfinex, exchange:bitstamp, exchange:btc-e.com, exchange:coinbase, exchange:localbitcoins, exchange:matbea.com, exchange:poloniex, exchange:coins.co.th, exchange:exmo, exchange:zxxx.biz, mixer:bitcoinfog, payment:bitpay.com, payment:coinpayments, payment:webmoney, tormarket:nucleusmarket
	samsam	6 exchange:bitcoin.de, exchange:cubits.com, exchange:localbitcoins, exchange:poloniex, exchange:yobit.net, tormarket:alphabaymarket
	wannacry	2 exchange:changelly, exchange:shapeshift

Table 3.12: Summary of relationships found. Tags with an asterisk at the end were found by multi-input clustering on the seeds, the rest were found by the exploration.

3.6 Relationships

Tagged addresses in a transaction graph may capture relations between the campaign under analysis and external services (e.g., exchanges, mixers, gambling, markets) and other malicious campaigns. Those relations are identified by the graph analysis module as paths that detail the specific chain of transactions capturing money flow from the campaign under analysis to an external entity, or vice-versa. Such paths constitute evidence that can be shared with third parties to allow independent validation of the relations.

Table 3.12 summarizes the relations found by back-and-forth exploration. Table 3 in the Appendix summarizes forward-only results. In 93% (28/30) of the explorations at least one relation is identified, which we believe reflects that our tag database provides good coverage. Tags may have been assigned to addresses found by MI clustering on the seeds (9.2%) (marked with asterisk at the end of the tag) or during the exploration (90.8%). The large percentage difference highlights the need to perform transaction tracing (beyond MI clustering) to discover relations. The most common relations are with exchanges used by the operators for seeding C&C signaling and cashing out profits. We observe 44 exchanges, the most popular one being *localbitcoins* used by 9 families, followed by *coinbase* (5), *poloniex* (5), *cryptonator* (5), *btc-e* (5), and *binance* (5). Ransomware families use a larger variety of exchanges compared to clippers, e.g., the *cryptxxx* ransomware uses 13 exchanges. We also observe 5 payment processors, also used for cashing out: *coinpayments* (9), *bitpay* (4), *webmoney* (3), *easywallet* (1), and *payeer* (1). These services used for cashing out include some that are not amongst the most popular ones [202], which may be due to relaxed KYC measures. For example, *localbitcoins* is a peer-to-peer exchange that allows users to transact between them over-the-counter in a fairly anonymous manner. *Coinpayments* is a payment gateway that merchants can use to receive cryptocurrency with just an email and password. Both services did not establish KYC measures until mid-2019 [42, 120]. Furthermore, the owner of the *btc-e* exchange was arrested in July 2017 for money laundering with the indictment mentioning the lack of KYC measures [210].

Other common relations are gambling sites and mixers, likely used for money laundering. Six families are observed transacting with 11 gambling sites (e.g., *satoshidice*) with the *cryptotorlocker2015* ransomware using a record 5 gambling sites. Gambling sites are also possible attribution points since they may be subjected to legislation mandating KYC practices such as the Licence Conditions and Codes of Practice (LCCP) in the UK [44]. Two mixers appear across 4 families: *wasabi* (3) and *bitcoinfog* (1). Wasabi [214] is an open-source Bitcoin wallet implementing CoinJoin shuffling over Tor for anonymity. The operator of *bitcoinfog* was arrested on April 2021 on counts of unlicensed money transmission and money laundering [54]. Other relations found include three malware families using profits for buying products from four Tor hidden markets (*alphabaymarket*, *evolutionmarket*, *nucleusmarket*, *silkroad2market*) and two families transacting with four mining pools (*btcc-pool*, *cointerra*, *deepbit*, *eobot*). Finally, several relations between malicious campaigns are found, as detailed next.

Phorpiex-tldr and SaveYourself. The exploration from *phorpiex-tldr* reveals multiple connections with the *saveyourself* sextortion campaign. Figure 3.5 shows the paths capturing these relations, other paths are excluded for readability. There are six *saveyourself* tagged

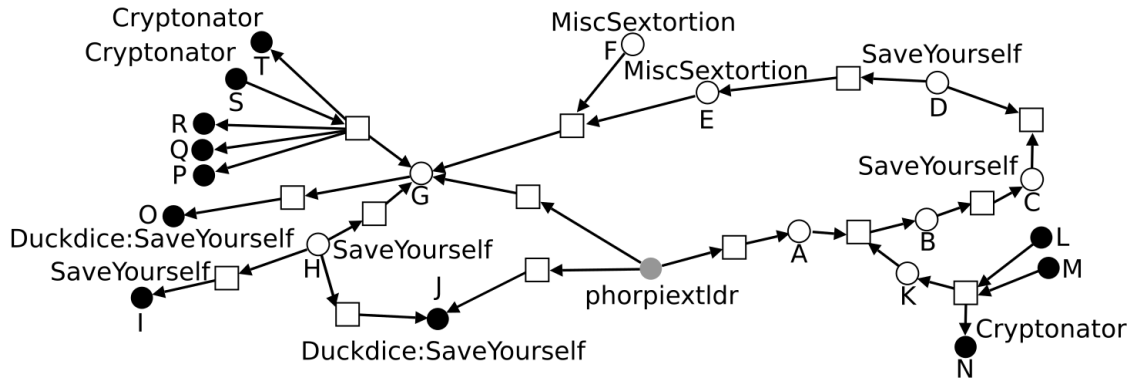


Figure 3.5: Phorpiex-tldr – SaveYourself relations.

addresses in the graph, corresponding to addresses reported by Bitcoin Abuse users as appearing in sextortion emails with a common SaveYourself pattern in the subject and content. The clipper’s profits flow towards multiple *saveyourself* addresses through different paths, e.g., to tagged address C from the seed (in gray) through addresses A and B, to tagged address J from the seed, and through collector node G to tagged address O. Addresses O and J correspond to online wallets in the *duckdice.io* gambling service, managed by SaveYourself operators. Some connections are found through forward transaction tracing, but others such as the path going from the seed to address D through A, B, and C or the path going from the seed through G and H to reach node I require exploring backward. This illustrates the benefits of back-and-forth exploration. A report [123] mentions that the Phorpiex malware sends sextortion emails. Checking the Bitcoin addresses in that report on Bitcoin Abuse confirms the link between Phorpiex and the SaveYourself campaign. Furthermore, the multiple connections between both campaigns and how the clipper profits flow into the sextortion addresses makes us conclude that the sextortion campaign is run by the malware operators, as opposed to the Phorpiex operators renting their botnet for sending third-party spam.

MrPr0gr4mmer clipper and scam. The MrPr0gr4mmer exploration, summarized in Figure 3.6, starts from two seeds (A, B) in the same MI cluster, each with multiple clipper reports in Bitcoin Abuse. Address F is reported in sextortion scam emails [18]. The sender of the scam emails is “Mr Robot”. Manual searches uncover that seed A appears in the @MrPr0gr4mmer Telegram channel, where software to send spam through the telegram API [161] is advertised. The contact address starts again with the “Mr.” prefix. Seed B appears as donation address for a user of the [cracked.to](#) forum and backward tracing finds that collector E receives deposits from the *paxful* exchange. We conclude that the MrPr0gr4mmer operator could be attributed through the *paxful* exchange and the [cracked.to](#) forum and is involved in multiple malicious activities including selling spam tools, sending scam emails, and operating a clipper.

Mekotio, MekotioN40 and MrPr0gr4mmer. A seed from *mekotio* and a seed from *mekotion40* appear as inputs to the same transaction indicating that both families are indeed the same. While the relationship was already hinted by the names assigned in the two reports

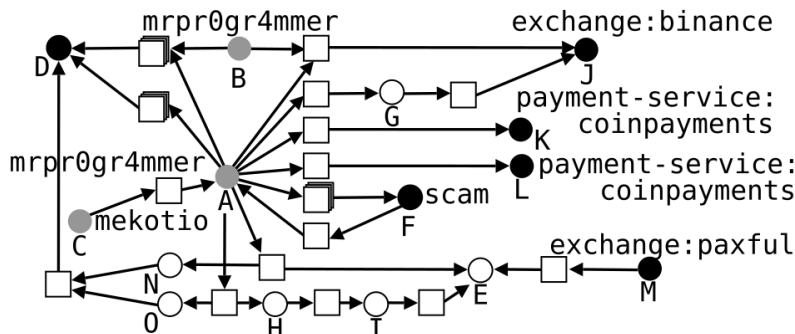


Figure 3.6: Mekotio-MrPr0gr4mmer relations.

from where we obtained the seeds, those reports do not explicitly mention a connection. In addition, both graphs show cash outs to the *mercadobitcoin.br* exchange and transactions to the *wasabi* mixer in the non-shared components, further reinforcing that both campaigns have the same operators. Another relation is a payment from a *mekotio* seed to the seed of *mrpr0gr4mmer* used for selling the Telegram spam tool (see Figure 3.6). Thus, *mekotio* operators are using the clipper profits to pay for a spam tool possibly used as part of their malware distribution.

PredatorTheThief and Masad. The *predatorthethief* clipper seed makes a transaction to an address that a report identifies as belonging to the Masad Clipper and Stealer malware [105]. Masad is offered as a malware kit in forums. It has a free version and a full version that costs \$85. The transaction on June 2020 is for 0.00910768 BTC, roughly \$85.31 USD using the conversion at that time. We believe this relation captures how *predatorthethief* operators bought a license of *masad*, possibly to upgrade their own malware.

DMALocker and Slave. The *dmalocker* ransomware graph contains a transaction to an address tagged as belonging to the Slave banking trojan [34]. Slave receives 2.3 BTC from DMALocker, more than 1.3K USD at that time. Unfortunately, in this case we are not able to conclude what type of relation the transaction captures.

Skidmap and Sysupdate. The *skidmap* cryptominer backward exploration (forward transaction tracing is not possible as the seed is an online wallet) identifies a *cryptojacking:sysupdate* address that sends funds to the seed six times. This address belongs to a mining pool used by a Linux cryptomining malware to receive the shares produced by the botnet’s mining activity [119]. We believe this malware is an older *skidmap* version (prior to using Bitcoin for C&C) and that the relation shows that funds from Skidmap’s cryptomining were used to seed the new Bitcoin C&C signaling. The Sysupdate address cashes out in three exchanges (*coinbase*, *huobi*, *bitmex*) that can be used as attribution points and are only found exploring backward.

CryptoTorLocker2015 and Scam Campaigns. The MI cluster of the *cryptotorlocker2015* ransomware seeds contains 10 addresses beginning with short English words involved in

multiple scam campaigns in 2014-2015. Four of those imitate satoshidice gambling addresses beginning with “1DiCE”, another four (e.g., 1BonUS, 1sYSTEM) were used in scams that promise to double the BTCs sent [213], and two addresses beginning with “1bet” and “1shop” were used in campaigns claiming to have hacked the bitcointalk forum [25]. Those scams are likely run by the ransomware operators.

3.6.1 Limited Ground Truth Validation

Ground truth is not available for cybercrime operations and can only rarely be obtained, e.g., through leaks [129, 111]. Still, we tried hard to find some ground truth, even if limited, encountering two major challenges. First, prior analysis focuses on forward-only explorations and thus does not mention relations only reachable by exploring backward. Second, analysis reports rarely exhaustively list relations, mentioning only those of interest to the report. We found only one report that partially addresses those challenges, which we use as a limited ground truth. It corresponds to the affidavit filed in April 2021 by the US Internal Revenue Service to justify the takedown of the *bitcoinfog* mixer [11]. The affidavit mentions *bitcoinfog* received deposits from 35 darknet markets, but only names five: *agora*, *alphabay*, *evolution*, *silkroad*, and *silkroad2*.

Our database has one address tagged as *bitcoinfog* belonging to a MI cluster with 244K addresses. Forward exploration from the mixer addresses does not work as the mixer’s goal is to hamper forward transaction tracing. Thus, we perform a backward-only exploration using as seed all addresses in the *bitcoinfog* MI cluster. The exploration finds a wealth of relations including all 15 darknet markets in our database. Those 15 markets include the five named in the affidavit and another 10: *abraxas*, *babylon*, *blackbank*, *bluesky*, *cannabisroad*, *doctord*, *middleearth*, *nucleus*, *pandora*, and *sheep*.

Since all 15 tagged darknet markets are found after one backward step, we stop the exploration at that point. In addition to the darknet markets, the backward exploration identifies relations with 130 services and 15 malicious operations not mentioned in the affidavit: 4 ransomware (*cryptolocker*, *locky*, *xlocker*, *xtplocker*), 3 scams (*mintpal*, *mcrnow*, *bitoomba*), 3 thefts (*binance-hack*, *bips-hack*, *flexcoin-hack*). 3 mixers (*bitlaunder*, *wasabi*, and an unknown mixer), a malware (*slave*), and a ponzi scheme (*nanoindustryinv*).

This experiment shows backward exploration finds correct relations, although it may miss relations for which no tags are available. Unfortunately, we cannot evaluate FPs as the affidavit does not list all markets, and other services and malicious operations.

3.7 Discussion

Backward exploration. Our results show that back-and-forth exploration can discover important relations that forward transaction tracing cannot. One limitation is that backward exploration is not suitable for every transaction. However, we show how it is possible to identify the problematic cases. For example, we provide a heuristic to identify dust attacks that may deposit small amounts to many addresses. We also provide an option to disable backward exploration on seeds, which requires an easy decision by the analyst: use it when

seeds may get deposits from other addresses belonging to the same campaign (e.g., C&C signaling) whereas avoid it when deposits come from victims (e.g., clippers, ransomware). Furthermore, even if a transaction was included that it should not, the transaction and its dependencies can later be pruned from the graph.

Exchange defenses. 90% of malware families analyzed had at least one relation with an exchange. This includes 10 malware families where addresses hardcoded in the malware correspond to online wallets in exchanges. These exchanges are either unaware about the abusive nature of those addresses, or they simply do not care. This creates an opportunity for intervention where public malware analysis reports could be automatically analyzed to extract tagged Bitcoin addresses (e.g., using indicator extraction approaches [117, 228]). The identified addresses could then be automatically explored, and those identified as online wallets or receiving cash-outs from the malware activities could be reported to the exchanges. This would prevent exchanges from claiming they were unaware of the abuse, and allow evaluating their reaction to abuse reports.

Address classification. Tagging Bitcoin addresses is fundamental to identify relations. A well-known tagging limitation is low coverage, as only a very small percentage of addresses is tagged. Furthermore, tagging is often delegated to commercial services (e.g., [36, 59]), which use proprietary approaches to build their tag databases and do not disclose how they handle challenges such as double-ownership and potentially conflicting tags. We believe open research on building large, accurate, tag databases is fundamental to allow tagging to sustain challenges likely to appear in court cases. Furthermore, we have shown how to build classifiers to identify certain types of addresses such as those belonging to exchanges and those used by malware for C&C signaling. A possible avenue of future work is to build classifiers for many other address types.

Evasion. An attacker could evade our approach by making its addresses look uninteresting, e.g., like exchanges. However, feature analysis shows that the high volume of transactions is a key feature of the exchange classifier, so impersonating an exchange would be expensive. Alternatively, an attacker could pollute our tag database so that some of its addresses are considered benign services. Evasion is also possible by using mixers since they prevent forward exploration. However, as shown, backward exploration reveals their clients. Another evasion would pollute the graphs with unrelated transactions, e.g., by using sophisticated dusting techniques that our approach may fail to filter. This would make the graphs less readable, but would not prevent automated extraction of relations.

Other blockchains. Our approach generalizes to other cryptocurrencies following Bitcoin’s model that are already supported by BlockSci: Bitcoin Cash, Dash, Litecoin, Namecoin, and ZCash. Supporting those would require extending our tag database with addresses for those blockchains and to retrain the exchange classifier.

3.8 Conclusion

Our study of the financial relationships of 30 malware families sheds light on cybercrime’s financial inner workings. Our novel back-and-forth exploration identifies important relations missed by forward transaction tracing, such as previously unknown C&C signaling addresses, a wealth of relations to external services and other cybercrime operations, and new attribution points. The results of this work can benefit the community in two main ways: (i) the open-source tool and tag database we will provide may serve as the basis for future works, and (ii) the detailed graphs and relations our approach produces could be used as evidence to warrant interventions.

Chapter 4

The Impact of Methodology and Coverage in Revenue Estimations

4.1 Introduction

The Bitcoin ecosystem has attracted cybercriminal activities such as ransomware [196, 116, 45, 157, 90, 151, 199], thefts [130], scams [152, 9, 130, 222, 10, 115], human trafficking [160], clippers [73], cryptojacking [89, 201], hidden marketplaces [38, 113, 175], and money laundering [134]. Cybercriminals often request payments in bitcoins from victims that fall for their scams or are infected with their malware. The public nature of the Bitcoin ledger has been leveraged by multiple works to estimate the revenue cybercriminals obtain from victims [89, 196, 116, 45, 90, 9, 113, 151, 152, 222, 146, 115]. An accurate estimation of the financial impact on victims is fundamental for understanding the cybercrime ecosystem, e.g., for comparing the revenue of different types of cybercrime, such as ransomware versus sextortion. It is also critical for triaging, i.e., assigning adequate investigative resources to each cybercriminal group based on their impact. Underestimating the financial impact of a cybercriminal group may convey the wrong impression that it is unimportant, removing resources from its analysis and defense and thus allowing its operations to continue unfettered. On the other hand, overestimating the financial impact may wrongly identify small players as dominant, assigning them unnecessary investigative resources.

Starting from a set of *seed* Bitcoin addresses, known to belong to the same cybercriminal group (e.g., ransomware family, Ponzi scheme) or to the same type of cybercrime (e.g., ransomware, sextortion), estimation works apply different methodologies to quantify the financial impact on the victims. Some works simply add deposits to the input seed addresses, while others apply a combination of expansions to discover additional payment addresses (e.g., multi-input clustering [137, 6, 174, 130] and change address heuristics [130, 6, 76, 61, 101]) and filters to remove unrelated addresses and deposits. Estimations focusing on the same target often do not agree, due to the use of different methodologies, seed addresses, and time periods (i.e., block heights). Furthermore, these factors make it very challenging to understand to what degree each methodology step is responsible for differences in the estimation.

In this work, we perform the first systematic analysis on the estimation of cybercrime bitcoin revenue. We quantify the impact in the estimation of the methodology used and the limited seed coverage. In detail, we survey prior estimation works providing a detailed analysis of their methodologies and concrete takeaways. We implement a tool that can replicate the different methodologies. We apply our tool to estimate the same target using 15 methodologies, while fixing the set of seeds and the blockchain height. This allows us to quantify how differences in the methodology affect the estimation. In contrast to what is widely believed, we show that estimations do not always underestimate the revenue. There exist estimation methodologies that can introduce huge overestimation such as those that do not filter seeds that are online wallets in exchanges and those using change address heuristics. We collect 30,424 cybercrime Bitcoin payment addresses from publicly available sources. We use this dataset to compare, using a consistent methodology, the impact of 6 abuse types (ransomware, clippers, sextortion, Ponzi schemes, giveaway scams, exchange scams) and of 141 groups running Ponzi schemes, ransomware, and clippers (which replace addresses copied into the clipboard with their own). We observe that the top cybercrime groups by revenue are dominated by ransomware and the most successful ones operate in the ransomware-as-a-service model [131], although there are also Ponzi schemes and clipper families for which we observe more than one million USD revenue. There are also 14 groups whose estimated revenue is below \$20, likely indicating limited coverage on their payment addresses. Our results highlight the limited effectiveness of existing expansions. The popular multi-input (MI) clustering fails to discover additional addresses for 40% of the 141 groups, likely indicating cybercriminals are actively evading it, while change address expansion introduces large false positives.

An intrinsic issue in cybercrime revenue estimations is that they underestimate the real revenue because they start from a limited set of seeds (oftentimes only one) while the cybercriminals may use a large number of addresses to receive victim payments. To this day, no work has quantified the impact of the (lack of) coverage on the estimation. This requires a vantage point that allows to observe all victim payments. We perform the first quantification of this issue. For this, we focus on the DeadBolt server ransomware family, which infects network-attached storage (NAS) devices [1]. We start by collecting 4,997 DeadBolt payment addresses from two Internet scan engines [190, 32]. We estimate DeadBolt’s conversion rate from infections to payments to be 0.7%. A unique characteristic of DeadBolt is that it releases the decryption key on the blockchain upon receiving the victim’s payment. We propose two novel techniques, leveraging unique characteristics of DeadBolt’s key release transactions, to obtain very high coverage, possibly nearly complete, on the payments DeadBolt receives from victims. Using the 34 seeds with victim payments collected from the scan engines (which MI clustering cannot expand) we would have estimated a very modest revenue of 2.826 BTCs or \$63K. By applying our coverage-expanding techniques, we instead estimate 98.350 BTC, 35 times higher, and our USD estimation is \$2.47M, 39 times higher. The vantage point provided by the scan engines only identifies 2.6% of victim payments due to issues such as scan frequency or infections happening before the scanning begins. Still, Internet scan engines arguably provide higher coverage on server ransomware (i.e., 34 DeadBolt seeds) that is typically available for other groups such as desktop ransomware (i.e., a median of one seed per group). Thus, for other groups, the coverage impact may be larger than what we have measured.

Our coverage results critically indicate that even if a family or campaign is estimated to have

Work	Year	Crime	Platform	Blk Height	Seeds					Expand			Filtering					Methodology
					Seeds	Labels	Micro Payments	Seed Clustering	Seeds Available	Multi-input (MI)	Change Address (CA)	Exploration	Value Filtering (VF)	Time Filtering (TF)	Online Wallets (OW)	Double-Counting (DC)	Payment day rate	
Huang et al. [89]	2014	Cryptojacking	-	2013-11-30	290	10	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	DD+MI
Spagnuolo et al. [196]	2014	Ransomware	Bitlodge	2013-12-15	≥12	1	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	DD+MI+CA-VF
Liao et al. [116]	2016	Ransomware	-	2014-01-31	2	1	✗	✗	[116]	✓	✓	✗	✓	✓	✗	✗	✓	DD+MI+CA-VF-TF
Conti et al. [45]	2018	Ransomware	-	2017-12-07	128	20	✗	✗	[145]	✓	✓	✗	✓	✓	✗	✗	✓	DD+MI+CA-VF-TF
Huang et al. [90]	2018	Ransomware	BlockSci	2017-08-31	25	10	✓	✗	✗	✓	✗	●	✓	✗	●	✗	✓	DD-OW+MI-VF
Bartoletti et al. [9]	2018	Ponzi	-	-	32	32	✗	✗	[21]	✓	✗	✗	✗	✗	✗	✗	✓	DD+MI
Lee et al. [113]	2019	Dark Web	BlockSci	2018-04-30	85	-	✗	✗	✗	✓	✓	✗	✗	✗	●	✗	✓	DD-OW+MI+CA
Paquet-Clouston et al. [151]	2019	Ransomware	GraphSense	489,181	7,118	35	✗	✗	[165]	✓	✗	✗	✗	✓	✗	●	✓	DD+MI-TF-DC
Paquet-Clouston et al. [152]	2019	Sextortion	GraphSense	573,989	245	-	✗	✓	[197]	✓	✗	✗	✓	✗	●	●	✓	DD-OW+MI-VF-DC
Xia et al. [222]	2020	Exchange scams	-	2019-09-23	66	-	✗	✓	[48]	✗	✗	✗	✗	✗	✗	✗	✓	DD
Oosthoek et al. [146]	2022	Ransomware	-	-	7,321	87	✗	✗	[30]	✗	✗	✗	✗	✗	✗	✗	✓	DD
Li et al. [115]	2023	Giveaway scams	-	2022-07-01	860	-	✗	✓	[56]	✗	✗	✗	✗	✗	✗	✗	✓	DD

Table 4.1: Related work on estimating cybercrime Bitcoin financial impact. A tick (✓) indicates a property is implemented by the work, a cross (✗) no support, a half-filled circle (●) partial support, and a dash (-) that the paper does not specify it.

low revenue, it could still have a significant non-measured financial impact on victims. Thus, estimations should also consider other impact metrics beyond the revenue, e.g., the number of family samples observed.

This work provides the following main contributions:

- We perform the first systematic analysis of cybercrime bitcoin revenue estimations. We build a tool that implements the different estimation methodologies and use it to quantify the impact of each methodology step in the estimation. We show that some methodologies can produce huge overestimation.
- We quantify, for the first time, the impact of the (lack of) coverage on the estimation. For this, we propose two novel techniques to achieve high, possibly complete, coverage of the victim payments received by the DeadBolt server ransomware. The USD revenue DeadBolt collects from victims is 39 times larger than what would have been estimated by collecting seeds from two popular Internet scan engines.
- We compare the bitcoin revenue obtained by 6 abuse types and 141 cybercriminal groups. Cybercriminals may be actively evading the popular multi-input clustering, which does not discover additional addresses for 40% of groups.
- We release our tool and datasets [73].

4.2 Bitcoin Revenue Estimation

This section surveys prior works that estimate cybercrime bitcoin *revenue* [89, 196, 116, 45, 90, 9, 113, 151, 152, 222, 146, 115]. These works do not estimate the *profit* the cybercriminals made, as that would require subtracting from the revenue the unknown expenses they incurred [204]. Estimations may target a specific cybercriminal group (e.g., malware family, Ponzi scheme, scam campaign) or a (type of) cybercrime (e.g., ransomware) by aggregating revenue of multiple groups in the same cybercrime (e.g., multiple ransomware families).

At a high level, all the works follow the same approach. They take as input a set of *seed* addresses known to receive victim payments. They optionally expand the seed addresses to obtain an *expanded set* of addresses that belong to the same owners as the seeds. If no expansions are used, the expanded set only contains the seeds. Then, they obtain all the deposits to addresses in the expanded set. Next, they optionally apply filtering to remove unrelated addresses and deposits. This process results in an estimation of BTC revenue. Finally, the BTC amount is converted to fiat currency (typically US Dollars). While the general process is shared, there exist methodological differences at each of the above steps that can lead to differing estimations for the same input seeds.

Table 4.1 summarizes the 12 surveyed works. To select these works, we first examined papers published since 2009 (the initial Bitcoin protocol release year) in top computer security venues. To identify other articles published in smaller venues and pre-print repositories, we analyzed the references of those initial works. Additionally, we searched on engines like Google Scholar for combinations of keywords, including *bitcoin*, *payments*, *ransomware*, and *scam*. We limit the selection to papers published on peer-reviewed venues (and technical reports with public datasets) that include an estimation of cybercriminal bitcoin revenue. We exclude works that analyze Bitcoin abuse by cybercriminals but do not include a financial estimation (e.g., [201, 73]), those performing estimations on other cryptocurrencies (e.g., Monero [86, 12]), and those estimating revenue using transaction data not from the Bitcoin blockchain (e.g., [112]). The rest of this section details the above steps using the different parts of Table 4.1.

4.2.1 Seeds

We call *payment addresses* to Bitcoin addresses where victims are requested to send their payments. Payment addresses may be obtained from social media (e.g., [196]), threat intelligence reports by security companies (e.g., [45]), by running malware samples on a sandbox (e.g., [90]), from scam emails (e.g., [152]), from Tor hidden services (e.g., [113]), and from scam websites (e.g., [115]). Payment addresses can be split into those that have received some deposit, which are called *seeds*, and those that have not (yet). This determination, and generally the whole estimation, needs to be performed at some particular height of the Bitcoin blockchain. Unfortunately, only two works provide the specific block height used [151, 152]. Most works mention a day, but there are roughly 144 blocks in a day, as a new block is minted every 10 minutes.

Cybercriminals may or may not reuse payment addresses across victims. In the extreme, they could use a single payment address for all victims, or generate a different address for each victim. Oftentimes, a middle ground is used with a (potentially large) pool of payment addresses that are reused with some (potentially low) probability. Payment addresses with deposits are the input seeds to the estimation. The number of seeds ranges from only one up to 7,036 addresses. However, the high number of seeds in two works [151, 146] is due to the outlier Locky ransomware with over 7K addresses. Overall, the median number of seeds per group is 1 (measured on the dataset in Section 4.3.4). Seeds are the critical starting point for any estimation. To enable replicability, authors should release the seeds used, which happens for 8 of the 12 works.

One challenge when each victim is assigned a unique payment address is that only a small fraction of payment addresses receives payments and thus can be used as seeds. For example, in their sextortion paper, Paquet Clouston et al. [152] extracted 12,533 payment addresses, but only 2% had deposits. Furthermore, for some address collection methods like running malware on a sandbox or collecting emails marked as spam (prior to the user receiving them), if the address is unique for each victim then none of the collected payment addresses would have payments, as there are no real victims. To address this situation, Huang et al. [90] perform micro-payments to payment addresses. Micro-payments simulate the payment of a victim, but their value is much smaller than the requested payment value. Their goal is incentivizing the cybercriminals to move the micro-payment so that it can reveal other payment addresses through the expansions presented in Section 4.2.2. For example, a micro-payment of a few hundred satoshis is small enough that to move those funds the recipients need to combine them with other Unspent Transaction Outputs (UTXOs) in a multi-input (MI) transaction so that the transaction fee can be paid. This enables discovering those additional inputs through MI clustering, as described in Section 4.2.2. The larger the micro-payment, the larger the incentive for the cybercriminals to move the gifted funds. However, cybercriminals could ignore any payments not matching the expected values.

Seeds may be labeled with the group to which they belong, which allows to perform separate estimations for each group. For example, the label may capture the malware family that uses the seed, obtained from external reports or from the AV labels of the samples [186]. But, some collection methods such as visiting Tor hidden services [113], visiting scam websites [222, 115], and examining spam emails [152] may collect addresses that belong to different groups. To perform per-group estimates, these works first need to cluster the addresses using external information such as the content of the websites or the spam emails. When seeds are unlabeled and no clustering is performed, it is only possible to provide an estimation of the type of cybercrime all addresses are associated with (e.g., the Dark Web [113]). The quality of the labels is fundamental to the estimation. If a seed is incorrectly labeled, that will inflate the estimation of its group.

Some works extract Bitcoin addresses from Web pages using regular expressions [113, 222, 115]. Since Bitcoin addresses are hashes, such regular expressions may generate false positives. It is important to validate the matches by verifying the checksum embedded in valid addresses, e.g., using online blockchain explorers or by using tools that perform such validation during the extraction [121].

4.2.2 Expansions

Given a set of seeds, the simplest estimation consists of adding all direct deposits the seeds have received, i.e., for each transaction where a seed address appears in an output slot, accumulate the value of the seed's output slot. This simple estimation is used in three surveyed works [222, 146, 115]. However, if the campaign uses a large number of payment addresses, the seeds may only receive a small portion of the revenue received from victims, i.e., there may exist many unknown payment addresses that also received victim payments. To discover previously unknown payment addresses, it is common to apply expansions to the seeds that identify additional addresses that also belong to the seed owners. Then, the

same estimation as above is performed on the expanded set of addresses (i.e., seeds plus additional addresses the expansions identified). The surveyed works use two main expansions: 8 works use *multi-input clustering* and 4 works combine multi-input clustering with one or more *change address* heuristics. We also discuss an *exploration* expansion, which generalizes the approach used in one work for one group.

Multi-input clustering. The most popular expansion is multi-input (MI) clustering [137, 6, 174, 130]. It assumes that input addresses to the same transaction have the same owner because their private keys are used together to sign the transaction. Clusters can be created transitively. If a transaction has addresses A and B as inputs, and another transaction has B and C as inputs, then A , B and C are all clustered together and have the same owner. One exception is CoinJoin [128] transactions where a group of users creates a single transaction that simultaneously spends all their inputs into a shuffled list of outputs. For this reason, before computing MI clustering, it is common to apply proposed heuristics to identify CoinJoin transactions [76, 99]. Kappos et al. [101] recently proposed a machine learning (ML) classifier to reduce the false negatives of the CoinJoin heuristics. However, their classifier has not yet been used in estimations.

Once the MI clusters are computed, the seeds and all addresses in the MI cluster of a seed are added to the expanded set. If the seeds belong to different MI clusters, the expanded set contains the union of all addresses in those MI clusters (including the seeds). Of the 9 works using MI clustering, 5 use open-source platforms to compute it: one uses BitIodine [22], two use BlockSci [99], and two use GraphSense [84]. The rest use their own implementation. As far as we know, no work has compared that different implementations of MI clustering indeed produce the same results.

While MI clustering is an extremely popular and reliable expansion [140, 83], there are two caveats that could affect the estimation. First, MI clustering captures *same ownership*. If the same cybercriminals run multiple campaigns, MI clustering may add to the expanded set addresses from campaigns different from the one the seeds belong to. If the goal is to estimate the revenue of a specific campaign, rather than the revenue of a cybercriminal group, or of a type of cybercrime, this may introduce overestimation.

Second, *double ownership* is common in services (e.g., exchanges) offering online wallets for their users. In that scenario, the service owns the address (i.e., has the private key), but the address is handled (and thus indirectly owned) by the user for which it is created. Performing MI clustering on an online wallet address can bring into the expanded set thousands (and even millions) of addresses that also belong to the service, but are unrelated to the estimation as they belong to other customers of the service. For seeds that are online wallets, only their direct deposits should be considered, i.e., other addresses in their MI cluster should not be added to the expanded set. Failure to filter seeds that are online wallets, can make the estimation hugely overestimate the financial impact. We discuss the filtering of online wallets in Section 4.2.3.

Change address. Bitcoin’s UTXO-based model does not allow the partial spending of transaction outputs. Since the sum of input values to a transaction may be larger than

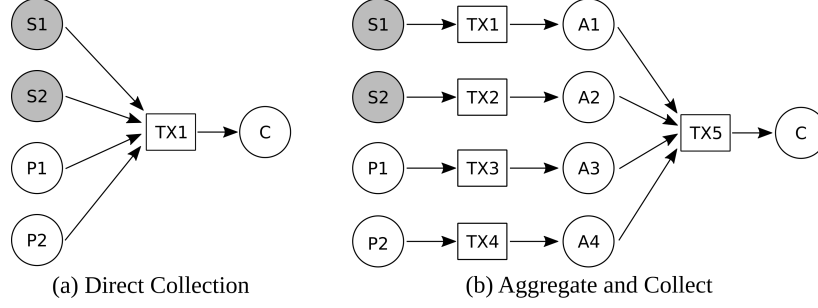


Figure 4.1: Two approaches to collect payments from two seeds S_1, S_2 (in gray) and two unknown payment address P_1, P_2 into a collector address C . On the left, MI clustering discovers the unknown addresses P_1, P_2 , while on the right it does not.

the amount that needs to be paid, a *change address* can be used by the owner of the input addresses to collect back the change. Several heuristics have been proposed for identifying which output slot in a transaction is the change address [130, 6, 76, 61, 101]. Once identified, the change address (and other addresses in its MI cluster) are added to the expanded set. Three works ([196, 116, 45]) use the change address heuristic by Androulaki et al., which checks transactions with two output addresses. If one output address is fresh (i.e., never used before) and the other is not, the fresh address is considered the change address. Instead, Lee et al. use one of the variants implemented by BlockSci [99], but do not detail which one. Recently, Kappos et al. [101] compared different change address heuristics, showing that most have high false positives (FPs). They proposed a new heuristic to reduce FPs, which has not yet been used by any estimation.

Exploration. Cybercriminals may accumulate the received payments prior to cashing them out or sending them to a mixer to obfuscate their origin. MI clustering may capture such aggregation if multiple payment addresses are used as input to the same transaction. However, the effectiveness of applying MI clustering on the seeds depends on how the cybercriminals withdraw their funds. Consider the example in Figure 4.1(a), where the cybercriminals use two seeds (S_1, S_2) and two unknown payment addresses (P_1, P_2) as input to a withdrawal transaction (withdrawal for short) that accumulates the payments into a *collector* address C . In this case, MI clustering would identify the previously unknown payment addresses (P_1, P_2). Instead, in Figure 4.1(b) the funds are first moved from the 4 payment addresses into 4 intermediate aggregation addresses ($A_1 - A_4$) in separate transactions, and then collected using a MI transaction into collector C . In this case, MI clustering on the two seeds would not identify the unknown payment addresses or the aggregators.

Withdrawals from the seeds evade MI clustering if they use a single input address. We can classify withdrawals based on their number of distinct input and output addresses. We focus on distinct addresses rather than on transactions slots because a transaction can use multiple input slots for the same address. For example, a payment address may be reused for multiple victims; each victim payment generates a different UTXO and all UTXOs are consumed by the same withdrawal. We say a transaction is *1-to- n* ($n \geq 1$) if it has a single input address, regardless of its number of input slots. In Section 4.3.5 we measure that 40% of groups only

use 1-to-n withdrawals from their seeds, defeating MI-clustering completely. This result also indicates that such evasion is widely used.

Figure 4.1(b) captures how the Cerber ransomware operated. To handle this family, Huang et al. proposed to manually add the aggregators (A_1, A_2) into the set of seeds. That allows MI to discover the other two aggregators (A_3, A_4) and thus perform a more accurate estimation. In Section 6.7 we discuss how we believe their approach could be generalized to address MI clustering evasion.

4.2.3 Filtering

Not all deposits to the expanded set are necessarily victim payments. The seeds and the additional addresses the expansions identify could be used for other purposes, e.g., for aggregating funds. For these reasons, 8 of the 12 works apply filters to exclude deposits that do not look like victim payments. We describe them next.

Value filtering. If the amounts the victims should pay are known, deposits for other amounts can be excluded. One consideration is that some victims may ignore that, in addition to the requested amount, they also need to pay a transaction fee. This results in some victim payments not reaching the requested amount since the transaction fee is discounted prior to depositing the funds. Some payments can also be slightly higher than the requested amount because the victim conservatively increases the amount to make sure any fees are covered. To account for these small deviations in amounts, some works apply an epsilon around the known payment values [116, 45]. A limitation of value filtering is that, due to limited coverage, it may not be possible to know all valid amounts (e.g., each victim could be given a different amount). Thus, some victim payments could be incorrectly excluded. For example, 13 of the 20 ransomware families considered in [45] have no seeds matching the expected ranges, and thus cannot be estimated. Another limitation is that value filtering does not apply to cases where the victim decides how much to pay, e.g., giveaway scams [115].

Time filtering. If the time periods when a campaign was active are known, deposits to the expanded set outside of those periods can be excluded. This helps exclude other uses of the payment addresses before or after a campaign happened. Similar to value filtering, due to limited coverage, it may not be possible to know all periods when a campaign was active, which may exclude valid victim payments.

Some works combine value and time filtering by taking as input a list of time ranges, each associated with a list of values victims could pay on the period. This handles campaigns where the amounts change over time, e.g., they are lowered when the BTC conversion rate spikes to avoid exorbitant fees that discourage victims to pay.

Online wallet filtering. If a seed is an online wallet in a service, only direct deposits to that seed should be considered. Other addresses in its cluster should not be included in the expanded set. Two works [152, 113] filter exchange clusters by looking for clusters that are outliers, in terms of large number of addresses, large total amount received, and cluster

age. Unfortunately, these filters are not detailed. In addition, two works identify exchange clusters using tag databases that associate addresses with additional information like their owner. Lee et al. [113] use the public WalletExplorer [212] tag database and Huang et al. [90] a proprietary database from Chainalysis [36]. However, Huang et al. [90] do not use the tags to identify seeds that are online wallets. Instead, they filter deposits to the expanded set that do not originate from exchanges, since most victims likely do not own BTCs and need to purchase them from exchanges. However, they discarded this filter due to concerns about the coverage of the tag database. To address this intrinsic limitation of tag databases, Gomez et al. [73] recently proposed to complement tag databases with an ML classifier to identify (untagged) exchanges. However, their exchange classifier has not been used in any estimation.

Double-counting. Simply adding all deposits to addresses in the expanded set, without considering their precedence, may lead to double-counting, i.e., counting the same payment multiple times, thus overestimating the revenue. The simplest such double-counting can occur when, in a transaction, one input address also appears in the outputs, e.g., an input address is used as a change address. While this is not recommended for privacy reasons, it often happens in practice. Imagine Alice has two BTCs in one UTXO from a previous payment and wants to send one BTC to Bob. Instead of creating a fresh change address, Alice uses the sender address as the change address. After the payment, the balance of Alice's address will be one BTC, but the total amount deposited to her address will be three BTCs, two BTCs from the original payment that created the UTXO plus one BTC she received as change from this transaction.

More generally, double-counting can occur when a transaction that deposits into an address in the expanded set has an input address that also belongs to the expanded set. Imagine two deposits. In the first one, address A which is not in the expanded set deposits some funds into address B which is part of the expanded set. In the second one, address B moves the received funds to address C which is also in the expanded set. The funds B received from A are first counted. Then, if we add the second deposit that only moves funds between addresses inside the expanded set, we are double counting the funds that B received from A as also being received by C .

Only the two works by Paquet-Clouston et al. [151, 152] try to avoid double-counting by filtering transactions. In [151], the authors define collectors as addresses receiving deposits from at least two addresses in the expanded set. Then, the collectors found in the expanded set are filtered to avoid double-counting. However, transactions reusing one of the inputs as the change address may still be double counted. In [152], they exclude all transactions with at least one input address and one output address within the expanded set of all sextortion addresses, which indicates money flowing among spammers instead of victim payments. This filtering removes all possible double-counting deposits among all target groups. However, it may also remove genuine payments to each other (e.g., if both groups are estimated together), like those identified in previous works [73]. In this dissertation, we generalize the DC filtering and use it per group in a one-versus-the-world fashion. To estimate the revenue of a group, we filter out any deposit having at least one input address exclusively in the group's expanded set. Estimating all groups together (e.g., all sextortion addresses in [152]) would replicate the filtering proposed by Paquet-Clouston et al.

One issue to keep in mind is that any filter that tries to minimize double-counting will be affected by the coverage issue, i.e., it will not be able to filter deposits between two MI clusters that belong to the group if one of the two MI clusters has not been included in the expanded set, i.e., no seed is available from that MI cluster.

4.2.4 Conversion Rates

Adding the amounts of the filtered deposits to the expanded set produces an estimation of BTC revenue. Due to Bitcoin’s highly volatile conversion rate, the estimated BTCs can have very different values over time. Converting the BTC amount into US Dollars makes it easier to understand the revenue and to compare it with other cybercrime, e.g., those not abusing Bitcoin. To perform this conversion, 9 of the 12 works use the conversion rate from BTC to USD on the day the deposit was received. This conversion rate captures the financial impact on victims, i.e., how much it cost the victims to pay assuming they bought the BTCs from an exchange just prior to paying. Two works [196, 222] use instead the conversion rate on a specific day, e.g., the day of the analysis. Since cybercriminals may not cash out victim payments immediately, this approach includes the rise or depreciation in value of the payments from the day the payment was received until the day of the conversion rate. Finally, one work uses the highest and lowest conversion rates in their analysis period [115], providing a USD range.

4.3 Methodology Impact

This section quantifies how the different methodology options presented in Section 6.6 impact the estimation. We first introduce the analyzed methodologies and datasets in Section 4.3.1, then quantify their impact on the CryptoLocker ransomware in Section 4.3.2, and on our whole dataset in Section 4.3.3.

4.3.1 Methodologies and Datasets

Given the 2 main expansions (excluding the exploration as it is only used for one ransomware family in one work) and 4 filters detailed in Table 4.1, 64 different methodologies can be used for the estimation. We have created a naming scheme to refer to those methodologies. The name always starts with *DD* which is an abbreviation for direct deposits. Then, it can have at most 6 parts for multi-input clustering (*MI*), change address expansion (*CA*), online wallet filtering (*OW*), value filtering (*VF*), time filtering (*TF*), and double-counting filtering (*DC*). We prefix expansions with a plus sign and filters with a minus sign. The last column in Table 4.1 captures the methodology used by each work using our naming scheme.

We have implemented an extension to the WYB platform [73], itself built on top of BlockSci [99], which can produce any of the estimations. Given as input a set of seeds, a block height, a set of value and time filtering ranges, and a methodology string using our naming scheme; it produces the estimation using that methodology. We will release our extension after publication. While our extension implements all previously used methodologies, the results it produces are not guaranteed to be the same as those obtained in the works in Table 4.1,

even when using the same methodology on the same seeds. Differences may happen because the implementation of the expansions and filters may not exactly match the ones in those works. For example, we use BlockSci to obtain clusters using multi-input and change address expansions, while other works use other open-source platforms or their own implementations.

Selected methodologies. We analyze 15 selected methodologies, which cover those used in the surveyed works and additional ones illustrating interesting cases. The first 5 estimations test the filters without applying expansions. We exclude the filter of online wallets as it only applies to expansions. *DD* is the simplest estimation that sums all deposits to the seeds. *DD-VF*, *DD-TF*, and *DD-VF-TF* apply value filtering, time filtering, and both, respectively, on the direct deposits to the seeds using the 7 ranges proposed by Conti et al. [45]. *DD-DC* removes overlaps in the direct deposits caused by transactions where the seed appears as both input and output.

The next 6 estimations correspond to expanding using only MI clustering and then applying the different filters. The final 4 estimations correspond to expanding with both MI clustering and change address expansions and then applying the different filters. MI clustering and change address expansions both use the clusters that BlockSci precomputes for the given block height. MI clustering excludes CoinJoin transactions using the heuristics in BlockSci. *DD+MI* and *DD+MI+CA* first query BlockSci for the clusters of the seeds (precomputed using only MI clustering or both expansions, respectively) and add the seeds and other addresses in the clusters of the seeds to the expanded set. Then, they obtain all deposits to the expanded set and sum their value without any filtering. *DD-OW+MI* and *DD-OW+MI+CA* first query BlockSci for the clusters of the seeds. Then, they check whether the seed clusters appear in the WYB tag database marked as service clusters. For any seed cluster identified as a service, only the seed is added to the expansion set. For other seed clusters, the seed and other addresses in the cluster are added to the expanded set. Then, they obtain all deposits to the expanded set and sum their value without applying any filtering. *DD-OW+MI-DC* applies *DD-OW+MI* but avoids double-counting by filtering deposits to the expanded set that have an input address belonging to any of the clusters in the expanded set. All other estimations correspond to the ones above with value filtering, time filtering, or both, applied on the deposits to the expanded set.

Datasets. We collect all publicly available labeled datasets of cybercrime Bitcoin seeds that we are aware of. These include all datasets mentioned in Table 4.1, as well as an additional dataset of 9,478 addresses used by clippers [73]. Since the ransomware datasets overlap, we use the Ransomwhere dataset [30], which contains the seeds used in prior works. For scam datasets containing addresses for different blockchains [48, 56] we focus exclusively on the Bitcoin addresses. Other sources of Bitcoin cybercrime seeds exist such as abuse databases (e.g., [15]), but they do not provide a reliable classification of cybercrime type. The six datasets used, summarized in Table 4.5, in total comprise 30,424 cybercrime Bitcoin addresses, of which 8,816 have deposits at block height 785,100 (April 12, 2023). Some works clustered the seeds, but their datasets do not include the cluster identifier for each seed. To estimate individual groups, we use a subset of 8,021 labeled seeds belonging to 141 groups: 88 ransomware families, 22 clipper families, and 31 Ponzi schemes.

Work	Blk Height	Methodology	Seeds	Clust.	Adrs.	Payments	BTC	USD
Spagnuolo et al. [196]	2023-12-15	DD+MI+CA-VF	≥ 12	≥ 12	2,118	771	1,226.0000	1,100,000
Liao et al. [116]	2014-01-31	DD+MI+CA-VF-TF	2	1	968	795	1,128.4000	310,472
Conti et al. [45]	2017-12-07	DD+MI+CA-VF-TF	4	1	956	804	1,403.7548	449,275
Huang et al. [90]	2017-08-31	DD-OW+MI-VF	2	2	4,457	-	-	667,000
Paquet-Clouston et al. [151]	489,181	DD+MI-TF-DC	2	1	944	-	1,511.7100	519,991

Table 4.2: Previous estimations on CryptoLocker.

Methodology	Addr.	Deposits	BTC	USD
DD	2	54	100.8608	\$12,532
DD-VF	2	48	94.8625	\$11,786
DD-TF	2	53	100.8607	\$12,531
DD-VF-TF	2	47	93.8625	\$11,667
DD-DC	2	54	100.8608	\$12,532
DD+MI	968	1,101	1,544.9144	\$528,046
DD+MI-VF-TF	968	803	1,130.1121	\$309,935
DD-OW+MI	968	1,101	1,544.9144	\$528,046
DD-OW+MI-VF-TF	968	803	1,130.1121	\$309,935
DD-OW+MI-DC	968	1,077	1,511.7579	\$520,238
DD-OW+MI-VF-TF-DC	968	801	1,110.3121	\$304,791
DD+MI+CA	-	-	-	-
DD-OW+MI+CA	2	54	100.8608	\$12,532
DD+MI+CA-VF-TF	-	-	-	-
DD-OW+MI+CA-VF-TF	2	47	93.8625	\$11,667

Table 4.3: CryptoLocker estimations using different methodologies starting from the two seeds used in [116, 151]. A dash indicates the estimation did not finish in 5 days.

4.3.2 Impact on CryptoLocker

We first analyze the CryptoLocker ransomware because its revenue has been estimated in five prior works with varying results, as illustrated in Table 4.2. For each prior estimation, the table shows the maximum block height (or date) considered in the estimation, the methodology used, the number of seeds, the number of MI clusters for the seeds, the total addresses in the expanded set, the number of victim payments identified after filtering, and the estimation in BTC and USD. The table illustrates that the estimations can widely vary, e.g., the highest estimation of \$1.1M is 3.5 times larger than the lowest estimation of \$310K. But, the estimations use different seeds, methodologies, and block heights. Thus, it is complicated to isolate the impact of each methodology decision on the estimation. To address this issue, we perform estimations using different methodologies, but on a fixed set of seeds and on the same block height. This allows us to analyze the impact of each methodology decision.

Of the 5 prior estimations, two do not specify the seed addresses used [196, 90]. Two use the same pair of seeds [116, 151], and one [45] uses 4 seeds (the two in [116, 151] and two additional ones). We fix the block height at 498,150 (December 7, 2017) and perform two estimations: one starting with the two seeds in [116, 151] and another from the four seeds in [45]. We convert from BTC to USD using the rate at the time of each payment, obtained from CoinDesk [40].

Two seeds. Table 4.3 presents the estimations on the two seeds from [116, 151]. The estimations without expansions are pretty consistent in the range of \$11.6K–\$12.5K. MI clustering is quite successful at discovering additional CryptoLocker addresses increasing the estimations 26–42 times to \$304.7K–\$528.0K. Among filters, value filtering has the largest

Methodology	Addr.	Deposits	BTC	USD
DD	4	321	70,426.0298	\$11,442,063
DD-VF	4	59	123.6568	\$18,006
DD-TF	4	288	57,987.8966	\$10,094,385
DD-VF-TF	4	50	106.8525	\$15,062
DD-DC	4	227	41,614.1073	\$6,339,388
DD+MI	12,770,529	27,617,860	134,110,936.7365	\$26,405,734,619
DD+MI-VF-TF	12,770,529	39,015	54,091.2840	\$32,786,920
DD-OW+MI	970	1,368	71,870.0835	\$11,957,577
DD-OW+MI-VF-TF	970	806	1,143.1021	\$313,330
DD-OW+MI-DC	970	1,208	34,771.1654	\$5,453,205
DD-OW+MI-VF-TF-DC	970	797	1,112.8791	\$305,080
DD+MI+CA	-	-	-	-
DD-OW+MI+CA	4	321	70,426.0298	\$11,442,063
DD+MI+CA-VF-TF	-	-	-	-
DD-OW+MI+CA-VF-TF	4	50	106.8525	\$15,062

Table 4.4: CryptoLocker estimations using different methodologies starting from the four seeds used in [45]. A dash indicates the estimation did not finish in 5 days.

impact, especially after MI clustering, indicating that some deposits may not correspond to victim payments. Online wallet filtering has no effect as none of the seeds are online wallets. The double-counting filter reduces the estimations by 1.5%–1.7%.

Surprisingly, we are unable to compute some estimations with change address expansion. The cluster with the seeds returned by BlockSci has 160M addresses. Due to the huge number of deposits to that cluster (261M), the estimation does not finish in 5 days and we stop it. Three works in Table 4.2 claim to use change address expansion. However, their estimates are very close to ours using only MI clustering. One possibility is that the change address expansion they implemented was not transitive, and thus avoided the explosion affecting BlockSci’s implementation. Similar to what has been observed in prior work [101], our results indicate that change address heuristics can have high false positives. We further examine this issue for all groups in Section 4.3.3. Once online wallets are filtered, the estimations with change address expansion finish. This happens because the huge cluster that contains the seeds also contains multiple addresses tagged as exchanges. Thus, the cluster is considered to belong to an exchange, and both seeds are (incorrectly) considered online wallets. Thus, *DD-OW+MI+CA* matches the results for *DD* and *DD-OW+MI+CA-VF-TF* matches the results for *DD-VF-TF*.

Four seeds. Table 4.4 presents the estimations on the four seeds from [45]. In this case, the estimations without expansions widely vary from \$15K up to \$11.4M, indicating that the additional seeds directly receive millions of USD in deposits. Value and time filtering remove most of those deposits as likely not being victim payments resulting in a \$15K estimate that is only 30% higher than with two seeds. After applying MI clustering the expanded set grows to 12.7M addresses and the estimation is an astronomical 26 Billion USD. The reason for this is uncovered using the online wallet filter, which identifies one of the additional seeds as an online wallet in the btc-e exchange, and the other as belonging to an incorrect huge cluster due to an address whose private key is the empty string [73]. Considering both additional seeds as online wallets reduces the estimation to \$11.9M and value and time filtering brings it down to \$313K. A key question is whether the two additional seeds are indeed related to CryptoLocker. We are inclined to think they are not but have no definite proof. Still,

Cybercrime	Dataset	Seeds	Labels	OW	BTC	USD
Ransomware	[146]	7,352	88	12	31,434.4069	\$115,513,569
Giveaway scams	[115]	494	-	17	386.0907	\$12,207,462
Ponzi schemes	[9]	32	31	7	7,956.3913	\$5,029,029
Clippers	[73]	637	22	23	891.5331	\$2,814,443
Sextortion	[152]	248	-	5	305.9584	\$1,504,437
Exchange scams	[222]	53	-	2	117.2569	\$1,021,764
Total		8,816	141	66	41,091.6373	\$138,090,704

Table 4.5: Cybercrime type estimates on April 12, 2023 using the seeds from public datasets and a *DD-OW+MI-DC* estimation.

this experiment illustrates how failing to consider seeds that are online wallets or belong to other problematic MI clusters, can hugely overestimate the financial impact. Value filtering significantly ameliorates the impact of incorrectly applying MI clustering on online wallets, but it cannot entirely correct the problem as some fraction of the very large number of deposits will match the expected value ranges.

4.3.3 Impact on the Whole Dataset

This section quantifies the impact of expansions and filters on the whole dataset. We exclude the value and time filters since we only have ranges for a few ransomware families. All experiments are performed on block height 785,100 (April 12, 2023).

Online wallet filter impact. We use BlockSci to precompute the MI clusters and query the 8,816 seeds to obtain their cluster identifier. Then, we use the WYB tag dataset to identify tagged clusters. This step flags 58 seeds. Of those, 57 are online wallets in exchanges. The other is the only seed for the Razy family in the Ransomwhere dataset, which really is an FBI address used to seize the SilkRoad funds after its takedown. We remove Razy from our datasets.

An intrinsic issue in tag databases is the limited coverage, i.e., only a fraction of all MI clusters belonging to services will be tagged. Indeed, when we perform an initial *DD-OW+MI-DC* estimation for each cybercrime type, the estimation for giveaway scams returns an astronomical 6.1 Billion USD. This indicates that the WYB tag database likely missed some service clusters. To identify missed (i.e., untagged) service clusters, we run all MI seed clusters with at least 1K addresses through the exchange classifier provided by WYB. The classifier identifies 8 of those 11 MI clusters as exchanges: 5 in the giveaway scams dataset, 2 in the Ransomwhere dataset, and 1 in the exchange scams dataset. We consider the 9 seeds in these 8 clusters to be online wallets for all other estimations. Had we not identified these 9 online wallets, we would have grossly overestimated the financial impact of three cybercrime types: \$6.1B for giveaway scams (503 times higher than our final estimation), \$543M for ransomware (5 times higher), and \$231M for exchange scams (226 times higher).

While only 0.7% of seeds are online wallets, they can introduce huge overestimation. To avoid missing online wallets, we recommend complementing tag databases with ML classifiers.

MI clustering impact. We use BlockSci to precompute the MI clusters and query each of the labeled seeds that are not online wallets to obtain their cluster. The seeds belong to 968 clusters, with an average cluster size of 19.5 addresses, and a median size of 1 address. Four clusters contain seeds from more than one group. One cluster holds the seeds for *mekotio* and *mekotion40*, which are known to be run by the same operators [73]. Another holds Towerweb and Cyptohitman seeds and was already reported in [151]. The other two clusters hold Jigsaw and Cryptowall, and TripleM and APT, respectively. We have not found reports linking the groups in these two clusters. However, the small size of those two clusters (3 and 4 addresses, respectively) makes us think they may be true relations or incorrect labels on the seeds. Clustering FPs typically create a snowball effect that quickly grows the clusters, as we show next for the change address expansion.

Change address impact. We use BlockSci to precompute the MI+CA clusters and query each of the 8,021 labeled seeds from the 141 groups to obtain their cluster. The 8,021 seeds belong to 251 clusters with an average cluster size of 2.2M addresses, and a median size of 2 addresses. The higher average is due to a huge cluster with 543M addresses. This cluster contains seeds from 91 of the 141 groups (64.5%). For 70 groups, all their seeds are in this cluster. It also contains addresses tagged by WYB as belonging to a variety of different services such as exchanges, gambling sites, and mixers. All other clusters with seeds contain one group at most, except for one cluster that contains two malware families known to belong to the same operators (*mekotio* and *mekotion40*) [73].

The fact that one cluster contains 91 unrelated groups and many unrelated services indicates the change address heuristic is introducing FPs. Because of this, we suggest that estimations do not use change address expansion. We further discuss this in Section 6.7.

Double-counting filter impact. To quantify the impact of the DC filter, we perform two estimations on all 8,816 seeds using the DD-OW+MI and DD-OW+MI-DC methodologies, respectively. DD-OW+MI estimates a total revenue of 171,994.9777 BTC, while DD-OW+MI-DC estimates 41,091.6373 BTC. Thus, the DC filter reduces the BTC estimate by 76.1%. This is much higher than the reduction observed in CryptoLocker, as the impact of the DC filter quickly grows as the volume and fund movement increases. In detail, the DC filter drops 43.7% of deposits to the expanded set and only 80.8% of the addresses in the expanded set receive funds from outside the expanded set.

4.3.4 Cybercrime Estimations

All works in Table 4.1 have estimated one or multiple groups of the same type of cybercrime. In this section, we instead compare revenue across abuse types. We perform all estimations at the same block height of 785,100 (April 12, 2023), using the same *DD-OW+MI-DC* methodology, and the conversion rate on the day of each payment. We select this methodology because we have shown that change address expansion introduces many false positives and we only have value and time filtering ranges for some ransomware families.

Name	Seeds	OW	Clust	Addr.	BTC	USD
R:Netwalker	66	0	65	329	3,131.7562	\$28,262,120
R:Conti	26	0	26	35	410.8025	\$18,869,852
R:REvil	7	0	6	77	369.5363	\$12,540,689
R:DarkSide	3	0	3	3	158.7086	\$9,124,091
R:Locky	7,036	0	2	7,094	15,396.6934	\$7,833,613
R:Ryuk	26	0	26	39	866.8069	\$4,844,149
R:RagnarLocker	1	0	1	1	414.0007	\$4,547,241
R:MountLocker	1	0	1	1	298.5000	\$4,226,277
R:BlackMatter	1	0	1	1	96.3863	\$4,068,295
R:Makop	1	0	1	996	316.0883	\$3,853,713
R:Egregor	9	0	9	9	197.9554	\$3,129,599
R:Tejodes	1	0	1	178	345.5527	\$2,798,239
P:Leancy	1	0	1	1	3,092.8482	\$1,896,583
R:CryptXXX	1	0	1	1,742	3,338.5464	\$1,877,833
R:DMALockerv3	9	0	3	177	1,833.9286	\$1,757,464
R:MedusaLocker	3	0	3	26	56.1546	\$1,454,776
R>HelloKitty	1	0	1	1	32.3529	\$1,070,653
R:Bitpaymer	1	0	1	1	90.0004	\$1,058,347
C:cliptomaner	1	1	1	1	38.7643	\$1,004,397
P:Cryptory	1	1	1	1	1,677.5167	\$886,689
Total	8,021	42	383	17,040	40,282.3312	\$123,357,041

Table 4.6: Top-20 groups by USD financial impact on April 12, 2023 using a *DD-OW+MI-DC* estimation. Ransomware families are prefixed by R:, clipper families by C:, and Ponzi schemes by P:. The last row captures total revenue of all 141 labeled groups.

Cybercrime comparison. The right side of Table 4.5 summarizes the BTC and USD estimations for the 6 abuse types. The largest estimated revenue is \$115.5M for ransomware, followed by giveaway scams (\$12.2M), Ponzi schemes (\$5.0M), clippers (\$2.8M), sextortion scams (\$1.5M), and exchange scams (\$1.0M). Giveaway scams collect lower BTCs (386.0907) than Ponzi schemes (7,956.3913) and clippers (891.5331). However, their USD financial impact on victims is larger due to their seeds being collected in 2022, when the conversion rate BTC-USD was higher than for older seeds.

Group comparison. We also estimate the 141 groups. Table 4.6 shows the top-20 groups by USD revenue and the 141 groups are detailed in Table 4 in the Appendix. We prefix each name with R: for ransomware families, C: for clipper families, and P: for Ponzi schemes. Of the top-20 groups, 17 are ransomware families, two are Ponzi schemes and one is a clipper family. The highest revenue is for the *Netwalker* ransomware which collects \$28.2M. The highest revenue among Ponzi schemes is *Leancy* with \$1.8M and the highest revenue among clippers is *Cliptomaner* with \$1.0M. The average revenue per operation is \$874K, and the median is \$17.5K. There are 19 groups with revenues higher than one million USD and 14 groups (13 ransomware families and one Ponzi scheme) with revenue below \$20, likely due to limited seed coverage for those groups.

Similar to what we observe in the cybercrime comparison, higher BTC revenue does not necessarily imply a higher financial impact on victims, measured using fiat currency, due to the wide oscillations of the Bitcoin conversion rate. For example, Locky collected the most BTCs (15,396.6934). However, its financial impact measured in USD is below that of DarkSide, which received nearly two orders of magnitude fewer BTCs (158.7086). This last happens because, as shown in Table 4, Locky was active from the beginning of 2016 to mid-2017, when the value of 1 BTC was \$780 on average. In contrast, DarkSide was active

during the first half of 2021, when the conversion rate ranged from \$29k to \$61k (\$55.2k on average). This example highlights the importance of using the conversion rate at the time of the payment for more accurate estimations of the financial impact on the victims.

The top-4 earners in Table 4.6 (Netwalker [46], Conti [200], REvil [77], DarkSide [198]) leverage the ransomware-as-a-service (RaaS) model, where the ransomware operators recruit affiliates to take care of the infections [131]. Affiliates are paid a fraction of the victim’s ransom payment or a fixed fee. Outsourcing infections to affiliates helps ransomware gangs target more victims. These families focus on high-value targets such as large companies and hospitals, e.g., DarkSide was behind the Colonial Pipeline attack [167]. They often use different ransom values for each victim, threaten victims to release their data publicly, and may even perform double extortion, i.e., demand a second payment for not leaking the data. Targeted victims may not want to disclose the attacks, making it difficult to obtain seeds. For example, there are reports of DarkSide having at least 90 victims in the US [167], but we only have 3 seeds for that group and MI clustering does not reveal additional payment addresses. We examine the impact of the lack of coverage in Section 4.4.

4.3.5 Multi-Input Evasion

This section examines how useful MI clustering on the seeds is, and whether cybercriminals may be using evasive techniques to defeat it. For this, we examine the withdrawal transactions from the seeds of the 141 groups, and their MI clusters. For each group, we first remove seeds identified as online wallets. Then, we compute the total number of withdrawals from the remaining seeds and the fraction of *1-to-n* withdrawals. Next, we compute the expanded set of each group by doing the union of the addresses in the MI clusters of the seeds that are not online wallets and recompute the above numbers for all withdrawals from the expanded set of each group.

Figure 4.2 shows the cumulative distribution of 1-to-n withdrawals across the 141 groups. 40% of the groups exclusively move funds from their payment addresses using MI-defeating 1-to-n withdrawals. Three-quarters of the groups use at least 50% of 1-to-n withdrawals, indicating their preference for such MI-defeating withdrawals. Only 9% of the groups do not use 1-to-n withdrawals at all, but most of these have not withdrawn any funds yet. The fact that 40% of the groups exclusively use MI-defeating withdrawals and 75% use them more than half of the time, likely indicates that cybercriminals are aware of the limitations of MI clustering and are actively taking steps to defeat it. We discuss how to address this evasion in Section 6.7.

4.4 Coverage Impact

A recurring issue in cybercrime estimations is that they start from a small number of seeds (a median of one seed per group) and it is not clear how many other payment addresses with victim deposits may exist. Expansions are used to try to discover missing payment addresses, but as shown in Section 4.3 their effectiveness is limited. As far as we know, no prior work has tried to quantify the impact of the lack of coverage in the estimations, due to the complexity of obtaining the needed vantage point. In this section, we quantify this issue for the first

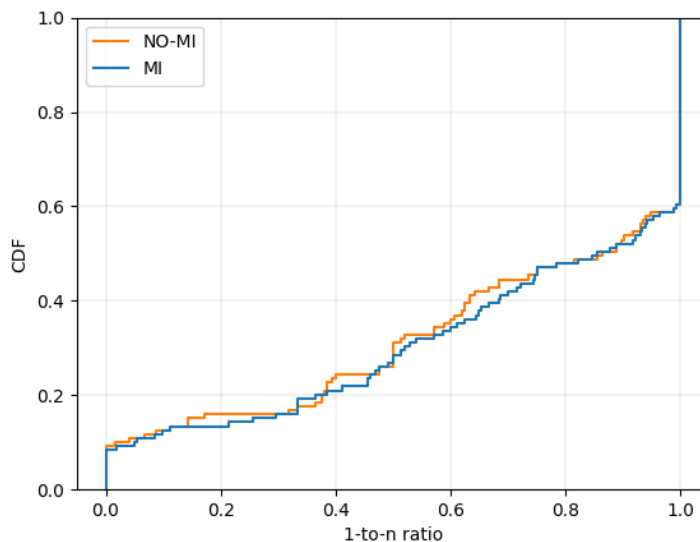


Figure 4.2: Cumulative distribution of the proportion of withdrawals with one input address (1-to-n) over all the withdrawals across the 141 groups for both withdrawals of seeds (orange line) and withdrawals of their MI clusters (blue line).

time using two novel techniques that allow us to obtain very high coverage, possibly nearly complete, on the DeadBolt server ransomware [1]. We introduce DeadBolt in Section 4.4.1, describe our DeadBolt datasets in Section 4.4.2, present our techniques to expand DeadBolt’s coverage in Section 4.4.3, and compare the estimations obtained from different vantage points in Section 4.4.4.

4.4.1 DeadBolt

On January 25th, 2022, BleepingComputer first reported a new server ransomware strain that called itself *DeadBolt* and was spreading by exploiting vulnerabilities in network-attached storage (NAS) devices [1]. DeadBolt encrypted specific data directories and file extensions and hijacked the login page of the NAS to display a ransom note titled “WARNING: Your files have been locked by DeadBolt”. The ransom note requested a Bitcoin payment of 0.03 BTC (1,100 USD at the time) to be sent to a Bitcoin payment address to obtain the decryption key. In mid-June 2022, DeadBolt introduced a new ransom value of 0.05 BTC, possibly due to a drop in the BTC-USD conversion rate at that time, and both amounts are used therefore.

One novel feature in DeadBolt is that after the victim pays the ransom, the decryption key is automatically posted to the Bitcoin blockchain by the DeadBolt operators. Figure 4.3 illustrates the two transactions involved. First, the victim (address V) pays the ransom of 0.03 BTC or 0.05 BTC to the payment address P . Then, the DeadBolt operators perform a deposit of value 0.0000546 BTC (\$1.2) from a *key release address* K to the payment address P . This second transaction has an OP_RETURN output that stores the AES-128 decryption key on the blockchain.

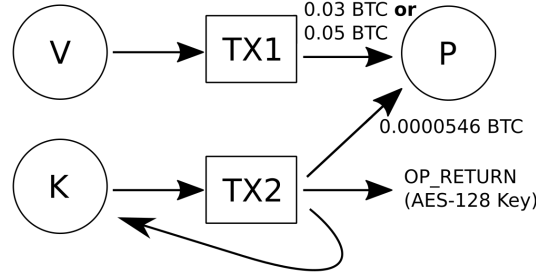


Figure 4.3: DeadBolt key release transactions. First, the victim deposits the ransom of 0.03 BTC or 0.05 BTC to the payment address. Then, the cybercriminals release the decryption key to the blockchain through a deposit to the payment address.

Engine	Events	IP	Port	ASN	CC	Notes	Addr.	Seeds
Shodan	23,413	9,999	171	769	89	4,938	4,940	64
Censys	15,886	6,147	66	671	87	-	4,014	49
All	39,299	11,282	199	780	98	4,938	4,997	64

Table 4.7: DeadBolt datasets summary.

In October 2022, the Dutch National Police tricked the DeadBolt operators into handing over 155 decryption keys by performing ransom payments with a tiny fee when the blockchain was heavily congested and canceling the payment transactions before they appeared in a block [69]. Since the decryption key was posted automatically to the blockchain without waiting for the payment transaction to appear in a block, the police recovered 155 keys before the attackers realized. Then, they set up a webpage that outputs the decryption key of any of the 155 addresses [169]. After that event, the DeadBolt operators adjusted the key release procedure in Figure 4.3 to wait for the payment transaction to appear in a block before releasing the decryption key. From this on, the time difference between victim payments and key release transactions starts varying significantly, reaching even 3 days in some cases, indicating the key release transactions are now manually triggered.

4.4.2 DeadBolt Datasets

DeadBolt exploited network-facing vulnerabilities in the NAS and hijacked the NAS login page to display the ransom note. A NAS had to be connected to the Internet to get infected. Thus, the ransom note was (potentially) visible to Internet scanners. However, Internet scanners may not observe all victims since an infected NAS may have been disconnected or cleaned before being scanned. We obtain data about DeadBolt infections from two Internet scanners: Censys [32], and Shodan [190]. The datasets are summarized in Table 4.7.

Censys. Censys publishes servers infected with DeadBolt in a Google Datastudio [33]. Each entry in the dataset represents an infection and contains the IP address, the port number, the country code (CC) and ASN of the IP address, the payment address, the ransom amount, and the DeadBolt variant. The raw content of the ransom notes is not available. We collect Censys data for four months, from December 13, 2022, until March 13, 2023. We obtain

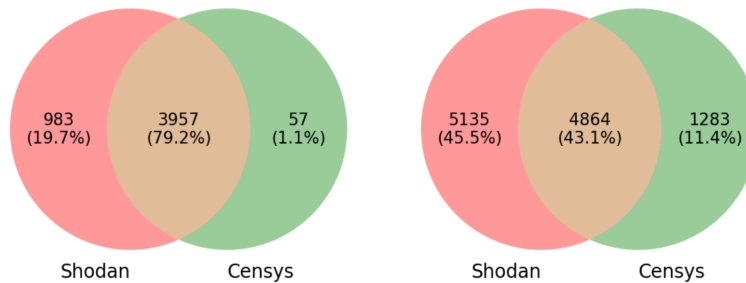


Figure 4.4: Overlap in DeadBolt datasets. Left: Bitcoin addresses. Right: IP addresses.

4,014 DeadBolt payment addresses from 6,147 IP addresses. The large number of payment addresses likely indicates each infected server is given a different one. The number of IP addresses is larger than the number of payment addresses likely due to infected servers that change IP addresses.

Shodan. We identify DeadBolt events in Shodan by querying its API with the dork `http.title:"Your files have been locked by DEADBOLT."`. Each query returns events from the last 30 days, so we query once a month, from December 13, 2022, until March 13, 2023. From each event, we extract the timestamp, IP address, port number, country code and ASN of the IP address, and the ransomware note in HTML format. We obtain 4,938 distinct ransom notes from 9,999 IP addresses. We extract 4,940 payment addresses from the ransom notes using the *iocsearcher* tool [121].

Both scanners observe different sets of infected servers, mainly due to their scanning happening at different times. Figure 4.4 shows that 79.2% of the payment addresses appear in both datasets, 19.7% only in Shodan, and 1.1% only in Censys; and also that 43.1% of the IP addresses appear in both datasets, 45.5% only in Shodan, and 11.4% only in Censys. Thus, our Shodan dataset has slightly better coverage on DeadBolt, but the set of payment addresses and infected IPs observed from both vantage points significantly overlap. The larger overlap in payment addresses is likely due to servers appearing multiple times on different IP addresses.

4.4.3 Increasing the Coverage

Of the 4,997 DeadBolt payment addresses collected from Shodan and Censys, only 64 addresses had received deposits as of April 12, 2023. For 30 of those 64 seeds, the Bitcoin ledger does not show the victim’s payment, but only the key release transaction, all of them on October 13, 2022. These 30 addresses are part of the 155 addresses for which the Dutch police recovered their decryption key. Since the ransom payment was withdrawn after the key was released, the payment transactions do not appear on the ledger. The fact that our datasets only have 30 of the 155 addresses for which the Dutch police recovered the keys indicates that we only have partial coverage. The 34 seeds do not have withdrawals. Thus, the received payments have still not been moved and expansions can not reveal new addresses. In the remainder of this section, we present two novel techniques that leverage unique characteristics of DeadBolt, which allow us to obtain very high coverage, possibly nearly complete, on the payments

performed by DeadBolt victims.

Key release analysis. All key release transactions to the 64 seeds have the same format, illustrated in Figure 4.3, with one input key release address and three output slots that correspond to (a) the payment address that receives 0.0000546 BTC, (b) the input key release address which is used as change address and (c) the OP RETURN script that stores the decryption key on the blockchain.

Critically, all key release transactions for the 64 seeds originate from one of two key release addresses: *bc1qh6* or *bc1q62*. We identify Bitcoin addresses by their first 6 characters and provide the full addresses in Table 1 in the Appendix. The reuse of key release addresses to post the decryption keys of many victims allows us to identify ransom payments to previously unknown payment addresses (i.e., not in our datasets) that correspond to infected servers that the Internet scanners did not observe (e.g., NAS devices disconnected from the Internet right after infection). For this, we first query BlockSci for all withdrawal transactions from the two key release addresses. We obtain 2,064 withdrawals for *bc1qh6* and 417 for *bc1q62*. All these transactions match the expected pattern. Output addresses in these transactions receiving 0.0000546 BTC are (potentially unknown) payment addresses. This technique identifies 2,418 previously unknown payment addresses.

The 2,482 payment addresses with deposits we know at this point contain 154 addresses with only the key release transaction, but no victim payment. This almost matches the 155 keys the Dutch Police were able to extract in October 2022. Still, there is one missing address indicating we may still be missing some victim payments. One possible reason for this is that there could exist other key release addresses that we have not observed in our datasets.

Key release signature. We propose a novel technique to identify additional DeadBolt key release addresses. We leverage that DeadBolt key release transactions are quite distinct, build a signature for them, and scan all transactions in the Bitcoin ledger for over 15 months to search for unknown key release addresses. Given a transaction, our signature checks that it has one input and three outputs slots, where the output slots correspond (in any order) to (1) an OP RETURN address, (2) an address receiving exactly 5460 satoshis (0.0000546 BTC), and (3) an address that is the same as the sending address. We apply the signature to all Bitcoin transactions from block 716,591 (2022-01-01) up to block 785,100 (2023-04-12). The search takes 16 minutes to complete. We decode the OP RETURN payload data from the matched transactions and observe that several decoded values start with the string *omni*, indicating that they are part of the OMNI protocol [144]. After filtering the OMNI transactions, all remaining transactions originate from 3 addresses, two of them are the known key release addresses, and the third one (*bc1q3g*) is a previously unknown key release address. The newly discovered key release address has delivered the decryption key to 18 (previously unknown) payment addresses. Of those, 17 receive the expected payments of 0.03 BTC or 0.05 BTC. The other address only received the key release transaction, but no victim payment. This address receives the missing key of those recovered by the Dutch police, as confirmed by using the public Web service.

The fact that we have scanned all Bitcoin transactions since DeadBolt started to operate and

Collection	Coverage	Seeds	Addr.	Dep.	BTC	USD
Censys	2.0%	49	49	104	2.48620746	54,136.66
Shodan	2.6%	64	64	114	2.82608808	63,077.10
Censys+Shodan	2.6%	64	64	114	2.82608808	63,077.10
Key rel. analysis	99.3%	2,482	2,483	2,596	97.83718238	2,453,532.39
Key rel. signature	100.0%	2,500	2,501	2,615	98.35008368	2,472,845.02

Table 4.8: DeadBolt estimations from different vantage points.

have not identified further key release addresses, together with the fact that we observe all 155 keys recovered by the Dutch Police, gives us strong confidence that we have achieved nearly perfect coverage on DeadBolt’s victim payments. Of course, our key release signature assumes that there is a unique pattern of key release transactions. If there were multiple such patterns we could miss some victim payments.

4.4.4 Coverage Comparison

Table 4.8 summarizes the coverage and revenue estimation on DeadBolt from different vantage points: using only Censys, only Shodan, both Censys and Shodan, and extending our coverage using the key release analysis and the key release signature cumulatively. The table shows the fraction of payment addresses found at each step (Coverage), the number of payment addresses with deposits (Seeds), the number of addresses found after applying MI clustering (Addr.), the number of victim deposits (Dep.), the BTC estimation on block height 785,100 using a *DD-OW+MI-DC* methodology, and the USD estimation using the conversion rate at the time of each payment.

Had we focused only on the payment addresses observed in the union of the Censys and Shodan datasets, we would have identified only 64 seeds, 2.6% of all payment addresses finally identified. Of those 64, only 34 had victim payments, and the MI expansion fails to identify additional addresses since the funds have not yet been moved from those 34. Thus, we would have estimated a very modest revenue of 2.826 BTCs or \$63,077.

Instead, by applying our two coverage-expanding techniques, we identify 39 times more seeds. The estimation jumps to 98.350 BTC, 35 times higher than the 2.826 BTC. The estimated USD revenue is \$2.47M, 39 times higher than the \$63,077 using the Internet scanners datasets. Thus, the estimation using only the Shodan and Censys datasets would have been 2.5% of the total USD estimation, highlighting the huge impact of the (lack of) coverage in the estimation. This in turn means our estimations in Section 4.3.4 may significantly underestimate the financial impact of Bitcoin-related abuse types.

Figure 4.5 shows, on the left, daily BTC amounts received by DeadBolt when using our most complete coverage. On the right, it shows how the revenues accumulate over time. As shown, DeadBolt had a significant payment spike in September 2022. A likely reason for this spike is an infection campaign launched using a zero-day vulnerability affecting NAS devices (CVE-2022-27593) [164].

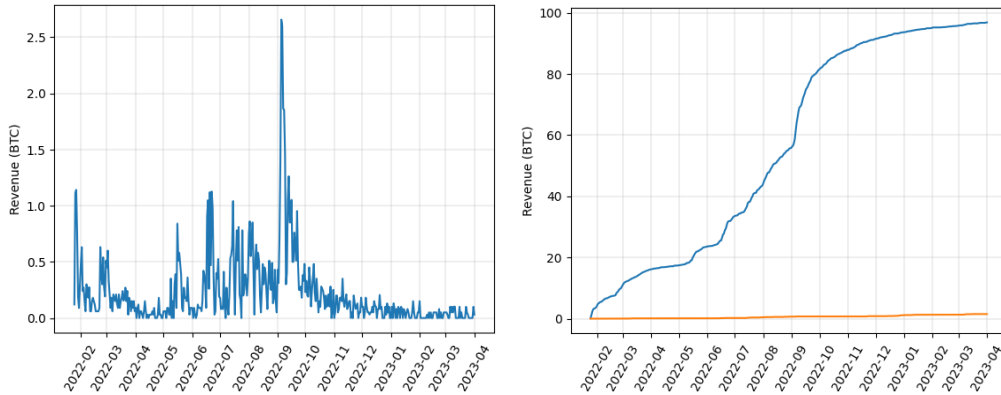


Figure 4.5: DeadBolt’s revenue using full coverage. On the left is the daily BTC revenue. On the right, the cumulative revenue over time using full coverage (blue line) and only scan engines (orange line).

4.5 How To Estimate

It is widely accepted that estimated cybercrime Bitcoin revenue is a lower bound on the actual revenue due to the impact of the lack of coverage on the seeds. Indeed, we have quantified the coverage impact to be very significant on the DeadBolt ransomware. However, our results show that some methodology steps, e.g., handling online wallets in services and using change address heuristics, may introduce huge overestimation. In some cases, such methodological errors can outweigh the underestimation caused by the lack of coverage, thus overshooting the actual revenue.

This section presents our recommendations on how to perform estimations. At a high level, we recommend performing two BTC revenue estimations: DD and DD-OW+MI-DC. First, as a conservative estimation, we recommend authors to provide the sum of direct deposit on seeds (DD). This estimation does not use expansions. Thus, it is easy to compute without needing specialized platforms, e.g., deposit transactions to seeds are available from blockchain explorer webpages and APIs. On the other hand, it may significantly underestimate revenue if seed coverage is low. Overestimation only happens if seeds are incorrectly labeled or reused for other purposes. We discuss the latter below in the discussion of value and time filtering.

As a tighter estimation, we recommend using DD-OW+MI-DC, which we used for the comparison in Section 4.3.4. This estimation uses the MI clustering expansion (with special handling for seeds that are online wallets in services), avoids change address expansion, and removes duplicate deposits. We have shown that identifying online wallets in services like exchanges is critical to prevent huge overestimation when using MI clustering. For such identification, it is possible to use publicly available tag databases like those in WYB [73] and GraphSense [84]. But, tag datasets have limited coverage. For example, our initial giveaway scams estimation was an astronomical 6.1 Billion USD (503 times higher than the final estimation) due to seeds in five exchanges not identified by the WYB tags. Because of that, an ML classifier such as the one in WYB should be used to complement tag databases. An alternative is using heuristics to exclude outlier clusters. But, it is hard to select reliable

thresholds on cluster size (e.g., Locky’s MI cluster has 7,094 addresses) or BTC volume (given the highly volatile BTC conversion rate). The best option may be to exclude clusters with multi-million USD revenue, but this may prevent the analysis of large operations. To prevent overestimation, MI clustering should use detection heuristics for CoinJoin mixing transactions [99, 76], available in open-source platforms [99, 84]. This estimate avoids the change address expansion due to its potentially high false positives, which prevented us from obtaining some results in Section 4.3.2. Recently, Kappos et al. [101] proposed a new change address heuristic with lower false positives than prior heuristics. This heuristic has not been used for estimations yet, but we would like to evaluate it in future work. This estimate also avoids double counting deposits, which we have shown can introduce overestimation. In particular, we have measured that applying the double-counting filter on all 8,816 seeds removes 76% of the estimated revenue.

Value and time filtering. The proposed estimations do not use value and time filtering for several reasons. First, these filters are helpful only if payment addresses are reused for multiple purposes (e.g., different campaigns or abuse types) and the estimation targets just one (e.g., a specific campaign). Second, they are specific to each group, and the selection of a proper filtering range is affected by the group’s coverage. In addition, value filtering does not apply to cases where the payment amount is not pre-determined (e.g., clippers and giveaway scams) and when each victim is assigned a different amount (e.g., some ransomware families). When authors believe value and time filtering (or any other group-specific filtering) should be applied, we recommend providing estimations with and without those filters.

Other blockchains. Three surveyed works also collect seeds for other blockchains such as Cardano, Ethereum, Monero, and Ripple [113, 222, 115]. We have focused on estimating Bitcoin revenue for two reasons. First, prior works apply exclusively a DD estimation on other blockchains. Thus, there are no methodologies to compare. Besides, our platform currently only supports Bitcoin. To estimate the revenue of a cybercriminal activity that solicits victim payments on multiple blockchains, we recommend performing a separate DD estimation on each blockchain (e.g., using the webpage or API of blockchain explorers) and adding the estimations. We expect future research to investigate more sophisticated estimation methodologies for other blockchains (e.g., novel expansions).

Conversion rate. The above recommendations focus on estimating BTC revenue. To estimate the financial impact on victims using fiat currency (e.g., USD) we recommend applying the BTC to fiat conversion rate on the day each payment was received. This conversion accounts for a victim’s loss at the time it happened. Conversions using a later fixed date are difficult to interpret due to the high variability of the BTC conversion rate over time. In particular, we have shown how older cybercriminal campaigns that collected high amounts of BTC (e.g., Locky) had a lower USD financial impact on victims than more recent campaigns that collected fewer BTCs (e.g., DarkSide).

Enabling replicability. Bitcoin transactions are immutable and public, so in theory, estimation results should be easy to replicate. In practice, it is often not so due to missing

information. To make results replicable, we recommend that estimation works estate their estimation target (e.g., a cybercrime type, a cybercriminal group, or a specific campaign) and release their seeds and the block height used for the estimation. If applicable, they should also release filtering ranges and clustering results. We also recommend releasing other payment addresses that have not received deposits yet, as that could change at higher block heights. If using expansions, authors should mention which platform they are using to implement them and release their code if custom. If using change address expansion, authors should specify the specific variant employed.

4.6 Further Discussion

This section discusses other insights and future improvements.

Evasion. MI clustering fails to find new addresses for 40% of the 141 groups, arguably indicating cybercriminals are actively evading this expansion. Huang et al. [90] addressed this issue for the Cerber ransomware by observing that the seeds sent the funds to aggregators, on which MI clustering could be successfully applied. We believe their approach can be generalized using recent platforms for tracing cybercriminal flows [73]. In particular, it may be possible to perform a 1-step forward exploration from the seeds and apply heuristics to determine if the discovered addresses belong to the cybercriminals. For example, if the attackers use 1-to-1 withdrawals then the no change heuristic [20] would determine the destination address belongs to the seed owners. Cybercriminals can also evade the estimation by hiding their payment addresses. For example, some ransomware families do not provide the payment address and value in their ransom notes, but only a contact email or IM address [60]. That extra level of indirection complicates the collection of seeds.

Another common evasion is the use of mixers. While their goal is to obfuscate the destination of funds, mixing transactions can also introduce estimation errors. First, undetected mixing transactions may lead to overestimation with MI clustering by bringing unrelated addresses from other entities into the expanded set. Second, undetected mixing deposits to the expanded set addresses, coming from unknown addresses of the same entity, won't be removed by the DC filter, thus introducing overestimation. We use mixer tags available in WYB [73] and CoinJoin detection heuristics implemented by BlockSci [99, 76] to address both issues. However, we acknowledge that both approaches are largely incomplete. Heuristics in BlockSci target CoinJoin, but there exist a variety of other mixing protocols (e.g., CoinShuffle [178]) and mixing techniques [221]. Tags can only identify known mixing services and may miss their mixing transactions, e.g., if mixing commission fees are not aggregated. Future work should develop newer mixing detection techniques to incorporate into blockchain analysis platforms.

Generality of DeadBolt results. Our two coverage-increasing techniques are specific to DeadBolt. Thus, the coverage impact may differ for other groups. We believe the coverage impact will likely be larger in other cases because Internet scanners should have better coverage on server ransomware than most security vendors would on a malware family. Furthermore, our two techniques generalize beyond the DeadBolt coverage measurement. The key release

analysis technique is an instance of doing one backward exploration step from a payment address to identify a key release address, followed by one forward exploration step from the key release address to discover other payment addresses. It is related to the exploration Huang et al. [90] performed for Cerber, although that case was one forward step from the seeds followed by one backward step. We believe such forward+backward or backward+forward exploration will be critical to improving estimation methodologies (e.g., to address MI clustering evasion). The key release signature technique builds a signature for a distinctive malicious transaction and scans all blockchain transactions in a time range to identify more transactions matching the signature. We believe it should be possible to build signatures for other types of malicious Bitcoin transactions, such as those used in C&C signaling [73].

DeadBolt conversion rate. Measuring the conversion rate of a cybercrime, i.e., the fraction of targets that pay, is challenging and may require developing novel methodologies [100]. The fact that DeadBolt’s ransom notes are publicly available on the Internet allows us to identify infected servers (i.e., victims), obtain their payment addresses, and check what fraction received a deposit in the expected ranges ($[0.029, 0.031]$ or $[0.049, 0.051]$). Since we collect 4,997 payment addresses from Censys and Shodan, and only 34 of those have received valid payments, this indicates that 0.7% of the victims paid the ransom. This estimate relies on each server being assigned a unique payment address. Among the 2,500 payment addresses with victim payments, only 13 (0.52%) receive more than one deposit in the expected ranges. Thus, while not every server may have been assigned a unique address (it is unlikely that the same victim paid multiple times), most seem to have their own, and the 0.7% ratio is likely a good estimate of DeadBolt’s conversion rate. Given the nature of DeadBolt targets, i.e., NAS devices possibly storing much data (including data backups), this conversion rate may be very specific to DeadBolt and may not represent other (e.g., desktop) ransomware, which may have lower conversion rates.

4.7 Conclusion

We have presented the first systematic analysis on the estimation of cybercrime bitcoin revenue. Our analysis has quantified the impact in the estimation of the methodology used and the limited seed coverage. We have built a tool able to replicate the different methodologies and have collected a dataset of 30,424 cybercrime payment addresses. We have used them to quantify the impact of different methodology steps and to compare the revenue obtained by 6 abuse types and 141 cybercriminal groups. We show that some methodologies may produce large overestimation by introducing addresses unrelated to the campaign, through undetected online wallets in services like exchanges, or by double-counting deposits. We have measured that for 40% of the groups MI clustering fails to discover additional addresses, and that ransomware dominates the Bitcoin-payment cybercrime scene with a revenue almost 10 times larger than other abuse types. For the first time, we have quantified the impact of the lack of coverage in the estimation. We propose two techniques to achieve possibly complete coverage of victim payments to the DeadBolt server ransomware. From our privileged vantage point, we estimate a revenue of \$2.47M, 39 times higher than estimated from the vantage point provided by two popular Internet scan engines.

Chapter 5

A Comparison Methodology for Indicator Extraction Tools

5.1 Introduction

Cyber Threat Intelligence (CTI) provides information on attacker behavior that allows gaining visibility into the fast-evolving threat landscape; to understand the techniques, tactics, and procedures (TTPs) attackers use; and to timely identify and contain attacks. CTI is a multi-billion dollar industry, expected to keep growing to more than 16 billion USD by 2026 [126]. An essential piece of CTI is extracting and sharing *indicators of compromise* (IOCs), forensic artifacts that when observed on a device or network indicate it may have been compromised, e.g., malicious IPs, domains, and file hashes. IOCs are an actionable piece of CTI, as they can be fed to security systems (e.g., NIDS, firewall, HIDS, blocklists) to detect and block attacks.

IOCs can be distributed through open and commercial feeds [114, 24], which provide structured IOC data following standardized formats (e.g., STIX [148], OpenIOC [72]). However, much CTI is distributed through unstructured threat reports, written in natural language and published through a wealth of security blogs and social media platforms. For example, Twitter has become widely used for exchanging and spreading cybersecurity information, not only by cybersecurity companies but also by experts that sometimes rush to share their discoveries [179, 4]. Natural language reports often contain IOCs and typically provide more detailed contextual descriptions about the IOCs compared to IOC feeds. In addition, while IOC feeds usually focus on a small set of indicator types (i.e., IP addresses, domain names, file hashes), unstructured threat reports frequently provide more varied indicator types such as vulnerability identifiers, email addresses, blockchain addresses, Tor onion addresses, fuzzy hashes (e.g., SSDeep [108]), social handles, and target countries.

There exist multiple indicator extraction tools that can identify IOCs from unstructured threat reports in natural language using regular expressions [172, 26, 173, 191, 96, 141]. Even recent works that apply natural language processing (NLP) techniques to analyze threat reports still use regular expressions for identifying indicators as part of their pipeline [117, 94, 228, 183]. Regular expressions for extracting indicators can easily become complex, making it difficult to

understand what they match. Also, small differences between regular expressions for the same indicator type may significantly affect the extraction results. Furthermore, regular expressions can be affected by catastrophic backtracking introducing ReDoS vulnerabilities [53]. Despite the popularity of regular expression based indicator extraction tools [172, 26, 173, 191, 96, 141] no previous work has systematically evaluated them to understand which tool is more accurate for each indicator type and how accurate their extraction is. The main challenge to evaluate indicator extraction accuracy is the difficulty of building large-scale ground truth datasets from real threat reports. To address this challenge, this work presents a novel majority vote methodology for comparing the accuracy of indicator extraction tools, which does not require a manually-built ground truth.

Another challenge is that in a large and fast-evolving landscape of threats, coverage is difficult to achieve by any single entity, as shown by different sources having little IOC overlap [114, 24]. Thus, it is not sufficient to rely on a single source, or a small set of sources, to build an accurate, comprehensive, and up-to-date IOC list. To overcome this limitation, organizations and security analysts can resort to extracting IOCs from the threat reports collected from multiple sources. However, there exists a myriad of sources through which threat reports are disseminated including hundreds of blogs, Twitter accounts, and Telegram channels. To address this challenge, we present a modular threat report collection pipeline that can collect reports from a wealth of *sources* in a unified manner.

Our collection pipeline currently supports 6 diverse sources: RSS, Twitter, Telegram, and three report datasets Malpedia [158], APTnotes [8], and ChainSmith [228]. For RSS, Twitter, and Telegram it takes as input a list of *origins* (RSS feeds, Twitter accounts, Telegram channels) to monitor. It periodically visits those origins to identify new *entries* (blog posts, tweets, messages). For entries that contain a URL to a report, it downloads the report’s *document*. The selected sources are complementary. RSS allows collecting reports from hundreds of blogs from cybersecurity companies and individual experts, while Twitter and Telegram cover social media distribution. In addition, crowd-sourced datasets such as Malpedia and APTnotes allow identifying previously unknown blogs and Twitter accounts that should be monitored, as well as collecting reports from blogs without an RSS feed.

We have implemented our novel methodology for comparing indicator extraction tools and our collection pipeline into an automated platform called GoodFATR. GoodFATR supports a variety of open-source indicator extraction tools but also provides its own IOCSEARCHER tool, which takes as input a document in HTML, PDF, or plain text format and applies regular expressions to extract 41 indicator types. When processing reports, indicator extraction tools may identify malicious indicators (e.g., C&C domains), as well as benign ones (e.g., URL references or contact emails for security companies). We call malicious indicators IOCs and benign indicators *generic*. GoodFATR provides a filtering module that removes generic indicators extracted from the documents to output only the IOCs. Its filtering module builds a dynamic blocklist from the monitored sources and origins. This approach avoids hard-coded blocklists used by prior tools, which cannot adapt to the different sources and origins each user may monitor.

We use GoodFATR to compare the accuracy of 7 popular open-source indicator extraction tools [172, 26, 173, 191, 85, 96, 141], as well as GoodFATR’s own IOCSEARCHER tool, on

4,420 reports. The results show IOCSEARCHER being the most accurate tool on 11 of the 13 indicator types supported by multiple tools. We also identify ReDos vulnerabilities in two of the tools, one new and one previously reported. We also evaluate the filtering module provided by GoodFATR against the filtering rules used in two prior tools. The filtering by GoodFATR achieves an F1 score of 0.91 compared to 0.47 (IOC_PARSER) and 0.46 (CACADOR).

We have used GoodFATR to collect 472,891 reports over 15 months from 3,226 origins distributed through 6 sources; extract 978,151 indicators from the reports; and identify 618,217 IOCs. We analyze the collected data to identify the top IOC contributors and the IOC class distribution. Finally, we detail how we have leveraged GoodFATR on a novel approach for tracking cybercrime relations on the Bitcoin blockchain [73].

This work provides the following contributions:

- We present a novel majority-vote methodology for evaluating the accuracy of indicator extraction tools, which does not require a manually-built ground truth. We apply our methodology to compare the accuracy of 7 popular tools and our own IOCSEARCHER over 4,420 reports. The results show IOCSEARCHER is the most accurate tool on 11 of the 13 indicator types supported by multiple tools.
- We present GoodFATR, an automated platform to collect threat reports from a variety of sources in a unified manner, extract the indicators in the collected reports, and filter generic indicators to output only the IOCs.
- We use GoodFATR to collect 472,891 reports over 15 months from the 6 sources; extract 978,151 indicators from the reports; and identify 618,217 IOCs.
- We apply GoodFATR in a case study to identify previously unknown Bitcoin addresses from threat reports and produce unclassified tags for them, an approach that allows incrementing tag databases.
- We release the source code of the IOCSEARCHER tool [121].

The remainder of this Chapter is organized as follows. Section 5.2 provides an overview of our platform. Section 5.3 details the sources and approach used in the threat report collection. Section 5.4 performs a systematization and comparative study of 8 indicator extraction tools. Section 5.5 analyzes the reports collected over 15 months, the IOCs extracted from those reports, and the top contributing sources and origins. Section 5.6 presents our novel accuracy comparison methodology and applies it to compare the 8 indicator extraction tools on 4,420 threat reports. Section 5.7 presents an application of our platform to build a database of tagged Bitcoin addresses. Section 5.8 discusses limitations and future improvements. Finally, Section 7.1 concludes.

5.2 Overview

An *indicator* represents an artifact such as an IP address, a domain name, a URL, a file, a registry key, or a blockchain address. An indicator is typically represented as a pair of a type (e.g., *email*) and a string value (e.g., `contact@test.com`). An *indicator of compromise* (IOC)

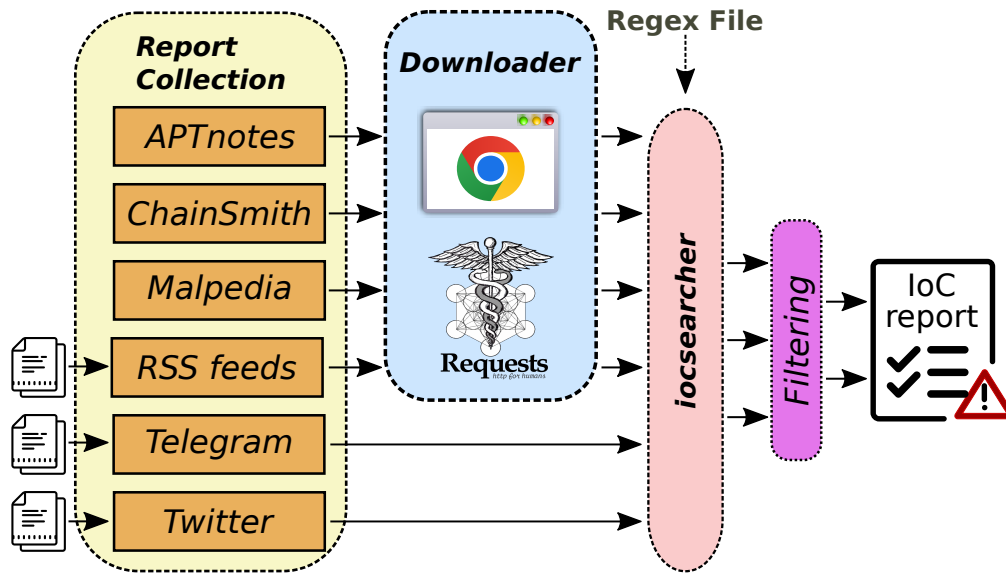


Figure 5.1: The architecture of GoodFATR.

is a malicious indicator whose presence on a device or network indicates the device might have been compromised.

This chapter presents GoodFATR, a platform for collecting threat reports from a wealth of sources and extracting IOCs from the collected reports. Figure 5.1 depicts the architecture of GoodFATR. It consists of four modules: *Report Collection*, *Downloader*, *IOCSEARCHER*, and *Filtering*. The Report Collection gathers documents from a wealth of sources including RSS feeds, Telegram channels, Twitter accounts, and threat report datasets like Malpedia [158], APTnotes [8], and ChainSmith [228]. The collection outputs a list of *entries* each corresponding to the observation of a report in a source. Each entry contains the URL of the report or the content of a tweet or Telegram message. In addition, it may contain optional metadata about the report (e.g., publication date, author, language) and the source where it was observed (e.g., RSS feed, Twitter account). The Downloader leverages an instrumented Chrome browser (and Python’s requests library as backup) to fetch the document pointed by an entry’s URL and store it to file together with download metadata (e.g., download date, HTTP status code, redirection chain followed). For Twitter and Telegram, the Downloader is not used, as their entries already contain the content of the posts. *IOCSEARCHER* takes as input a document in HTML, PDF, or plain text format, extracts its text (for HTML and PDF documents), and applies regular expressions to identify 41 indicators. We choose regular expressions because they are an efficient technique for identifying, in a given string, indicators with some intrinsic structure such as email addresses or URLs. To extract indicators, *IOCSEARCHER* applies a match-and-validate approach. Matching applies each of the regular expressions to identify candidate indicators in the input string. Validation uses a function specific to each indicator type to ensure the candidate is indeed an indicator. Separating matching from validation allows the validation to perform computations that are not possible within regular expressions such as checking a checksum embedded in an indicator value (e.g., blockchain addresses, bank account numbers). It also prevents regular expressions from becoming too complex. While validation minimizes the set of incorrect indicators output, it is possible that indicators

present in the text are not IOCs, but rather correspond to benign indicators (e.g., the email of the report’s author). Thus, before generating the final list of IOCs, the Filtering removes generic indicators that are not IOCs.

Throughout its pipeline, GoodFATR maintains traceability. From an IOC an analyst can identify the documents that contained it, from a document the entries from where it was collected, and from an entry the sources where it was observed.

5.3 Threat Report Collection

Our threat report collection module has been designed to support a variety of threat report sources. We collect information from RSS, Twitter, Telegram, and publicly available report datasets such as the Malpedia malware encyclopedia [158], the APTnotes repository [8], and the ChainSmith database [228]. To handle diverse sources in a unified manner, the collection is structured around three concepts: *origin*, *entry*, and *document*.

An origin captures the distribution vector through which a report is disseminated, at a finer granularity than a source. Each source typically has many origins; for RSS, an origin corresponds to the feed from a specific blog, for Twitter to a user account, and for Telegram to a channel. For report datasets (Malpedia, APTnotes, ChainSmith) we use as origin the organization that authored the report (e.g., Norton, TrendMicro). The same threat report might be distributed through multiple origins within the same source (e.g., through different blogs), as well as through different sources (e.g., a blog and a Twitter account). We use the origins to produce fine-grained statistics about the report distribution, e.g., to measure which Twitter accounts, Telegram channels, and RSS feeds provide more threat reports. We format the origin to also include the source in the form *source:origin*. For example, *rss:nakedsecurity* and *twitter:nakedsecurity* correspond to the RSS feed of the Naked Security blog by Sophos, and its Twitter account [@NakedSecurity](#), respectively.

An entry captures a specific mention of a threat report through an origin. It corresponds to a post in a Twitter account, Telegram channel, or RSS feed. For report datasets, an entry is a record in the report metadata, namely a line in the report index file of APTnotes, a row in the database of ChainSmith, and a BibTex entry in the Malpedia bibliography. Each entry has an origin and contains either a download URL pointing to the report or in the case of social media, a string with the post’s text. An entry may also contain additional report metadata as provided by the source: the report author, its original publication timestamp, and the last update timestamp.

A document is an HTML, PDF, or text file that has been collected by GoodFATR. A document is identified by the file’s SHA256 hash. A document is generated every time the Downloader downloads a URL and also from the text content of each Twitter and Telegram entry. A document corresponds to an instance of a threat report. The distinction between *threat report* and *document* is important to understand how GoodFATR handles report distribution and changes to the report over time. Multiple documents collected by GoodFATR may correspond to the same report. There are three main reasons for this. First, a threat report may be distributed in different formats. For example, an APT report originally distributed as a

Source	Origin	Input Origins	Cumulative	Download URL	Content	IOCs
aptnotes	organization	✗	✓	✓	✓	✗
chainsmith	organization	✗	✓	✓	✗	✓
malpedia	organization	✗	✓	✓	✗	✗
rss	feed	✓	✗	✓	✗	✗
telegram	channel	✓	✓	✗	✓	✗
twitter	account	✓	✓	✗	✓	✗

Table 5.1: Sources used for collecting threat reports.

webpage in a blog will be stored by APTnotes as a PDF document. When GoodFATR collects the report from the blog’s RSS feed it will obtain an HTML document. When GoodFATR collects the report from APTnotes, it will obtain a PDF document. Both documents are different instances of the same threat report. Second, for reports distributed through URLs, if GoodFATR downloads the URL at different points in time, it might collect a different HTML document (i.e., different SHA256) each time. This may happen if the webpage contains dynamic content that changes over time. Each of those HTML documents is an instance of the same threat report. Third, a threat report may have multiple versions. For example, the original blog entry for the report may be updated a day later to fix an errata. If GoodFATR downloads the URL of the report on both days, it will collect two different HTML documents, which correspond to different versions of the same threat report.

Identifying which collected documents correspond to the same threat report is not necessary for the operation of GoodFATR. All collected documents go through IOC extraction and the extracted IOCs are aggregated and deduplicated. If two documents are different versions of the same report, GoodFATR will extract the same (or very similar) IOCs from those documents and will remove the duplicated IOCs. GoodFATR provides information that can help identify documents that correspond to the same report, e.g., by querying for all documents downloaded from the same URL or by searching for all documents with the same title (extracted from the PDF and HTML document metadata). But, this problem is not addressed in this dissertation. We further discuss this issue in Section 5.8.

5.3.1 Threat Report Sources

Table 5.1 summarizes the six report sources currently supported by GoodFATR. Each source has a dedicated collection submodule. The RSS, Twitter, and Telegram submodules take as input a file that specifies the origins (i.e., feeds, accounts, channels) that should be monitored. The report datasets do not require an input origin list as the whole dataset is downloaded. We manually created our initial origin lists for RSS, Twitter, and Telegram by including resources from prominent companies, cybersecurity news websites, and well-known security experts. We continuously add new origins as we identify interesting RSS feeds, Twitter accounts, and Telegram channels. We discuss the update process in Section 5.8. Table 5.4 captures the total number of origins in each source at the time of writing: 300 RSS feeds, 383 Twitter accounts, 9 Telegram channels, 2,368 organizations in Malpedia, 155 organizations in APTnotes, and 11

blogs in ChainSmith.

For each source, Table 5.1 shows how the origin is defined and 5 properties: whether it requires an input list of origins; whether the reports are cumulative (i.e., old entries are always available) or entries expire after some time; whether it provides a URL for each entry from where the report needs to be downloaded; whether it includes the actual report or only the URL to the report; and whether it includes IOCs extracted from the report.

RSS. We use RSS feeds to collect reports from 300 security blogs. Each RSS feed is uniquely identified by a URL pointing to the feed’s XML file. The owner of an RSS feed configures it to provide a maximum number of the latest entries, *e.g.*, the 5, 10, or 100 most recent ones. To avoid missing entries, the RSS module needs to visit a feed frequently enough that the feed has not posted more entries than the maximum available since the RSS module last visited it. For example, the *Hexacorn* blog only provides the last 5 posts in its feed, but they only post twice per month. Thus, visiting its feed every two months would be enough to avoid missing entries. For each feed, the RSS module monitors the maximum number of entries it has obtained in a visit (*i.e.*, the configured maximum unless the feed has posted less than the maximum since its creation), as well as the maximum daily rate at which posts are added. While we could configure a different visit frequency for each feed, we have empirically observed that no feed posts more than their configured maximum in a single day. Thus, for simplicity, the RSS module queries each feed on a daily basis. The module is configured to throw a warning if any feed publishes more than its maximum number of available entries in a single day. So far, the warning has not been observed, so no entries have been missed.

Since the RSS module is configured not to miss feed entries, two consecutive visits to the same feed will provide some overlapping entries. The RSS module has an incremental approach where feed entries that were already collected (*i.e.*, older than the last feed visit) are ignored. Only the new entries are passed to the Downloader to collect their content. The incremental approach avoids downloading the same content multiple times.

Telegram & Twitter. The Telegram and Twitter modules leverage the official APIs to query each service. Using these APIs they can fetch not just the most recent messages, but any message that was ever posted on an account or channel. Each module takes as input a list of origins that should be monitored, and connects to each origin on a daily basis to fetch all new posts. Using the API, our modules collect each post from an account, or channel, only once. However, it is possible to obtain posts with the same content from multiple origins, *e.g.*, if different users re-tweet the same original message.

Malpedia. Malpedia offers a manually-curated BibTex file with reports related to malware. BibTex entries for new reports are added on a nearly daily basis [158]. The BibTex file is generated dynamically upon request and includes all reports in the database in a cumulative manner. Each bibliography entry contains the following report fields: author, title, publication date, organization (which we use as origin), URL, language, and the date when the report was added to Malpedia. The reports’ documents are not provided, so the URLs are passed to the Downloader to collect them.

Tool	Language	Stars	Inputs			Indicator Extraction				Filtering
			Text	HTML	PDF	Rearm	Validate	Dedup.	IOCs	
JAGER [172]	Python	69	●	●	●	○	○	●	11	○
IOC_PARSER [26]	Python	378	●	●	●	●	○	●	11	●
CACADOR [173]	Go	119	●	○	○	●	○	●	12	●
CYOBSTRACT [191]	Python	66	●	○	○	●	●	○	25	○
IOC-FINDER [85]	Python	96	●	○	○	●	○	●	25	○
IOCEXTRACT [96]	Python	356	●	○	○	●	○	○	8	○
IOC-EXTRACTOR [141]	JavaScript	35	●	○	○	●	○	●	18	○
IOCSEARCHER [121]	Python	-	●	●	●	●	●	●	41	●

Table 5.2: Comparison of indicator extraction tools.

APTnotes. APTnotes is a GitHub repository with manually-curated APT reports [8]. It provides an index file with report metadata containing the report’s title, source (i.e., author organization which we use as origin), publication date, the filename and file hash of the report document, and the URL where the document can be obtained. Reports in webpages are converted from HTML to PDF. All reports are stored in the [box.com](#) service as PDF files.

ChainSmith. The ChainSmith project [228] makes available an SQLite3 database, updated every week since 2018, containing the reports it has downloaded, their metadata, and the IOCs it has extracted from them. The database consists of one table where each row corresponds to an IOC extracted from a report, including the report URL, the source (i.e., the blog identifier we use as the origin), the report title, and the publication date. Reports’ documents are not provided. Thus, we extract the document URLs and give them to the Downloader.

5.3.2 Downloader

The Downloader takes as input an entry and tries to download the content pointed to by the entry’s URL. It uses Selenium, a popular framework used for testing Web applications [194]. We instrument Selenium to render URLs using a fully-fledged instance of Google Chrome. The Downloader is able to follow redirects, it supports dynamic content executed with JavaScript, and, in addition to HTML pages, it can also download plain text documents as well as other MIME types such as PDFs. In case our instrumented browser did not succeed in retrieving the content, the Downloader makes an additional attempt with Python’s *requests* library [163]. For each successfully downloaded URL, the Downloader stores the document to a file and it updates the entry with the download information including the download timestamp, the document hash, the HTTP status code, and the redirection chain followed. In the final step, the downloaded documents are filtered to remove resources with HTTP status code errors and HTML content where the title states the webpage was not found.

5.4 Indicator Extraction: A Comparative Study

In this section, we perform a survey and functionality comparison of indicator extraction tools including our own. We further quantitatively evaluate the tools in Section 5.6.

To identify the tools, we search for open-source projects for extracting indicators. For this, we query GitHub for projects related to the *ioc* keyword. Then, we manually examine each

matching project to select those that correspond to tools that extract indicators. We keep only popular tools with at least 30 stars, which helps avoid the forks of more popular projects. If an identified tool references any other indicator extraction tools, we also include those in our search. This process identifies 7 popular open-source tools for extracting indicators. We also include our own IOCSEARCHER tool, for a total of 8 tools in Table 5.2.

The 8 tools follow the same model comprising three steps, two of which are optional. The first optional step is *text extraction*, which given an HTML or PDF document, extracts its text. The second step, present in all tools, is *indicator extraction*, which extracts the indicators present in an input string by applying a number of regular expressions, grammars, and rules. Finally, some tools apply an optional *filtering* step whose goal is to remove indicators that are benign (e.g., domains of security vendors), and thus cannot be considered IOCs. The dominant language for implementing the tools is Python used by 6 tools including the two most popular tools (IOC_PARSER and IOCEXTRACT) and our own IOCSEARCHER tool. CACADOR is written in Go and IOC-EXTRACTOR in JavaScript.

In Table 5.2, a solid circle (●) indicates full support, and a half-filled circle (◐) partial or optional support, and an empty circle (○) no support. Next, we detail each of the three steps.

5.4.1 Text Extraction

Since most threat reports are distributed as HTML and PDF documents, text extraction is an important step in the complete indicator extraction process. However, it can be performed as a separate pre-processing step that takes as input an HTML or PDF document and outputs its content as text. Content extraction is an independent process from identifying indicators, and this is likely the reason why 5 of the tools do not support it and assume that the input is a string, e.g., containing the full text of a report. In summary, all 8 tools can take an input string, read text from standard input or from a file, and extract indicators in the string. In addition, three tools including our own, accept directly input HTML and PDF files, including the text extracted from the input document, before extracting indicators. This is convenient for the user since it does not need to write its own text extraction code and allows the tool developer to customize the text extraction.

Text extraction is a common step in many Natural Language Processing (NLP) pipelines such as those used to analyze privacy policies (e.g., [229, 192, 230, 80, 5]). Text extraction can have a significant impact on the extracted indicators. For example, the text extraction process may pre-pend or append text to the indicator value, making the extraction miss the indicator or output an incorrect indicator with extraneous characters. It is also possible for the extracted text to split indicators across multiple lines, causing them to be missed. For example, PDF text extraction libraries oftentimes split long URLs appearing in footnotes across multiple lines in the output text (e.g., URLs that do not fit the length of a column).

The only three tools that support all three input types are JAGER, IOC_PARSER, and IOCSEARCHER. These three tools use the pdfminer.six [189] library for PDF text extraction. JAGER and IOC_PARSER use the BeautifulSoup [170] library for HTML text extraction, while our IOCSEARCHER tool supports both BeautifulSoup and Readability.js [135]. Recently, Hosseini et al. [88] analyzed 7 HTML text extraction approaches used in privacy policy

analysis showing wide variability among the produced text. They concluded that the best-performing HTML text extraction libraries for privacy policies were Boilerpipe [106] and Readability.js [135], while the worst-performing one was BeautifulSoup [170]. BoilerPipe is written in Java and Readability.js in JavaScript, but both have Python wrappers available.

5.4.2 Indicator Extraction

Given an input string, indicator extraction identifies indicators present in the string. Seven of the tools use regular expressions for the extraction, while IOC-FINDER uses grammars instead. The tools using regular expressions process the text in a loop, each time applying one regular expression at a time to the whole text. Each regular expression is associated with one indicator type that is assigned to each of the regular expression matches. Most tools have one regular expression for each supported indicator type, although some tools (e.g., IOCEXTRACT, IOCSEARCHER) support multiple regular expressions for the same indicator.

The middle part of Table 5.2 captures four properties of the indicator extraction step: whether defanged indicators are supported and rearmed, whether tools split the extraction process into matching and validation, whether the indicators are deduplicated, and the supported indicator types. We describe each property next.

Defanged and rearmed indicators. It is common for threat reports to *defang* malicious indicators, (i.e., IOCs) in case a user inadvertently clicks on them in a navigation tool like a browser. For example, IP address 9.9.9.9 may appear in a threat report as 9[.]9[.]9[.]9, while URL <http://example.com/badfile> may appear as [http://example\(.\)com/badfile](http://example(.)com/badfile). Such indicators are often called *defanged* to indicate that they have been converted into less harmful ones, and are not dangerous anymore. These transformations are similar to the ones applied to email addresses to limit automated collection by spammers, e.g., “contact_at_somewhere[.]com”. Of the 8 tools, only JAGER does not identify defanged indicators. The other tools support a subset of the following defang transformations: (I) replacing the dot in IPv4 addresses, domain names, URLs, emails, and filenames, (II) replacing the @ sign in emails, (III) replacing the scheme in URLs (e.g., using [http://](http://example.com) instead of [http://](http://example.com)), and (IV) replacing the backslash in URLs. Two tools are marked as having partial support: IOC_PARSER only supports the dot in URLs and domain names, while CACADOR only supports the dot in IPv4 addresses. The other tools are marked as full support. However, they may not exactly support the same transformations, e.g., only IOC-EXTRACTOR supports the replacement of the backslash in URLs. The space of defang transformations that users can apply is very large and the extraction tools only support the limited set presented above, which is clearly incomplete. Intuitively, the defang transformations most important to support are those most often used by threat report authors (or by users when analyzing other sources like webpages). However, it is hard to know the most popular transformations a priori, so it is typical to add support for new defang transformations as they are observed in the wild.

The tools use two approaches to handle defanged indicators. The most popular approach, used by six tools, is broadening the regular expressions used to identify the indicators to cover common defang operations. For example, the regular expression should support that an IP address optionally contains brackets or parenthesis around the dots. Once the defanged

indicators have been matched by the regular expression, they can be *rearmed* (or *refanged*) to output the original indicator values. Four tools rearm the defanged indicators by default, while another two (IOC_PARSER, IOCEXTRACT) allow the user to choose if defanged or rearmed values should be returned. The alternative approach used by IOC-FINDER and IOC-EXTRACTOR is to first apply a rearm transformation to the raw text before applying the regular expressions. This approach does not require broadening the regular expressions to handle different defang transformations. On the other hand, blindly rearming the text before applying the regular expressions could incorrectly modify the text (e.g., rearming some text that is not part of an indicator). It also prevents returning the defanged value as it appears in the text since it has been rewritten.

Match and validate. One goal of indicator extraction is to minimize false positives (FPs), i.e., avoid outputting indicators that are not real indicators, such as two tokens concatenated with a period, that is not a fully qualified domain name. This requires making the regular expressions as narrow as possible, without introducing false negatives. For example, it is typical that a regular expression for domain names will check that the top-level domain (TLD) is one of the IANA-approved TLDs. For this, it is common for tools to include a long list of valid TLDs inside the regular expression. Unfortunately, the IANA list already contains over 1,500 TLDs and may continue to grow over time making the regular expression cumbersome. An alternative approach is to split the indicator extraction process into two steps: regular expression matching and validation. In this model, the regular expression can be a bit wider (i.e., produce more matches) because the validation, which is specific to each indicator type, will discard incorrect matches. This match and validate process was first used by CYOBSRACT, with basic checks such as validating that ASN numbers are in hard-coded ranges or that the length of a domain name is below 160 characters (which is not entirely correct as the maximum length is 255 octets).

Our IOCSEARCHER tool implements a more complete match-and-validate process where each indicator type has an optional validation function that checks the returned regular expression matches. For example, in IOCSEARCHER the *fqdn* regular expression does not need to include the list of valid TLDs because there is a *fqdn* validation function that ensures the matched value indeed contains a valid TLD. The advantage is that the list of IANA-approved TLDs can be kept in a separate file, which can be updated without modifying the regular expression. More importantly, the match and validate approach allows to perform Turing-complete processing on the matches such as validating an embedded checksum in a bank account number or a Bitcoin address. Such validation is not possible using only a regular expression, and it can significantly reduce FPs. For example, it is common for MD5 hashes to match a Bitcoin regular expression, but it is highly unlikely those spurious matches will pass the checksum validation. It is worth mentioning, that the implementation in our IOCSEARCHER tool performs validation after the indicator has been rearmed so that the validation does not need to handle defang transformations.

Deduplication. It is possible for the same indicator to appear multiple times in the same string at different locations. We say that a tool *deduplicates* if it removes duplicated indicators from its output. For example, if the same URL appears twice in the input string, the output

only contains the *url* indicator once. Tools can be classified into those that always deduplicate (●), those that do not deduplicate (○), and those where deduplication is optional (◐). None of the prior tools return the position (i.e., start offset) at which the indicator is matched. Thus, the main value of not deduplicating is to know how many times an indicator appeared in the input string. The most flexible approach is to make deduplication optional, as done by `IOC_PARSER` and `IOC-FINDER`.

Our `IOCSEARCHER` tool offers two different APIs. The raw API does not deduplicate. It returns all indicators identified, with their indicator type, starting offset, raw value, and rearmed value. In contrast, the deduplicated API first invokes the raw API, and then it deduplicates the received values by removing the starting offset and the raw values, i.e., returns only deduplicated rearmed indicators. Providing all matches with their starting offset allows the raw API to be used in additional scenarios. For example, Gao et al. [68] propose IOC protection to handle IOCs that contain dots (e.g., URLs, domains, IPs) in NLP pipelines. Such indicators negatively impact sentence tokenization in NLP libraries that leverage dots to identify the end of sentences. IOC protection first identifies the indicators (e.g., using regular expressions), replaces their value in the text with a keyword that does not contain dots, tokenizes the text into sentences, and finally replaces back the keyword with the original indicator value. IOC protection requires the starting offset and the raw value of the indicator in the input string, precisely what our raw API provides.

Indicators. One key difference between indicator extraction tools is the set of indicator types they support, i.e., the set of indicators for which they have regular expressions, grammars, or extraction rules. Overall, we have identified 63 indicators that the tools extract, as summarized in Table 5.3. We group indicator types into 16 classes: network (13 indicators), social (12), blockchain (9), cryptographic hashes (6), analytics (3), attack (3), contact (2), file (2), intellectual property (2), organization (2), payment (2), vulnerability (2), fuzzy hash (1), information sharing (1), location (1), registry (1), and yara (1).

The largest class corresponds to network-related artifacts such as domain names (*fqdn*), URLs (*url*), IP addresses (*ip4*, *ip6*), IP subnets in CIDR form (*ip4cidr*, *ip6cidr*), IP address ranges (*ipv4range*, *ipv6range*), AS numbers (*asn*), MAC addresses (*macaddress*), Tor onion addresses used to access hidden services (*onionAddress*), Internet Content Provider numbers that uniquely identify the owner of a Chinese website (*icp*), and HTTP User-Agent strings (*userAgent*). The social category comprises of user handles for social networks (Facebook, Instagram, LinkedIn, Pinterest, Twitter), open source repositories (GitHub), Instant Messaging tools (Jabber, Telegram, WhatsApp), as well as YouTube usernames and channel identifiers. The blockchain category comprises addresses for 9 popular blockchains (Bitcoin, BitcoinCash, DashCoin, DogeCoin, Ethereum, Litecoin, Monero, Tezos, ZCash). Cryptographic hashes include MD5, SHA1, SHA256, and SHA512, as well as their application on specific parts of a Windows executable such as the import table (*importHash*) and the whole executable without Authenticode code-signing fields (*authentihash*). There is also a fuzzy hash (*ssdeep*) used to identify files with similar content [108]. The analytics category includes three Google identifiers: Google Adsense, Google Analytics, and Google Tag Manager. The attack category contains MITRE ATT&CK techniques, tactics, and procedures, a list of attack-related keywords (e.g., DoS, spam), and antivirus labels (*avLabel*). The payment category includes

Indicator	jager	ioc_parser	cacador	cyobstrack	ioc-finder	iocextract	ioc-extractor	iocsearcher	NumTools
asn				✓	✓		✓		3
asnOwner				✓					1
attackType				✓					1
attCk					✓				1
authentihash					✓				1
avLabel				✓					1
bitcoin					✓		✓	✓	3
bitcoincash								✓	1
copyright								✓	1
country				✓					1
cve	✓	✓	✓	✓	✓		✓	✓	7
dashcoin								✓	1
dogecoin								✓	1
email	✓	✓	✓	✓	✓	✓	✓	✓	8
ethereum							✓	✓	2
facebookHandle								✓	1
filename	✓	✓	✓	✓					4
filepath		✓		✓	✓				3
fqdn	✓	✓	✓	✓	✓		✓	✓	7
githubHandle								✓	1
googleAdsense					✓		✓	✓	3
googleAnalytics					✓		✓	✓	3
googleTagManager								✓	1
iban								✓	1
icp								✓	1
importHash					✓				1
incident				✓					1
instagramHandle								✓	1
ip4	✓	✓	✓	✓	✓	✓	✓	✓	8
ip4cidr				✓	✓			✓	3
ip4range				✓					1
ip6			✓	✓	✓	✓	✓		5
ip6cidr				✓					1
ip6range				✓					1
isp				✓					1
linkedinHandle								✓	1
litecoin								✓	1
macAddress					✓		✓		2
md5	✓	✓	✓	✓	✓	✓	✓	✓	8
monero					✓		✓	✓	3
onionAddress								✓	1
phoneNumber					✓			✓	2
pinterestHandle								✓	1
regKey		✓		✓	✓			✓	3
sha1	✓	✓	✓	✓	✓	✓	✓	✓	8
sha256	✓	✓	✓	✓	✓	✓	✓	✓	8
sha512	✓		✓			✓	✓		4
ssdeep	✓		✓	✓	✓		✓		5
telegramHandle								✓	1
tezos								✓	1
tlpLabel					✓				1
trademark								✓	1
twitterHandle								✓	1
url	✓	✓	✓	✓	✓	✓	✓	✓	8
userAgent				✓	✓				2
webmoney								✓	1
whatsappHandle								✓	1
xmppHandle					✓				1
yara						✓			1
youTubeChannel								✓	1
youtubeHandle								✓	1
zcash								✓	1

Table 5.3: Indicators extracted by different tools.

IBAN bank account numbers and WebMoney addresses [217]. The vulnerability category includes CVE identifiers and also a variety of identifiers for other vulnerability sources (e.g., BugTrack, Microsoft Bulletins) grouped into a generic *incident* indicator by CYOBSTRACT. The information-sharing category comprises a single indicator for the traffic light protocol (*tlpLabel*) that controls the dissemination of information [63]. Indicators in the remaining categories are straightforward and capture: contact (*email*, *phoneNumber*), files (*filename*, *filepath*), intellectual property (*copyright*, *trademark*), organization names (*asnOwner*, *isp*), locations (*country*), Windows registry keys (*registryKey*), and Yara rules (*yara*).

There are only 6 indicators that are extracted by all 8 tools: IPv4 addresses (*ip4*), emails, hashes (*md5*, *sha1*, *sha256*), and URLs. Another two indicators are extracted by 7 tools: CVE vulnerability identifiers (*cve*) and domain names (*fqdn*). Next come two indicators extracted by 5 tools: IPv6 addresses (*ip6*) and SSDeep fuzzy hashes. However, the majority corresponds to 38 (61%) indicators extracted by a single tool.

Among the indicators in Table 5.3 there are some that deserve discussion. There are three indicators (*country*, *tlpLabel*, *attackType*) that are identified using regular expressions that are a disjunction of keywords. For example, TLP labels can have only four values: *TLP:white*, *TLP:green*, *TLP:amber*, *TLP:red*. Since the set of possible TLP values is finite, the regular expression can identify all possible indicator values. Similarly, there is only a finite number of recognized countries. However, *attackType* includes a number of attack-related keywords that CYOBSTRACT identifies. Such a list only includes attack topics of interest for the tool authors and may not include attack topics other users are interested in. The approach of building a regular expression from a set of keywords can be applied to any taxonomy of terms (e.g., family names, exploit kit names). CYOBSTRACT provides support for building such regular expressions. The main limitation of this technique is that it cannot identify new terms (e.g., new family names or exploit kits). CYOBSTRACT extracts two indicators that aim to capture organization names (*asnOwner*, *isp*). However, organization names do not have a clear structure save when using company-related suffixes (e.g., Ltd., Inc.) and thus are typically extracted using Named Entity Recognition (NER) techniques based on machine learning classifiers. Furthermore, trying to separate whether an organization name corresponds to an ISP or ASN owner is a challenging problem that is better suited for NLP techniques. There are also some indicators that one could argue should be split into multiple indicators. For example, the regular expression for the *incident* indicator extracted by CYOBSTRACT captures a disjunction of regular expressions, each for a different vulnerability report identifier (e.g., BugTrack, Microsoft Bulletins). Such disjunction is more efficient than having a separate regular expression for each identifier but does not allow users to extract only those identifiers they are interested in. Another interesting case is cryptographic hashes like *authentihash* and *importHash* which are really *subtypes* of other cryptographic hashes. For example, *importHash* is the MD5 of the import table of a PE executable [122], while *authentihash* is the SHA256 of a PE executable excluding Authenticode code signing fields. Thus, it is not possible for a regular expression to differentiate an *importHash* from a *md5* or an *authentihash* from a *sha256* except if the subtype indicators appear with some specific keywords before or after. Such regular expressions are very specific, they rarely produce FPs but they will introduce FNs when the expected keywords do not appear before or following the hashes. On the other hand, regular expressions for the parent types will also identify indicators subtypes (e.g., the

md5 regular expression will also identify *importHash* indicators), but classifying an MD5 into what the MD5 captures (e.g., a file hash, a certificate hash, an import table hash) is a challenging problem that is may be easier to tackle through NLP. Two more indicators correspond to ranges of IP addresses (*ip4range*, *ip6range*). Similar range indicators could be defined for any integer indicators, e.g., ASN numbers. The last interesting case is Yara rules extracted by IOCEXTRACT. Yara rules can be long and have an internal structure with mandatory and optional fields. Thus, some readers may not consider them indicators.

5.4.3 Filtering.

While most tools in Table 5.2 label themselves as IOC extraction tools, in reality, what they extract are indicators. Not all indicators are IOCs; an indicator is an IOC only if it represents a malicious artifact. For example, threat reports often contain hyperlinks to prior reports by other security vendors. Those references may be extracted as *url* indicators, but they can hardly be considered IOCs. We use the term *generic indicators* to (emphatically) refer to indicators that are not IOCs, *i.e.*, not malicious.

Determining whether an indicator is benign or malicious, *i.e.*, whether it is an IOC, is challenging. Most tools in Table 5.2 do not try to perform such determination and simply output all indicators they find. But, three tools (IOC_PARSER, CACADOR, and our IOCSEARCHER) implement a per-indicator filtering step, whose goal is to remove generic indicators leaving only the IOCs. IOC_PARSER and CACADOR both make use of hard-coded blocklists (or regular expressions) for benign indicators. Overall, the blocklists capture domains (or emails and URLs that contain those domains) from cybersecurity companies, news outlets, law enforcement agencies, and a few other trusted sites such as github.com. CACADOR uses a single blocklist with 15 domains, 12 of them from cybersecurity companies. IOC_PARSER uses 11 blocklists, one per each indicator type it supports. However, only 4 of those blocklists (*email*, *fqdn*, *ip4*, *url*) have entries. The *email* blocklist contains 6 domains of cybersecurity companies. The *fqdn* blocklist contains 143 domains of cybersecurity companies, news outlets, and law enforcement agencies. The *url* blocklist contains 79 root URLs for popular cybersecurity blogs. Finally, the *ip4* blocklist contains 6 reserved IP address ranges.

The main difference in our filtering module is that its blocklist is dynamically generated, *i.e.*, created from the collected documents. Indicators are added to the blocklist if they satisfy at least one of the following 5 rules: (i) the indicator is a *fqdn*, *url*, or *email*, whose domain is in the list of domains from where a document was collected; (ii) the indicator was extracted from at least 20 documents from the same origin; (iii) the indicator is a *fqdn* or *url* whose domain (excluding the “www.” prefix if present) is in the top-100k of the Tranco domain popularity list [159]; (iv) the indicator appears in more than 90% of all the collected documents; or (v) the indicator is a private IP address. The first two rules aim to filter indicators that belong to the origins from where documents are collected, *e.g.*, email contact addresses for security companies such as *contact@trendmicro.com*. Instead of hard-coding domains of cybersecurity companies and blogs, we infer them from the origins of the collected documents. This allows each user to personalize the filtering to the list of origins he chooses to monitor. The next two rules target popular indicators that appear in most documents or in a list of popular domains. The frequently updated Tranco list is supplemented with a

Source	Orig.	Start Date	Entries	Indicators	
				Extracted	IOCs
aptnotes	155	2022/07/18	629	44,540	41,031
chainsmith	11	2022/07/18	3,792	48,046	41,246
malpedia	2,368	2022/07/18	11,363	271,495	237,011
rss	300	2021/04/01	97,882	314,351	210,149
telegram	9	2021/09/09	61,892	44,534	31,445
twitter	383	2021/10/28	397,333	338,967	126,081
<i>ALL</i>	<i>3,226</i>	<i>2021/04/01</i>	<i>472,891</i>	<i>978,151</i>	<i>618,217</i>

Table 5.4: Data collection summary.

dynamically generated list of popular indicators appearing in the collected documents, which again can be personalized for each user.

Section 5.5 provides an evaluation of our filtering module and the filtering from CACADOR and IOC_PARSER on a manually generated ground truth.

5.5 Evaluation

We have applied GoodFATR to collect reports from six sources over the last 15 months. We started our collection with RSS on April 1st, 2021. We then added Telegram on September 9th 2021, and Twitter on October 10th, 2021. On July 18th, 2022, we added the three report datasets (APTnotes, ChainSmith, Malpedia), but these datasets are cumulative so GoodFATR can also process their past entries.

Table 5.4 summarizes the data collection and IOC extraction results for each source. It captures the number of origins in the source at the end of the collection, the collection start date, the number of collected entries, as well as the number of extracted indicators (before filtering) and IOCs (after filtering). The most diverse source by the number of origins is Malpedia with reports from 2,368 organizations, followed by Twitter with 383 accounts, RSS with 300 feeds, and APTnotes with reports from 155 organizations. In contrast, ChainSmith includes reports from only 11 blogs, and GoodFATR tracks only 9 Telegram channels.

Twitter is the source with the largest number of collected entries despite being introduced later than RSS and Telegram, but its entries are tweets and thus short compared to full reports collected from RSS and the three datasets. RSS is the most stable contributor with an average of 6.1K new report URLs per month. Among the three report datasets, Malpedia provides the most reports (11,363), with ChainSmith providing one-third (3,792), and APTnotes only 629 reports. The relatively small size of APTnotes is due to its more focused goal of only collecting reports for APTs.

IOC extraction. The last two columns in Table 5.4 show the indicators extracted by IOCSEARCHER (before filtering) and the final IOCs (after filtering). Overall, GoodFATR extracted 978k indicators of which 618K (63%) are IOCs. Thus, filtering is extremely important for discarding over one-third of indicators that are generic and thus are not IOCs. Filtering affects each source differently. In the extreme case of Twitter, two-thirds of the indicators are filtered. Across all sources, domain names and IPv4 addresses account for 99% of the generic indicators. Filtered domain names are those matching the domain from where the report was

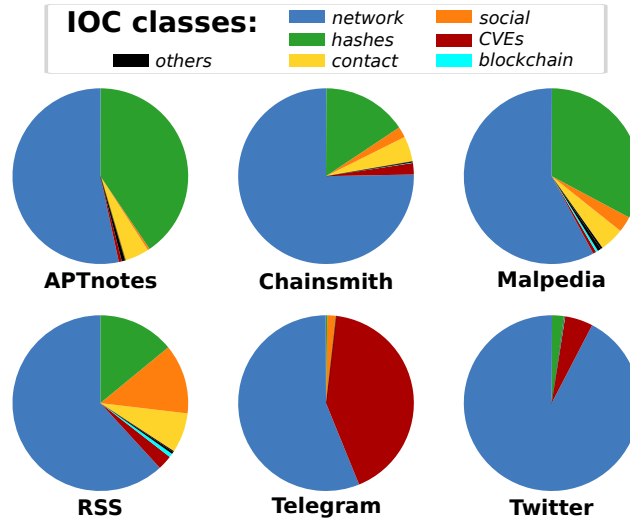


Figure 5.2: Distribution of IOC classes grouped per source.

collected and those included in the Tranco top-100K list. Filtered IPv4 addresses are those of private and local networks,

The average number of IOCs per report is highest for the report datasets: APTnotes (66), Malpedia (22), and ChainSmith (11). Of those, APTnotes and Malpedia are manually curated, which prevents the collection of non-technical reports. The collection in ChainSmith is automated, but it leverages a small number of blogs from large security companies known for their high-quality technical content. In contrast, reports from RSS have a lower ratio of IOCs (2.2), likely due to the variety of feeds and reports they disseminate. For example, some feeds may focus on technology news for the wider public (and thus provide fewer IOCs), while blogs from security companies may mix technical reports with less technical reports that focus on the virtues of their products. Still, the large number of RSS feeds GoodFATR monitors compensates for the lower ratio, making RSS a consistent IOC contributor. Finally, Telegram (0.5) and Twitter (0.3) have the lowest ratios, which is expected due to the limited content in each entry.

IOC class distribution. To understand what type of indicators each source distributes, we group indicators using the categories introduced in Section 5.4.2. Figure 5.2 reports the filtered IOCs split by class and source. The plots include only the six most popular classes, while we aggregate the rest in the *other* category. Across all sources, *network* indicators dominate. Malpedia, Telegram, and Twitter are the biggest contributors to the *network* category, with more than 110K IOCs of this category per source. The *hash* category ranks second, with most indicators coming from Malpedia (82K file hash IOCs). In third place, we find *contact* and *social* indicators, both having similar distributions, with the vast majority coming from RSS and Malpedia. These two sources are also the highest contributors of IOCs that survive the filtering and exhibit the highest variety in terms of indicator types. For example, RSS and Malpedia provide 94% of the *blockchain* IOCs and over 61K email addresses, phone numbers, and social network handles. Telegram and Twitter show a significantly smaller variety of IOC types. For example, the union of the *contact* and *social* classes represents only 1.6% of the

Rank	APTnotes	ChainSmith	Malpedia	RSS	Telegram	Twitter
#1	Kaspersky (13.11%)	Webroot (27.92%)	Palo Alto Networks (4.16%)	Dancho Danchev's Blog (26.21%)	cybsecurity (76.70%)	ecarlesi (76.42%)
#2	Palo Alto Networks (7.18%)	Sucuri (22.07%)	Trend Micro (4.11%)	Blockchain on Medium (9.06%)	VulnerabilityNews (8.28%)	threatmeter (4.73%)
#3	Norman (6.11%)	Malwarebytes (15.01%)	Kaspersky (2.92%)	Cybersecurity on Medium (8.63%)	Cyber_Security_Channel (7.33%)	malwrhunterteam (2.54%)
#4	Symantec (5.24%)	Virus Bulletin (14.12%)	ESET (2.82%)	Cisco Talos (6.57%)	cKure (4.01%)	bgpstream (2.02%)
#5	ClearSky (4.74%)	Sophos (4.88%)	Proofpoint (1.68%)	BleepingComputer News (2.44%)	malwr (2.18%)	YourAnonRiots (1.54%)
#6	FireEye (4.24%)	Forcepoint (4.60%)	Cisco Talos (1.66%)	Cointelegraph.com News (2.33%)	androidMalware (0.84%)	cryptolaemus1 (1.44%)
#7	ESET (3.72%)	ESET (4.34%)	FireEye (1.65%)	F5 Labs (1.28%)	ckuRED (0.48%)	ActorExpose (1.32%)
#8	Trend Micro (2.67%)	TaoSecurity (4.15%)	BitDefender (1.44%)	Schneier on Security (1.26%)	canyoupwne (0.16%)	MalwarePatrol (0.98%)
#9	Citizen Lab (2.40%)	Hexacorn (1.70%)	360netlab (1.39%)	Malwarebytes Labs (1.12%)	itsecalert (0.03%)	lzzr4h (0.97%)
#10	PwC UK Blogs (1.88%)	Roger McClinton (0.90%)	Microsoft (1.34%)	contagiodump (1.08%)	-	dubstard (0.74%)

Table 5.5: Top-10 origins that contribute with the highest number of IOCs, grouped per source. In brackets we report the percentage of IOCs associated to each origin.

IOCs in Telegram, and they only account for 92 of the IOCs extracted from Twitter. One possible reason is that Twitter accounts are more specialized, with some accounts focusing only on certain indicator types, such as malicious file hashes or vulnerability identifiers.

Top origins. Table 5.5 shows the top-10 origins for each source, ranked by the percentage of IOCs the origin contributes to the source. Overall, we notice a long tail distribution, with the top-10 origins generating 50% of all IOCs. This pattern is consistent across all sources and IOC classes. All three report datasets are dominated by large security companies, with the exception of the Citizen Lab at the University of Toronto (rank 8 in APTnotes) and three personal blogs (ranks 8–10 in ChainSmith). A surprising contributor is Price Waterhouse Coopers (rank 10 in APTnotes), one of the largest financial accounting companies, but less known for their security services. In general, personal blogs exhibit less activity compared to company blogs with 7% of all IOCs coming from personal blogs. RSS has more variety in the origins. Surprisingly, the top contributor is the personal blog from Dancho Danchev, followed by two Medium blogs that aggregate blockchain and cybersecurity news. The RSS top-10 is rounded by the research labs of three large companies (Cisco, F5, Malwarebytes), two other personal blogs (Bruce Schneier, contagiodump), and two magazines (Cointelegraph.com and BleepingComputer). Interestingly, two of the RSS top 10 origins focus on blockchain. We added blockchain-specific sources motivated by recent work that requires tagged Bitcoin addresses [73]. Telegram is largely dominated by the [cybsecurity](#) channel, which provides 76% of all Telegram IOCs. Similarly, Twitter also has one dominant account [@ecarlesi](#) with more than 140K tweets, 76% of all Twitter IOCs. Second, but far behind with 4.7% of IOCs, comes [@threatmeter](#), an automated bot that publicizes vulnerabilities and accounts for 90% of *vulnerability* IOCs.

5.6 Evaluation of IOC Extraction Tools

To evaluate the indicator extraction tools, we design a majority-vote methodology that runs the 8 tools in Table 5.2 on the text of the same set of threat reports. Then, we compare the indicators extracted by the different tools on the same report, assuming that the correct indicators are those extracted by a *majority* of tools. The advantage of such evaluation is that it can be run on a large set of reports bypassing the challenge of building a ground truth that is representative of the wealth of indicators the tools extract. The disadvantage is that this approach works only when indicators are extracted by multiple tools, and in some occasions it is possible that the *minority* of tools is actually correct. We detail our methodology and

the obtained results next.

We first build a document dataset by extracting the text using GoodFATR from all reports in APTnotes and ChainSmith. We choose these two sources because they cover different report file types (APTnotes has PDF reports and ChainSmith mostly HTML reports) and because both sources have a high average of IOCs per report, as shown in Section 5.5. In four PDF reports the text extraction failed: two were Excel spreadsheets that our text extraction does not support and the other two were corrupted. We evaluate the tools using the 4,420 successfully extracted text documents.

Algorithm 1 Accuracy comparison methodology.

```

1: procedure COMPARE(tools, docs)
2:   accuracy  $\leftarrow$  {}
3:   for tool in tools do
4:     accuracy[tool]  $\leftarrow$  (0, 0, 0, 0)
5:   end for
6:   for doc in docs do
7:     extracted  $\leftarrow$  {}
8:     all  $\leftarrow$  set()
9:     for tool in tools do
10:      iocs  $\leftarrow$  Run(tool, doc)
11:      iocs  $\leftarrow$  Normalize(iocs)
12:      all  $\leftarrow$  all  $\cup$  iocs
13:      for ioc in iocs do
14:        extracted[ioc].add(tool)
15:      end for
16:    end for
17:    for ioc in all do
18:      found  $\leftarrow$  extracted[ioc]
19:      supported  $\leftarrow$  canExtract(ioc.type)
20:      missed  $\leftarrow$  supported - found
21:      if len(found) > len(missed) then
22:        for tool in found do
23:          accuracy[tool].tp += 1
24:        end for
25:        for tool in missed do
26:          accuracy[tool].fn += 1
27:        end for
28:      end if
29:      if len(missed) > len(found) then
30:        for tool in found do
31:          accuracy[tool].fp += 1
32:        end for
33:        for tool in missed do
34:          accuracy[tool].tn += 1
35:        end for
36:      end if
37:    end for
38:  end for
39:  return accuracy
40: end procedure

```

Methodology. Algorithm 1 details our accuracy comparison methodology. We run all tools on each document, saving the indicators each tool extracts to a separate file. To compare all tools in a similar setting, we disable filtering for tools that support it (IOC_PARSER, CACADOR, IOCSEARCHER) and deduplicate indicators. Since each tool may assign slightly different names to indicator types (e.g. `ipv4addr`, `ip`, and `ipv4` for IPv4 addresses), we normalize them to match the names in Table 5.3. Additionally, some tools transform case-insensitive indicator values (e.g., domain names, email addresses) to lowercase or uppercase, while others output them as they appear in the text. To address such differences, we normalize the values of the following indicators to lowercase: hashes (*md5*, *sha1*, *sha256*, *sha512*, *ssdeep*), *regkey*, *ip6*, *fqdn*, and *email*. Finally, we normalize AS numbers to the format *AS1234* (e.g., from *asn1234* to *AS1234*) and URLs by prepending “`http://`” when no scheme is present.

For each tool, the evaluation keeps counters for true positives (TPs), false positives (FPs), false negatives (FNs), and true negatives (TNs). The evaluation processes one document at a time, updating the counters with the document results. For each document, it examines all indicators extracted from the document by at least one tool. For each indicator, it generates three sets: the *found* set captures the tools that identified the indicator in the document, the *missed* set captures the tools that support the indicator type but did not identify it, and the *unsupported* set captures the tools that do not support the indicator type. If the size of the found set is larger than the size of the missed set, then the evaluation assumes the majority is correct and therefore the indicator was indeed present in the document. Thus, it adds a TP for each tool in the found set and a FN for each tool in the missed set. If the size of the missed set is larger than the size of the found set, then the evaluation assumes the majority is correct and therefore the indicator was not present in the document. Thus, it adds a FP for each tool in the found set and a TN for each tool in the missed set. If the size of the found and missed sets is the same, then there is no majority. In such cases, we skip the indicator and do not update the counters. In future work, we plan to investigate how to break such ties. After processing all documents, we compute the precision, recall, and F1 score for each tool across all indicators, as well as separately for each indicator type.

Handling errors. When running all tools on the 4,420 documents we observed that IOCEXTRACT, the second most popular tool, needed hours to process some documents, compared to seconds or a few minutes for other tools. We suspected that one of their regular expressions had a ReDoS vulnerability, which caused a worst case that got triggered only in some documents [53]. We found an open issue in the IOCEXTRACT repository regarding catastrophic backtracking in the regular expression used to identify defanged URLs that modify the backslash [52]. To be able to complete the evaluation in a reasonable time, we configured IOCEXTRACT to avoid using the problematic regular expression. This change does not affect the accuracy results, as IOCEXTRACT is the only tool supporting that defang transformation. Similarly, there are two documents where IOC-EXTRACTOR does not terminate. We identify the root of this issue in the regular expressions used to extract domain names. Of the four domain regular expressions used, only one avoids the problem, while the others require hours to process the document. For these two documents, we added IOC-EXTRACTOR to the missed set for all indicators in the document. These results show that ReDoS vulnerabilities are a serious problem for indicator extraction tools, but oftentimes these worst cases appear only

Indicator	Count	jager			ioc_parser			cacador			cyobstruct			ioc-finder			iocextract			ioc-extractor			iocsearcher		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
asn	202	-	-	-	-	-	-	-	-	-	0.93	0.95	0.94	0.99	1.00	0.99	-	-	-	0.69	0.97	0.81	-	-	-
bitcoin	2,698	-	-	-	-	-	-	-	-	-	-	-	-	0.77	1.00	0.87	-	-	-	1.00	1.00	1.00	1.00	0.01	0.03
cve	2,135	0.96	0.99	0.98	1.00	1.00	1.00	-	-	-	0.98	0.97	0.97	0.98	1.00	0.99	-	-	-	0.96	1.00	0.98	1.00	1.00	1.00
email	1,947	0.60	0.85	0.70	0.89	0.75	0.81	0.60	0.73	0.66	0.99	0.98	0.98	0.93	1.00	0.96	0.75	0.97	0.85	0.97	0.99	0.98	0.99	1.00	1.00
filename	17,082	0.98	0.97	0.98	0.97	0.71	0.82	0.87	0.97	0.92	0.78	0.87	0.82	-	-	-	-	-	-	-	-	-	-	-	-
filepath	1,551	-	-	-	0.73	0.66	0.69	-	-	-	0.29	0.97	0.45	0.25	0.76	0.37	-	-	-	-	-	-	-	-	-
fqdn	41,360	0.48	0.06	0.10	0.99	0.91	0.94	0.56	0.39	0.46	0.97	0.96	0.97	0.95	0.99	0.97	-	-	-	0.92	0.99	0.95	0.98	1.00	0.99
googleAdSense	4	-	-	-	-	-	-	-	-	-	-	-	-	1.00	1.00	1.00	-	-	-	1.00	1.00	1.00	1.00	0.75	0.86
googleAnalytics	3	-	-	-	-	-	-	-	-	-	-	-	-	1.00	1.00	1.00	-	-	-	0.75	1.00	0.86	1.00	1.00	1.00
ip4	9,479	0.99	0.87	0.92	0.98	0.92	0.95	0.97	1.00	0.98	1.00	0.98	0.99	0.99	0.99	0.98	0.97	0.98	0.98	1.00	0.99	1.00	1.00	1.00	1.00
ip4cidr	287	-	-	-	-	-	-	-	-	-	1.00	0.99	0.99	0.97	1.00	0.99	-	-	-	-	-	-	1.00	0.99	0.99
ip6	967	-	-	-	-	-	-	0.50	0.91	0.65	0.87	0.66	0.75	0.91	0.12	0.21	0.15	0.80	0.25	0.51	0.97	0.66	-	-	-
macAddress	64	-	-	-	-	-	-	-	-	-	-	-	-	1.00	1.00	1.00	-	-	-	1.00	1.00	1.00	-	-	-
md5	14,635	1.00	0.98	0.99	1.00	1.00	1.00	0.45	1.00	0.62	1.00	0.97	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
monero	2	-	-	-	-	-	-	-	-	-	-	-	-	1.00	1.00	1.00	-	-	-	1.00	1.00	1.00	1.00	1.00	1.00
regkey	608	-	-	-	0.96	0.72	0.82	-	-	-	0.80	0.90	0.85	0.69	0.73	0.71	-	-	-	-	-	-	-	-	-
sha1	4,150	1.00	1.00	1.00	1.00	0.99	0.99	0.41	1.00	0.58	1.00	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
sha256	5,336	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	0.99	1.00	0.94	0.97	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
sha512	1	1.00	1.00	1.00	-	-	-	0.07	1.00	0.12	-	-	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-	-	-
ssdeep	74	1.00	0.28	0.44	-	-	-	0.16	0.30	0.21	0.81	0.88	0.84	0.55	0.91	0.69	-	-	-	0.47	1.00	0.64	-	-	-
url	14,818	0.61	0.56	0.59	0.53	0.56	0.54	0.51	0.52	0.52	0.76	0.96	0.85	0.62	0.83	0.71	0.60	0.99	0.75	0.70	0.80	0.74	0.77	1.00	0.87
All	117,403	0.87	0.57	0.69	0.91	0.85	0.88	0.61	0.70	0.65	0.92	0.96	0.94	0.87	0.96	0.91	0.77	0.99	0.87	0.90	0.96	0.93	0.95	0.97	0.96

Table 5.6: Precision (P), recall (R), and F1 score achieved by each indicator extraction tool on the comparative evaluation over 4,420 documents. Only indicator types extracted by at least two tools are included.

when examining a large number of documents. We examine whether ReDoS vulnerabilities exist in the regular expressions used by IOCSEARCHER with the *rat* ReDoS checker [153]. The tool implements a sound static analysis (*i.e.*, no false positives) that identifies regular expressions with exponential time worst case [154]. Except for 3 regular expressions the *rat* tool could not handle due to negative lookbehinds, all other IOCSEARCHER regular expressions are free from exponential time worst case.

There are also a few documents where a tool throws an exception and thus extracts no indicators. This happens for CYOBSTRACT in 41 documents and for JAGER in 20. For CYOBSTRACT, only the indicators of the type causing the exception are missed. We included JAGER in the missed set for all indicators in those documents.

Results. Table 5.6 summarizes the results for the 21 indicator types extracted by two or more tools. The columns show the indicator type, the number of indicators of that type considered TPs, as well as the precision (P), recall (R), and F1 score for each tool. We do not include indicators extracted only by a single tool, as there is no concept of majority for those.

Overall, IOCSEARCHER is the tool that achieves both the best precision (0.95) and F1 score (0.96), while IOCEXTRACT achieves the best recall (0.99). The highest overall F1 scores are for IOCSEARCHER (0.96), CYOBSTRACT (0.94), IOC-EXTRACTOR (0.93), and IOC-FINDER (0.91). The lowest F1 scores are for CACADOR (0.65) and JAGER (0.69). These results seem to indicate that recent tools (IOCSEARCHER, IOC-EXTRACTOR, IOC-FINDER) perform better than those released earlier (JAGER, CACADOR), possibly because newer tools could use older ones for comparison.

The table also presents the accuracy results for each indicator type. We observe perfect agreement for MAC address extraction (although only 64 MAC addresses are found by two tools) and nearly perfect agreement on the hashes, with the exception of CACADOR which has low F1 scores for *sha512* (0.12), *sha1* (0.58), and *md5* (0.62). IOCSEARCHER performs best in 11 of the 13 (85%) indicator types it supports in the table, IOC-EXTRACTOR in 8

Indicator	Count	Tools
fqdn	41,360	7
attackType	26,109	1
country	17,654	1
filename	17,082	4
url	14,818	8
md5	14,635	8
ip4	9,479	8
sha256	5,336	8
avLabel	4,660	1
sha1	4,150	8

Table 5.7: Top-10 indicator types extracted by all tools.

out of 17 (47%), and IOC-FINDER in 9 out of 20 (45%). CYOBSRACT is the best tool for extracting registry keys (0.85) and SSDeep hashes (0.84); JAGER for extracting filenames (0.98), IOC_PARSER for filepaths (0.69), and IOCSEARCHER for emails (1.0), domain names (0.99), IPv4 addresses (1.0), and URLs (0.87). These results are useful for future work in this area, to identify which prior tool may have the best regular expression for an indicator type. For example, if we were to add registry key support to IOCSEARCHER we would start by looking at CYOBSRACT and for filenames to JAGER.

The indicator types with the lowest agreement are IPv6 addresses with F1 score ranging 0.21–0.75; filepaths (0.37–0.69); URLs (0.52–0.87); and registry keys (0.71–0.85). These are arguably the indicator types for which regular expressions are harder to build. We observe that most IPv6 addresses extracted are actually FPs caused by a regular expression that matches serial numbers in certificates and certificate fingerprints.

One surprising result is that IOCSEARCHER has an F1 score of 0.03 on Bitcoin addresses, while it achieves the best F1 score on most other indicators it extracts. We manually checked the extracted Bitcoin addresses and observed that IOCSEARCHER is actually always correct in identifying Bitcoin addresses. Both IOC-EXTRACTOR and IOC-FINDER return hashes that are erroneously included as Bitcoin addresses. IOCSEARCHER avoids those FPs by examining the checksum in the regular expression matches, which do not validate. This is an example of the minority of tools being correct and the majority being wrong. We discuss this issue in Section 5.8.

Table 5.7 shows the Top 10 most popular indicator types across the 4,420 documents. The count column represents the number of TPs, and the last column reports the number of tools that extract each indicator. Among those, 7 correspond to indicators supported by most tools, namely domain names, file names, URLs, IPv4 addresses, and hashes (*md5*, *sha256*, *sha1*). On the other hand, the other three indicators are extracted only by CYOBSRACT and correspond to attack-type keywords, countries, and AV labels. A total of 16 indicator types are not found in the 4,420 reports. These include *attribution*, *groupName*, *authentihash*, *ip6range*, *useragent*, *iban*, *icp*, 7 blockchain addresses (*dashcoin*, *dogecoin*, *ethereum*, *litecoin*, *tezos*, *webmoney*, *zcash*), and two social handles (*telegramHandle*, *whatsappHandle*). These correspond to less popular indicators in APTnotes and ChainSmith reports. However, other sources may differ. For example, RSS collection includes blockchain-related blogs, which leads to IOCSEARCHER extracting indicators for all blockchain addresses in Section 5.5 (e.g., 726 *ethereum* addresses).

A	B	p-unc	p-corr	eta-square
CACADOR	CYOBSTRACT	7.6527e-03	1.0203e-02	3.6639e-04
CACADOR	IOCEXTRACT	3.1698e-21	1.4792e-20	1.2091e-03
CACADOR	IOC-EXTRACTOR	2.9591e-18	1.1836e-17	8.6447e-04
CACADOR	IOC-FINDER	1.1236e-17	3.9328e-17	8.4822e-04
CACADOR	IOC_PARSER	7.1188e-05	1.4237e-04	4.7131e-05
CACADOR	JAGER	1.0000e+00	1.0000e+00	0.0000e+00
CACADOR	IOCSEARCHER	9.5401e-23	8.9041e-22	1.3045e-03
CYOBSTRACT	IOCEXTRACT	1.0013e-02	1.2744e-02	2.4565e-04
CYOBSTRACT	IOC-EXTRACTOR	1.1071e-01	1.2916e-01	1.0536e-04
CYOBSTRACT	IOC-FINDER	1.1717e-01	1.3123e-01	9.9363e-05
CYOBSTRACT	IOC_PARSER	4.2949e-04	7.5162e-04	6.7494e-04
CYOBSTRACT	JAGER	2.1251e-02	2.5871e-02	3.6767e-04
CYOBSTRACT	IOCSEARCHER	5.4571e-03	8.0421e-03	2.8976e-04
IOCEXTRACT	IOC-EXTRACTOR	5.9633e-03	8.3486e-03	2.9454e-05
IOCEXTRACT	IOC-FINDER	3.5693e-03	5.5523e-03	3.2923e-05
IOCEXTRACT	IOC_PARSER	3.6201e-25	5.0681e-24	1.7284e-03
IOCEXTRACT	JAGER	2.6479e-10	7.4141e-10	1.2132e-03
IOCEXTRACT	IOCSEARCHER	2.4354e-08	5.2456e-08	1.8189e-06
IOC-EXTRACTOR	IOC-FINDER	6.2505e-01	6.4820e-01	9.3784e-08
IOC-EXTRACTOR	IOC_PARSER	3.8698e-22	2.7089e-21	1.3122e-03
IOC-EXTRACTOR	JAGER	1.1041e-08	2.8105e-08	8.6746e-04
IOC-EXTRACTOR	IOCSEARCHER	6.1777e-04	1.0175e-03	4.5934e-05
IOC-FINDER	IOC_PARSER	1.4453e-21	8.0938e-21	1.2927e-03
IOC-FINDER	JAGER	1.3778e-08	3.2149e-08	8.5116e-04
IOC-FINDER	IOCSEARCHER	3.6823e-04	6.8737e-04	5.0259e-05
IOC_PARSER	JAGER	1.3860e-01	1.4927e-01	4.7302e-05
IOC_PARSER	IOCSEARCHER	1.0825e-26	3.0310e-25	1.8419e-03
JAGER	IOCSEARCHER	5.4592e-11	1.6984e-10	1.3090e-03

Table 5.8: Post-hoc t-tests between groups summary.

Statistical significance. We evaluate whether differences in the extracted indicators across tools are statistically significant by first performing an analysis of variance test and then measuring the strength of the differences using pairwise t-tests between the tools. The independent variable is the tool used to produce each set of IOC detections. There are eight groups accounting for the source of variability, one per tool. The dependent variable is the number of detections (TPs) produced by each tool. We perform a Repeated Measures One-Way ANOVA test because all tools are evaluated on the same samples (*i.e.*, documents). We focus on the 6 indicator types extracted by all tools (*email*, *ip4*, *md5*, *sha1*, *sha256*, *url*).

The test results indicate a significant statistical difference between the detections produced by the tools, with a p-value lower than 10^{-5} . We apply pairwise t-tests among groups to identify which pairs of tools exhibit significant differences. We use the Benjamini/Hochberg FDR correction method for the p-values because of the multiple statistical tests produced. Table 5.8 shows the results of this test. A statistically significant difference between detections is found among every pair of tools (shown by the *p-corr* column), except for 5 pairs (CACADOR–JAGER, CYOBSTRACT–IOC-EXTRACTOR, CYOBSTRACT–IOC-FINDER, IOC-EXTRACTOR–IOC-FINDER, and IOC_PARSER–JAGER). Figure 5.3 shows the strength of the differences between tools, measured by the *eta-square* value resulting from the tests. The results show that detections produced by IOCSEARCHER are statistically significant compared to all other tools.

Runtime. Table 5.9 shows the total time in seconds each tool took to extract indicators on the 4,420 text documents, the number of indicator types supported, and the average time by indicator type. In our tests, the fastest tool is CACADOR, likely due to being implemented in

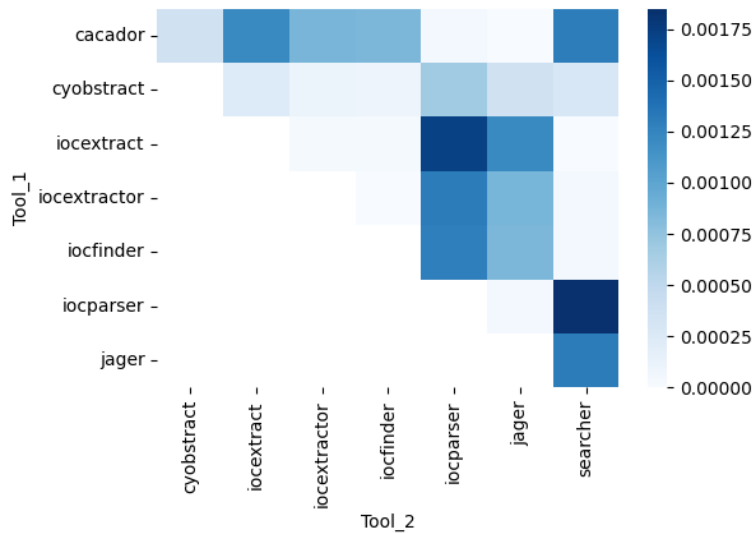


Figure 5.3: Eta-square values resulting from pairwise comparison of tools.

Tool	Runtime (sec)	Ind.	Avg. per Ind. (sec)
CACADOR	489.23	12	40.76
IOCEXTRACT	1350.69	8	168.83
IOC_PARSER	1563.33	11	142.12
IOC-EXTRACTOR	1978.51	18	109.91
JAGER	2360.02	11	214.54
IOCSEARCHER	2847.98	41	69.46
CYOBSTRACT	6945.03	25	277.80
IOC-FINDER	96596.23	25	3863.84

Table 5.9: Indicator extraction runtime and average by indicator type (in seconds).

Go and compiled to native code. The slowest tool is IOC-FINDER, which is 14 times slower than the second slowest tool (CYOBSTRACT). This is probably because IOC-FINDER is the only tool that uses grammars for the extraction, highlighting the efficiency of using regular expressions to extract indicators. All tools perform one pass on the text for each regular expression and they typically use one regular expression for each indicator type. Thus, the runtime is highly influenced by the number of indicator types extracted. When normalizing the total runtime by the number of extracted indicator types, CACADOR is still the fastest with 40.8 seconds, followed by IOCSEARCHER (69.5), and IOC-EXTRACTOR (109.9).

Social handle validation. We cannot apply our majority-based evaluation to 38 indicators extracted by only one tool. Among those, there are 11 social indicators only extracted by IOCSEARCHER, of which 9 appear in APTnotes and ChainSmith reports (we found no Telegram and WhatsApp handles in these sources). We perform an alternative validation for social network handles, which checks whether an account *currently* exists in the social network for that handle. For this, we use the Blackbird [150] open-source tool. Given a username, it checks if an account with that handle currently exists on 143 different social networks. Blackbird supports 6 social networks IOCSEARCHER extracts handles for (Facebook,

Indicator	Total	Validated
facebookHandle	150	124
gitHubHandle	258	242
instagramHandle	10	8
pinterestHandle	1	1
youTubeHandle	24	24
youTubeChannel	6	6
twitterHandle	516	431
All	965	836

Table 5.10: Social indicators validation.

Tool	TP	FP	FN	TN	Prec.	Recall	F1
CACADOR	29	69	0	8	0.30	1.00	0.46
IOC_PARSER	29	66	0	11	0.31	1.00	0.47
IOCSEARCHER	29	6	0	71	0.83	1.00	0.91

Table 5.11: Accuracy of filtering approaches on a manually-built ground truth.

GitHub, Instagram, Pinterest, Telegram, and YouTube). In addition, we expanded Blackbird to support *youtubeChannel* indicators. For Twitter handles, we leverage the official Twitter API for the same purpose [209]. LinkedIn does not allow searching for account names.

Table 5.10 summarizes the results. Our automated approach successfully validated 87% of the social indicators. We manually inspect the 102 indicators that we could not validate with Blackbird, using a web browser to search for a particular handle on each social network. For Twitter, 15 of the 85 usernames that did not validate, were linked to suspended accounts (e.g., [@MalwareSigs](#)). In all of the remaining cases, the social networks returned a page to inform us that the particular account does not exist. For 75 of the 87 accounts for which we could not find the user profile page, the identifier contained a meaningful sequence of characters (e.g., “sucuri_security”, “avast_antivirus”), suggesting that the identifier could belong to an old account that was closed. Overall, this evaluation suggests that over 90% of the social identifiers IOCSEARCHER extracted are likely true positives. For the rest, we are not able to determine if they are false positives or correspond to accounts that have been closed since they appeared in the sources.

Filtering. To evaluate the filtering, we manually built a ground truth (GT) of 106 indicators. We randomly selected those indicators among those extracted by IOCSEARCHER across all datasets and prior to the filtering. Then, an analyst manually reviewed the documents from where the indicators were extracted to determine whether each indicator was benign or malicious. The analyst made a determination based on the indicator context in the document, i.e., by reading the parts of the document that mentioned the indicator.

To measure the filtering accuracy, we first apply the filtering approach of IOC_PARSER, CACADOR, and our platform to the 106 indicators. Then, for each approach, we compare the indicators that survive the filtering with the GT. We consider a true positive (TP) a malicious indicator in the GT that correctly survives the filtering and a TN is a benign indicator in the GT that was correctly filtered out. A false positive (FP) is a benign indicator in the GT that incorrectly passes the filtering, while a false negative (FN) is a malicious indicator in the GT incorrectly removed by the filtering. From those counts, we compute the precision,

recall, and F1 score. Note that it does not matter that indicators in the GT were selected among those extracted by IOCSEARCHER, as we do not extract indicators using CACADOR and IOC_PARSER in this experiment. We only run their filtering rules on the GT indicators.

Table 5.11 summarizes the results. All three filtering approaches have zero FNs and thus perfect recall, i.e., they do not remove any truly malicious indicators. On the other hand, precision significantly differs; our filtering achieves more than twice higher precision (0.83) than the filtering used by IOC_PARSER (0.31) and CACADOR (0.30). The low precision by CACADOR and IOC_PARSER is due to its filtering being fairly incomplete, allowing many benign indicators to be output as malicious IOCs. The final F1 score is 0.91 for our filtering, followed by IOC_PARSER (0.47) and CACADOR (0.46). The results indicate that the static blocklists used by IOC_PARSER and CACADOR fail to filter many of the benign indicators in the documents. Furthermore, despite the IOC_PARSER blocklist being an order of magnitude larger than the CACADOR blocklist, the recall improvement is marginal. This shows that it is hard to predict the sources from where a user will collect reports, making static blocklists largely incomplete. The use of a dynamic blocklist in our filtering module significantly improves the recall by adjusting the filtering to the collected sources. For example, it is very common for tweets to contain URLs to external references, including links to documents from other sources (e.g., an already monitored blog or RSS feed). The blocklists of CACADOR and IOC_PARSER would assume that such a URL is an IOC, possibly because the tool developers did not consider Twitter as a source. IOCSEARCHER correctly flags those generic indicators as FPs, since it relies on a blocklist that is dynamically expanded each time the user adds a new source that will be monitored. In summary, the dynamic blocklist allows filtering a larger amount of benign indicators than static blocklists without losing any malicious IOCs.

5.7 Case Study: Blockchain Address Tagging

Cryptocurrencies, such as Bitcoin, are abused by cybercriminals. A key property of Bitcoin is that its transaction ledger is public, which allows it to inspect and analyze all transactions. Related work has leveraged this visibility on transactions for tracking payments from victims of ransomware [90, 151] and scams [152, 130]; buying and selling of illegal items in underground marketplaces [38, 113]; and identifying malicious communication channels hosted on the blockchain [157, 199]. Bitcoin addresses are pseudo-anonymous, but sometimes it is possible to identify a set of addresses owned by an entity, e.g., by transacting with a service [130] or by analyzing malware samples [58]. Related work that analyzed malicious Bitcoin transactions leveraged Bitcoin tag databases. These databases contain Bitcoin addresses labeled with information on whether the address is malicious or benign and who the owner is. Tag databases are a cornerstone of many commercial blockchain analysis companies [36, 59, 50].

Building and updating tag databases is currently a largely manual process, and analysts are in desperate need of tools to assist them. This case study shows how we leveraged GoodFATR to assist us in updating the Bitcoin tag database proposed in Chapter 3, which was essential for identifying relationships between cybercriminals that abuse Bitcoin [73].

We deliberately included blockchain-related blogs in the RSS origins list input to GoodFATR to collect Bitcoin addresses, e.g., Blockchain on Medium and Cointelegraph.com News, as

illustrated in Table 5.5. GoodFATR collects new RSS entries daily, downloads the documents using the entry URLs, and extracts the indicators. Every week, a cron job selects the Bitcoin addresses identified over that week, tags them with the URLs of the reports containing the addresses, filters those already present in our tag database, and sends an email with the untagged addresses to an analyst.

Notice that we do not know if the addresses reported in the email were used only in legitimate transactions, as our Bitcoin address validation step only discards indicators with an invalid checksum, with no further filtering. We are interested in collecting malicious and benign addresses, as cybercriminals often abuse addresses of legitimate services (e.g., cryptocurrency exchanges). After receiving the list of new Bitcoin addresses, the analyst examines the documents downloaded by GoodFATR (or visits the report URLs) and manually tags each address. To this end, the analyst searches for the address in the report text and reads the section mentioning the address to tag it accordingly.

This approach allows GoodFATR to gather a continuous feed of fresh Bitcoin addresses, showing how to identify and tag addresses quickly with low effort, e.g., within a few days since they first appear in a monitored origin. The tagged addresses are added to the database used by our automated system [73]. Malicious addresses can also be used as input to our system to trace their transactions. In 1.3 years (68 weeks), our approach automatically identified 854 new Bitcoin addresses, for an average of 12.6 Bitcoin addresses per week, a median of one address, a maximum of 497, a minimum of zero, and a standard deviation of 60.4. In Chapter 6, we investigate NLP techniques and show how to leverage LLMs to automate the tagging of Bitcoin addresses using information about their context, e.g., criminal activity described in abuse reports, which could avoid any manual steps by the analyst.

5.8 Discussion

Threats to validity. Our methodology for comparing indicator extraction tools assumes that the majority of the tools will be correct. However, in some cases, the minority of the tools may be correct instead, as illustrated by the *bitcoin* results where IOCSEARCHER is wrongly assigned FNs due to FPs from two other tools. In general, the more tools support an indicator, the higher confidence we have in the verdict of the majority. An alternative for evaluating tools is using a ground truth (GT) dataset. However, the manual effort required to build a GT with thousands of documents would be overwhelming, and it would be challenging to cover different indicator types. For this reason, regular expression evaluations usually leverage synthetic examples (e.g., [127]). Unfortunately, synthetic examples may not represent results on real-world documents.

Our evaluation could be biased due to document selection, unknown normalizations, expired social accounts, and the size of the ground truth used for evaluating the filtering. For example, we use exclusively English documents. While regular expressions should mostly be language-agnostic, some language-specific features (e.g., special characters) may affect the results. Our comparison methodology may negatively evaluate a tool if we miss some normalization it performs on its outputs. The validation of social handles may underestimate the IOCSEARCHER’s accuracy since extracted handles may correspond to accounts since

deleted. Finally, we perform the filtering evaluation on a small ground truth of 106 indicators. A larger ground truth may uncover more challenging cases where accuracy drops.

Adding origins. Adding new origins is currently the only step where GoodFATR requires human involvement. GoodFATR can automatically identify new origins to be considered for inclusion. For this, it examines the report URLs in the crowd-sourced datasets (Malpedia, APTnotes) ranking domains in the report URLs by the number of report URLs where they appear. Then, it filters domains already in the list of RSS origins. Finally, the top-ranked domains are flagged as potential cybersecurity blogs, so that an analyst can examine them and identify their RSS URL. To identify candidate new Twitter accounts to monitor, GoodFATR examines the re-tweets in the monitored accounts, building a ranking from the original tweeting account to the number of collected re-tweets from the account. The top re-tweeted accounts, not yet monitored, are output so that a human can analyze them for inclusion. In future work, we would like to explore integrating techniques to identify valuable Twitter accounts by analyzing their posts [139].

Report variants. If the same document (i.e., same SHA256 hash) is collected from different origins, GoodFATR deduplicates it to store only one copy of the document and updates the document’s traceability information to capture all origins from where the document was collected. As introduced in Section 5.3, it is possible that GoodFATR collects multiple documents corresponding to different instances of the same threat report. This may happen if a report is distributed in different formats, in case GoodFATR downloads multiple times the URL of a report that points to a webpage with dynamic content, and if a report generates multiple versions over time (e.g., by fixing some errata). Currently, the user can identify multiple instances of the same report by querying for all documents downloaded from the same URL or for all documents with the same title. However, this may miss some cases, e.g., when the title has changed. To address this issue, GoodFATR could store a similarity hash of the text content (extracted from the PDF or HTML document), which could be used to identify small variations of the same content.

In this work, GoodFATR was configured to download a URL only once (unless an error happened). This configuration reduces the number of needed downloads and removes some of the above cases. On the other hand, it may fail to collect different versions of the same report that appear over time. If the user wants to collect multiple versions of a report, he can easily change this configuration, at the cost of increased network bandwidth and runtime. In future work, we would like to explore leveraging the ETag header to keep track of the document pointed by a URL and use an HTTP HEAD request to check if the document was updated since the last time we retrieved it. With this approach, the report would only be re-downloaded if a new version exists. However, not all websites provide the ETag header or support HTTP HEAD requests.

5.9 Conclusions

This chapter presents GoodFATR, an automated platform for collecting threat reports from 6 sources (RSS, Twitter, Telegram, Malpedia, APTnotes, ChainSmith) that compares the

accuracy of indicator extraction tools on the collected reports. GoodFATR implements a novel majority vote methodology that removes the need for a manually built ground truth. It continuously monitors the sources, downloads new threat reports, extracts indicators from the reports, and filters generic indicators to produce a list of IOCs. GoodFATR includes the IOCSEARCHER tool for extracting 41 indicator types from HTML, PDF, and text files using regular expressions. We have run GoodFATR for over 15 months, collecting 472,891 reports from the 6 sources; extracting 978,151 indicators from the reports using IOCSEARCHER; and identifying 618,217 IOCs. We analyzed the collected data to determine the top IOC contributors and the IOC class distribution. Then, we applied GoodFATR to compare 7 popular indicator extraction tools with IOCSEARCHER and assess their accuracy using our novel majority-vote methodology. Finally, we presented a case study on how GoodFATR can assist in tracking cybercrime relations on the Bitcoin blockchain.

Chapter 6

Automatic Classification of Cryptocurrency Abuse Reports

6.1 Introduction

Cryptocurrencies have become a popular payment and financial investment mechanism. Unfortunately, they are also frequently abused by cybercriminals, for example, to receive payments from victims of criminal activities such as ransomware [196, 116, 45, 90, 151], sextortion [152, 143], and scams [9, 130, 222, 10, 115]. A valuable source of data on cryptocurrency abuse is reporting services that provide online forms for users to fill out with information about the abuse they have observed (e.g., date, description, type, amount lost), the abuser (e.g., email address, Bitcoin address), and the reporters themselves (e.g., country, age). While some abuse reporting services specialize in cryptocurrency abuse (e.g., BitcoinAbuse [15], ChainAbuse [35]), others collect all abuse reports, regardless of whether they involve cryptocurrencies, such as the Better Business Bureau (BBB) Scam Tracker [184], the Australian Competition and Consumer Commission (ACCC) ScamWatch [185], and the Federal Trade Commission (FTC) ReportFraud [166].

Abuse reports provide critical information on abusive Bitcoin addresses, prevalent cryptocurrency abuse types, and financial losses suffered by victims. Blockchain explorers [75, 50] link Bitcoin addresses to their abuse reports in BitcoinAbuse so that users can avoid known malicious addresses. Unfortunately, abuse reporting services can suffer data pollution due to their crowd-sourced nature. A fraction of the submitted reports may be useless, including advertisements, trolling, reports aiming to pollute the database, and reports not describing abuse. Some services manually validate the reports to address data pollution. Unfortunately, manual validation is costly and does not scale. Thus, a critical challenge is how to identify valid reports automatically.

Abuse reports are also fundamental for analyzing the prevalence and financial impact of different abuse types (e.g., sextortion [152], investment scams [9]). BBB and FTC publish well-known annual reports with statistics on the received abuse reports [43, 125], which prior work often cites in their motivation [14, 109, 110]. For example, in 2023, the FTC received

47K abuse reports involving cryptocurrency payments, totaling \$1.4 billion in losses [43]. Those statistics rely on an accurate classification of the abuse reports into abuse types, for which two approaches exist. The first approach requires the reporter to select the abuse type from a service-specific list of options. However, reporters may be under stress when reporting abuse and not care about accurately classifying the reports, may not have enough background to differentiate abuse types, or may even be malicious. An alternative approach is for human analysts to examine the submitted reports, inferring the abuse type from the textual description where reporters detail the abuse in their own words. This approach is manual and does not scale well. Furthermore, humans can be inconsistent in repetitive tasks, and multiple analysts may classify similar reports differently.

We perform the first analysis of data pollution in cryptocurrency abuse reporting services. To enable our analysis, we collect cryptocurrency abuse reports from two services. We obtained 287K abuse reports for 92K Bitcoin addresses received by BitcoinAbuse [15] over 6 years. We also collected 2,321 cryptocurrency abuse reports received by Scam Tracker [184] over 5 years.

We answer the following three research questions:

(RQ1) What is the extent and impact of pollution in abuse reporting services?

We examined three types of pollution: spam reports, incorrect abuse types reported by users, and benign addresses reported as malicious. We focused on BitcoinAbuse, where there was no filtering for submitted reports. Then, we checked if the identified issues also affected Scam Tracker. We identify two prevalent types of spam reports in BitcoinAbuse: advertisements for funds recovery services and reports that aim to discredit cryptocurrency exchanges by claiming they are involved in terrorism. At least 10.6% of all BitcoinAbuse reports are spam, with the ratio reaching 75% of submitted reports prior to the service stopping operation. In contrast, Scam Tracker manually validates submitted reports. Still, we observe a handful (0.4%) of funds recovery spam reports missed by their filtering.

To measure the accuracy of user-reported abuse types, we develop a taxonomy of 19 popular abuse types and build a ground truth with 19,443 BitcoinAbuse descriptions labeled using the taxonomy. We compare this ground truth with the 6 abuse types that reporters could select in the online form (which we call *BA types*). Reports labeled with three BA types (Bitcoin-Tumbler, Darknet-Market, Ransomware) were almost entirely spam. Two other BA types (Blackmail-Scam, Other) were too broad. The only accurate BA type was Sextortion. Even after removing spam reports, users often selected an incorrect abuse type, with 22% of Ransomware-labeled reports being indeed *sextortion*. In Scam Tracker, analysts classify the reports. However, all cryptocurrency abuse reports are placed in the same class, preventing the analysis of fine-grained abuse types.

Out of the 92,151 Bitcoin addresses reported to BitcoinAbuse, we identify 91 (0.1%) benign addresses, most often addresses of exchanges. 72% of reports for benign addresses are spam, but we also observe users trying to track the destination of their funds and ending up incorrectly reporting an exchange address. Even if benign addresses are a minority, their impact is overwhelming, with those 91 addresses being responsible for 60% of all received funds. While Scam Tracker does not have a specific field for reporting cryptocurrency addresses, the 96 Bitcoin addresses found in the text of reports are not known to be benign.

Previous works that analyzed BitcoinAbuse [37, 27, 176] did not filter out spam reports, considering that all reported addresses were malicious (and thus that their deposits corresponded to cybercriminal revenue) and that the reporter-selected abuse types were correct. Our results cast a shadow over their conclusions.

(RQ2) Can the identification of valid reports and their classification into abuse types be automated? We propose a novel classifier that leverages a large language model (LLM) for interpreting the textual description in an abuse report and assigning it one of 19 abuse types in our taxonomy. Our unsupervised LLM-based classifier does not need training. It relies on natural language definitions for the abuse types, making it easy to extend by adding definitions for new abuse types. We examine multiple designs for the classifier covering the impact of the definitions, three query approaches, and six LLMs: two open models (llama3, llama3.1) and four commercial models (gpt-4, gpt-4o, gpt-4o-mini, gpt-3.5). We compare our LLM-based classifier with a baseline supervised machine learning classifier.

The best design for the LLM-based classifier achieves an F1 score of 0.95 compared to 0.42 for a naive design used as a baseline. In out-of-distribution (i.e., previously unseen) data, it achieves a precision of 0.92, a recall of 0.87, and an F1 score of 0.89, whereas the baseline supervised model achieves an F1 score of 0.55. Furthermore, it filters spam reports with an F1 score of 0.99. Finally, we show the ease of extending our classifier with a new abuse type.

(RQ3) What is the financial impact of different cryptocurrency abuse types? Two approaches exist to estimate the financial impact of cryptocurrency abuse types: adding losses disclosed by victims in their abuse reports and analyzing transactions in the blockchain that deposit funds to reported addresses. Both approaches require accurate classification of abuse types. Unfortunately, Scam Tracker does not sub-classify cryptocurrency abuse reports, and we show that BitcoinAbuse types are untrustworthy.

To answer this question, we classify 2.3K Scam Tracker reports and 5K BitcoinAbuse reports using our LLM-based classifier. Using the classification, we unveil that victims often fall for *investment* scams, with 61% of victims of that scam type reporting financial losses, four times higher than the next abuse type. The median loss is highest for *fundsrecovery* scams at \$1,700 per report, followed by *romance* (\$1,500) and *investment* scams (\$1,300). To estimate revenue from address deposits, we propose a majority vote to aggregate reports for the same address and use it to classify 1,930 addresses reported in BitcoinAbuse. The deposits to those addresses total \$453M and confirm that *investment* scams have the highest financial impact (44% of the revenue). Despite their low conversion rate, extortions collect nearly \$10M in revenue due to their massive email campaigns. We compare both estimation approaches on 71 Scam Tracker reports mentioning 96 addresses. Victim-reported losses heavily underestimate revenue, with the estimation from deposit transactions being 29 times higher.

6.2 Abuse Reporting Services

Abuse reporting services receive reports through online forms, which include a *description* field for the reporter to describe the abuse in natural language. Beyond that, services may

Dataset	Reports	Desc.	Addr.	Start	End
BitcoinAbuse	287,823	224,059	92,151	03/2017	06/2023
Scam Tracker	2,321	2,311	96	02/2017	05/2024

Table 6.1: Abuse report datasets used.

differ. First, some abuse reporting services, such as BitcoinAbuse [15], ChainAbuse [35], and Scam Tracker [184], make the abuse reports public, while others, such as FTC’s Report-Fraud [166] and ACCC’s ScamWatch [185] do not. We focus on BitcoinAbuse and Scam Tracker because they make the abuse reports public. We skipped ChainAbuse (which acquired BitcoinAbuse [208]) because free access to their API is very restrictive, and they declined our request for research access to their commercial API. Second, services may differ in their focus. BitcoinAbuse focused on cryptocurrency abuse, accepting reports from any geographical location written in different languages. Scam Tracker covers any abuse type affecting users in the USA and Canada, regardless of the involvement of cryptocurrencies. Thus, reports are in English. Third, some services post the submitted reports immediately online without validation (e.g., BitcoinAbuse) or with some undisclosed filtering (e.g., ChainAbuse). Others, like Scam Tracker, manually validate the submitted reports before posting them, determining if they are valid and cleaning up their contents (e.g., anonymizing personal information). Fourth, the information requested in the online form varies. For example, Scam Tracker allows the reporter to disclose the losses suffered, while BitcoinAbuse did not. They also differ in the requested abuser’s data; BitcoinAbuse only requested the abuser’s blockchain address, while Scam Tracker also requests URLs, email addresses, and phone numbers. However, Scam Tracker does not request blockchain addresses, which only appear in the report’s text.

As summarized in Table 6.1, we collect abuse reports from two services: BitcoinAbuse and BBB Scam Tracker.

BitcoinAbuse. Originally, BitcoinAbuse allowed to download the whole database of abuse reports through its API. We downloaded a complete database dump on February 1, 2021. Later, the API changed to only provide the reports of the last 30 days, and we have monthly dumps from February 2023 until early June 2023, when BitcoinAbuse stopped accepting reports and started redirecting users to ChainAbuse [35]. To fill the collection gap, we built a crawler for the BitcoinAbuse webpage, which lists all reported addresses. We query the API for each address in the webpage that was not in our dump. The query returns the last 10 reports for the address. Thus, our dataset contains reports on all addresses in BitcoinAbuse but may miss some reports for addresses with more than 10 reports. The BitcoinAbuse webpage provides the total number of reports in the database for each year, which we use to compute the fraction of all database reports in our dataset. Overall, we have collected 86% of all reports and have reports for 100% of the addresses.

Each report contains the report creation date and four fields from the online form: the reported Bitcoin address, the *abuse type* selected from a drop-down list with 6 options, an *other-abuse* short string optionally used to describe other types of abuse, and the description text for freely describing the abuse (2,000 characters maximum).

The dataset comprises 287,823 reports for 92,151 Bitcoin addresses, an average of 3.1 re-

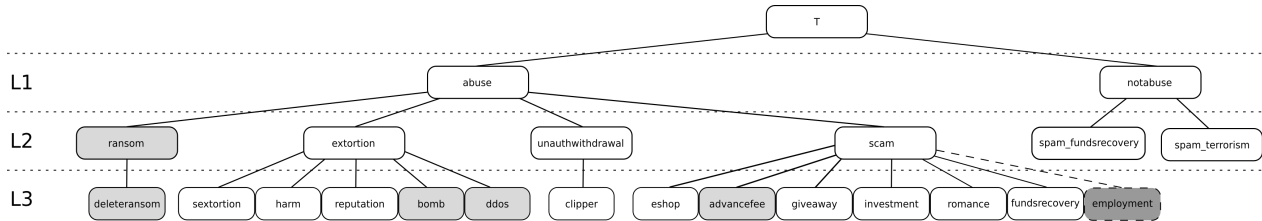


Figure 6.1: Our taxonomy. White nodes come from the initial taxonomy. Light gray solid nodes were identified by the clustering. Dark gray dashed nodes were not originally part of the taxonomy but were added during our evaluation.

ports per address. The 287,823 reports have 224,059 unique descriptions (by SHA256 of the description text). Multiple reports may have the same description, e.g., due to short descriptions such as “Sextortion” or “Scam”. We identify the description’s language using the *langdetect* tool [51]. Most descriptions are written in English (80.5%), followed by Russian (1.7%), French (1.7%), Spanish (1.6%), and German (1.6%).

Scam Tracker. We build a crawler to download abuse reports from the Scam Tracker website. Scam Tracker groups all reports related to cryptocurrencies in the same *Cryptocurrency* abuse type. In May 2024, we downloaded all 2,321 Cryptocurrency reports. For each report, we obtain the scam ID, report date, description, victim’s location, dollars lost (optional, may be zero), and attacker’s indicators (URL, phone, email) if provided. The 2,321 reports have 2,311 descriptions written in English. The online form does not collect cryptocurrency addresses, but we use the IOCSEARCHER tool [121] to extract 96 Bitcoin addresses present in 71 reports.

6.3 Abuse Type Taxonomy and Ground Truth

To answer RQ1 and RQ2, we build a taxonomy of prevalent abuse types and ground truth datasets of descriptions annotated with their abuse type in the taxonomy. Section 6.3.1 describes the design of our taxonomy, and Section 6.3.2 details the process for generating the taxonomy and ground truth.

6.3.1 Taxonomy

Figure 6.1 shows our tree-structured abuse taxonomy, where nodes are abuse types, and edges capture parent-child relationships, e.g., *investment* is a subtype of *scam*. The lower a node, the more specific abuse it captures.

The *first level* (L1) underneath the root node (T) captures whether the report is valid (*abuse*) or useless (*notabuse*). Valid abuse reports describe abuse suffered or observed by the reporter, regardless of whether the abuse was successful (i.e., lost funds). In contrast, *notabuse* reports may instead advertise a service, try to pollute the reporting database (e.g., trolling), or offer no usable information (e.g., provide a personal opinion of the reporter). The two L1 nodes cover all reports, i.e., a report is either an abuse report or not.

The *second level* (L2) captures the four main abuse types identified and two prevalent types of spam reports. A *scam* is an attempt to defraud by gaining the victim’s confidence. The term *scam* is sometimes used to refer to any abuse type, i.e., as a synonym of *abuse*. However, in this work, we use *scam* to refer only to attacks that abuse the user’s confidence. In contrast, *extortion* captures coercion, i.e., the victim is threatened with some future harm if a payment to the extortionist is not made. An extortionist does not try to gain the victim’s confidence as a scammer does but directly threatens the victim. Some attacks demand a *ransom* payment from the victim, e.g., to recover access to encrypted or deleted data. The difference with *extortion* is that there is no threat of future harm. Instead, in *ransom*, the damage has already happened, i.e., the data has already been encrypted or deleted. Finally, *unauthwithdrawal* captures thefts where funds belonging to a victim are transferred without authorization or consent, e.g., by stealing addresses’ private keys. This level also includes two prevalent types of *notabuse* spam reports. The first is reports advertising funds recovery services (*spam_fundsrecovery*) where reporters state that they lost funds to some scam and then used a funds recovery service to get the funds back. The report provides a contact (i.e., email, URL, phone) for the funds recovery service. Given the difficulty of recovering Bitcoin funds, even for law enforcement, these services are with all probability scams. The other are reports claiming that the reported addresses are involved in terrorism (*spam_terrorism*), but those addresses belong to well-known exchanges (e.g., Binance, HTX).

The *third level* (L3) captures prevalent subtypes of the main abuse types. We identify 5 subtypes of *extortion*. In *sextortion*, the extortionist threatens to release the victim’s data of sexual character, most often a video. But, the extortionist may also threaten to damage the victim’s *reputation* or personal integrity (*harm*), make some resources of the victim unavailable (*ddos*), or damage other persons known to the victim by planting a *bomb*. We identify 7 prevalent *scam* subtypes. The attacker may try to convince the victim to participate in a supposedly highly profitable *investment* or in a *giveaway* that promises victims they will receive back twice (or more) the amount they send to the scammers. Scammers may set up a fake online *eshop* that does not deliver the goods bought by the victim, request a fee in advance for a service that will not be delivered (*advancefee*), try to establish a friendship or loving relation to later request money from the victim (*romance*), offer shady jobs that are not paid (*employment*), or promote useless funds recovery services (*fundsrecovery*). Note that *fundsrecovery* corresponds to valid reports of victims scammed by funds recovery services, while *spam_fundsrecovery* are advertisements posted by scammers. For *unauthwithdrawal*, we only observe one prevalent subtype corresponding to the victim’s device becoming infected with a *clipper* malware that monitors the clipboard of the infected host. When the user copies a blockchain address, the malware replaces it with an address controlled by the attacker. If the victim uses the (replaced) address as the destination in a transaction, the attacker will receive the funds instead of the intended recipient. For *ransom*, we only observe one prevalent subtype that deletes data, typically from a database with no or weak authentication, to request a ransom to return the deleted data (*deleteransom*).

Previous works focus on a single cryptocurrency abuse type such as ransomware [196, 116, 45, 90, 151], sextortion [152], giveaway scams [115], and investment scams [9]. In contrast, our taxonomy contains 4 L2 and 14 L3 abuse types. However, any abuse taxonomy is necessarily incomplete as cybercriminals frequently introduce new abuse types and variations. Our

Type	gt_initial	gt_final	gt_bbb
sextortion	44 (25.1%)	16,112 (82.8%)	23 (11.5%)
extortion	28 (16.0%)	1,417 (7.3%)	2 (1.0%)
giveaway	6 (3.4%)	740 (3.8%)	6 (3.0%)
spam_fundsrecovery	1 (0.6%)	174 (0.9%)	-
harm	1 (0.6%)	170 (0.9%)	-
deleteransom	-	162 (0.8%)	1 (0.5%)
investment	11 (6.3%)	143 (0.7%)	102 (51.0%)
advancefee	-	74 (0.4%)	8 (4.0%)
notabuse	27 (15.4%)	73 (0.4%)	8 (4.0%)
spam_terrorism	3 (1.7%)	64 (0.3%)	-
ddos	-	62 (0.3%)	-
scam	31 (17.7%)	60 (0.3%)	48 (24.0%)
reputation	1 (0.6%)	54 (0.3%)	-
bomb	-	52 (0.3%)	-
romance	2 (1.1%)	34 (0.2%)	-
clipper	1 (0.6%)	33 (0.2%)	-
unauthwithdrawal	7 (4.0%)	7 (0.0%)	2 (1.0%)
eshop	6 (3.4%)	6 (0.0%)	-
abuse	5 (2.8%)	5 (0.0%)	-
fundsrecovery	1 (0.6%)	1 (0.0%)	-
ransom	-	-	-
All	175 (100%)	19,443 (100%)	200 (100%)

Table 6.2: Number of descriptions for each abuse type in ground truth datasets.

hierarchical taxonomy is designed to handle such missing abuse types. For example, any scam report that does not match the 7 L3 scam subtypes can be labeled using the L2 *scam* type, indicating that it captures another type of scam. Examining the reports assigned an L2 type (or the L1 *abuse* type) may reveal new prevalent abuse types that should be added to the taxonomy. For example, *employment* scams were not originally part of the taxonomy but were discovered by examining the reports classified as *scam*, in Section 6.5.4.

6.3.2 Taxonomy and Ground Truth Construction

Since we cannot manually examine the 224,059 descriptions in the BitcoinAbuse dataset, we design a two-step approach to build our taxonomy in Figure 6.1 and the ground truth datasets in Table 6.2. First, we manually analyze 175 descriptions to produce an initial ground truth (gt_initial) with its corresponding initial taxonomy. Then, we cluster the 224K descriptions by the similarity of their text and manually label a fraction of the clusters. The cluster analysis identifies additional prevalent abuse types not in gt_initial, which we add to the taxonomy. It also produces a much larger ground truth (gt_final) with 19,443 descriptions.

Initial ground truth and taxonomy. We built an initial taxonomy by manually labeling 175 descriptions. To select these descriptions, we split the 287K reports in the BitcoinAbuse dataset per year of the report and the six BA types, producing 35 buckets: one per year (from 2018 to 2023) and abuse type (except for Sextortion in 2018 since it was created in 2019). Then, we randomly selected 5 unique descriptions from each bucket, avoiding duplicates.

Two analysts manually classified each of the 175 descriptions by reading their text and freely assigning labels to the descriptions based on their experience. After completing the first labeling round, the analysts discussed the selected labels, normalized their names, and merged duplicates, coming up with the 16 abuse types in white in Figure 6.1. Thus, despite the small

number of descriptions in `gt_initial`, this analysis identifies 16 out of the 22 entries in the final taxonomy. Then, the analysts performed a second joint labeling round to assign one of the 16 identified abuse types to the 175 descriptions.

Clustering. To discover additional abuse types not present in `gt_initial` and to produce a larger ground truth, we cluster the 224K descriptions by the syntactical similarity of their text. We select the *all-mpnet-base-v2* pre-trained embedding from the Hugging Face [91] repository, a general-purpose model that achieves the highest average performance for sentence similarity in the Hugging Face evaluation [92].

To cluster the embeddings, we use *hierarchical agglomerative density-based clustering* (HDBSCAN) [31] because it can find clusters of arbitrary shape and does not require knowing the number of clusters in advance. We use Euclidean distance and set the minimum core point neighborhood size to 2 and the minimum cluster size to 5. The clustering produces 5,265 clusters containing one-third of the descriptions. The largest cluster has 3,814 descriptions. The remaining two-thirds (66.3%) of descriptions end in singleton clusters.

One of the analysts examines the top 150 clusters by size. Most clusters (95) are for *sextortion* campaigns, followed by 9 *giveaway* clusters capturing reports by security vendors or analysts. This process identifies four new abuse types: *deleteransom*, *advancedfee*, *ddos*, and *bomb*. We additionally introduce *ransom* as the parent of *deleteransom*, to capture ransomware reports.

To check the smaller clusters for other previously unknown abuse types, we produce a word cloud [220] for each of the 5,265 clusters using an open-source library [220]. The analyst visually examined the word clouds, selecting 30 clusters to analyze because they contain words that differ from those in the top 150 clusters. However, all 30 clusters corresponded to abuse types already in the taxonomy.

We manually examined 175 singletons selected using the same approach as `gt_initial` (i.e., 35 buckets, 5 descriptions from each). Of the 175 descriptions, 99 (56%) belonged to a fine-grained L3 abuse type, and 26 (15%) were *notabuse*. Among the 50 (29%) descriptions assigned an L1 or L2 abuse type, there were no new abuse types. Thus, the examined singletons correspond to syntactic variations of abuse types already in the taxonomy.

Final taxonomy and ground truth. Figure 6.1 shows the final taxonomy, where the light gray nodes correspond to the 5 abuse types introduced by the cluster analysis. Table 6.2 describes the final ground truth (`gt_final`), which contains all `gt_initial` descriptions, plus those in the clusters manually labeled by the analyst. In total, `gt_final` contains 19,443 descriptions. The top abuse type is *sextortion* with 16,112 (82.8%) descriptions, followed by *extortion* (7.3%), *giveaway* (3.8%), and *spam_fundsrecovery* (0.9%).

Scam Tracker ground truth. We apply the process for generating `gt_initial` to 200 Scam Tracker descriptions (`gt_bbb`). We also clustered all Scam Tracker descriptions, which resulted in only two small clusters. We use `gt_bbb` to validate our classifiers.

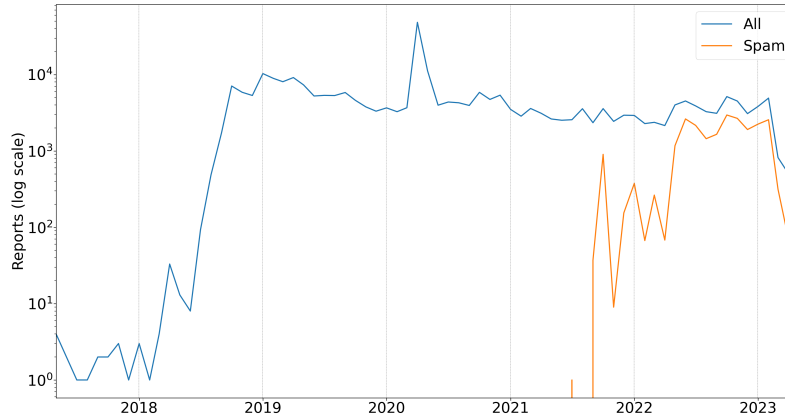


Figure 6.2: Volume of submitted reports and spam reports in BitcoinAbuse (log-scale) per month. Spam first appears in 2021 and dominates all reports since June 2022.

6.4 Data Pollution in Abuse Reporting Services

This section answers RQ1 by analyzing data pollution in abuse reporting services. We focus on BitcoinAbuse because it did not validate the submitted reports. Additionally, we examine which of the identified issues affect Scam Tracker.

6.4.1 Spam Reports

Section 6.3.2 identified two prevalent spam report types in BitcoinAbuse: advertisements for funds recovery services (*spam_fundsrecovery*) and fake terrorism reports (*spam_terrorism*). In this section, we measure the volume of spam reports across the whole BitcoinAbuse dataset.

Spammers may post the same spam report hundreds or even thousands of times. For example, the 174 *spam_fundsrecovery* descriptions in *gt_final* (compared to 16,112 for *sextortion*) appear in 21,070 reports (compared to 17,837 reports for *sextortion*), an average of 121 reports per description (compared to an average of 1.1 for *sextortion*).

To identify *spam_fundsrecovery* reports across all the BitcoinAbuse data, we extract a blacklist of contact indicators from the 174 *spam_fundsrecovery* descriptions in *gt_final*. The blacklist contains 27 domains, 17 email addresses, 12 phone numbers, 10 URLs, and one Telegram handle. Then, we scan the 287K BitcoinAbuse reports using the blacklist. We conservatively match domains so that if the domain summitrecoup.com is in the blacklist, we match any reports containing the string *summitrecoup*. This process identifies 208 (0.1%) *spam_fundsrecovery* descriptions used in 22,180 (9.9%) reports. To identify *spam_terrorism* reports across all the BitcoinAbuse data, we manually examine the 119 descriptions not in *gt_final* that contain the string *terror*. This process identifies 79 (0.03%) *spam_terrorism* descriptions used in 1,500 (0.7%) reports.

The volume of spam identified is a lower bound, as smaller spam types may also exist. Figure 6.2 shows the volume of spam reports and all reports over time. Spam reports first appear in February 2021. Spam has dominated since mid-2022, reaching 75% of all submitted reports between June and December 2022. Submitted reports have declined since

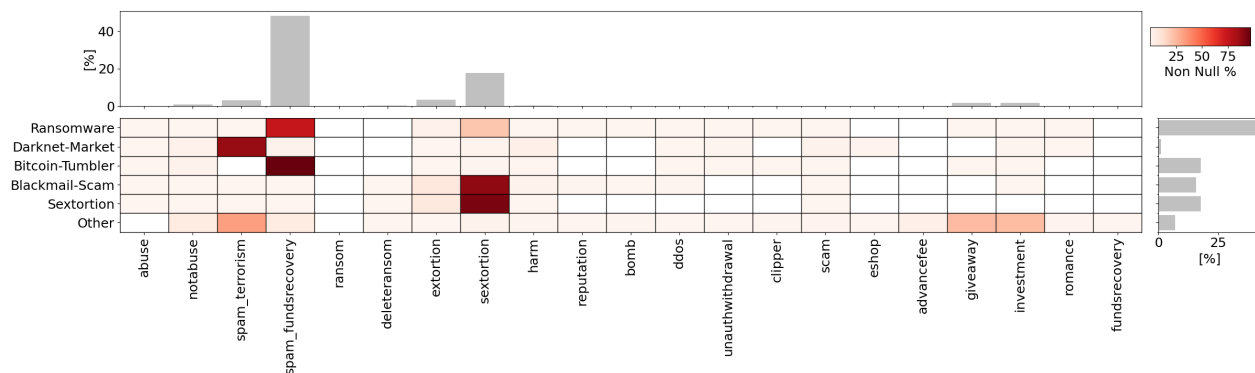


Figure 6.3: Heat-map capturing the fraction of the 43K evaluation reports assigned the BA type on the y-axis and tagged with the taxonomy abuse type on the x-axis. White means the ratio is zero, light orange is a small ratio, and dark red is a high ratio.

February 2023 as BitcoinAbuse started redirecting users to ChainAbuse, eventually stopping the reception of new reports in June 2023.

Our results show that, when no filtering is applied, spam will eventually dominate the submitted reports. In contrast, the Scam Tracker ground truth does not include funds recovery or terrorism spam reports, likely because the manual validation filters most of them from the dataset.

6.4.2 Accuracy of User-Reported Abuse Types

Next, we evaluate the accuracy of user-reported BA types by comparing them with the taxonomy abuse types in `gt_final`. The evaluation is performed on reports because the same description may correspond to multiple reports with different BA types. Thus, we first propagate the taxonomy types from the 19,443 descriptions in `gt_final` to 43,693 reports (15.2% of all 287,823 reports) where those descriptions appear.

Figure 6.3 shows a heat map where the heat captures the fraction of the 43K reports assigned the BA type on the y-axis and tagged with the taxonomy abuse type on the x-axis. White means the ratio is zero, light orange means a small ratio, and dark red a high ratio. The bars on top capture the fraction of the 43K reports assigned to each taxonomy abuse type. The bars on the right capture the fraction of the 43K reports assigned to each BA type.

Spam impact. Four of the six BA types are dominated by spam reports, namely Bitcoin-Tumbler (98%), Darknet-Market (91%), Ransomware (75%), and Other (43%). For these four BA types, any conclusions drawn without filtering out the spam reports would be heavily biased toward the spam reports and thus would be incorrect.

Sextortion. Of the 7,766 reports with the Sextortion BA type, 97.4% are indeed *sextortion*, and 2.5% are *extortion*. Among the 2.5% *extortion* reports, some are likely in reality *sextortion* where the reporter only copied a small part of the received email in the description that did not mention the sexual character of the extortion, making our analysts label it *extortion*.

Thus, the Sextortion BA type correctly captures *sextortion* abuse. On the other hand, the most common misclassification (after excluding spam reports) is 22% Ransomware reports that are instead *sextortion*. A possible reason is that reporters who did not know or did not care about the abuse type just selected the first option in the drop-down list (Ransomware). We conclude that reporters could correctly identify *sextortion* abuse. However, most reporters do not care about selecting the right abuse type, which indicates that this task needs to be automated or performed by human analysts.

Ransomware. `gt_final` has no *ransom* reports, but there are 196 *deleteransom* reports. Of those, 173 (88.3%) possess the correct BA type (Ransomware). However, *deleteransom* is only 1% of Ransomware reports. Thus, the Ransomware BA type is untrustworthy, even after removing spam reports. We conclude that BitcoinAbuse was not commonly used to report ransomware, but when it was (e.g., *deleteransom*), users identified accurately such abuse type.

Blackmail-Scam. This BA type was too broad, covering all taxonomy entries underneath *extortion* and *scam*. Of the 6,830 Blackmail-Scam reports, 92% are *sextortion*. Of these, 19% correspond to reports before the introduction of the Sextortion BA type in February 2019, when Blackmail-Scam was the best option for *sextortion*. In the rest, the reporter may have considered that Blackmail-Scam fit the bill and perhaps did not even realize there was another more specific Sextortion entry further down the list. We conclude that broad abuse types should be avoided as they may cause errors due to overlapping more specific abuse types.

Bitcoin-Tumbler and Darknet-Market. Our taxonomy and ground truth do not have abuse types equivalent to Bitcoin-Tumbler and Darknet-Market, indicating that victims rarely reported such services to BitcoinAbuse. This makes sense as users of mixers and darknet markets may not be inclined to report the services they use. Thus, these BA types are untrustworthy, as all reports in these BA types (even after excluding spam reports) were incorrectly classified by the reporters.

Other. Among the 3,051 Other reports, 43.1% are spam reports, 24.0% are *giveaway*, 23.6% are *investment*, and 9% are scattered across 14 abuse types. The widespread presence of *giveaway* and *investment* scams may indicate that many reporters know how to identify those abuse types. However, since the platform did not offer such types to its users, they opted for the generic type Other. We conclude that it is hard to predict what abuse types victims will report and, thus, which types the online form should offer.

Impact on previous works. Previous works analyzing BitcoinAbuse reports rely on the BA types [37, 27, 176]. Unfortunately, we have shown that user-reported BA types are most often incorrect. Conclusions drawn from the Bitcoin-Tumbler, Darknet-Market, and Ransomware BA types will be incorrect as those BA types are dominated by spam, with the remaining reports being mostly misclassifications. On the other hand, conclusions drawn from Sextortion should be trustworthy as most reporters accurately use that BA type for *sextortion* abuse. Blackmail-Scam and Other are too broad. Blackmail-Scam is dominated by *sextortion*, so conclusions drawn on this BA type should match those of Sextortion.

Scam Tracker places all cryptocurrency abuse reports under the same category. Thus, we cannot evaluate the accuracy of the abuse type selection by analysts.

6.4.3 Benign Addresses Reported

BitcoinAbuse indexed reports by the reported Bitcoin address. Next, we check whether users may report addresses known to be benign, e.g., belonging to services like exchanges. For this, we leverage a variety of sources that tag addresses, such as the open-source GraphSense [84] and WatchYourBack [73] platforms and blockchain explorers [75, 7]. Of the 92,151 reported Bitcoin addresses, 91 (0.1%) are tagged in those sources as internal service addresses or belonging to benign organizations. These are mainly hot wallets used for the daily operations of services and cold wallets used for long-term storage of the service reserves. Of the 91 addresses, 78 belong to exchanges, 5 to mining pools, 3 to the FBI, 2 to payment-processors, 2 to gambling services, and one to a lending provider. These 91 addresses have 780 reports. Of those, 436 (56%) are *spam_terrorism*, claiming an exchange is involved in terrorism and typically reporting a hot wallet of the exchange. Another 123 (16%) are *spam_fundsrecovery* reports, posted on frequently reported addresses where victims are most likely to see them, i.e., the reported address is unrelated to the funds recovery service. Among the other 28% of reports, we observe two main reasons. First, victims may try to track down where their stolen funds are going and end up reporting an internal service address (most often an exchange’s hot wallet) that received the money after the attackers cashed out. Victims should learn to report only addresses they directly sent funds to (i.e., payment addresses), without trying to track down the funds. Because the information in the blockchain is immutable, tracking can be done later by experienced analysts. In other cases, the reporter seems to want to pollute an exchange’s reputation similar to *spam_terrorism*, e.g., one report asks for a donation to a cold wallet of the HTX exchange.

The Scam Tracker online form does not have a field to report cryptocurrency addresses. However, we extracted 96 Bitcoin addresses from the description text. None of those addresses are benign, according to the tagging sources used.

Impact on revenue estimation. Internal service addresses may receive large amounts of bitcoins unrelated to cybercrime. Thus, any revenue estimation that does not exclude them can hugely overestimate cybercrime revenue. To evaluate this impact, we use WatchYourBack [73] to compute the amount of BTC received by each reported address from the beginning of the Bitcoin blockchain until block 838,800 (April 11, 2024). We calculate a *DD-DC* estimation per address, which filters out self-deposits to avoid double counting [78]. We use the BTC conversion rate to USD on the day of each deposit. Of the 92,151 reported addresses, 58,218 (63.2%) have no transactions until April 11, 2024. Of the rest, 21,429 (23.2%) addresses have received at most \$10K, 10,441 (11.3%) between \$10K and \$1M, 1,565 (1.7%) between \$1M and \$100M, and 498 (0.5%) over \$100M. The 33,933 addresses with transactions have collectively received 123,655,891 BTCs, corresponding to \$2,477,761,387,918. When excluding the 91 internal service addresses, the estimation reduces to 48,682,682 BTC (\$950,739,558,245). Thus, even if benign addresses are only 0.1% of reported addresses, their impact on cybercrime revenue estimation is overwhelming. Excluding them reduces the estimation by 60% from

Query 1: Default Prompt

You are a cybersecurity expert with extensive knowledge about scams and abuses. You will help me classify abuse reports given by users, based exclusively on the content of the given TEXT. Do not infer or assume facts that are not described in the TEXT.

The following is a LIST OF DEFINITIONS of abuse classes. Read carefully the list and use it to classify the TEXT at the end, and to answer the QUESTION.

LIST OF DEFINITIONS ### <definition_list>

TEXT: <report_description_text>

QUESTION: Given the LIST OF DEFINITIONS above, classify the TEXT in one of the following classes: <class_list>. Answer only with the name of the class that clearly matches one of these definitions and justify your answer by filling the next JSON structure:

```
{"answer": "", "reasoning": ""}
```

Figure 6.4: LLM prompt. Macros in angle brackets are replaced with abuse type definitions and the description text.

\$2.4 trillion to \$950 billion. The latter is still likely an overestimation since additional internal service addresses unknown to our tagging sources could remain unfiltered.

6.5 Abuse Report Classification

This section answers RQ2 by analyzing different designs for a classifier to automate identifying valid reports and their classification into fine-grained abuse types.

We approach this problem as a multi-class classification that given the text in a report's description returns the more specific (i.e., lowest) applicable abuse type in our taxonomy. If a report has traits of two abuse types (e.g., romance and investment [47]), our classifier selects only one. The alternative would be using multiple Boolean classifiers, one for each abuse type. But, this is problematic. For example, in L1 of our taxonomy, a report should not be abuse and notabuse. Furthermore, double-counting a report in two abuse types is problematic when computing financial impact.

We investigate two different types of classifiers: an unsupervised classifier that uses an LLM (described in Section 6.5.1) and a machine learning (ML) supervised classifier trained using our ground truth (detailed in Section 6.5.2).

6.5.1 LLM-Based Unsupervised Classifier

We propose a novel unsupervised LLM-based classifier that does not require training. Many design options exist to build a LLM-based classifier. We investigate three design aspects: the format of the queries, the sequence of queries to classify a description, and the LLM model to be used. We show that naive baseline designs do not work well, but we propose designs that achieve good accuracy.

Query 2: Basic Prompt

You are a cybersecurity expert with extensive knowledge about scams and abuses. You will help me classify abuse reports given by users, based exclusively on the content of the given TEXT. Do not infer or assume facts that are not described in the TEXT.

TEXT: <report_description_text>

QUESTION: Classify the given TEXT in one of the following abuse classes: <class_list>.

Answer only with the name of the class that clearly matches one of these classes and justify your answer by filling the next JSON structure: {"answer": "", "reasoning": "" }

Figure 6.5: LLM prompt used in the baseline. Macros in angle brackets are replaced with a list of abuse types and the description text. Abuse type definitions are not included.

Query format. The intuition behind an unsupervised LLM classifier is that instead of using a labeled training dataset, we can provide an LLM with a list of natural language definitions for the abuse types and let the LLM interpret the description text to choose the best-matching definition. Queries to the LLM use the prompt in Figure 6.4, which instructs the LLM to take the role of a cybersecurity expert and contains three input macros that are replaced with the description to be classified (<report_description_text>), the list of abuse types (<class_list>), and the abuse type definitions (<definition_list>). Capitalized words (e.g., TEXT, QUESTIONS) are text references within the prompt. The prompt instructs the LLM to avoid assuming facts not mentioned in the description to minimize hallucinations [97]. The question at the end directs the LLM to output (in JSON format) both the selected class (i.e., abuse type) and an explanation of the reasoning for the selection. Prompting the model to generate its *reasoning* along with the answer is a common prompting technique [218, 224] that can improve the accuracy of the provided answer and help understand why the model selected the output abuse type.

While constructing the abuse taxonomy, the analysts generated an initial list of abuse type definitions. Those initial definitions were refined by classifying a small subset of descriptions in the training dataset of the supervised classifier (see Section 6.5.2). The analysts used the classification results and the explanation in the LLM answers to reduce overlaps between the definitions. Table 5 in the Appendix details the definitions used by the LLM-based classifiers.

To analyze the importance of the definitions, we analyze a simplified prompt that does not include the abuse definitions as a baseline, as shown in Figure 6.5. It only lists the names of the abuse types and lets the LLM interpret those based on its pre-existing knowledge.

Query approaches. We test three approaches to query the LLM. Regardless of the approach used, queries follow the format in Figure 6.4. Differences between the approaches are in the number of queries to classify a description and in the definition list provided in each query.

(a) *One-query.* This approach queries the LLM once per description. Each query provides all the definitions and asks the LLM to select the most appropriate abuse type for the description. This approach minimizes the number of queries, making it the fastest and reasonably inexpensive. But, it forces the LLM to distinguish abuse subtypes (e.g., *investment*) from their parents (e.g., *scam*), which may be challenging as they semantically overlap.

(b) *Top-to-bottom*. This approach classifies the input description at each taxonomy level from top to bottom. It performs between one and three queries per description, making the classification more specific with each query. An L1 query is performed for all descriptions by providing only the *abuse* and *notabuse* definitions. If the L1 label is *notabuse*, the classification outputs *notabuse*. Otherwise, it queries the LLM again with the definitions of the four L2 abuse types, plus an additional *tbb_other* class that states that the description does not match the other definitions. If the L2 label is *tbb_other*, the classification outputs *abuse*, i.e., since none of the more specific children types matches, it returns the parent type. Otherwise, it performs a final query using only the child definitions for the assigned L2 label plus the additional *tbb_other* class. For example, if the L2 label was *extortion*, only the 5 *extortion* children and *tbb_other* are included in the query. The returned L3 label is output as the description’s abuse type. The top-to-bottom approach eases the distinction between a parent and its children, but it is more expensive in terms of LLM queries. Most importantly, errors at the top levels accumulate, i.e., outputting an accurate label may require correctly classifying the description up to three times.

(c) *Bottom-to-top*. This approach classifies the input description at each taxonomy level, from bottom to top. An L3 query is performed for all descriptions. It provides the 13 leaf definitions plus an additional *btt_other* class that states that the description does not match the given definitions and may not even describe an abuse. If the L3 label is not *btt_other*, the classification outputs the L3 label. Otherwise, an L2 query is performed using the four L2 definitions and the additional *btt_other*. If the L2 label is not *btt_other*, the classification outputs the L2 label. Otherwise, a final L1 query is performed using only the *abuse* and *notabuse* definitions, and the classification outputs the assigned L1 label. This approach is typically the most economical. While it performs more queries than the one-query approach, the queries are smaller as only a subset of definitions is included in each query. However, it requires distinguishing more classes compared to the top-to-bottom approach, which may impact the accuracy.

Models. We evaluate six LLM models. We download and run locally two open models from HuggingFace [132]: Meta-Llama-3-70B-Instruct (llama3 for short) and Meta-Llama-3.1-70B-Instruct (llama3.1). We also evaluate four commercial models from OpenAI accessible via API: the older and cheaper gpt-3.5-turbo-0125 (gpt-3.5), the more recent and more expensive gpt-4-0125-preview (gpt-4), and two faster models with intermediate cost gpt-4o-2024-05-13 (gpt-4o) and gpt-4o-mini-2024-07-18 (gpt4-4o-mini).

Classifier evaluation. Table 6.3 summarizes the classification results for each classifier design on a balanced testing dataset with 1,049 descriptions (detailed in Section 6.5.2). For this, we use both weighted average (classes are weighted by the number of samples in the class) and macro average (all classes have the same weight) scores. We first compare the baseline approach, which does not include the abuse type definitions in the prompt, with the three query approaches, all using the gpt-3.5 model. The results show that the baseline works worst, achieving a weighted F1 score of 0.42, illustrating that a naive LLM-based design does not work and the importance of the definitions for the classification. Among the three query approaches, top-to-bottom works best, achieving a weighted F1 score of

Approach	Definitions	Model	Weighted Average			Macro Average			Cost		
			Prec.	Rec.	F1	Prec.	Rec.	F1	Queries	Tokens	USD
one query	✗	gpt-3.5	0.47	0.47	0.42	0.27	0.27	0.20	1,049	416,295	\$0.26
one-query	✓	gpt-3.5	0.57	0.54	0.52	0.46	0.42	0.33	1,049	1,927,752	\$1.02
bottom-to-top	✓	gpt-3.5	0.65	0.67	0.60	0.59	0.65	0.56	1,126	1,461,260	\$0.79
top-to-bottom	✓	gpt-3.5	0.85	0.82	0.81	0.63	0.72	0.65	2,871	1,993,112	\$1.13
top-to-bottom	✓	llama3	0.74	0.70	0.68	0.71	0.77	0.68	1,719	-	-
top-to-bottom	✓	llama3.1	0.83	0.78	0.77	0.68	0.65	0.59	2,698	-	-
top-to-bottom	✓	gpt-4o-mini	0.84	0.83	0.82	0.79	0.80	0.76	2,837	2,074,957	\$0.40
top-to-bottom	✓	gpt-4o	0.95	0.92	0.92	0.79	0.73	0.71	2,830	2,090,951	\$12.76
top-to-bottom	✓	gpt-4	0.96	0.95	0.95	0.83	0.82	0.80	2,901	2,218,064	\$28.08

Table 6.3: LLM-based classifier results on 1,049 BitcoinAbuse descriptions. Our best design in the bottom row more than doubles the F1 score compared to the baseline in the top row.

0.81, followed by bottom-to-top (0.60) and one-query (0.52). Since top-to-bottom works best, we compare the four LLMs using that approach. The results show that gpt-4 is the best-performing model, achieving a weighted precision of 0.96, recall of 0.95, and F1 score of 0.95. The macro average results (where abuse types have the same weight) are lower for the best-performing top-to-bottom approach using gpt-4, achieving a macro precision of 0.83, recall of 0.82, and F1 score of 0.80. The lower macro scores are due to errors in infrequent abuse types that disproportionately impact this average. For example, *fundsrecovery*, *abuse*, and *unauthwithdrawal* have only 1–2 samples in the testing dataset, so their results are not very representative but greatly affect the macro average scores.

The open Llama models are the cheapest, as we run them locally. But, they achieve the worst weighted average. The cost of the OpenAI API depends on the number of queries performed, the total size (measured in tokens) of the input and output text (including the reasoning), and the model used [149]. The cost difference between GPT models is significant. Both gpt-4 models are an order of magnitude more costly than gpt-3.5 and gpt-4o-mini, and the best-performing gpt-4 is twice as expensive as gpt-4o. The most expensive model better interprets the input descriptions, providing better classification results.

6.5.2 Supervised Classifier

We also build a supervised classifier based on transfer learning as a baseline. We select the pre-trained DistilBERT model [181] available from Hugging Face [91] and fine-tune it for 5 epochs with the following default hyper-parameters: `learning_rate=2e-5` and `weight_decay=0.01`.

Supervised classifiers require labeled data for building training and evaluation datasets. For this, we leverage our labeled `gt_final` dataset, detailed in Table 6.2. `gt_final` has two challenges. First, it is unbalanced towards *sextortion*, which provides 86.3% of the labeled descriptions. For this reason, we experiment with two dataset-building approaches. First, we apply a 70/30 split on the whole `gt_final` dataset, generating unbalanced training (d_u^t with 13,609 samples) and evaluation (d_u^e with 5,833 samples) datasets. Second, we build balanced training and evaluation datasets by selecting from each abuse type 70% of the samples labeled with that type, except for *sextortion* and *extortion* which we downsize by selecting 1/20 samples and 1/2 samples respectively, with the goal of having a final number of samples similar to the next larger abuse type. The balanced training dataset has 2,446 samples (d_b^t) and the balanced evaluation dataset has 1,049 samples (d_b^e). Second, some abuse types in `gt_final` have very

Model	Weighted Average			Macro Average		
	Prec.	Rec.	F1	Prec.	Rec.	F1
distillbert-17-unb	0.99	1.00	0.99	0.79	0.79	0.79
distillbert-17-bal	0.99	0.99	0.99	0.80	0.80	0.80
distillbert-14-unb	1.00	1.00	1.00	0.76	0.75	0.75
distillbert-14-bal	0.99	0.99	0.99	0.72	0.75	0.73

Table 6.4: Accuracy for the supervised classification models.

Classifier	Weighted Average			Macro Average		
	Prec.	Rec.	F1	Prec.	Rec.	F1
distillbert-17-unb	0.66	0.51	0.52	0.30	0.26	0.21
distillbert-17-bal	0.63	0.58	0.53	0.38	0.40	0.29
distillbert-14-unb	0.64	0.59	0.55	0.27	0.25	0.20
distillbert-14-bal	0.73	0.57	0.51	0.41	0.33	0.25
top-to-bottom (llama3)	0.80	0.69	0.70	0.34	0.43	0.34
top-to-bottom (llama3.1)	0.86	0.75	0.76	0.44	0.55	0.44
top-to-bottom (gpt-3.5)	0.79	0.62	0.61	0.28	0.39	0.26
top-to-bottom (gpt-4o-mini)	0.71	0.69	0.69	0.27	0.39	0.30
top-to-bottom (gpt-4o)	0.88	0.80	0.80	0.45	0.47	0.42
top-to-bottom (gpt-4)	0.92	0.87	0.89	0.67	0.76	0.70

Table 6.5: Classification results on Scam Tracker ground truth dataset with 199 labeled descriptions.

few samples. In particular, we have no samples for *ransom* and less than 10 samples for four classes (*unauthwithdrawal*, *eshop*, *abuse*, *fundsrecovery*). Thus, we also build a classifier with 14 classes that removes the four under-represented classes, as they do not have enough samples to train a classifier.

Table 6.4 compares the four supervised models on the balanced testing dataset. The top two models have 17 classes trained with unbalanced (distillbert-17-unb) and balanced (distillbert-17-bal) datasets. The bottom two models have 14 classes trained with unbalanced (distillbert-14-unb) and balanced (distillbert-14-bal) datasets. We observe that down-sampling has almost no impact and that the samples of abuse types for which the classifier is not trained still heavily impact the macro average F1. The best classifier is distillbert-17-bal which achieves a nearly perfect weighted average of 0.99 and up to 0.80 macro average F1 score. The results of the supervised classifier are so good that we believe it may be overfitting the data. To check this, we next evaluate all classifiers on gt_bbb.

6.5.3 Scam Tracker Classification

One challenge with classifiers is how well they generalize to datasets different from those used to build them. To examine this issue, we perform an out-of-distribution experiment where we apply all classifiers on gt_bbb. One of the 200 descriptions exceeds the maximum token length, so we removed it, leaving 199 samples. As shown in Table 6.5, the best classifier is the LLM-based with top-to-bottom and gpt-4, which achieves a weighted average precision of 0.92, a recall of 0.87, and an F1 score of 0.89. The F1 score is only 6 percentile points below that achieved on the BitcoinAbuse testing dataset, indicating that the model generalizes well. In contrast, the supervised classifiers perform worst with F1 scores of 0.51–0.55, showing that they were overfitting in the BitcoinAbuse dataset.

Most errors happen due to *investment* reports labeled as *advancefee* by the classifier. Misclas-

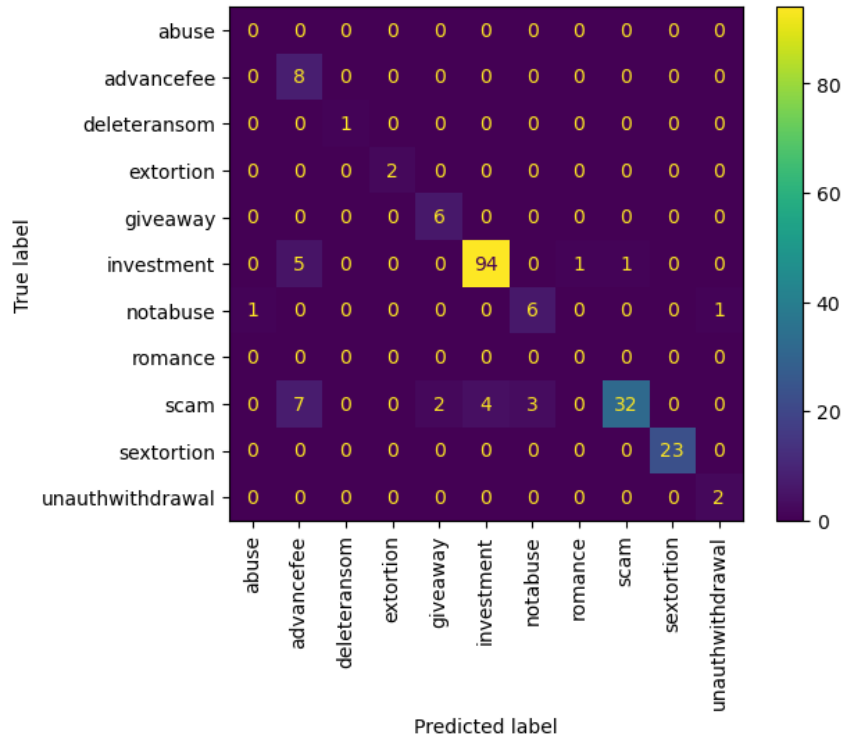


Figure 6.6: Confusion matrix of the top-to-bottom classification using gpt-4 on the Scam Tracker dataset of 199 samples.

sifications happen because the *investment* reports mention upfront fees, e.g., “[...]Then when trade happened he stated that there will be a fee that I have to pay for taxes upfront[...]”. Indeed, most investment scams request additional fees when the user wants to withdraw its profits. The difference with *advancefee* is subtle in that in an *investment* scam, the victim first makes an investment deposit, and the fee is only requested when the victim attempts to withdraw the investment profits. Instead, in *advancefee* scams, the fee is requested prior to the victim making any deposit. Our *advancefee* definition does not explicitly capture this timing requirement. Figure 6.6 shows the confusion matrix for the top-to-bottom gpt-4 classifier.

6.5.4 Updating the Taxonomy and Classifier

Scam Tracker bundles all cryptocurrency abuse reports into one category, preventing comparing finer-grained cryptocurrency abuse types. To address this issue, we apply the top-to-bottom gpt-4 classifier to entire Scam Tracker dataset. We exclude two descriptions that are too large to be classified by gpt-4. Column All in Table 6.6 captures the number of reports of each type identified. 85% of the reports are for scams with half being for *investment* scams. Despite its name, Scam Tracker also contains small percentages of extortions (5.4%) and unauthorized withdrawal (2%) reports. While there is only one *fundsrecovery* report in gt_final and none in gt_bbb, the classifier found 16 across the Scam Tracker dataset. We manually validated that these reports are correctly classified. This shows that given a good abuse definition the LLM-based classifier can correctly classify infrequent abuse types.

Abuse	Reports		Loss		
	All	w/Loss	Total	Avg.	Med.
investment	1,161 (50%)	957 (61%)	\$22,502,025	\$19,382	\$1,300
scam	452 (19%)	243 (16%)	\$3,072,803	\$6,798	\$100
advancefee	175 (8%)	127 (8%)	\$1,191,216	\$6,807	\$226
notabuse	140 (6%)	69 (4%)	\$1,105,560	\$7,897	\$0
sextortion	104 (4%)	0 (0%)	\$0	\$0	\$0
giveaway	75 (3%)	38 (2%)	\$149,831	\$1,998	\$20
romance	57 (2%)	34 (2%)	\$2,181,882	\$38,279	\$1,500
abuse	44 (2%)	21 (1%)	\$211,436	\$4,805	\$0
unauthwith.	42 (2%)	27 (2%)	\$224,270	\$5,340	\$220
eshop	32 (1%)	28 (2%)	\$99,226	\$3,101	\$600
fundsrecovery	16 (1%)	12 (1%)	\$91,077	\$5,692	\$1,710
extortion	16 (1%)	2 (0%)	\$2,552	\$160	\$0
reputation	3 (0%)	0 (0%)	\$0	\$0	\$0
harm	2 (0%)	2 (0%)	\$1,100	\$550	\$1,000
All	2,319 (100%)	1,560 (67%)	\$30,832,979	\$13,297	\$500

Table 6.6: Scam Tracker reports classified into each abuse type, the subset reporting a financial loss, the total loss in reports of the type, and the average and median loss per report.

As described in Section 6.3.1, we have designed our hierarchical taxonomy to capture previously unknown abuse types in the L2 types and the L1 *abuse* type. Thus, we manually examine the reports assigned those types (i.e., 452 *scam*, 44 *abuse*, 42 *unauthwithdrawal*, and 16 *extortion*) to check for new abuse types. Among the 452 *scam* reports, we identify a new abuse type for *employment* scams. We produced a definition for this abuse type and ran the top-to-bottom gpt-4 classification on the Scam Tracker descriptions again. The new classification results identify 56 *employment* descriptions. Manual verification identifies 54 as true positives and two as false positives. We also examine all descriptions containing related keywords (e.g., *job*, *employment*), finding no false negatives. This process illustrates the ease of extending our taxonomy and LLM-based classifier with new abuse types.

6.5.5 Spam Detection

Next, we check how accurately our classifier identifies spam reports. For this, we leverage the 287 spam descriptions identified in Section 6.4.1 across the entire BitcoinAbuse dataset. We classify those descriptions using the L1 query of the top-to-bottom gpt-4 classifier. The classifier achieves a precision of 1.0, a recall of 0.99, and an F1 score of 0.99. It correctly identifies 284 (99%) descriptions as *notabuse*. The other three descriptions were considered *spam_fundsrecovery* in Section 6.4.1 because their text contains contact IOCs from funds recovery services. However, the classifier correctly determines that they are valid abuse reports where victims describe how they were scammed by a funds recovery service (i.e., their correct abuse type is *fundsrecovery*).

We also examine the 140 Scam Tracker reports classified as *notabuse*. Among those, we identify 9 *spam_fundsrecovery* reports (0.4% of all Scam Tracker reports) that Scam Tracker’s manual validation failed to filter but are correctly identified as invalid reports by our classifier.

The results show that our classifier can accurately filter spam reports, including some missed by manual validation. Furthermore, our LLM-based classifier correctly distinguishes useless *spam_fundsrecovery* advertisements from valid *fundsrecovery* scam reports by victims.

6.6 Abuse Type Revenue Estimation

This section answers RQ3 by analyzing the financial impact of different abuse types. Two approaches exist for estimating the financial impact of cryptocurrency abuse. The first one adds losses disclosed by victims in abuse reports. Unfortunately, victims often do not report the abuse suffered [193, 3, 206], making this estimation a conservative lower bound. The alternative is analyzing transactions in the blockchain ledger that deposit funds to reported malicious addresses. This approach enables observing deposits from victims who did not report an attack. However, it may over-estimate due to benign service addresses (see Section 6.4.3) and if addresses are used in multiple abuse types.

6.6.1 Victim-Reported Financial Losses

Scam Tracker reports contain an optional field with victim-reported losses in US dollars. While Scam Tracker bundles all cryptocurrency abuse reports in one category, we can use the classification results on the Scam Tracker descriptions, introduced in Section 6.5.4, to compare the financial impact of different cryptocurrency abuse types.

Table 6.6 summarizes the results. Overall, 67% of the cryptocurrency abuse reports include a financial loss, the total reported loss was \$30.8M, with an average loss of \$13K and a median of \$500 per report. But, there are significant differences across abuse types. Financial loss is reported in 61% of *investment* reports, nearly four times larger than any other abuse type. Thus, *investment* scam targets are most likely to become victims. The highest median loss is for *fundsrecovery* scams (\$1,710), followed by *romance* scams (\$1,500), and *investment* scams (\$1,300). The largest average is for *romance* (\$38,279) and *investment* (\$19,382) as these categories contain the reports with the largest losses. In contrast, extortions seem particularly inefficient at obtaining revenue from victims. For example, no losses are reported for *sextortion* and *reputation*. Extortions typically leverage massive email campaigns to reach large numbers of targets to compensate for their low conversion rate. Another category with small median losses is *giveaway* (\$20) where victims choose how much to send, and may conservatively select small test amounts. Of the 9 *spam_fundsrecovery* advertisements underneath *notabuse*, 7 report a loss, for a total of \$548,457. Thus, spam reports can also pollute financial estimations by reporting fictitious losses.

6.6.2 Estimation from Address Deposits

We propose a three-step approach to estimate the revenue obtained by each abuse type from the transactions in the blockchain. First, we use our LLM-based classifier to classify reports into abuse types. Second, for each address that is not a benign address, we assign an abuse type to the address through a majority vote on the classification results on the reports for the address. Finally, we use the WatchYourBack platform to perform a DD-DC estimation [73] of the funds received by addresses of the same type. We convert the Bitcoin estimation to US dollars using the conversion rate on the day of each deposit.

Our tagging procedure assigns an abuse type to an address by performing a majority vote on the classifier results for its reports. Given an address, it iterates on all the reports for the

Abuse	Rep.	Tagged Addresses		Revenue	
		All	w/Dep.	BTC	USD
investment	209	202	201	8,060.2558	\$199,717,207
scam	323	236	229	4,392.8819	\$121,183,928
unauthwithdrawal	60	57	57	2,729.4449	\$65,097,775
abuse	106	49	48	1,651.4201	\$20,317,573
giveaway	103	102	102	1,135.9124	\$11,151,614
romance	17	17	17	359.4312	\$10,951,634
advancefee	46	42	42	390.6354	\$10,449,521
extortion	1,979	915	462	362.5466	\$7,099,341
eshop	26	25	25	80.4185	\$2,610,870
sextortion	2,141	767	698	285.4095	\$2,474,572
fundsrecovery	5	5	5	50.8731	\$1,542,469
ransom	111	62	35	19.7539	\$453,233
clipper	8	7	7	10.0543	\$350,226
harm	2	1	1	4.5950	\$271,174
deleteransom	49	24	24	18.9006	\$174,852
ddos	1	1	1	0.0865	\$4,558
reputation	27	4	4	0.1691	\$1,533
bomb	2	2	2	0.0000	\$0
All abuse	5,215	2,518	1,960	19,552.7888	\$453,852,080

Table 6.7: Revenue estimation for 1,960 addresses tagged using the classification of 5K BitcoinAbuse descriptions.

address. For each report, it obtains the abuse type output by the classifier for the description text. If the report’s abuse type is *notabuse*, the report is ignored. Otherwise, it increases a counter for the abuse type. After processing all reports for the address, the abuse type with the highest counter is output as the abuse type for the address. If all reports were *notabuse*, no abuse type is output.

BitcoinAbuse estimation. Classifying all 224K BitcoinAbuse descriptions would cost roughly \$5,000, a budget we currently do not have. Instead, we classify 5K descriptions using the LLM-based top-to-bottom with gpt-4 classifier, with a cost of \$109. Those 5K descriptions are used in 7,322 reports for 3,668 addresses. We exclude 33 benign addresses and apply our tagging procedure to the rest, generating tags for 2,518 (69%) addresses. The rest of the addresses are not tagged because their reports are classified as *notabuse*. Table 6.7 presents the estimation for the 1,960 tagged addresses with at least one deposit. Due to the fluctuating BTC conversion rate, it is possible for abuse types with smaller BTC revenue to have larger USD revenue (e.g., *eshop* vs *sextortion*). Similar to Scam Tracker, *investment* brings the most profit, and extortions like *bomb*, *reputation*, or *ddos* are fairly ineffective. However, we identify \$2.4M *sextortion* revenue due to this abuse type having the largest number of tagged addresses (698). This again points to extortions having a low conversion rate, but obtaining revenue through massive spam campaigns.

Scam Tracker. Although Scam Tracker does not have a field to report cryptocurrency addresses, there are 71 reports mentioning 96 Bitcoin addresses in the description text. Using these 71 reports, we can compare the victim-reported losses with the estimation from the deposits to the mentioned addresses. Table 6.8 shows the classification of the reports and addresses, the losses victims disclosed in the reports, and the estimation from the address deposits applying the same approach used for BitcoinAbuse. While victims disclosed losses of \$520K in the 71 reports, the revenue computed from the deposits to the 96 addresses

Abuse	Rep.	Addr.	Rep. Loss	BTC	USD
unauthwithdrawal	1	1	\$76,000	1,321.1525	\$6,608,765
investment	27	38	\$129,160	192.8219	\$6,479,591
advancefee	5	20	\$233,455	30.7403	\$1,102,368
scam	6	6	\$8,360	19.0552	\$767,115
giveaway	5	5	\$73,090	4.4790	\$177,599
notabuse	3	3	\$300	1.8235	\$28,414
reputation	1	1	-	0.3088	\$5,907
sextortion	20	19	-	1.1376	\$5,634
extortion	2	2	-	-	-
abuse	1	1	-	-	-
All	71	96	\$520,365	1,571.5188	\$15,175,393

Table 6.8: For a set of 71 Scam Tracker reports mentioning 96 Bitcoin addresses, comparison of victim-reported losses (Rep. Loss) with revenue computed from the address deposits (BTC, USD).

is \$15.1M, 29 times higher. The extreme case is one *unauthwithdrawal* address with \$76K victim-reported losses that has received \$6.6M dollars, i.e., victim-reported losses capture only 1.1% of the total revenue. While no losses are reported for *reputation* and *sextortion*, the estimation from the blockchain shows that those addresses indeed received payments from other victims (\$5.6K–\$5.9K). Our results show that the victim-reported losses provide a very conservative lower bound of the revenue achieved by the reported cybercriminal operations, with the real revenue being up to 29 times higher.

6.7 Discussion

This section discusses our results, limitations, and provides recommendations.

ChainAbuse. BitcoinAbuse has been acquired by ChainAbuse [208] and the BitcoinAbuse reports have been added to the ChainAbuse database. The acquisition shows that despite the pollution issues we analyze, the BitcoinAbuse abuse reports were considered valuable. To check whether the BitcoinAbuse pollution was cleaned in ChainAbuse, we queried in November 2024 the 91 benign addresses through the ChainAbuse webpage. ChainAbuse had removed all reports for 63 addresses, but another 28 still appear in their database. The reports for those addresses still include some funds recovery and terrorism spam reports. This shows that identifying benign addresses and spam reports is challenging even for specialized commercial services.

Addressing pollution. We show that, if left unchecked, spam reports will eventually flood an abuse reporting service like BitcoinAbuse. Instead, Scam Tracker’s manual validation prevented spam reports from flooding the database, but 0.4% of reports are spam that was not filtered, possibly due to analyst fatigue or inconsistency. We also do not observe benign addresses in Scam Tracker. However, we believe this is not due to successful filtering (which is hard even for specialized services like ChainAbuse), but due to Scam Tracker not providing a field for reporting cryptocurrency addresses and only 96 addresses appearing in 71 reports.

Our recommendation to abuse reporting services is to deploy two filtering steps for useless reports. One filter should examine the description text. This filtering scales best using

a classifier such as ours, which also avoids analyst fatigue. The other filter should check reported addresses against open [84, 73] or commercial databases of tagged addresses. While incomplete, those databases help identifying well-known benign addresses. The identified problematic reports could be rejected, ignored, or flagged to be analyzed by a human.

We also recommend revisiting the design of online reporting forms to avoid some data pollution. Some ChainAbuse reports state that the reported address is bogus and only added because the reporter is forced to include an address even if he wants to report a URL. And, some Scam Tracker reports have a cryptocurrency address in a wrong field since there is no field for addresses.

Evasion. Even if our classifier can accurately filter spam and other reports that do not describe abuse, an attacker could craft fake reports that look legit and cannot be identified by the classifier, or even a human analyst. Attackers could even leverage LLMs for this task. Identifying such fake reports is an open challenge for crowd-sourced datasets.

Abuse type classification. Abuse reports are fundamental to compare abuse types and their financial impact, but this requires accurate abuse type classification. Scam Tracker does not sub-classify cryptocurrency reports and BitcoinAbuse and ChainAbuse leave the classification to the reporters. We show the latter is problematic. Even if spam reports are filtered, reporters may not care about the classification or may lack the needed experience. We recommend that user-reported abuse types be replaced by a classifier such as ours that automates and homogenizes the classification.

Taxonomy completeness. Previous works focus on a single cryptocurrency abuse type, while our taxonomy covers 19. Two abuse types analyzed in prior work that we have not observed are cryptojacking that typically mines Monero [201] and initial coin offering (ICO) scams that mostly affect Ethereum [205]. Others appear with different names such as Ponzi schemes [9] (*investment*) and tech support scams [2] (*fundsrecovery*). We have designed our taxonomy and classifier to be easily extensible. As illustrated in Section 6.5.4, adding a new abuse type simply requires a new definition.

Generalization. We focus on Bitcoin, the most used blockchain and also the one with the most abuse. For example, ChainAbuse lists 261K reported Bitcoin addresses, eight times more than the second-ranked Ethereum with 33K addresses. Still, we believe our classifier could be generalized to handle reports on other cryptocurrencies, as well as non-cryptocurrency abuse reports (e.g., phishing, credit card scams). Furthermore, our classifier could be applied to less noisy datasets, e.g., victim complaints directly filed with LEAs. Since LEAs may not be able to submit complaints to commercial LLMs, we could deploy our classifier on-site using the open models, albeit at a reduced accuracy.

Cost. The cost of commercial models may currently limit the classification of large datasets. Luckily, the cost drops quickly as newer models appear, e.g., gpt-4o-mini costs an order of magnitude less than gpt-4.

6.8 Conclusion

We have performed the first analysis on data pollution in cryptocurrency abuse reporting services. Using 289K abuse reports from two popular services, we have answered three important research questions. First, we have measured the extent and impact of data pollution. We have developed a taxonomy of 19 popular abuse types and have built a ground truth with 19,443 abuse descriptions. Using them, we have shown that, if left unchecked, spam reports will eventually flood an abuse reporting service; that a small fraction of reported addresses may be benign (0.1%) but those are responsible for a majority (60%) of the deposited funds; and that user-reported abuse types cannot be trusted. Second, we have shown that the identification of useful reports and their classification into abuse types can be automated. We have proposed a novel LLM-based classifier that, on previously unseen data, achieves an F1 score of 0.89 compared to 0.55 for a baseline supervised classifier. And, it identifies spam reports with a 0.99 F1 score. Finally, we have compared the financial impact of different cryptocurrency abuse types showing that victim-reported losses largely under-estimate, with the revenue estimation from deposit transactions being 29 times higher. We show that *investment* scams have the highest financial impact, that *fundsrecovery* and *romance* scams have the highest revenue per report, and that extortions have a very low conversion rate.

Chapter 7

Conclusions & Future Work

7.1 Conclusions

In this dissertation, we have proposed novel techniques and tools to extract cybercrime intelligence from Bitcoin. We proposed on-chain techniques for analyzing the data in the blockchain and off-chain techniques for extracting information from threat and abuse reports that enable such analysis, e.g., cybercrime-related addresses, their illegal activities and groups. We have developed back-and-forth exploration, a transaction tracing technique for uncovering financial relationships and attribution points of cybercrime groups. We implemented our technique in a tool called WatchYourBack, which we use to reveal relations of diverse cybercrime groups that abuse Bitcoin, including 30 malware families and a mixer used for money laundering.

We have performed the first systematic comparison of Bitcoin revenue estimation methodologies to identify the most accurate. We implemented 15 methodologies in WatchYourBack, which we used to compare the financial impact of diverse cybercrime groups and abuse types. We showed that some methodologies introduce massive overestimation instead of underestimating the results. We generalized a technique for identifying and filtering out self-deposits that reduced overestimation by 75% on average. We additionally proposed two techniques for uncovering (possibly-)all addresses of the DeadBolt ransomware, shedding light on the problem of address coverage and its impact on the estimation for the first time.

Finally, we have proposed address collection and classification techniques that contribute towards automating tag generation. We have built a threat intelligence platform for collecting IOCs in CTI reports, including cryptocurrency addresses related to cybercrime. We have proposed a novel methodology for evaluating IOC extraction. We have collected data from two online services where victims report cryptocurrency abuse to build an abuse report classifier, which interprets abuse reports written in natural language to give them an abuse type. Our classifier allowed us to filter pollution in crowd-sourced data, label the reported addresses with a more accurate abuse type, and produce refined economic analyses of cybercrime.

Bitcoin tracing. Tracing Bitcoin transactions permits identifying financial relationships between cybercrime groups and the services they abuse, which is crucial for attribution. At the same time, it allows locating more addresses controlled by cybercriminals and possible connections to other malicious operations. As shown with some malware families, tracing the seeds forward may miss attribution points. Plus, malicious online wallets cannot be traced forward. Thus, we have proposed back-and-forth exploration to search for relationships missed by forward-only explorations. Our technique traces coins backward from addresses and transactions, uncovering more attribution points. Due to the predominance of payments coming from services (e.g., payments from victims) and the low coverage of tag databases, backward tracing is challenging. To avoid exploring service-owned addresses, we complemented our tag database with a machine learning (ML) exchange classifier, minimizing the instances of graph explosion. To assess our tracing technique, we have proposed operation oracles, classifiers that detect addresses that abuse Bitcoin transactions to signal information to their botnets. We leveraged operation oracles to confirm that our approach identifies previously unknown malicious addresses. To show the benefits of our back-and-forth exploration, we used it to study the financial relationships of 30 malware families, uncovering relations for 93% of the families with at least 70 services, 80% being exchanges and gambling sites. Those services are good attribution points due to KYC/AML policies enforced by local governments. We discover new attribution points for 13% of the families, previously unknown C&C signaling addresses used by malware operators to control botnets, and financial relations between cybercrime groups (e.g., payments to buy malware tools, and the use of services like mixers for money laundering).

Revenue estimation. Several methodologies exist to estimate the revenue obtained by cybercriminals using information from the blockchain. However, no systematic comparison existed to understand the impact of using different methodologies. For example, previous works estimating the revenue of the same ransomware operation obtained very different results. Thus, we systematically analyzed 15 revenue estimation methodologies used by previous works. To quantify the impact of the selected methodologies, we collected 30,424 addresses controlled by cybercrime and estimated their revenue. We showed that some methodologies overestimate the revenue by incorrectly identifying service addresses as part of the operation or by double counting deposits: our double-counting (DC) filter reduced the estimations by 75% on average. We also showed that underestimation due to lack of coverage is a common issue as attackers actively evade co-spending clustering techniques. We have proposed two estimation methodologies that include the DC filtering, one based exclusively on the seeds, considered a lower bound, and another that uses address expansion and thus can be used as a tighter bound of the revenue obtained. We used both to compare 6 abuse types and 141 cybercrime groups, showing that ransomware dominates with revenues 10 times larger than other abuse types included in the comparison. We have built an operation oracle to uncover the addresses controlled by the DeadBolt ransomware family, showing that operation oracles can adapt to different use cases. Our findings allowed us to quantify the impact of the lack of address coverage for the first time. We found that the real revenue may be 39 times higher than estimated from public sources.

Automating tag generation. Finally, we contributed to the automation of the tag generation process, which is vital for attribution and, in general, for blockchain analyses. We divided this task into two: address identification and address classification. We first developed GoodFATR, a platform to extract cybercriminal-related indicators from public CTI reports, including addresses from 9 cryptocurrencies. We used GoodFATR to collect 472,891 threat reports from 6 sources, extract 978,151 indicators, and identify 618,217 IOCs. We additionally proposed a majority voting methodology that does not need ground truth data to evaluate indicator extraction tools. We used it to compare IOCSEARCHER, GoodFATR’s IOC extraction tool, against 7 publicly available IOC extraction tools on 4,420 threat intelligence reports. We showed that IOCSEARCHER is the most accurate tool on 11 of 13 indicator types supported by different tools. For the classification task, we designed and implemented an abuse report classifier. Given a cryptocurrency abuse report written by a victim, our classifier assigns it one of 19 abuse types. We have collected 289K abuse reports from two popular services to evaluate our classifier and answer three important research questions. First, we have measured the extent and impact of data pollution in abuse reporting services. We have developed a taxonomy of 19 popular abuse types and have built a ground truth with 19,443 abuse descriptions to show that spam reports could eventually flood abuse reporting services, that a small fraction of benign addresses reported are responsible for 60% of the deposited funds, and that abuse types reported by users are untrustworthy. Second, we have shown how to automate both the filtering of spam reports to identify useful reports and their classification into abuse types. Our novel LLM-based classifier achieved an F1 score of 0.89 on previously unseen data, compared to 0.55 for a baseline supervised classifier. It identified spam reports with an F1 score of 0.99. Finally, we have shown that the revenue estimations that leverage the blockchain can be 29 times higher than those derived from victim-reported losses.

7.2 Future Work

To conclude this dissertation, we list some potential research lines for future work.

Other blockchains. Some of our techniques already apply to multiple blockchains, e.g., the automatic collection and extraction of attribution data from public sources and the classification of natural text. Some of our techniques, namely the double-counting filtering and the operation oracles, are directly applicable to other blockchains implementing the UTXO transaction model, like Litecoin and BitcoinCash. Additionally, it should be possible to generalize our techniques (i.e., for tracing coin flows between addresses, for producing oracles by analyzing transaction patterns, and for estimating address revenue) to support other non-UTXO blockchains based on different transaction models, like Ethereum [66] or Solana [65]. The attribution data produced in this dissertation could serve to build tag databases that enable the analysis of other blockchains in the future.

Cross-chain transactions. Previous works showed how malicious actors use different strategies to launder their funds. One strategy is sending funds between blockchains in *cross-chain* transactions by abusing third-party services that provide them, such as cryptocurrency exchanges, payment platforms, and DeFi services. Analyzing how cybercriminals abuse

cross-chain transactions and developing new techniques for tracing them is another interesting future research line.

Address expansion. As we showed in Chapter 4, cybercriminals actively evade MI clustering, the most used heuristic for address expansion in UTXO-based blockchains like Bitcoin. Thus, new techniques for this task are needed. Recently, authors in [101] developed a clustering technique based on *peeling-chain* tracing, a transaction pattern commonly used for money laundering. Systematically evaluating this and new clustering heuristics for estimating cybercrime revenue, as well as proposing address expansion techniques for other blockchains, is future work.

Attribution data. In addition to their abuse type, it would be valuable to extract additional information about addresses appearing in reports written in natural language, like the entity owning the mentioned addresses, information that is key for building attribution data.

Operation Oracles. In Chapters 3 and 4, we have built operation oracles by manually analyzing the transactions produced by malware operators. We characterized such transactions and derived rules to detect Bitcoin addresses producing them. A possible research line is automating this process by deriving ML classifiers from transactions produced by malicious addresses that behave similarly on-chain.

Bibliography

- [1] Lawrence Abrams. *New DeadBolt ransomware targets QNAP devices, asks 50 BTC for master key*. <https://www.bleepingcomputer.com/news/security/new-deadbolt-ransomware-targets-qnap-devices-asks-50-btc-for-master-key/>. Jan. 2022.
- [2] Bhupendra Acharya et al. “Conning the Crypto Conman: End-to-End Analysis of Cryptocurrency-based Technical Support Scams”. In: *IEEE Symposium on Security and Privacy*. 2024.
- [3] Citizens Advice. *How to communicate about scams in an effective and engaging way*. https://www.cas.org.uk/system/files/citizens_advice_scams_awareness_toolkit2018b.pdf. 2018.
- [4] Fernando Alves et al. “Follow the Blue Bird: A Study on Threat Data Published on Twitter”. In: *ESORICS*. 2020.
- [5] Benjamin Andow et al. “PolicyLint: Investigating Internal Privacy Policy Contradictions on Google Play”. In: *USENIX Security*. 2019.
- [6] Elli Androulaki et al. “Evaluating User Privacy in Bitcoin”. In: *Financial Cryptography and Data Security*. 2013.
- [7] Arkham Intel. <https://intel.arkm.com>. 2024.
- [8] Kiran Bandla and Santiago Castro. *APTnotes*. <https://github.com/aptnotes/data>. 2022.
- [9] Massimo Bartoletti, Barbara Pes, and Sergio Serusi. “Data Mining for Detecting Bitcoin Ponzi Schemes”. In: *Crypto Valley Conference on Blockchain Technology*. June 2018.
- [10] Massimo Bartoletti et al. “Cryptocurrency Scams: Analysis and Perspectives”. In: *IEEE Access* 9 (2021), pp. 148353–148373.
- [11] Devon Beckett. *Bitcoin Fog Affidavit by Internal Revenue Service*. https://storage.courtlistener.com/recap/gov.uscourts.dcd.230456/gov.uscourts.dcd.230456.1.1_1.pdf. Apr. 2021.
- [12] Hugo LJ Bijmans, Tim M Booi, and Christian Doerr. “Just the Tip of the Iceberg: Internet-Scale Exploitation of Routers for Cryptojacking”. In: *ACM SIGSAC Conference on Computer and Communications Security*. 2019.
- [13] Binance. *Our Commitment To Transparency*. <https://www.binance.com/en/blog/community/our-commitment-to-transparency-2895840147147652626>. Nov. 2022.
- [14] Marzieh Bitaab et al. “Beyond Phish: Toward Detecting Fraudulent e-Commerce Websites at Scale”. In: *IEEE Symposium on Security and Privacy*. 2023.
- [15] *Bitcoin Abuse*. <https://www.bitcoinabuse.com>. 2022.

- [16] *Bitcoin Explorer*. <https://www.blockchain.com/explorer>. 2022.
- [17] *Bitcoin OTC*. <https://bitcoin-otc.com>. 2022.
- [18] *Bitcoin Porn Scam and Sextortion*. <https://www.onlinethreatalerts.com/article/2020/4/9/bitcoin-porn-scam-and-sextortion/>. Apr. 2020.
- [19] *Bitcoin Talk Forum*. <https://bitcointalk.org>. 2022.
- [20] *Bitcoin Wiki: no change heuristic*. https://en.bitcoin.it/wiki/Privacy#Exact_payment_amounts_.28no_change.29. 2023.
- [21] *BitcoinPonziTool*. <https://github.com/bitcoinponzi/BitcoinPonziTool/>. 2018.
- [22] *BitIodine*. <https://github.com/mikispag/bitiodine>. 2023.
- [23] *Blockchain Luxembourg*. <https://blockchain.info>.
- [24] Xander Bouwman et al. “A different cup of TI? The added value of commercial threat intelligence”. In: *USENIX Security Symposium*. 2020.
- [25] *BTC-E and Bitcointalk.org Hacked?* <https://www.reddit.com/r/Bitcoin/comments/2mc8nc>. Nov. 2014.
- [26] Armin Buescher. *ioc_parser*. https://github.com/armbues/ioc_parser/. 2017.
- [27] David Buil-Gil and Patricia Saldaña-Taboada. “Offending concentration on the internet: An exploratory analysis of bitcoin-related cybercrime”. In: *Deviant Behavior* 43.12 (2022), pp. 1453–1470.
- [28] Elie Bursztein, Kylie McRoberts, and Luca Invernizzi. *Tracking desktop ransomware payments*. <https://www.blackhat.com/docs/us-17/wednesday/us-17-Invernizzi-Tracking-Ransomware-End-To-End.pdf>. 2017.
- [29] Juan Caballero et al. “The Rise of GoodFATR: A Novel Accuracy Comparison Methodology for Indicator Extraction Tools”. In: *Future Generation Computer Systems* 144 (Apr. 2023), pp. 74–89. DOI: [10.1016/j.future.2023.02.012](https://doi.org/10.1016/j.future.2023.02.012).
- [30] Jack Cable. *Ransomwhere: A Crowdsourced Ransomware Payment Dataset*. Version 1.0.0. May 2022. DOI: [10.5281/zenodo.6512123](https://doi.org/10.5281/zenodo.6512123).
- [31] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. “Density-Based Clustering Based on Hierarchical Density Estimates”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 2013.
- [32] *Censys*. <https://censys.io/>. 2022.
- [33] *Censys DeadBolt*. <https://datastudio.google.com/reporting/f8d38b6c-9997-4bba-be93-19cf57d7371a/page/DcGtC>. 2022.
- [34] CERT Polska. *Case Study of Malicious Actors: Going Postal*. https://maldroid.github.io/docs/The_Postal_Group.pdf. Oct. 2015.
- [35] *ChainAbuse*. <https://chainabuse.com>. 2024.
- [36] *Chainalysis*. <https://www.chainalysis.com/>. 2022.
- [37] Jinho Choi et al. “A large-scale bitcoin abuse measurement and clustering analysis utilizing public reports”. In: *IEICE TRANSACTIONS on Information and Systems* 105.7 (2022), pp. 1296–1307.
- [38] Nicolas Christin. “Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace”. In: *The World Wide Web Conference*. 2013.
- [39] Richard Clayton, Tyler Moore, and Nicolas Christin. “Concentrating Correctly on Cybercrime Concentration”. In: *Workshop on Economics of the Information Society*. 2015.

-
- [40] *CoinDesk: Bitcoin, Ethereum, Crypto News and Price Data*. <https://api.coindesk.com/v1/bpi/historical/close.json>. 2023.
 - [41] *Coinhive, the Monero mining service, is closing down*. <https://www.pandasecurity.com/mediacenter/news/coinhive-mining-closes>.
 - [42] *Coinpayments - Service Restriction Notice*. <https://bitcointalk.org/index.php?topic=5156686>. June 2019.
 - [43] Federal Trade Commission. “Consumer Sentinel Network - Data Book 2023”. In: (Feb. 2024). https://www.ftc.gov/system/files/ftc_gov/pdf/CSN-Annual-Data-Book-2023.pdf.
 - [44] UK Gambling Commission. *LCCP Condition 17.1.1 - Customer identity verification*. <https://www.gamblingcommission.gov.uk/licensees-and-businesses/lccp/condition/17-1-1-customer-identity-verification>. Oct. 2020.
 - [45] Mauro Conti, Ankit Gangwal, and Sushmita Ruj. “On the Economic Significance of Ransomware Campaigns: A Bitcoin Transactions Perspective”. In: *Computers & Security* 79 (2018), pp. 162–189.
 - [46] Nathan Copping. *Netwalker Ransomware Guide: Everything You Need to Know*. <https://www.varonis.com/blog/netwalker-ransomware>. Feb. 2022.
 - [47] Cassandra Cross. “Romance baiting, cryptorom and ‘pig butchering’: an evolutionary step in romance fraud”. In: *Current Issues in Criminal Justice* (2023), pp. 1–13.
 - [48] *Cryptocurrency Exchange Scams Dataset*. <https://cryptoexchangescam.github.io/ScamDataset/>. 2020.
 - [49] *CryptoScamDB*. <https://cryptoscamdb.org/>. 2022.
 - [50] *Crystal Lite*. <https://lite.crystalintelligence.com>. 2024.
 - [51] Michal Danilák. <https://github.com/Mimino666/langdetect>. 2021.
 - [52] DaveCrim. *Catastrophic backtracking in BACKSLASH_URL_RE*. <https://github.com/InQuest/python-iocextract/issues/52>. 2021.
 - [53] James C Davis et al. “The Impact of Regular Expression Denial of Service (ReDoS) in Practice: An Empirical Study at the Ecosystem Scale”. In: *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2018.
 - [54] Nikhilesh De. *US Officials Arrest Alleged Operator of \$336M Bitcoin Mixing Service*. <https://www.coindesk.com/us-officials-arrest-alleged-operator-of-336m-bitcoin-mixing-service>. Apr. 2021.
 - [55] Mark Ellzey, Aidan Holland, and Ryan Lindner. *The QNapping of QNAP Devices*. <https://censys.io/the-qnapping-of-qnap-devices/>. Jan. 2022.
 - [56] *Double and Nothing: Understanding and Detecting Cryptocurrency Giveaway Scams (Dataset)*. <https://double-and-nothing.github.io/>. 2023.
 - [57] dree12. *List of Major Bitcoin Heists, Thefts, Hacks, Scams, and Losses*. <https://bitcointalk.org/index.php?topic=576337>.
 - [58] Kobi Eisenkraft and Arie Olshtein. *Pony’s C&C servers hidden inside the Bitcoin blockchain*. <https://research.checkpoint.com/2019/ponys-cc-servers-hidden-inside-the-bitcoin-blockchain/>. Oct. 2019.
 - [59] *Elliptic*. <https://www.elliptic.co/>. 2022.
 - [60] Mark Ellzey and Emily Austin. *The Evolution of ESXiArgs Ransomware*. <https://censys.io/the-evolution-of-esxiargs-ransomware>. Feb. 2023.

- [61] Dmitry Ermilov, Maxim Panov, and Yury Yanovich. “Automatic Bitcoin Address Clustering”. In: *IEEE International Conference on Machine Learning and Applications*. 2017.
- [62] Europol. *Multi-million euro cryptocurrency laundering service Bestmixer.io taken down*. <https://www.europol.europa.eu/newsroom/news/multi-million-euro-cryptocurrency-laundering-service-bestmixerio-taken-down>. May 2019.
- [63] FIRST. *Traffic Light Protocol (TLP). FIRST Standards Definitions and Usage Guidance - Version 1.0*. <https://www.first.org/tlp/docs/tlp-v1.pdf>. 2016.
- [64] *Forced address reuse*. https://en.bitcoin.it/wiki/Privacy#Forced_address_reuse.
- [65] Solana Foundation. *Web3 Infrastructure for Everyone | Solana*. <https://solana.com/>. Nov. 2024.
- [66] The Ethereum Foundation. *What is Ethereum?* <https://ethereum.org/en/what-is-ethereum/>. Nov. 2024.
- [67] Fumko. *Let’s play (again) with Predator the thief*. <https://fumik0.com/2019/12/25/lets-play-again-with-predator-the-thief>. Dec. 2019.
- [68] Peng Gao et al. “Enabling Efficient Cyber Threat Hunting With Cyber Threat Intelligence”. In: *IEEE International Conference on Data Engineering*. 2021.
- [69] Sergiu Gatlan. *Police tricks DeadBolt ransomware out of 155 decryption keys*. <https://www.bleepingcomputer.com/news/security/police-tricks-deadbolt-ransomware-out-of-155-decryption-keys>. Oct. 2022.
- [70] Sergiu Gatlan. *Qlocker ransomware returns to target QNAP NAS devices worldwide*. <https://www.bleepingcomputer.com/news/security/qlocker-ransomware-returns-to-target-qnap-nas-devices-worldwide/>. Jan. 2022.
- [71] Sergiu Gatlan. *Massive ESXiArgs ransomware attack targets VMware ESXi servers worldwide*. <https://www.bleepingcomputer.com/news/security/massive-esxiargs-ransomware-attack-targets-vmware-esxi-servers-worldwide>. Feb. 2023.
- [72] Will Gibb and Devon Kerr. *OpenIOC: Back to the Basics*. <https://www.mandiant.com/resources/openioc-basics>. Oct. 2013.
- [73] Gibran Gomez and Pedro Moreno-Sanchez and Juan Caballero. “Watch Your Back: Identifying Cybercrime Financial Relationships in Bitcoin through Back-and-Forth Exploration”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’22. Association for Computing Machinery, 2022, pp. 1291–1305. DOI: [10.1145/3548606.3560587](https://doi.org/10.1145/3548606.3560587).
- [74] Martin Gill and Geoff Taylor. “Preventing Money Laundering or Obstructing Business? Financial Companies’ Perspectives on ‘Know Your Customer’ Procedures”. In: *British Journal of Criminology* 44.4 (2004), pp. 582–594.
- [75] *GlassChain*. <https://glasschain.org/>. 2024.
- [76] Steven Goldfeder et al. “When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies”. In: *PoPETs 2018* (2018), pp. 179–199.
- [77] Andrew Goldsmith. *What do we know about REvil, the Russian ransomware gang likely behind the Medibank cyber attack?* <https://theconversation.com/what-do-we-know-about-revil-the-russian-ransomware-gang-likely-behind-the-medibank-cyber-attack-194337>. Nov. 2022.

- [78] Gibran Gomez, Kevin van Liebergen, and Juan Caballero. “Cybercrime Bitcoin Revenue Estimations: Quantifying the Impact of Methodology and Coverage”. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’23. Association for Computing Machinery, 2023, pp. 3183–3197. DOI: [10.1145/3576915.3623094](https://doi.org/10.1145/3576915.3623094).
- [79] Ian W. Gray et al. “Money Over Morals: A Business Analysis of Conti Ransomware”. In: *APWG Symposium on Electronic Crime Research*. 2022.
- [80] Hamza Harkous et al. “Polisis: Automated Analysis and Presentation of Privacy Policies Using Deep Learning”. In: *USENIX Security*. 2018.
- [81] Mikkel Alexander Harlev et al. “Breaking Bad: De-Anonymising Entity Types on the Bitcoin Blockchain Using Supervised Machine Learning”. In: *Hawaii International Conference on System Sciences*. 2018.
- [82] David Harley and Aleksandr Matrosov. *TDL4 and Glupteba: Piggyback PiggyBugs*. <https://www.welivesecurity.com/2011/03/02/tdl4-and-glupteba-piggyback-piggybugs/h>. Mar. 2011.
- [83] Martin Harrigan and Christoph Fretter. “The Unreasonable Effectiveness of Address Clustering”. In: *IEEE International Conference on Ubiquitous Intelligence and Computing (ATC)*. July 2016.
- [84] Bernhard Haslhofer et al. “GraphSense: A General-Purpose Cryptoasset Analytics Platform”. In: *Arxiv pre-print* (2021). <https://arxiv.org/abs/2102.13613>.
- [85] Floyd Hightower. *IOC Finder*. <https://github.com/fhightower/ioc-finder>. 2018.
- [86] Geng Hong et al. “How You Get Shot in the Back: A Systematical Study about Cryptojacking in the Real World”. In: *ACM SIGSAC Conference on Computer and Communications Security*. 2018.
- [87] Jaromir Horejsi and Joseph C Chen Salgado. *Glupteba Hits Routers and Updates C&C Servers*. https://www.trendmicro.com/en_us/research/19/i/glupteba-campaign-hits-network-routers-and-updates-cc-servers-with-data-from-bitcoin-transactions.html. Sept. 2019.
- [88] Henry Hosseini et al. “Unifying Privacy Policy Detection”. In: *PoPETs*. 2021.
- [89] Danny Yuxing Huang et al. “Botcoin: Monetizing Stolen Cycles”. In: *Network and Distributed Systems Security Symposium*. 2014.
- [90] Danny Yuxing Huang et al. “Tracking Ransomware End-to-end”. In: *IEEE Symposium on Security and Privacy*. May 2018.
- [91] *Hugging Face*. <https://huggingface.co/>. 2023.
- [92] *Hugging Face Pre-trained models*. https://www.sbert.net/docs/pretrained_models.html. 2023.
- [93] Shane Huntley and Luca Nagy. *Disrupting the Glupteba operation*. <https://blog.google/threat-analysis-group/disrupting-glupteba-operation/>. Dec. 2021.
- [94] Ghaith Husari et al. “TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources”. In: *Annual Computer Security Applications Conference*. 2017.
- [95] Augusto Remillano II, Jakub Urbanec, and Wilbert Luy Saias. *Skidmap Malware Uses Rootkit to Hide Mining Payload*. https://www.trendmicro.com/en_us/research/19/i/skidmap-linux-malware-uses-rootkit-capabilities-to-hide-cryptocurrency-mining-payload.html. Sept. 2019.

- [96] InQuest. *iocextract*. <https://github.com/InQuest/python-iocextract>. 2019.
- [97] Ziwei Ji et al. “Survey of hallucination in natural language generation”. In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38.
- [98] Marc Jourdan et al. “Characterizing Entities in the Bitcoin Blockchain”. In: *IEEE International Conference on Data Mining Workshops*. 2018.
- [99] Harry Kalodner et al. “BlockSci: Design and Applications of a Blockchain Analysis Platform”. In: *USENIX Security Symposium*. 2020.
- [100] Chris Kanich et al. “Spamalytics: An Empirical Analysis of Spam Marketing Conversion”. In: *ACM conference on Computer and communications security*. 2008.
- [101] George Kappos et al. “How to Peel a Million: Validating and Expanding Bitcoin Clusters”. In: *USENIX Security Symposium*. 2022.
- [102] Kaspersky. *What is WannaCry ransomware?* <https://www.kaspersky.com/resource-center/threats/ransomware-wannacry>. June 2020.
- [103] Amin Kharraz et al. “Cutting the gordian knot: A look under the hood of ransomware attacks”. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. 2015.
- [104] Kim Nilsson. *The Missing MtGox Bitcoins*. <https://blog.wizsec.jp/2015/04/the-missing-mtgox-bitcoins.html>. Apr. 2015.
- [105] Paul Kimayong. *Masad Stealer: Exfiltrating using Telegram*. <https://blogs.juniper.net/en-us/threat-research/masad-stealer-exfiltrating-using-telegram>. Sept. 2019.
- [106] Christian Kohlschütter. *boilerpipe: Boilerplate Removal and Fulltext Extraction from HTML pages*. <https://github.com/kohlschutter/boilerpipe>. 2022.
- [107] Radhesh Krishnan Konoth et al. “MineSweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 1714–1730. DOI: [10.1145/3243734.3243858](https://doi.org/10.1145/3243734.3243858).
- [108] Jesse Kornblum. *ssdeep Project*. <https://ssdeep-project.github.io/ssdeep/index.html>. 2016.
- [109] Platon Kotzias et al. “Scamdog Millionaire: Detecting E-commerce Scams in the Wild”. In: *Annual Computer Security Applications Conference*. 2023.
- [110] Platon Kotzias et al. “Ctrl+Alt+Deceive: Quantifying User Exposure to Online Scams”. In: *Network and Distributed Systems Security Symposium*. 2025.
- [111] Brian Krebs. *Conti Ransomware Group Diaries*. <https://krebsonsecurity.com/2022/03/conti-ransomware-group-diaries-part-i-evasion/>. Mar. 2022.
- [112] Jochem van de Laarschot and Rolf van Wegberg. “Risky Business? Investigating the Security Practices of Vendors on an Online Anonymous Market using Ground-Truth Data”. In: *30th USENIX Security Symposium (USENIX Security 21)*. <https://www.usenix.org/conference/usenixsecurity21/presentation/van-de-laarschot>. USENIX Association, Aug. 2021, pp. 4079–4095.
- [113] Seunghyeon Lee et al. “Cybercriminal Minds: An Investigative Study of Cryptocurrency Abuses in the Dark Web”. In: *Network and Distributed Systems Security Symposium*. 2019.
- [114] Vector Guo Li et al. “Reading the Tea leaves: A Comparative Analysis of Threat Intelligence”. In: *USENIX Security*. 2019.

- [115] Xigao Li, Anurag Yepuri, and Nick Nikiforakis. “Double and Nothing: Understanding and Detecting Cryptocurrency Giveaway Scams”. In: *Network and Distributed Systems Security Symposium*. 2023.
- [116] Kevin Liao et al. “Behind Closed Doors: Measurement and Analysis of CryptoLocker Ransoms in Bitcoin”. In: *APWG Symposium on Electronic Crime Research*. June 2016.
- [117] Xiaojing Liao et al. “Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016.
- [118] Y. Lin et al. “An Evaluation of Bitcoin Address Classification based on Transaction History Summarization”. In: *IEEE International Conference on Blockchain and Cryptocurrency*. May 2019.
- [119] *Linux mining virus-sysupdate, notas de eliminación de servicios de red*. <https://programmerclick.com/article/47131149819/>. Nov. 2019.
- [120] *LocalBitcoins Statement on the Coming AML regulations and Compliance*. <https://localbitcoins.com/blog/aml-regulations-compliance>. Feb. 2019.
- [121] MaliciaLab. *iocsearcher*. <https://github.com/malicialab/iocsearcher>. 2023.
- [122] Mandiant. *Tracking Malware with Import Hashing*. <https://www.mandiant.com/resources/tracking-malware-import-hashing>. 2014.
- [123] Gil Mansharov and Alexey Bukhteyev. *In the footsteps of a sextortion campaign*. <https://research.checkpoint.com/2019/in-the-footsteps-of-a-sextortion-campaign/>. Oct. 2019.
- [124] Josh Marcus. ‘Bitcoin widow’ reveals how she fell victim to her own husband’s \$215m scam after his mysterious death. <https://www.independent.co.uk/tech/bitcoin-widow-gerald-cotten-quadriga-b2056019.html>. Apr. 2022.
- [125] Better Business Bureau Institute for Marketplace Trust. *BBB Scam Tracker Risk Report*. <https://bbbmarketplacetrust.org/wp-content/uploads/2024/04/2023-BBBScamTracker-RiskReport-US-040224.pdf>. Apr. 2024.
- [126] MarketWatch. *Threat Intelligence Market 2022 Report*. <https://www.marketwatch.com/press-release/threat-intelligence-market-2022-report-examines-latest-trends-and-key-drivers-supporting-growth-till-2030-2022-07-27>. 2022.
- [127] Mathias Bynens. *In search of the perfect URL validation regex*. <https://mathiasbynens.be/demo/url-regex>. 2021.
- [128] Gregory Maxwell. *CoinJoin: Bitcoin privacy for the real world*. <https://bitcointalk.org/index.php?topic=279249.0>. Aug. 2013.
- [129] Damon McCoy et al. “PharmaLeaks: Understanding the Business of Online Pharmaceutical Affiliate Programs”. In: *USENIX Security Symposium*. 2012.
- [130] Sarah Meiklejohn et al. “A Fistful of Bitcoins: Characterizing Payments among Men with No Names”. In: *Internet Measurement Conference*. 2013.
- [131] Per Håkon Meland, Yara Fareed Fahmy Bayoumy, and Guttorm Sindre. “The Ransomware-as-a-Service economy within the darknet”. In: *Computers & Security* 92 (2020), p. 101762. DOI: <https://doi.org/10.1016/j.cose.2020.101762>.
- [132] *Meta-Llama-3-70B-Instruct*. <https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct>. 2024.
- [133] *MINE.exchange*. <https://mine.exchange>. 2022.

- [134] Malte Möser, Rainer Böhme, and Dominic Breuker. “An Inquiry into Money Laundering Tools in the Bitcoin Ecosystem”. In: *APWG eCrime Researchers Summit*. Sept. 2013.
- [135] Mozilla. *Mozilla Readability.js library*. <https://github.com/mozilla/readability>. 2022.
- [136] Luca Nagy. *Glupteba: Hidden Malware Delivery in Plain Sight*. https://news.sophos.com/wp-content/uploads/2020/06/glupteba_final.pdf. June 2020.
- [137] Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System”. <https://bitcoin.org/bitcoin.pdf>. 2008.
- [138] *New report: Malware attacks Chilean banks and bypasses SmartScreen, by exploiting DLL Hijacking within popular software*. <https://business.blogthinkbig.com/new-report-malware-attacks-chilean>.
- [139] Amirreza Niakanlahiji et al. “IoCMiner: Automatic Extraction of Indicators of Compromise from Twitter”. In: *IEEE Big Data*. 2019.
- [140] Jonas David Nick. “Data-Driven De-Anonymization in Bitcoin”. MA thesis. Zurich, Switzerland: Distributed Computing Group, Computer Engineering and Networks Laboratory, ETH Zurich, Aug. 2015.
- [141] Manabu Niseki. *IoC extractor*. <https://github.com/ninoseki/ioc-extractor>. 2019.
- [142] Roberta Liggett O’Malley and Karen M Holt. “Cyber sextortion: An exploratory analysis of different perpetrators engaging in a similar crime”. In: *Journal of interpersonal violence* 37.1-2 (2022), pp. 258–283.
- [143] Frédérique Oggier, Anwitaman Datta, and Silivanxay Phetsouvanh. “An ego network analysis of sextortionists”. In: *Social Network Analysis and Mining* 10 (2020), pp. 1–14.
- [144] *Omni Layer*. <https://www.omnilayer.org>. 2023.
- [145] *On the Economic Significance of Ransomware Campaigns: A Bitcoin Transactions Perspective (Dataset)*. https://spritz.math.unipd.it/datasets/btcransomware/knowledge_base.zip. 2018.
- [146] Kris Oosthoek, Jack Cable, and Georgios Smaragdakis. *A Tale of Two Markets: Investigating the Ransomware Payments Economy*. 2022. arXiv: 2205.05028 [cs.CR].
- [147] Kris Oosthoek, Mark Van Staalduinen, and Georgios Smaragdakis. “Quantifying Dark Web Shops’ Illicit Revenue”. In: *IEEE Access* 11 (2023), pp. 4794–4808.
- [148] OASIS Open. *STIX: A structured language for cyber threat intelligence*. <https://oasis-open.github.io/cti-documentation/>. 2022.
- [149] *OpenAI Pricing*. <https://openai.com/api/pricing/>. 2024.
- [150] p1ngul1n0. *Blackbird*. <https://github.com/p1ngul1n0/blackbird>. 2022.
- [151] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benoit Dupont. “Ransomware Payments in the Bitcoin Ecosystem”. In: *Journal of Cybersecurity* 5.1 (2019).
- [152] Masarah Paquet-Clouston et al. “Spams Meet Cryptocurrencies: Sextortion in the Bitcoin Ecosystem”. In: *ACM Conference on Advances in Financial Technologies*. 2019.
- [153] Francesco Parolini and Antoine Miné. *rat - ReDoS Abstract Tester*. <https://github.com/parof/rat>. Apr. 2022.
- [154] Francesco Parolini and Antoine Miné. “Sound Static Analysis of Regular Expressions for Vulnerabilities to Denial of Service Attacks”. In: *International Symposium on Theoretical Aspects of Software Engineering*. 2022.

- [155] Sergio Pastrana and Guillermo Suarez-Tangil. “A First Look at the Crypto-Mining Malware Ecosystem: A Decade of Unrestricted Wealth”. In: *Proceedings of the Internet Measurement Conference*. IMC ’19. 2019, pp. 73–86. DOI: [10.1145/3355369.3355576](https://doi.org/10.1145/3355369.3355576).
- [156] PepperMalware. *Brief analysis of Redaman Banking Malware (v0.6.0.2) Sample*. <http://www.peppermalware.com/2019/11/brief-analysis-of-redaman-banking.html>. Nov. 2019.
- [157] Stijn Pletinckx, Cyril Trap, and Christian Doerr. “Malware Coordination using the Blockchain: An Analysis of the Cerber Ransomware”. In: *IEEE Conference on Communications and Network Security*. 2018.
- [158] Daniel Plohmann et al. “Malpedia: A Collaborative Effort to Inventorize the Malware Landscape”. In: *The Journal on Cybercrime & Digital Investigations* 3.1 (2017).
- [159] Victor Le Pochat et al. “Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation”. In: *NDSS*. 2019.
- [160] Rebecca S. Portnoff et al. “Backpage and Bitcoin: Uncovering Human Traffickers”. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017.
- [161] *pr0fessionalcrackers telegram channel*. <https://ru.tgchannels.org/channel/MrPrOgr4mmer>. July 2020.
- [162] The Monero Project. *What is Monero (XMR)?* <https://www.getmonero.org/get-started/what-is-monero/>. Sept. 2024.
- [163] Python Software Foundation. *Requests*. <https://github.com/psf/requests>. 2022.
- [164] *QSA-22-24: DeadBolt Ransomware*. <https://www.qnap.com/en/security-advisory/qsa-22-24>. Sept. 2022.
- [165] *Ransomware in the Bitcoin Ecosystem | Dataset Extraction*. <https://github.com/behas/ransomware-dataset>. 2019.
- [166] *ReportFraud*. <https://reportfraud.ftc.gov/>. 2024.
- [167] Trend Micro Research. *What We Know About the DarkSide Ransomware and the US Pipeline Attack*. https://www.trendmicro.com/en_us/research/21/e/what-we-know-about-darkside-ransomware-and-the-us-pipeline-attac.html. May 2021.
- [168] Trend Micro Research. *What Is RYUK Ransomware?* https://www.trendmicro.com/en_in/what-is/ransomware/ryuk-ransomware.html. Aug. 2023.
- [169] Responders.NU. <https://deadbolt.responders.nu/>. Oct. 2022.
- [170] Leonard Richardson. *Beautiful Soup*. <https://beautiful-soup-4.readthedocs.io/en/latest/>. 2022.
- [171] Ronald Rivest. *The MD5 Message-Digest Algorithm*. <https://datatracker.ietf.org/doc/html/rfc1321>. Apr. 1992.
- [172] Scott J. Roberts. *jager*. <https://github.com/sroberts/jager>. 2015.
- [173] Scott J. Roberts. *Cacador*. <https://github.com/sroberts/cacador>. 2016.
- [174] Dorit Ron and Adi Shamir. “Quantitative Analysis of the Full Bitcoin Transaction Graph”. In: *Financial Cryptography and Data Security*. 2013.
- [175] Dorit Ron and Adi Shamir. “How Did Dread Pirate Roberts Acquire and Protect his Bitcoin Wealth?” In: *Financial Cryptography and Data Security*. 2014.

- [176] Hampus Rosenquist et al. “On the Dark Side of the Coin: Characterizing Bitcoin Use for Illicit Activities”. In: *International Conference on Passive and Active Network Measurement*. 2024.
- [177] Jan Rubín. *Clipsa - Multipurpose password stealer*. <https://decoded.avast.io/janrubin/clipsa-multipurpose-password-stealer/>. Aug. 2019.
- [178] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. “CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin”. In: *Computer Security - ESORICS 2014*. Springer International Publishing, 2014, pp. 345–364.
- [179] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. “Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits”. In: *USENIX Security*. 2015.
- [180] Evyatar Saias. *Bitcoins, blockchains, and botnets*. <https://blogs.akamai.com/sitr/2021/02/bitcoins-blockchains-and-botnets.html>. Feb. 2021.
- [181] Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108* (2019).
- [182] *Satoshi DICE*. <https://satoshidice.com/>. 2022.
- [183] Kiavash Satvat, Rigel Gjomemo, and VN Venkatakrishnan. “Extractor: Extracting Attack Behavior from Threat Reports”. In: *Euro S&P*. 2021.
- [184] *ScamTracker*. <https://www.bbb.org/scamtracker>. 2024.
- [185] *Scam Watch*. <https://www.scamwatch.gov.au/>. 2024.
- [186] Marcos Sebastian et al. “AVClass: A Tool for Massive Malware Labeling”. In: *Research in Attacks, Intrusions, and Defenses*. 2016.
- [187] *Sheep Marketplace*. https://en.wikipedia.org/wiki/Sheep_Marketplace.
- [188] Hyejin Shin et al. “Twiti: Social Listening for Threat Intelligence”. In: *WWW*. 2021.
- [189] Yusuke Shinyama, Philippe Guglielmetti, and Pieter Marsman. *pdfminer.six*. <https://github.com/pdfminer/pdfminer.six>. 2019.
- [190] *Shodan*. <https://www.shodan.io/>. 2022.
- [191] Matt Sisk, Robin Ruefle, and Sam Perl. *Harvesting Artifacts: Improving Useful Data Extraction from Cybersecurity Incident Reports*. <https://github.com/cmu-sei/cyobstract>. 2018.
- [192] Rocky Slavin et al. “Toward a Framework for Detecting Privacy Policy Violations in Android Application Code”. In: *International Conference on Software Engineering*. 2016.
- [193] Russell G Smith. “Consumer scams in Australia: An overview.” In: *Trends & Issues in Crime & Criminal Justice* 331 (2007).
- [194] Software Freedom Conservancy. *Selenium*. <https://www.selenium.dev/>. 2022.
- [195] Kyle Soska and Nicolas Christin. “Measuring the Longitudinal Evolution of the Online Anonymous Marketplace Ecosystem”. In: *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Aug. 2015, pp. 33–48.
- [196] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. “BitIodine: Extracting Intelligence from the Bitcoin Network”. In: *Financial Cryptography and Data Security*. 2014.
- [197] *Spams meet Cryptocurrencies: Sextortion in the Bitcoin Ecosystem (Dataset)*. https://github.com/MatteoRomiti/Sextortion_Spam_Bitcoin. 2019.

-
- [198] US Department of State. *DarkSide Ransomware as a Service (RaaS)*. <https://www.state.gov/darkside-ransomware-as-a-service-raas>. Nov. 2021.
 - [199] Tsuyoshi Taniguchi, Harm Griffioen, and Christian Doerr. “Analysis and Takeover of the Bitcoin-Coordinated Pony Malware”. In: *ACM ASIA Conference on Computer and Communications Security*. 2021.
 - [200] Flashpoint Intel Team. *Conti Ransomware: The History Behind One of the World’s Most Aggressive RaaS Groups*. <https://flashpoint.io/blog/history-of-conti-ransomware>. Oct. 2022.
 - [201] Ege Tekiner et al. “SoK: Cryptojacking Malware”. In: *IEEE European Symposium on Security and Privacy*. 2021.
 - [202] Taylor Tepper and John Schmidt. *Best Crypto Exchanges For 2021*. <https://www.forbes.com/advisor/investing/best-crypto-exchanges/>. May 2021.
 - [203] *The 2023 Crypto Crime Report*. https://go.chainalysis.com/rs/503-FAP-074/images/Crypto_Crime_Report_2023.pdf. 2023.
 - [204] Kurt Thomas et al. “Framing dependencies introduced by underground commoditization”. In: *Workshop on the Economics of Information Security*. 2015.
 - [205] Milind Tiwari, Adrian Gepp, and Kuldeep Kumar. “The future of raising finance-a new opportunity to commit fraud: a review of initial coin offering (ICOs) scams”. In: *Crime, Law and Social Change* 73 (2020), pp. 417–441.
 - [206] TMJ4. *Victim blaming prevents consumers from coming forward and reporting fraud*. <https://www.tmj4.com/news/i-team/victim-blaming-prevents-consumers-from-coming-forward-and-reporting-fraud>. 2022.
 - [207] *Tor Project*. <https://www.torproject.org/>.
 - [208] TRM. *TRM Labs Acquires Bitcoinabuse.com*. <https://www.globenewswire.com/en/news-release/2023/10/11/2758400/0/en/TRM-Labs-Acquires-Bitcoinabuse-com.html>. Oct. 2023.
 - [209] Twitter. *Twitter API*. <https://developer.twitter.com/en/docs/twitter-api>. 2022.
 - [210] US Department of Justice. *Russian National And Bitcoin Exchange Charged In 21-Count Indictment For Operating Alleged International Money Laundering Scheme And Allegedly Laundering Funds From Hack Of Mt. Gox*. <https://www.justice.gov/usao-ndca/pr/russian-national-and-bitcoin-exchange-charged-21-count-indictment-operating-alleged>. July 2017.
 - [211] *W2 July | EN | Story of the week: Ransomware on the Darkweb*. <https://malware.news/t/w2-july-en-story-of-the-week-ransomware-on-the-darkweb/50791>. July 2021.
 - [212] *Wallet Explorer*. <https://www.walletexplorer.com/info>. 2022.
 - [213] *Warning: New email scam people are apparently falling for - "secret doubling address"*. <https://www.reddit.com/r/Bitcoin/comments/2fq3uc>. Sept. 2014.
 - [214] *Wasabi Wallet*. <https://wasabiwallet.io/>. 2022.
 - [215] *Watch your back repository*. <https://github.com/cybersec-code/watchyourback>. Sept. 2024.
 - [216] Shaun Waterman. *European police take down criminals behind two big ransomware strains*. <https://www.cyberscoop.com/ctb-locker-cerber-ransomware-arrests-europol-mcafee/>. Dec. 2017.

- [217] WebMoney. *WebMoney - Universal Payment System*. <https://www.wmtransfer.com/>. 2022.
- [218] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS ’22. New Orleans, LA, USA, 2024.
- [219] Dave Winer. *RSS 2.0 Specification*. <https://cyber.harvard.edu/rss/rss.html>. July 2003.
- [220] *word_cloud*. https://github.com/amueller/word_cloud. 2024.
- [221] Lei Wu et al. “Towards Understanding and Demystifying Bitcoin Mixing Services”. In: *Proceedings of the Web Conference 2021*. WWW ’21. Association for Computing Machinery, 2021, pp. 33–44. DOI: [10.1145/3442381.3449880](https://doi.org/10.1145/3442381.3449880).
- [222] Pengcheng Xia et al. “Characterizing Cryptocurrency Exchange Scams”. In: *Computers & Security* 98 (2020).
- [223] Pengcheng Xia et al. “Don’t Fish in Troubled Waters! Characterizing Coronavirus-themed Cryptocurrency Scams”. In: *2020 APWG Symposium on Electronic Crime Research (eCrime)*. 2020.
- [224] Shunyu Yao et al. *ReAct: Synergizing Reasoning and Acting in Language Models*. 2023. arXiv: [2210.03629](https://arxiv.org/abs/2210.03629) [cs.CL].
- [225] Haohua Sun Yin and Ravi Vatrupu. “A First Estimation of the Proportion of Cyber-criminal Entities in the Bitcoin Ecosystem using Supervised Machine Learning”. In: *IEEE International Conference on Big Data*. 2017.
- [226] Jun Zhao et al. “TIMiner: Automatically extracting and analyzing categorized cyber threat intelligence from social data”. In: *Computers & Security* (2020).
- [227] Andrey Zhdanov and Vladislav Azersky. *DeadBolt ransomware: nothing but NASTy*. <https://blog.group-ib.com/nas-under-threat>. Oct. 2022.
- [228] Ziyun Zhu and Tudor Dumitras. “ChainSmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports”. In: *IEEE European Symposium on Security and Privacy*. 2018.
- [229] Sebastian Zimmeck and Steven M Bellovin. “Privee: An Architecture for Automatically Analyzing Web Privacy Policies”. In: *USENIX Security*. 2014.
- [230] Sebastian Zimmeck et al. “Automated Analysis of Privacy Requirements for Mobile Apps”. In: *NDSS*. 2017.

Appendix

Operation	Short	Address
emptystring	3J98t1	3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy
binance	1NDyJt	1NDyJtNTjmwk5xPNhjaAMu4HDHigtobu1s
cerber	14ru2h	14ru2hbZyJzXADef285wxLVCLEUwnMtmzT
cerber	17gd1m	17gd1msp5FnMcEMF1MitTNSsYs7w7AQyCt
cerber	1CpTCV	1CpTCVckjajNKDd7PsApV3cAkunVd4McmT
cerber	1DiRvu	1DiRvuCvL7rYKuaP8NLEH3bj34wUEC5US
cerber	1GcnsL	1GcnsLs7C31uuroNmUHwwbB5xQeNvm63Ee
cerber	1HTDy9	1HTDy9SkfhwaNCXFA8wFCvN53f3iGpm8kb
cerber	1ML94w	1ML94w1SCudkiFHaEwYqTmKGTkywxVBuZg
glupteba	15y7ds	15y7dskU5TqNHXRtu5wzBpXdY5mT4RZNC6
glupteba	1CgPCp	1CgPCp3E9399ZFodMnTSSvaf5TpGiyM2N1
glupteba	1CUhaT	1CUhaTe3AiP9Tdr4B6wedoe9vNsymLiD97
glupteba	34Rqyw	34RqywhujshGVPNMedvGawFuffW9wWtbXC
glupteba	3NhC1b	3NhC1bvy1dVoJseK7V54VgLFLVNyFuNTVM
pony	19hi8B	19hi8BJ7HxKK45aLVdMbZ6E6oTSW5mGYC82
pony	1BkeGq	1BkeGqpo8M5KNVYXW3obmQt1R58zXAqLBQ
pony	1CeLgF	1CeLgFDu917tgtunhJZ6BA2YdR559Boy9Y
pony	1GUghN	1GUghNgnSbcnoR1CJwDkmco5e7MKfGBn8G
pony	1N9ALZ	1N9ALZUgqYzFQGDXvMY5j1c7PGMMGYqUde
pony	1NL8QT	1NL8QTrs1KEhpcpmBJU947nbcPtSZcpTqQg
pony	bc1q0n	bc1q0nw2g0dgm6shk0xazmaq2tzmwke7ypsz4upzps
pony	bc1qh9	bc1qh96q46mw72shp2j39uq3z0wh0gezguvk9q5js
pony	bc1qwl	bc1qwlmlh195l8fnc0s9xg7kse4sqc5mvj237gdy47
pony	bc1qzs	bc1qzsd9flsg4l4kq3awt23vkmte30n7skavkfle3
pony	35KxqL	35KxqLRwQPf5a17sPGDXDcQ8G9TKTtat7m
pony	3Dmcs6	3Dmcs6kKViugRBdtZwhdHTG4SampnURtbQ
phorpiextldr	phorpiextldr	1Bn4JYKoVgQpZ73doWVFSNZBbwKj3cpJNR
phorpiextldr	A	1GWuCbantYFJDr76ZbRCEjcrMrvqEqUA98
phorpiextldr	B	1Bs5Hc7CT2PCCCKQJeyVgRVyznGgJwMDMTu
phorpiextldr	C	1YRVEawzapE3N8CpPpXg46JbYd72Lfp7nB
phorpiextldr	D	12EMaHiZG75ztKjUjuPZhQDcyW89qRJVuR
phorpiextldr	E	1FJXSVbNmBu7z3RTNGixrvXN1N7pdpRZF3
phorpiextldr	F	1G1aLEVzBcPzqV7bbzfc8tyt4u7oUEZqZ
phorpiextldr	G	1K4Rb5FP3bcgubJzvCLHJ8vVDye1rQ3UNh
phorpiextldr	H	1CSDpCjyVHsuTb6i7zZ8dr8liUGL5ff7vM
phorpiextldr	I	14wmNiq9y9quQm69Y1bXZ7HFRB5mu1o72r
phorpiextldr	J	3GVBvLxVzDqBYxSKDi7uenKGw8xBpwwDgy
phorpiextldr	K	1QJoFaenGQ3bunZkMuN7i7c7GpzWXhmNqo
phorpiextldr	L	1Jkhy6eH9TPn7w9WxYK59a8FxpP4KtKGgLa
phorpiextldr	M	1Ny3nHNMkkxhLzas2ABfbd4ZNYunSV2bE
phorpiextldr	N	13e32uVdf6TSH47G67igN1sSkbrV6QATSY
phorpiextldr	O	3EEiBhtMgMzrrCnZnVxjKQnMWEeduYiyi5
phorpiextldr	P	3G5RektH9rucFBzemQQy5R9eB6RPtYehcX
phorpiextldr	Q	3PkdsSUfjdz6wDUAiFJDYB4bJTwmEZGq8y
phorpiextldr	R	3MTng3jxPihohziAFVSUu53BpifcSNT2EN
phorpiextldr	S	1KmaM9GZC1xTBiUndYSstW4YSrVLbgdsEj
phorpiextldr	T	1MWNkiFb3sqtmRwdEhPSkRYh3zUwusXWGD
mrpr0gr4mmer	A	1QDNWjJdBnNp8JNuQFhRWeQXL3fDb84cVS
mrpr0gr4mmer	B	1ASppKbVqcX1NxxQrsjNnkWLMmTQDnmvrX
mekotio	C	159cFxcSSpup2D4NSZiuBXgsGfgxWCHppv
mrpr0gr4mmer	D	1LWs9tvEmRqpiVYNny4CY912xnudVKsqXG
mrpr0gr4mmer	E	187g2Ppkb8F5suyqZwSExnZcDN9VZY7JhK
mrpr0gr4mmer	F	1E6qZkzbGZHh9hWF4dQcTudbmsYkvBYPrR
mrpr0gr4mmer	G	1M7aXuhpnyQ6r1xgjcNWHjzdnVLEDsMHh
mrpr0gr4mmer	H	1BRkLxWE888QTpV1eMVVHheQDdZkmkhSyT
mrpr0gr4mmer	I	1HTZmKHrjyFAJdtrE7P3gdmYLHjwpZpcYG
mrpr0gr4mmer	J	1BquFCDeLsYyL1SRykPaEgukh5EekwMGJ1
mrpr0gr4mmer	K	35oEFnFn3myZJKskhyEuaNcuGj6XDvRJKy
mrpr0gr4mmer	L	3B2j29iZJ64oT6VNrqw8m4EpVv6HPq4xaU
mrpr0gr4mmer	M	3LvUwiGHn5DCvpFmnQhVrE4tNhPkft4gLa
mrpr0gr4mmer	N	1Ga6NDFba13YSUq3GMz4hDQgzjTixZTFvb
mrpr0gr4mmer	O	1KRAgKpA1QjdACNQmtZUFcViEPB5T57Qsj
deadbolt	bc1qh6	bc1qh6pku7gg2d6pw87z3t4f6d4rk6c48ajvsmfjil
deadbolt	bc1q62	bc1q62rjm9a82s3qmjzff6uuytw25p3fpptl5zpd
deadbolt	bc1q3g	bc1q3gugv2yp5mzm7hfnr7zlg2unah9t6mjwyky72

Table 1: Mapping between short name and full address.

Class	Operation	Seeds			Graph				Classification		Relations	Runtime
		All	OW	Exp.	Comp.	Addr	Txes	Unexp.	Tagged	Exchanges		
C&C	cerber	4	0	4	1	448	904	0	5	320	0	1'00"
	glupteba	3	0	3	3	34	41	0	3	6	0	0'23"
	pony	4	0	4	2	27	1,548	1	17	2	0	0'33"
	skidmap	1	1	0	1	1	0	0	1	0	1	-
Clipper	aggah	1	0	1	1	7	8	1	3	2	1	0'21"
	androidclipper	1	1	0	1	1	0	0	1	0	1	-
	azorult	1	1	0	1	1	0	0	1	0	1	-
	bitcoingrabber	1	0	1	1	94	305	7	20	9	5	0'47"
	casbaneiro	1	0	1	1	59	96	2	32	11	4	1'03"
	clipbanker	1	1	0	1	1	0	0	1	0	1	-
	clipsa	9412	0	375	33	681	840	2	469	124	6	1'22"
	cliptomaner	1	1	0	1	1	0	0	1	0	1	-
	cryptoshuffler	1	0	1	1	507	2,331	18	134	180	12	4'10"
	masad	2	0	2	1	471	1,478	22	99	170	11	3'37"
	mekotio	3	0	2	2	146	385	8	35	53	17	1'12"
	mekotion40	3	0	3	3	145	311	1	25	46	8	7'29"
	mispadu	1	1	0	1	1	0	0	1	0	1	-
	mrpr0gr4mmer	2	0	2	1	577	1,570	7	274	145	2	1'21"
	n40	1	1	0	1	1	0	0	1	0	1	-
	phorpiex-tldr	15	14	1	1	11	55	0	6	2	3	0'21"
	phorpiex-trik	8	1	6	4	103	189	1	16	37	8	0'58"
	predatorthethief	1	0	1	1	410	1,627	19	91	139	9	1'45"
	protonbot	2	2	0	1	2	0	0	2	0	2	-
Ransomware	cryptorlocker2015	1	0	1	1	147	2,137	17	60	23	8	115'22"
	cryptxxx	1	0	1	1	621	1,836	9	115	233	7	2'38"
	dmalocker	11	0	11	1	482	1,372	1	91	145	4	1'22"
	globeimposter	3	0	1	1	601	5,240	15	72	249	4	3'01"
	locky	7076	0	7074	1	7,987	11,686	29	7,207	200	12	12'08"
	samsam	45	0	20	15	303	533	5	55	77	5	0'47"
	wannacry	6	0	5	2	30	520	0	13	10	0	0'24"

Table 2: Summary of forward-only explorations for the 30 malware families.

Class	Operation	Relations Found
C&C	cerber	0 -
	glupteba	0 -
	pony	0 -
	skidmap	1 exchange:coincola*
Clipper	aggah	1 exchange:binance
	androidclipper	1 exchange:luno.com*
	azorult	1 exchange:tradeogre*
	bitcoingrabber	5 exchange:bitmex, exchange:coinbase, exchange:okex, exchange:yobit.net, payment:payeer
	casbancero	4 exchange:bleutrade.com, exchange:exmo, exchange:localbitcoins, payment:coinpayments
	clipbanker	1 exchange:coinbase*
	clipsa	6 exchange:binance, exchange:bitfinex, exchange:changenow, exchange:localbitcoins, mixer:wasabi-fee, payment:coinpayments
	cliptomaner	1 exchange:tradeogre*
	cryptoshuffler	12 exchange:bittrex, exchange:btc-e.com, exchange:btctrade.com, exchange:cryptopay.me, exchange:cubits.com, exchange:exmo, exchange:poloniex, exchange:spectrocoin, mining:btcc-pool, payment:bitpay.com, payment:coinpayments, payment:webmoney
	masad	11 exchange:any-cash, exchange:binance, exchange:bitzlatto, exchange:coinbase, exchange:cryptonator, exchange:localbitcoins, exchange:whitebit, exchange:z-exchange.ru, gambling:1xbit, payment:payeer, payment:coinpayments
	mekotio	17 clipper:mekotio40*, exchange:binance, exchange:bitso, exchange:bittrex, exchange:btcturk, exchange:coinbase, exchange:hitbtc, exchange:localbitcoins, exchange:mercadobitcoin, exchange:paykrip, gambling:1xbit, miscabuse:mrpr0gr4mmer, mixer:wasabi-fee, payment:payeer, payment:coinpayments, service:gourl.io, service:muchbetter
	mekotio40	8 clipper:mekotio*, exchange:coingate, exchange:dsx-exchange, exchange:mercadobitcoin, gambling:foxbit, gambling:yolo-group, mixer:wasabi-fee, payment:coinpayments
	mispadu	1 exchange:mercadobitcoin*
	mrpr0gr4mmer	2 exchange:binance, payment:coinpayments
	n40	1 exchange:mercadobitcoin*
	phorpiex-tldr	3 exchange:cryptonator*, gambling:duckdice.io, misabuse:saveyourself
	phorpiex-trik	8 exchange:bitcoin.de, exchange:coinbase, exchange:cryptonator*, exchange:localbitcoins, exchange:matbea.com, exchange:poloniex, gambling:rollin.io, payment:webmoney
	predatorthethief	9 clipper:masad, exchange:binance, exchange:bitzlatto, exchange:btcbank.com.ua, exchange:cryptonator, exchange:localbitcoins, exchange:totalcoin, exchange:whitebit, payment:coinpayments
	protonbot	2 exchange:cryptonator*, exchange:hitbtc*
Ransomware	cryptotorlocker2015	8 exchange:bitcoin.de, gambling:betcoin-dice, gambling:fortunejack.com, gambling:luckybit, gambling:satoshibones, gambling:satoshidice, mining:deepbit, tormarket:silkroad2market
	cryptxxx	11 exchange:bitcoin.de, exchange:bitfinex, exchange:bitflyer, exchange:btc-e.com, exchange:coinbase, exchange:coingate, exchange:coins.co.th, exchange:huobi, exchange:localbitcoins, exchange:okcoin.com, exchange:xapo
	dmalocker	6 exchange:btc-e.com, exchange:localbitcoins, exchange:okex, exchange:poloniex, gambling:cloudbet.com, payment:coinpayments
	globeimposter	7 exchange:bittrex, exchange:btc-e.com, exchange:localbitcoins, gambling:coingaming.io, payment:bitpay.com, payment:easywallet, payment:webmoney
	locky	15 exchange:bitfinex, exchange:bitstamp, exchange:btc-e.com, exchange:coinbase, exchange:coins.co.th, exchange:exmo, exchange:localbitcoins, exchange:matbea.com, exchange:poloniex, exchange:xzxx.biz, gambling:cloudbet.com, mixer:bitcoinfog, payment:bitpay.com, payment:webmoney, tormarket:nucleusmarket
	samsam	6 exchange:bitcoin.de, exchange:cubits.com, exchange:localbitcoins, exchange:poloniex, exchange:yobit.net, tormarket:alphabaymarket
	wannacry	2 exchange:changelly, exchange:shapeshift

Table 3: Summary of relationships found with forward-only explorations. Tags with an asterisk at the end were found by multi-input clustering on the seeds, the rest were found by the exploration.

Algorithm 2 Cerber Oracle

```

1: procedure ORACLE(addr)
2:   for tx in addr.withdrawal_txes do
3:      $ins\_outs \leftarrow tx.ins\_count == tx.outs\_count == 1$ 
4:     if ins_outs then
5:        $o \leftarrow tx.outs[0].address$ 
6:        $n\_with \leftarrow o.withdrawal\_txes.size$ 
7:        $n\_dep \leftarrow o.deposit\_txes.size$ 
8:       if n_with == n_dep == 1 then
9:          $tx\_r \leftarrow o.withdrawal\_txes[0]$ 
10:         $txr\_ins \leftarrow tx\_r.ins\_count$ 
11:         $txr\_outs \leftarrow tx\_r.outs\_count$ 
12:        if txr_ins == txr_outs == 1 then
13:           $r\_addr \leftarrow tx\_r.outs[0].address$ 
14:          if r_addr == addr then
15:             $r\_val \leftarrow tx\_r.fee + tx\_r.out\_value$ 
16:            if tx.out_value == r_val then
17:              return True
18:            end if
19:          end if
20:        end if
21:      end if
22:    end if
23:  end for
24:  return False
25: end procedure

```

Algorithm 3 Pony Oracle

```

1: procedure ORACLE(addr)
2:    $ips \leftarrow []$ 
3:    $tx1, tx2 \leftarrow None, None$ 
4:   for tx in addr.txes do
5:     if tx in addr.deposit_txes then
6:       if tx.inputs.size <= 3 and tx.outputs.size <= 2 then
7:          $value \leftarrow tx.outs[addr].value$ 
8:         if tx1 then
9:            $tx2, tx2\_value \leftarrow tx, value$ 
10:           $ts1 \leftarrow tx1.block.timestamp$ 
11:           $ts2 \leftarrow tx2.block.timestamp$ 
12:           $delta \leftarrow (ts2 - ts1).total\_seconds$ 
13:          if delta <= 3600 then
14:             $ip \leftarrow decode(tx1\_value, tx2\_value)$ 
15:            if ip and is_public(ip) then
16:              ips.append(ip)
17:            end if
18:          end if
19:           $tx1 \leftarrow None$ 
20:        else
21:           $tx1, tx1\_value \leftarrow tx, value$ 
22:        end if
23:      else
24:         $tx1 \leftarrow None$ 
25:      end if
26:    else
27:       $tx1 \leftarrow None$ 
28:    end if
29:  end for
30:  return  $\frac{2 * ips.size}{addr.deposit\_txes.size} \geq 0.5$ 
31: end procedure

```

Algorithm 4 Glupteba Oracle

```

1: procedure ORACLE(addr, keys)
2:   for tx in addr.withdrawal_txes do
3:     for o in tx.outs do
4:       data  $\leftarrow$  None
5:       if hasattr(o.address, 'data') then
6:         data  $\leftarrow$  o.address.data.hex
7:       end if
8:       if data and len(data)  $\geq$  56 then
9:         for key in keys do
10:          iv  $\leftarrow$  unhexlify(data[: 24])
11:          tag  $\leftarrow$  unhexlify(data[-32 :])
12:          data  $\leftarrow$  unhexlify(data[24 : -32])
13:          plain_text  $\leftarrow$  aes_gcm_decrypt(iv, data, tag, key)
14:          if plain_text then
15:            return True
16:          end if
17:        end for
18:      end if
19:    end for
20:  end for
21:  return False
22: end procedure

```

Family	Seeds	OW	Clust	Addr.	Activity Period	BTC	USD	Family	Seeds	OW	Clust	Addr.	Activity Period	BTC	USD
R:Netwalker	66	0	65	329	20180216:20210528	3,131.7562	\$28,262,120	R:CryptoTorLocker2015	7	0	1	159	20131106:20170312	52.7342	\$15,068
R:Conti	26	0	26	35	20171204:20220207	410.8025	\$18,869,852	R:Black Kingdom	2	0	2	2	20200306:20210318	0.7233	\$15,015
R:Revil	7	0	6	77	20190823:20210611	369.5363	\$12,540,689	R:NotPetya	3	0	3	3	20170626:20230306	4.8104	\$13,770
R:DackSide	3	0	3	3	20210215:20210511	158.7086	\$9,124,091	C:casanero	1	0	1	19	20190307:20211109	1.6425	\$13,691
R:Locky	7,036	0	2	7,094	20160114:20170602	15,396.6934	\$7,833,613	R:XLockerV5.0	2	0	2	3	20170124:20170307	12.0000	\$12,568
R:Ryuk	26	0	26	39	20180811:20230322	866.8069	\$4,844,149	P:BTC-doubler.us	1	0	1	7	20170502:20180512	4.3549	\$12,527
R:RagnarLocker	1	0	1	1	20200727:20200728	414.0007	\$4,547,241	R:Xorist	3	1	3	11	20170523:20171001	4.3138	\$11,850
R:MoutLocker	1	0	1	1	20201103:20201104	298.5000	\$4,226,277	R:File-Locker	1	1	1	1	20120309:20230322	2.8771	\$10,832
R:BlackMatter	1	0	1	1	20210730:20210730	96.3863	\$4,068,295	C:cryptoctibale	8	0	4	13	20200228:20230308	0.7980	\$10,706
R:Mapok	1	0	1	996	20190617:20230408	316.0883	\$3,853,713	R:XTPLocker	1	0	1	4	20160509:20161110	19.1398	\$10,283
R:Eggregor	9	0	9	9	20201023:20201207	197.9554	\$3,129,599	R:StorageCrypter	1	0	1	5	20171110:20171208	0.9668	\$9,908
R:Tejodes	1	0	1	178	20171117:20200928	345.5527	\$2,798,239	P:bitcoindoubler.prv.pl	1	0	1	1	20160516:20220217	11.5914	\$9,758
P:Leancy	1	0	1	1	20131216:20171123	3,092.8482	\$1,896,583	R:Ranzy Locker	1	0	1	1	20210226:20210226	0.2100	\$9,729
R:CryptXXX	1	0	1	1,742	20160511:20161006	3,338.5464	\$1,877,833	R:Ransomix	1	0	1	1	20171121:20180704	1.2496	\$9,474
R:DMALockerV3	9	0	3	177	20160818:20211105	1,833.9286	\$1,757,464	P:120cycle	1	0	1	78	20140324:20140329	14.2823	\$8,264
R:MedusaLocker	3	0	3	26	20200603:20220121	56.1546	\$1,454,776	P:Minimalism10	1	0	1	1	20140227:20140327	13.0614	\$7,839
R:HelloKitty	1	0	1	1	20210712:20210712	32.3529	\$1,070,653	P:CRYPTOSX2	1	0	1	52	20150129:20180106	27.2141	\$7,490
R:Bitpaymer	1	0	1	1	20200813:20200817	90.0004	\$1,058,347	R:LamdaLocker	1	1	1	1	20170110:20170731	6.6692	\$7,430
C:cliptomaner	1	1	1	1	20191117:20230411	38.7643	\$1,004,397	R:ChupaCabra	1	1	1	1	20210513:20210923	0.1967	\$7,362
P:Cryptony	1	1	1	1	20130902:20150108	1,677.5167	\$886,689	R:AlbDecryptor	1	0	1	18	20200423:20211118	87.355	\$7,355
R:Qlocker	22	0	4	45	20210420:20220911	14.9621	\$786,335	C:protonbot	2	2	2	2	20180417:20210409	0.9430	\$7,349
R:Spora	1	0	1	2,129	20170105:20210102	616.2996	\$723,352	R:Vega	1	0	1	11	20210809:20210920	0.1463	\$6,989
R:SamSam	23	0	21	51	20160113:20170630	646.2379	\$583,508	P:doublebleitcoin.life	1	1	1	1	20151026:20201016	3.9236	\$6,853
R:CryptoLocker	2	0	1	968	20130907:20170710	1,511.7579	\$520,238	R:VenusLocker	2	0	2	5	20160719:20170217	6.8115	\$6,742
C:clipsa	576	37	784	284	20180821:20230324	25.9030	\$507,381	R:Phobos	1	1	1	1	20210720:20210720	0.1845	\$5,497
P:RockwellPartners	1	0	1	1	20140303:20150526	731.7512	\$383,136	R:Crypt Console	6	0	6	7	20160428:20170914	3.8135	\$4,254
P:bitcoindoubler.fund	1	1	1	1	20170803:20180127	49.5862	\$323,677	R:Predator	1	1	1	1	20180502:20190629	0.5559	\$3,522
P:MiniPonziCoin	1	0	1	1	20140218:20141022	445.9718	\$266,979	R:JigSaw	6	0	5	20	20160328:20170315	5.2538	\$3,369
P:Ponzi.io	2	0	1	33	20140204:20171030	369.3729	\$258,604	R:Qweuirksd	1	0	1	11	20181025:20181117	0.5407	\$3,301
P:BTC-doubler.com	1	0	1	1	20161226:20170723	176.3050	\$229,916	R:Black Ruby	1	0	1	1	20180205:20180213	0.3678	\$3,054
R:SunCrypt	1	0	1	1	20210719:20210719	7.4484	\$229,694	R:Bagi	1	0	1	8	20190503:20211114	0.3297	\$3,006
C:cliphanker	4	1	2	186	20160422:20230412	17.3412	\$210,100	R:XLocker	1	0	1	1	20170524:20170524	1.0000	\$2,476
C:phorpiextldr	15	14	2	18	20180531:20230302	19.8600	\$198,675	C:bitcoingrabber	1	0	1	16	20190130:20220807	0.2821	\$2,443
C:salve	1	0	1	1	20150102:20170714	651.4415	\$193,837	P:igjam.com	1	0	1	2	20131211:20170616	3.0101	\$2,291
C:n40	1	1	1	1	20170531:20191002	27.4139	\$187,461	R:HC6/HC7	6	0	6	6	20170729:20171127	0.3266	\$2,121
R:AES-NI	1	0	1	1	20140606:20210506	85.6786	\$183,968	P:world-btc.online	1	0	1	51	20170120:20170924	1.9167	\$2,082
R:NoobCrypt	1	0	1	28	20131018:20160715	550.4390	\$164,735	R:Bagi	1	0	1	8	20190409:20230403	0.9825	\$1,971
R:LockBit 2.0	2	0	2	51	20200929:20211008	4.9379	\$158,709	R:Cryptomat	1	0	1	4	20160429:20161116	2.6277	\$1,689
P:Nanondustryinv.com	1	0	1	2	20140816:20150213	480.1777	\$155,317	P:OpenPonzi	1	0	1	1	20150415:20160308	6.0855	\$1,490
C:mrp0gr4mmer	3	0	2	377	20191226:20230410	9.0049	\$130,166	R:ComradeCircle	1	0	1	1	20161018:20161018	2.0332	\$1,292
R:WannaCry	5	0	4	6	20170331:20230306	59.5107	\$119,094	R:Avaddon	1	0	1	1	20210512:20210512	0.0241	\$1,193
R:SynAck	1	0	1	1	20150329:20180122	99.2369	\$114,116	R:Cryptohitman	2	0	1	15	20160427:20190923	1.9001	\$1,049
R:GlobalmPoster	1	0	1	1	20141123:20171227	190.1388	\$97,266	R:Encryptd	1	0	1	1	20191209:20210704	0.1107	\$1,015
P:bitcoincopy.site123.me	1	1	1	1	20160725:20181112	17.9770	\$91,826	C:androidclipper	1	1	1	1	20180223:20181206	0.1287	\$942
P:1hourbtc.pw	1	0	1	433	20161031:20210104	43.4324	\$89,381	R:Ecovector	1	0	1	2	20160511:20160521	2.0041	\$888
R:AvosLocker	1	0	1	1	20210909:20210909	1.8047	\$83,722	R:Decryptiomega	2	1	2	2	20190718:20190807	0.0600	\$705
P:10PERCENTBTC	1	0	1	17	20140812:20230403	115.1086	\$81,443	P:bitcoinprofit2	1	0	1	1	20160221:20200301	0.5118	\$672
P:Ponzi120	1	0	1	1	20140225:20140301	144.9008	\$77,763	R:Kelly	1	0	1	1	20200909:20200909	0.0500	\$502
C:masad	2	0	2	114	20170117:20230327	5.6161	\$76,321	R:Gula	1	1	1	1	20180919:20181117	0.0425	\$268
C:cryptoshuffler	1	0	1	156	20160303:20210831	65.1258	\$71,890	R:DMALocker	1	0	1	1	20151228:20151228	0.6000	\$253
C:phorpiextrik	7	1	7	9	20160814:20230104	11.4488	\$68,726	P:7dayponzi	1	0	1	1	20150321:20150526	0.7735	\$202
P:GrandAgoFinance	1	0	1	1	20140924:20150129	178.7435	\$64,765	R:Vevelocker	1	0	1	5	20170923:20171116	0.0227	\$152
R:Ako	1	0	1	14	20200107:20210530	5.9315	\$54,487	C:gagab	1	0	1	1	20200518:20200826	0.0130	\$141
P:investorbitcoin.com	1	1	1	1	20160531:20171003	47.3833	\$51,444	R:Black Mamba	1	1	1	1	20170619:20170619	0.0299	\$105
P:1getpaid.me	1	0	1	2	20150607:20160429	176.7322	\$43,250	P:invest4profit	1	0	1	11	20161029:20170929	0.0317	\$42
R:APT	2	0	1	3	20160915:20171130	30.3040	\$39,348	R:LockOn	1	1	1	1	20170826:20170826	0.0050	\$22
R:Globev3	5	0	5	19	20160925:20180220	40.4265	\$36,493	R:BlackRouter	1	1	1	1	20181219:20190528	0.0023	\$15
P:LaxoTrade	1	1	1	1	20140820:20181218	69.7857	\$25,998	R:NullByte	1	0	1	1	20160903:20160903	0.0172	\$10
R:Flyper	2	0	1	31	20161004:20181013	7.4181	\$25,011	R:Exotic	1	0	1	1	20161028:20161028	0.0064	\$4
C:mekotio40	3	0	2	20	20180125:20221013	2.4035	\$24,950	R:Buchi	1	0	1	1	20160402:20160402	0.0090	\$4
P:Twelverized	1	0	1	1	20140302:20140413	38.8018	\$24,599	R:PopCornTime	1	1	1	1	20161214:20161230	0.0028	\$2
R:Globe	3	0	3	87	20160605:20170115	31.9565	\$23,577	R:Phoenix	1	0	1	1	20161214:20161214	0.0026	\$2
C:predatortheft	1	0	1	39	20190809:20201115	2.4541	\$23,461	R:CryptoHost	1	0	1	1	20160612:20160612	0.0020	\$1
C:mekotio	2	0	2	14	20170712:20220530	1.5535	\$22,189	R:WannaRen	1	0	1	1	20200408:20200408	0.0001	\$1
C:azurub	1	1	1	1	20180719:20190324	5.9413	\$21,872	R:7ev3n	1	0	1	1	20160429:20160429	0.0011	\$1
R:LockBit	1	0	1	1	20201201:20201214	1.1114	\$21,007	R:Chimera	1	0	1	1	20151107:20151108	0.0005	\$0
C:cliphboardwalletjacker	4	0	1	34	20180302:20220318	2.6740	\$18,721	R:WannaSmile	1	0	1	1	20171110:20171110	0.0000	\$0
C:mispadu	1	1	1	1	20190928:20220927	0.7798	\$18,473	R:CTB-Locker	1	0	1	1	20160313:20160418	0.0002	\$0
R:EDA2	2	0	2	33	20140406:20170725	10.8830	\$18,299	R:TeslaCrypt	1	0	1	1	20150315:20150315	0.0001	\$0
P:bestdoubler.eu	1	0	1	8	20161223:20170831	13.2392	\$18,149	P:btcgains	1	1	1	1	-	0.0000	\$0
R:Sam	1	0	1	1	20160927:20160927	29.0000	\$17,538	Total	8,021	42	383	17,040	20120309:20230412	40,282.3312	\$123,357,041

Table 4: Operations in the Ransomwhere [30], Ponzi [9], and Clipper [73] datasets, ordered by USD revenue using the *DD-OW+MI-DC* estimation. The last row captures total revenue of all 141 operations.

Category	Definition
abuse	An abuse report details instances of abuse or attempted abuse experienced by the person who authors the report, such as extortion, scams, demands for ransom, unauthorized withdrawals, replacement of addresses in the clipboard when copying/pasting cryptocurrency addresses, or theft. The report must point to how the abuse happened by providing a description or presenting evidence of the occurrence. It never promotes or recommends services of any kind.
notabuse	A report is notabuse if it does not describe an abuse attempt suffered by the reporter but rather presents a personal opinion, calls for action, advertises a service, claims that an address is involved in terrorism, or that it belongs to hackers. A notabuse report may initially describe an abuse to look more convincing, but its main message always turns to be promoting something like a service, a personal opinion, a call for action, a claim, or a defamation. In general, a notabuse report may sound phishy or seem like a scam.
scam	A scam is an attempt to defraud a person or group by gaining their confidence. Scammers are dishonest and deceive their victims to gain some benefit at the expense of the victim. Scammers may lie to the victims with texts that look legitimate but are hard to verify, commonly exploiting trust, compassion, greed, vulnerability, or lack of awareness in their victims. Popular scams involve investment scams, romance scams, giveaway scams, fake services like e-shops or funds recovery, and requests for donations to unverified addresses.
extortion	Extortion is the practice of obtaining benefit through coercion. The victim is threatened with something bad happening in the future if it does not make a payment to the extortionist. The threat may involve revealing compromising data from the victim, the victim suffering physical harm, or the extortionist damaging assets belonging to the victims. To make the threat more credible the extortionist may claim to know some password of the victim.
unauthwithdrawal	Unauthorized withdrawal refers to a third party withdrawing or transferring funds belonging to the victim without proper authorization or consent by the victim. Unauthorized withdrawal includes theft through credential stealing and the use of clipper malware to replace addresses copied into the clipboard with addresses belonging to the attacker when they are pasted.
ransom	Attacks that demand a ransom payment from the victim in order to recover access to data encrypted or deleted by the attacker. The data may have been stored in a device or server owned by the victim. The attackers leave a ransom note explaining how to make the payment or how to contact the attackers for instructions.
deleteransom	A ransom scheme where the attacker demands a ransom payment from the victim in order to recover data removed (and possibly exfiltrated) by the attacker, typically stored in databases.
clipper	Unauthorized withdrawals enabled by a clipper. A clipper is a type of malware that replaces cryptocurrency addresses copied to the victim's clipboard with an address belonging to the attacker, without the victim being aware of the replacement. When the victim pastes the address copied to the clipboard as destination of a cryptocurrency transaction, the funds are unwillingly transferred to the attacker's address.
bomb	A type of extortion where attackers threaten with detonating an explosive device. The attacker claims the bomb has been hidden in the building where the victim lives or works and that the explosion will affect many people.
harm	A type of extortion where attackers threaten with physically harming the victim unless a payment is made. Beyond moral damage, the attacker threatens to physically hurt the victim or destroy its property. The victim is supposed to pay for the attacker to remain inactive.
ddos	A type of extortion where attackers threaten with launching a distributed denial of service (DDoS) attack against a target server, website, or service. The DDoS attack will cause the service to be unavailable. The victim is requested to make a payment so that the DDoS attack does not happen.
reputation	A type of extortion where attackers threaten with damaging the reputation of the victim by sending threats with insults to several recipients (for instance, other web site owners or bloggers) on the victim's behalf.
investment	A type of scam that entices individuals to invest with the promise of high returns or guaranteed profits. Investment scams may use fake testimonials, fabricated success stories, or false celebrity endorsements to lure victims to invest. Scammers may create fake investment websites, platforms, or social media profiles to appear legitimate and trustworthy. After the initial investment, victims are not allowed to withdraw their profits or recover their original investment. Instead, attackers may request new fees and investments to allow withdrawal.
advancefee	A type of scam where scammers promise the victim with a significant share of a large sum of money, in return for a small advance fee. The fraudster claims the up-front payment will be used to obtain the large sum. The scammer may impersonate a wealthy individual such as a Nigerian prince or an organization such as a tax revenue service or the social security administration.
romance	A type of scam where victims are deceived by scammers to establish fake romantic relationships. Scammers create false personas, often online, and manipulate their victims' emotions to gain their trust and affection. Once an emotional bond is formed, the scammer demands payments from the victim under false excuses.

Category	Definition
eshop	A type of scam where scammers create an online shop website or e-commerce store where victims may place orders for buying items but receive nothing, or may receive a product that does not match the purchased one, e.g., a counterfeit or inferior product. The fake e-shop is made to look legitimate and lures buyers through unusually low prices or by offering illegal products. The e-shop may be offered as dark markets only accesible through Tor.
fundsrecovery	A type of abuse report in which the author specifically mentions to being a victim of a fake funds-recovery service. The victim may describe how the financial losses occured, or how the perpetrator was contacted, but the main purpose of the report must always be to give details about the fake funds-recovery service or person, and never promote a service or person that recovers funds.
sextortion	A type of extortion where attackers specifically claim to have a video or photos of the victim in sexual acts of any kind. The attackers may claim to have access to the victim’s webcam. The attackers must threaten to share the video or photos with explicit sexual content with friends, family, or colleagues of the victim unless a payment is made. The report is not sextortion if it does not clearly mention sex-related material.
giveaway	A type of scam where scammers promise that upon reception of a cryptocurrency deposit from the victim they will return to the sending address twice or more the received cryptocurrency amount. The scammers often lure victims to participate in the giveaway scam by impersonating famous persons, organizations, or cryptocurrency influencers, or by advertising the giveaway as a promotion or contest.
employment	Employment scams advertise fake job opportunities or services for assisting individuals in finding employment. They are also known as job scams. Victims of employment scams typically do not receive a payment for the tasks they completed, receive only partial payment for their work, or are required to pay a fee to be able to withdraw their earnings. The offered jobs are often shady such as writing fake reviews, generating traffic to websites, money-laundering, or reshipping packages.

Table 5: List of definitions for the LLM