Security Monitoring System Applied to IoT

Author: Marianne AYMARD
Supervisor: Prof. Dr. Iván PAU

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in the
ETSI Sistemas Informáticos
ETSI Sistemas de Telecomunicación

July 18, 2019
Declaration of Authorship

I, Marianne AYMARD, declare that this thesis titled, “Security Monitoring System Applied to IoT” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____________________________________________

Date: ______________________________________________
“The true mark of professionalism is the ability to respect everyone else for their styles and always find something positive in every dining experience and highlight it in your thoughts and words.”

Johnny Iuzzini
This project proposes a Security Monitoring System (SMS) applied to IoT applications, to monitor the system and analyze the information injected into the environment, in order to detect and prevent threats and attacks.

Because the scope of IoT is so vast, it facilitates the exchange of information and services, impacting the security and privacy of the involved parties. In consequence, the data it produces and collects is vital; making it the most valuable and desirable asset in today’s technological world.

Security Monitoring Systems focus on two approaches to reduce attacks and protect this data: reduce the environment vulnerabilities and reduce the amount of threats the environment can suffer.

The last method perfectly fits with IoT environments, being these systems perfect targets due to the vast amount of data they manage. Also, their inputs are completely variable which makes the vulnerabilities reduction an impossible task to execute.

The SMS developed in this project was built using out-of-the-box technologies like Elastic SIEM, presenting several challenges that were solved successfully.
Acknowledgements

This project was made in a very short period of time between challenges and difficulties, none of this would have been possible without the help and support of a very special group of people.

Thanks to Prof. Fernando Pescador who always supported me with every opinion I had, and for making possible this presentation.

Special thanks to Ramón López who helped me during the Master and significantly contributed in the development of this thesis.

Thank you to my classmates Andreas Beck, Javier Landaeta, Alejandro Mateo, Matías Suárez and Miguel Del Castillo, who became my friends and helped get through this program with a smile.

Thank you to my family, who has supported me since day one in all my decisions and achievements.

Thank to my Manager, Nieves González and my Mentor, Andrés Jiménez for supporting me during this development and believing in me no matter the circumstances.

Thank you my friends Ruchi Shah for editing part of this these and Susana Moreira for believing in me since the beginning.

Finally, thank you to my thesis advisor Prof. Iván Pau for proposing this thesis and giving me the opportunity to make it happen.
Contents

Declaration of Authorship iii
Abstract vii
Acknowledgements ix

1 Introduction 1
  1.1 Problem Statement .................................................... 1
  1.2 Objectives .......................................................... 2
  1.3 Limitations .......................................................... 2

2 State of the Art 3
  2.1 The Elastic Stack .................................................... 3
    2.1.1 Elasticsearch ................................................... 4
    2.1.2 Logstash ......................................................... 4
    2.1.3 Kibana .......................................................... 5
    2.1.4 Beats ........................................................... 5
    Types of Beats ....................................................... 5
  2.2 Elastic SIEM ........................................................ 5
    SIEM components ...................................................... 6
  2.3 Eclipse Mosquitto ................................................... 6

3 Description 7
  3.1 Methodology ......................................................... 7
  3.2 System Requirements ............................................... 7
  3.3 Analysis ............................................................. 8
    Security Monitoring System ........................................... 8
    How does a Security Monitoring System Works ..................... 10
    Benefits of Security Monitoring Systems .......................... 10
    Security Monitoring System’s SIEM Approach ....................... 10
    Security Information and Event Management ......................... 10
    How SIEM works ...................................................... 11
    SIEM Benefits ......................................................... 11
    SIEM Architecture and Components .................................. 12
    SIEM & IoT Integration ................................................. 15
    SIEM Comparison ...................................................... 17
    SIEM Solution Election and Justification ............................ 20
  3.4 Design ............................................................... 21
  3.5 Implementation ...................................................... 24
    Monitoring Environment Definition .................................. 25
    Technologies Installation and Configuration ......................... 26
    3.5.1 Hardware ......................................................... 26
    3.5.2 Software ........................................................ 27
xii

Installing and Configuring Technologies .................. 28
3.5.3 Results ........................................... 32
Monitoring IoT Sensors .................................... 32
Monitoring System Events ................................. 34

4 Conclusions ............................................. 41
  4.1 Conclusions .......................................... 41

Bibliography .............................................. 43
List of Figures

2.1 The Elastic Stack ........................................... 4
3.1 Security Monitoring System’s Philosophy ......................... 8
3.2 SIEM Logical Architecture ................................... 13
3.3 SIEM Physical Architecture ................................... 14
3.4 IoT Application Architecture ................................... 16
3.5 SIEM & IoT Integration Architecture ............................ 17
3.6 SIEM Gartner Magical Quadrant. October 2018 ................... 18
3.7 Security Monitoring System’s Design Defined in this Project .... 22
3.8 Itemized Security Monitoring System’s Architecture ............. 23
3.9 Modified SMS Design .......................................... 24
3.10 Messages being sent and received ............................... 33
3.11 Sensors logs shown in Elasticsearch ............................. 33
3.12 Sensor Data visualization in Kibana ............................. 34
3.13 Elasticsearch Index ........................................... 34
3.14 New SSH session .............................................. 35
3.15 Kibana Logs .................................................. 35
3.16 SSH Accepted Password Log .................................. 35
3.17 SSH Failed Password Log ...................................... 35
3.18 Elastic SIEM Events ........................................... 36
3.19 SIEM Overview ............................................... 36
3.20 SIEM Host Events ............................................. 37
3.21 SIEM Network Events .......................................... 37
3.22 Auditbeat Events ................................................ 38
3.23 Filebeat Events ................................................ 38
3.24 Elastic SIEM Timeline .......................................... 39
To my country, Venezuela, and all it's hard working migrants that wish to go back, someday…
Chapter 1

Introduction

The Internet has completely transformed human life. It has the ability to connect many everyday devices that a person can use, and form a massive network called the Internet of Things (IoT).

This network, connected to the Internet, is an emerging technology that commands devices that controls humans, and has already become the world’s number one data producer, collecting data from people’s actions and sharing it with other devices in the network.

The Internet of Things is present everywhere. It has affected human communications and interactions, and is even replacing certain actions and thoughts. Because the scope of IoT is so vast, it facilitates the exchange of information and services inside the network, impacting the security and privacy of the involved parties. In consequence, the data it produces and collects is vital; making it the most valuable and desirable asset in today’s technological world.

All this data needs to be processed and analyzed in order to extract information and be useful. To do this, data goes through several stages of collection, preparation, verification, processing, interpretation and storage. Each part of this procedure exposes the data to threats, making this critical information vulnerable. As a result, the usage of a monitoring system while processing data is increasingly necessary.

Monitoring Systems are used to obtain real time information about the data’s state. They are also used to detect, alert and prevent failures and attacks, as well as visualize the data in direct. Attacks occur the most in places where the monitoring systems are not completely secure.

Having a secure and protected environment where data processing can be done is very important in today’s industry. Protecting data has become the most significant challenge for organizations meanwhile data has evolved to be the most precious possession companies have.

Implementing Security Monitoring Systems is the present and future goal. Proposed in this project is the first step for this important and necessary solution.

1.1 Problem Statement

IoT applications are the most valuable asset in today’s technological world. The fact that they are involved in almost every aspect of a person’s life, makes them even more desirable. This applications are today’s number one data producers, carrying with them private, confidential, important and priceless data, that can be used in many ways with multiple purposes.

Most of the time, this data is not treated, moved and/or stored in a secure way; making it vulnerable and the perfect target for attackers.
To detect and prevent attacks, the failures and consequences they can cause; it is convenient to have a secure monitoring system. A way to control and analyze the application’s operation and performance, making it easy to detect, alert and prevent possible assaults.

This project intends to solve the problem described above, by providing a Security Monitoring System for IoT Applications, where sensor data, machine logs and general network traffic will be audited and analyzed in order to guarantee a secure and reliable environment.

1.2 Objectives

1. To advance on the secure monitoring of IoT applications.
   
   (a) To define a generic monitoring architecture.
   
   (b) To analyze the existing SIEMs and their application in IoT.
   
   (c) To theoretically design a Security Monitoring System to be applied to the generic architecture.
   
   (d) To implement the theoretical SMS based on SIEM.

1.3 Limitations

Until now, the way data monitoring has been done its considered to be reckless and unprotected. One of the main reasons of this practice is the lack of technologies that allow data owners to totally secure and protect their assets. This absence has caused millions of attacks, interventions, exposures and losses, that has affected the way users trust IoT applications; modifying how they use them, creating concern and paranoia around IoT and its different functions.

Lately, technologies promising a secure method to manage data has been released. However, they haven’t had enough time to prove themselves and show how they really work and provide secure data management and monitoring.

Although everyday that passes it seems that the problem will be solved sooner than later, and despite everything that’s known until now; there is no technology in today’s market that can guarantee 100% secure and protected data monitoring, being threats still present at the end of the day.
Chapter 2

State of the Art

2.1 The Elastic Stack

The Elastic Stack is the new name of what it was *The ELK Stack*, the acronym of three open source projects: Elasticsearch, Logstash and Kibana; commonly used for monitoring, troubleshooting and securing IT environments [16].

The ELK Stack project started with **Elasticsearch**, an open source, distributed, RESTful, JSON-based search engine; easy to use, scalable and flexible. It grew with **Logstash** and **Kibana** when it users wanted to ingest and visualize the logs they searched. The community got larger, new use cases appeared everyday and users kept pushing boundaries with the stack; so in 2015, a family of lightweight, single-purpose data shippers were introduced to the ELK Stack project; and they called them **Beats**.

This new addition turned the stack into a four legged project and led to a renaming; known today as **The Elastic Stack**[16].

The Elastic Stack is so popular because it fulfills a necessity in the log management and analytics space. It eases the application and IT infrastructure monitoring and analysis task and overcomes the challenges of monitoring highly distributed, dynamic and noisy environments. It also helps by providing users with a powerful platform that collects and processes data from multiple data sources, stores it in one centralized data store that can scale as data grows, and that provides a set of tools to analyze the data [16].

Figure 2.1 illustrates The Elastic Stack and its components, and it also shows how they communicate between them.
Chapter 2. State of the Art

2.1.1 Elasticsearch

Elasticsearch is the living heart and soul of what is today the world’s most popular log analytics platform \[1\]. The role it plays is so significant that its name has become a synonym with the name of the stack itself. Primarily used for search and log analysis, Elasticsearch has become one of the most popular database systems available today.

Elasticsearch is a modern search and analytics engine based on Apache Lucene \[21\]; it is completely open source and categorized as a NoSQL database. It stores data in an unstructured way and it allows the use of SQL statements to interact with the data.

Unlike most NoSQL databases, Elasticsearch has a strong focus on search capabilities and features; making it necessary to use its extensive REST API as the easiest way to get data.

Also, in the data analysis context, it is used together with the other ELK Stack components, as data indexing and storage.

2.1.2 Logstash

Logstash is the stack’s workhorse. Its log aggregator that collects data from various input sources, executes different transformations and enhancements and then dispatches the data to various supported output destinations \[1\].

Thanks to a large ecosystem of plugins, Logstash can be used to collect, enrich and transform a wide array of different data types.

Logstash is a crucial component of the Elastic Stack.
2.1.3 Kibana

Kibana is a browser-based user interface that can be used to search, analyze and visualize the data stored in Elasticsearch [1]. It is especially renowned and popular due to its rich graphical and visualization capabilities that allow users to explore large volumes of data.

2.1.4 Beats

Beats are a collection of open source log shippers that act as agents installed on the different servers in an environment for collecting logs or metrics. Written in Go, these shippers were designed to be lightweight in nature; they leave a small installation footprint, are resource efficient, and function with no dependencies [1].

The data collected by the different beats varies. Log files in the case of Filebeat, network data in the case of Packetbeat, system and service metrics in the case of Metricbeat, Windows event logs in the case of Winlogbeat, and so forth. In addition to the beats developed and supported by Elastic, there is also a growing list of beats developed and contributed by the community.

Once collected, a beat can be configured to ship the data either directly into Elasticsearch or to Logstash for additional processing. Some of the beats also support processing which helps offload some of the heavy lifting Logstash is responsible for.

Types of Beats

1. Filebeat: Used for collecting and shipping log files.
2. Packetbeat: Captures network traffic between servers, and can be used for application and performance monitoring.
3. Metricbeat: Collects and ships various system-level metrics for different systems and platforms. It supports internal modules for collecting statistics from specific platforms.
4. Winlogbeat: Designed specifically for collecting Windows Event logs. It can be used to analyze security events, updates installed, and so forth.
5. Auditbeat: Used for auditing user and process activity on Linux servers. It can be used to identify security breaches like file changes, configuration changes, malicious behavior, etc.
6. Functionbeat: Designed for monitoring cloud environments. It is defined as a “serverless” shipper that can be deployed as a function to collect and ship data into the ELK Stack.

2.2 Elastic SIEM

Security Information & Event Management with the Elastic Stack is the new solution presented by the Elastic Stack. It introduces a new set of data integrations for security use cases, and a new dedicated application in Kibana that lets security practitioners investigate and triage common host and network security workflows in a more streamlined way [9].
SIEM enables analysis of host-related and network-related security events as part of alert investigations or interactive threat hunting. It is built to support near real-time monitoring, powerful search capabilities, and responsive custom dashboards. Perform ad-hoc queries in rapid succession to test your hypotheses and quickly surface potential threats [4]. Additionally, it is cloud-native and deployable anywhere.

SIEM can collect from all relevant data sources, from the network infrastructure to endpoint agents. It also collects Linux audit framework data to monitor system details and file integrity.

**SIEM components**

1. Beats
2. Elasticsearch
3. Kibana

### 2.3 Eclipse Mosquitto

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers [3].
Chapter 3

Description

3.1 Methodology

The development methodology used in this project was The Waterfall Model, breaking down all the activities into linear sequential phases.

The Security Monitoring System development was divided in four stages:

- System Requirements
- Analysis
- Design
- Implementation

Where each phase depended on the deliverables of the previous one, in order to deliver a fully functional system with no performance errors.

The following sections explain in detail each of the four phases previously mentioned.

3.2 System Requirements

As is well known, technology’s evolution has made organizations more and more unprotected as days go by.

With IoT’s appearance and the invasion of its wide range of devices, companies has been forced to use different communication protocols that were not that common before the IoT era; developing new vulnerabilities and becoming a target for attackers.

For this reason, organizations need to implement certain measures that can guarantee them peace and quiet in a secure way.

Therefore, the Security Monitoring System (SMS) presented in this project intends to provide this tranquility by implementing some security methods that will allow the environment to function in a secure and protected way on its daily basis.

The system will have the following features:

1. It will be capable of obtaining real-time data about:
   
   (a) Communication protocols, in this case MQTT and HTTP.
   (b) IoT sensors logs.
   (c) System’s logs and configuration changes.
   (d) Security events, like authentication errors or failed login attempts.
2. It will analyze the collected data through different filters and analyzers.
3. It will process the data.
4. It will generate alerts when suspicious events are detected.
5. It will display the information on interactive dashboards.
6. It will generate reports based on the processed data.

These actions are vital for any organization functioning today. Every company should be supported by this kind of systems. This project will handle the exposure organizations are living, and will give them a peace of mind regarding who is accessing their system and making the right changes.

3.3 Analysis

Security Monitoring System

A Security System (SS) literally means “a method by which something is secured through a system of interworking components and devices”. [23] This definition is the foundation of the Security Monitoring System, which description is similar but its performance and functionality varies in a significant way.

A Security Monitoring System (SMS) is an automated procedure that process and analyses indicators of potential security threats, detecting suspicious behaviors or unauthorized changes on the system, network, etc., defining which types of behavior should trigger alerts, and taking action on alerts as needed, in order to enhance protection and trust [5]. Figure 3.1 illustrates a SMS’s philosophy. showing the main tasks it does and how it handles them.

![Figure 3.1: Security Monitoring System’s Philosophy](image)

Threats and Vulnerabilities are a system’s worst nightmare. It is true that both of them are always present, but there are different ways to reduce or even eliminate them. Although developers work hard to make their systems secure in every single way, attackers always manage to find vulnerabilities and send threats. If the system is not secure enough or is not properly protected, this threats can cause irreversible damage and incredible losses.
To fight this, Security Monitoring Systems focus on two (2) ways of eluding attacks, ergo eliminating harms.

1. Reducing attacks by decreasing vulnerabilities
2. Reducing attacks by decreasing threats

- **Reducing attacks by decreasing vulnerabilities**

  The **attack surface** is the total sum of resources exposed to exploit within a system. Defending this surface was less complicated when a defined "perimeter" existed, neatly separating the system’s assets from the outside world [22]. Today’s technologies (IoT, Cloud Computing and Software-Defined Networking) have dissolved the perimeter, making the attack surface grow exponentially; causing applications to face unwanted, deliberate or accidental events that may result in damage, often produced by exploiting one or more vulnerabilities [22].

  According to the National Institute of Standards and Technology (NIST), a vulnerability is, "a flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system’s security policy."[2]

  To reduce vulnerabilities and therefore prevent attacks, several approaches are proposed:

  1. Attack Patterns for testing purposes and to ensure that potential vulnerabilities are prevented. They can be used to highlight areas which need to be considered for security hardening in a software application [22].
  2. Eliminate Complexity to avoid poor policy management or incomplete information during rule creation, which can lead to technical policy mistakes and an elevated possibility of human error and risk [22].
  3. Anomaly Detection to identify rare events which raise suspicions by differing significantly from the majority of the data.
  4. Control Endpoints by gaining visibility into what’s happening on them. Independent process monitors keep all endpoints under constant surveillance and provide alerts when endpoint behaviors deviate from the norm. This will significantly reduce the impact on the attack surface [22].
  5. Visualizing vulnerabilities by creating a real-time model of what could happen in the context of network movement. Collecting suspicious data and submitting it to analysis in order to determine configuration assessments, traffic flow and quantitative risk.

- **Reducing attacks by decreasing threats**

  Lowering the amount of threats a system can receive is not an easy task, there are just too many threats out there to effectively prevent them all. Although is nearly impossible to guarantee a threat free environment, there are certain actions that help accomplish this job.

  1. Two-factor authentication to confirm user’s claimed identities by using a combination of two different factors: something they have and something they know.
  2. Encryption to encode information in such a way that only authorized parties can access it and those who are not authorized cannot.
How does a Security Monitoring System Works

Security Monitoring Systems work on the simple concept of securing entry points through different methods to reduce vulnerabilities and threats.
In this age, it is usual to use technology to supervise systems, equipment, and networks performance, that is fundamental to allow businesses operations [18]. But, even if technology is something essential for a company’s performance, that does not mean it’s perfect; this is the reason why Security Monitoring Systems incorporates human observation ad combine it with technology to provide a safer environment.

Vulnerabilities that originate critical situations may appear at any time. Therefore, in an environment where computer infrastructure is important, it will be necessary to control its correct operation, so that a possible error does not end up affecting the service given to users.
Security Monitoring Systems are responsible for controlling the technology used in order to analyze their real-time operation and performance, and to detect and alert about possible errors.

Benefits of Security Monitoring Systems

The main benefit of a SMS is that it increases productivity. This is accomplished because it guarantees certain actions that allow administrators to work in a panic free environment without being concerned about threats and attacks.
In addition, SMS have more benefits that help companies in many ways. Below are some of the reasons why a Security Monitory System is necessary [18]:

1. It improves the use of the hardware.
2. It prevents incidents.
3. It detect negative events in a faster way.
4. It reduces problem resolution time.

Security Monitoring System’s SIEM Approach

As mentioned until now, SMS has several approaches on how to monitor data and reduce threats in the best way.
Security Information and Event Management (SIEM) is a more complete and powerful method that will contribute to the fulfillment of the SMS’s main goal of analyzing data and preventing attacks.

Security Information and Event Management

The term SIEM was made up in 2005 by Mark Nicolett and Amrit Williams, in Gartner’s SIEM report, Improve IT Security with Vulnerability Management. They proposed a new security information system, on the basis of two previous generations.
Security Information and Event Management is the solution that provides the information of what is happening on a system and/or network in real-time, in order to help IT teams fight against security threats in a more proactive way. [7]
SIEM solutions are a combination of two security disciplines:
3.3. Analysis

1. Security Event Management (SEM): Analyzes event and log data in real-time to provide event correlation, threat monitoring and incident response[7].

2. Security Information Management (SIM): Retrieves and analyzes log data and generates a report[7].

This union has made SIEM a critical solution for organizations that want a complete vision and control over their systems in real-time. Being one of the most implemented security approaches in today’s technological world.

How SIEM works

SIEM is an approach that has been present for more than a decade. In the past, SIEMs required meticulous management at every stage of the data pipeline - data ingestion, policies, reviewing alerts and analyzing anomalies [6]. As time passes, SIEMs are evolving and getting smarter at pulling data together, and using AI techniques to understand what type of behavior constitutes a security incident.

SIEMs work by collecting log and event data that is generated by host systems, network, security devices and applications. They do this by deploying a collection of agents throughout the organization’s technology infrastructure, and then they gather all the data on a centralized platform.

SIEM software identifies this data (network anomalies, antivirus events, security incidents, firewall logs, and sorts it into categories, such as failed and successful logins, malware activity and other potentially malicious activity. When the software identifies events that could be a threat to the organization, alerts are generated to indicate a possible security problem. Alerts can be established with different priority levels using a set of pre-defined rules. Administrators can then set rules and thresholds to define what type of anomaly is considered a security incident [7].

SIEM Benefits

As mentioned before, SIEM is a powerful method for threat detection, real-time reporting and long-term analytics of security logs and events. This solution can be amazingly useful for protecting organizations of all sizes. Some of the benefits a SIEM solution provides, are [7]:

1. Efficiency enhancement.
3. Cost reduction.
4. Greater log analysis, retention and reporting.
5. IT Compliance.

Since SIEM solutions are capable of collection logs from multiple sources, they permit administrators provide a faster response to potential security breaches. This allows threats to be identified in an early stage, ensuring a minor impact on the organization. This kind of security solution allows IT teams to see the bigger picture and centralized protection to the environment.
SIEM Architecture and Components

A SIEM solution is formed by several components that collect log and event data from systems, and turn it into actionable security insights. To accomplish this task, a SIEM relies on the following elements and actions [6].

1. Data Aggregation: Collects and aggregates data from host systems, network devices, etc.
2. Threat Intelligence Feeds: Combines internal data about threats and vulnerabilities with third-party’s.
4. Analytics: Uses Machine Learning and statistical models to identify relationships between data.
5. Alerting: Analyzes events and sends alerts when an anomaly is detected.
6. Dashboards: Generates visualizations to show real-time data situation and allow administrators to identify patterns and anomalies.
7. Compliance: Gathers log data from different security standards and generates reports.
8. Retention: Stores long-term historical data.
10. Threat Hunting: Enables administrators to run queries on log and event data to uncover threats.
11. Incident Response: Helps identify and respond to security incidents.
12. SOC Automation: Automatically responds to incidents by orchestrating security systems.

As shown in the Figure 3.2, a SIEM solution has many constituents that make its purpose possible. This picture represents a SIEM’s Logical Architecture, showing not only components, but actions that determine how it functions.
As referred to before, SIEM solutions collect data from various systems, devices and applications into a single common format. This data is normalized and then run through a policy engine where defined policies enrich the events, de-duplicate them, and in some cases create new ones to promote it into the SOC operator console. SIEM solutions are typically built in a hierarchical way. As shown in Figure 3.3, the SIEM Central Engine is the piece in charge of filtering, analyzing, monitoring logs, applying policies and creating alerts. Also, it sends logs to storage and ships the produced information to the Presentation Layer, to be visualized and analyzed by the users.
Chapter 3. Description

Figure 3.3: SIEM Physical Architecture

- Log Source: A Log Source is any device that can capture information and is capable of generation logs. Some of them could be, sensors, routers, switches, servers, etc.

- Log Collection: Process where all records are shipped to the SIEM Central Engine.

- Log Filtering: Process where all records are normalized into a common format.

- Alerts Policies-Correlation: Process where alerts are generated and linked to events, keeping in mind certain policies that will avoid the creation of false alarms that can lead to panic and waste of time.

- Real-time Analysis: Process where records are analysis as they’re coming.

- Monitoring: Final phase of the SIEM Central Engine. After all records are collected and processed, this phase allows the access to stored information, and also facilitates security policies development that will allow information extraction from events that are being process.

- Log Storage: Logs are stored in a DataBase. Depending on the quantity, they can be stored in various nodes forming a cluster.
SIEM & IoT Integration

As IoT devices are ruling our world, their utilization doesn’t represent an elevate cost. Because of this, their presence is incredibly high and security and communication standards are evolving to be implemented in either the physical devices or in applications that communicate with them.

In a typical IoT Solution Architecture, the IoT device will use a common IoT protocol, like MQTT or HTTP, to communicate with a gateway or an IoT Hub. If the IoT device communicates within a proprietary method with its gateway, then that particular aggregated device can be used to communicate back to the IoT Central Hub using common IoT messaging protocols [20].

At the same time, as soon as the device’s connectivity is formed, it is then engaged with the IoT Central Hub. Next, the IoT device can establish a bi-directional communication with the IoT Hub. This permits the device to perform as it should and to send requested information back to the Central Hub.

Since IoT devices can come in different shapes and sizes, some of them are sensors that are not able to hold complex code to initiate communications. In that case, IoT devices are authenticated to their field gateways, which periodically check into the devices to gather the last data and then forward it to the Central Hub [20].

The data is normalized in the Central Hub.

Because the data comes from thousands of devices, and needs to be analyzed and stored as its coming, to save IoT messages, Big Data storage is used. This allows for analytics to be added on the front-end of IoT applications.

Figure 3.4 represents an IoT Solution Architecture intentionally designed to integrate IoT devices, as its shown below (sensors, low power devices, IoT devices in general). This was made with the purpose of handling login attempts from unknown devices, rejected connections and even above/below frequency values coming from the devices.

The picture also illustrates the process described above.
Since IoT devices are everywhere, alarms and logs generated by them are being integrated into SIEM solutions.

To complete this union, the following should be done [20]:

- Use the existing Big Data storage from IoT solutions.
- Develop a feed into the SIEM environment after analytics are performed on the centralized data.

This will ensure important information is captured from the IoT messages.

Following this plan will avoid that duplicate logs of IoT messages are pushed into the SIEM. It will also prevent the analysis to be done on the IoT application side and on the SIEM side.
Figure 3.5: SIEM & IoT Integration Architecture

source: https://medium.com/dtembe/iot-and-siem-integration-pt-1-6645a012bdc

Figure 3.5 shows how IoT and SIEM solutions can be integrated. In this diagram, the IoT solution has bi-directional communications with the IoT Big Data storage. This permits the application of complex analytics on the captured data. The Big Data IoT storage normalizes the data coming from the IoT devices. Analytics are applied and security related messages are created. The messages can be pushed into the SIEM using standard probes. Most of the time, this is a unidirectional interaction with the SIEM [20].

The picture also shows that the SIEM solution preserves its original role and design; and that the IoT devices are integrated into the SIEM solution with the purpose of monitoring security events produced after the IoT data has been analyzed in the Big Data storage.

It is important to point out how the SIEM solution integration into the IoT world, provides a significant change on how security issues are manage. Because IoT devices are the principal vulnerability a system can have today, SIEM’s presence will help decrease the number of attacks they can cause. Being this a considerable contribution to the field.

SIEM Comparison

Choosing a SIEM solution is not an easy task. There are several points to consider and basically, the selection will depend on the use cases where it is going to be applied. Organizations used to acquire overpriced SIEMs, with a false belief that it will solve all its problems thanks to its uncountable features and functionalities. The truth is, that not all SIEM solutions are overpriced or over-advertised. Open source solutions that provide the same performance, are also an option when it comes to making a decision. Beside cost, a SIEM solution should cover all of the organization’s necessities. It
should adapt to its infrastructure and provide an agile and flexible search engine that include real-time visualizations, reports and alerts generation.

In today’s market, there are a significant amount of SIEM solutions. To have an idea of what organizations can find, the Gartner Magical Quadrant published on October 2018, will be used as a reference.

Figure 3.6 shows that SIEM solutions’ leaders are Splunk. IBM QRadar, LogRhythm, Dell Technologies RSA, Exabeam, McAfee and Securonix. Some of these solutions will be described and compared, in order to decide which one better satisfies this project’s necessities.

1. **Splunk**

   SIEM solution that combines network analysis with log management with an excellent analysis tool [8].
## 3.3. Analysis

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Powerful Data Analytics application</td>
<td>Some queries can run slow</td>
</tr>
<tr>
<td>Excellent and very informative dashboard</td>
<td>Difficult setup time if logs aren’t stored in common locations</td>
</tr>
<tr>
<td>Defining Field Extractor is very easy to use</td>
<td>Searches can be difficult to interpret</td>
</tr>
<tr>
<td>Can analyze large amount of data very effectively</td>
<td>High cost</td>
</tr>
<tr>
<td>OS: Windows and Linux</td>
<td>Designed for large companies</td>
</tr>
</tbody>
</table>

### 2. IBM QRadar

SIEM solution that offers a suite of log management, analytics, data collection, and intrusion detection [8].

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has more than 400 built-in log source types</td>
<td>Can’t be integrated with TSM</td>
</tr>
<tr>
<td>DSM Editor. Allow events to be parsed as the user wish</td>
<td>Some searches are not very intuitive</td>
</tr>
<tr>
<td>Integration with Vulnerability Manager and Risk Manager</td>
<td>It is not possible to export reports from the vulnerability manager add on</td>
</tr>
<tr>
<td>Built-in Rules, Offences and Reports</td>
<td>OS: Windows</td>
</tr>
</tbody>
</table>

### 3. LogRhythm

One of the most complete SIEM solutions, it offers behavioral analysis, log correlation and artificial intelligence [8].

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatible with a massive range of devices and log types</td>
<td>High cost</td>
</tr>
<tr>
<td>Provides a variety of deployment options</td>
<td>Design for large companies</td>
</tr>
<tr>
<td>Easy to scale</td>
<td>Hard to use</td>
</tr>
<tr>
<td>OS: Windows and Linux</td>
<td></td>
</tr>
</tbody>
</table>

### 4. Dell Technologies RSA

Evolved SIEM and threat detection and response solution that allows security teams to rapidly detect and respond to any threat, anywhere [17]. It focuses on real-time threat detection, forensic analysis and fast incident response [19].
Pros | Cons
---|---
Has the ability to investigate into network traffic | High cost
Has the ability to write rules and triggers for network communication | Design for large companies
Initial setup is time consuming | Hard to use
Good Technical Support | OS: Windows

5. **McAfee Enterprise Security Manager**

One of the best SIEM solutions in terms of analytics [19].

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capability to correlate different events from different platforms</td>
<td>Multiple bugs</td>
</tr>
<tr>
<td>It has an adaptive mode where it learns for itself</td>
<td>Consumes too much computer resources</td>
</tr>
<tr>
<td>Good Technical Support</td>
<td>Designed for mid-large companies</td>
</tr>
<tr>
<td>OS: Windows and Mac</td>
<td></td>
</tr>
</tbody>
</table>

6. **The Elastic Stack and Elastic SIEM**

Group of open source products designed to help users search, analyze and visualized real-time data coming from any source.

As mentioned in Chapter 2, The Elastic Stack is formed by a search engine, a log aggregator, a web-based user interface and a collection of log shippers. All this components are open source.

Although The Elastic Stack is not officially a SIEM solution, it acts like one when all its constituents work together.

Despite almost all of The Elastic Stack components are open source, the security and reporting features of X-Pack are not included in that list. Until now, their code in opened but a license is needed to use them.

In June 2019 The Elastic Stack added a new member to its family, Elastic SIEM. Its the official SIEM solution presented by Elastic, and it enables analysis of host-related and network-related security events as part of alert investigations or interactive threat hunting.

Because of its late appearance in today’s market, it is not included in the Gartner Quadrant site above.

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Components can be installed separately</td>
<td>Not all components are open source</td>
</tr>
<tr>
<td>Has a powerful search engine</td>
<td>Free version has no Support</td>
</tr>
<tr>
<td>It can be install in servers or desktops</td>
<td></td>
</tr>
<tr>
<td>OS: Windows, Linux and Mac</td>
<td></td>
</tr>
</tbody>
</table>

**SIEM Solution Election and Justification**

After evaluating and comparing multiple SIEM solutions, choosing only one is a tough call.
3.4 Design

As mentioned in Section 3.2 organizations are becoming unprotected due to technology’s evolution. This is a serious problem that has to be solved as soon as possible and in the best way. Meanwhile, companies turn into targets for security attacks, an unmeasurable amount of data is compromised and millions of dollars jeopardize. Quoting Former Cisco CEO John Chambers, “There are two types of companies: those that have been hacked, and those who don’t yet know they have been hacked” meaning that is almost impossible not to be attacked. For this reason, this project’s aim is to implement a Security Monitoring System that can contribute to the solution of the presented problem.

The SMS is going to be implemented using The Elastic Stack, and it will capable of monitoring and analyzing system logs, configuration files, security events and network packages. These actions will be possible by using Beats, specifically Filebeat for logs and files, Auditbeat for security events, and Packetbeat for network packages. The environment will be formed by various hosts, each one with a different Elastic Stack component installed and running. The hosts will be able to communicate between them, representing each one a stage of a Monitoring System. An IoT application with multiple sensors will send data to the environment to process. Five hosts will be defined in the development of this project:

1. Host 1: Has a MQTT broker and the previously mentioned beats installed. It is receiving sensor data from the IoT application and has a direct communication with Host 3. Here, MQTT logs, configuration and communication events will be send for monitoring and analysis. In addition, sensor logs and network events will be send too.
2. Host 2: Has an HTTP server and the previously defined beats installed. This host is also receiving data from the IoT application and has a direct communication with Host 3. In this environment, web server logs and file configuration will be send for analysis. Also, sensor logs, communication and network events will be shipped for monitoring.

3. Host 3: This host has The Elastic Stack Log Aggregator, Logstash, installed. It has direct communication with Hosts 1 and 2, from whom it is collecting the data from the different sources and will ship to Host 4, with whom also has a direct communication.

4. Host 4: This host is the system’s search engine, it has installed Elastic Stack component, Elasticsearch. It has direct communication with Host 3, from whom it is receiving the collected and parsed data that will be send to Host 5, with whom has also a direct communication.

5. Host 5: This host has The Elastic Stack user interface and the Elastic SIEM installed. It has a direct communication with Host 4, from whom it is taking the stored data and it will analyze, process, display and generate reports about. Also, it will allow the information visualization in a user friendly interface.

**Figure 3.7: Security Monitoring System’s Design Defined in this Project**

Figure 3.13 represents the project’s architecture. As its shown, every piece is connected to build a Security Monitoring System. For a better understanding the architecture will be itemize, illustrated in Figure 3.8.
Explaining in detail how the system is divided and the way the parts function and provide the expected result.

**Part 1**

Part 1 is composed by an IoT Application, a MQTT broker, a HTTP server and Beats.

The IoT Application has various sensors that are introducing data into the system. This data is a very valuable asset, it will be processed and analyzed to obtain meaningful information.

The MQTT broker is receiving data from the sensors and acknowledging the process.

Filebeat is a log shipper, its role is to generate the log files, tail them, and forward the data to Logstash. Also it will locate specific logs and files and apply basic processing to them.

Packetbeat is a network packet analyzer, it tracks the network traffic, decodes the protocols, and records data for each transaction. It will also send the collected data to Logstash.

Auditbeat is used for auditing user and process activity on a Linux server. It identifies breaches like file changes, configuration changes and malicious behavior. It also ships the data to Logstash.

The HTTP server is handling requests from the IoT application’s clients. At the same time, it is generating logs that will be process by Filebeat.

This Part of the system represents the first stage of a SMS, where the sources inject...
the data, it is collected and sent to the next stage where it will be processed.

**Part 2** is formed by Logstash. As defined before, Logstash is a log aggregator capable of pulling and receiving data from multiple sources, transforming it into a meaningful set of fields and eventually streaming the output to a defined storage destination. This represents the second stage of a SMS, where data is filtered, correlated and analyzed to finally be sent to a storage location. Thanks to Logstash’s filters that processes and transforms the received data, this task is accomplished.

**Part 3** is made up by Elasticsearch, Kibana and Elastic SIEM. Here the data is stored and it can be queried, filtered, aggregated and visualized. Elasticsearch is more than just a powerful search engine, it allows data to be searched and analyzed in every way, it rebalances and routes data automatically and it acts as a coordinator to delegate operations. Kibana is the visualization tool that allows data to be graphically seen. Using pie charts, maps or scatter plots, data can be represented for study and analysis. Elastic SIEM is a part of Kibana, it shows security events and alerts related to the processed data. This represents the final stage of a SMS, where the data is now transformed to information and displayed so administrators can know what is happening in their system.

### 3.5 Implementation

After analyzing how a Security Monitoring System is formed, how it works and how it can be implemented using The Elastic Stack, this section will show the detailed procedure to obtain the desired result.

First of all, it is important to mention that the architecture proposed in Section 3.4 was slightly modified due to increased resource consumption. Being implemented the architecture shown in Figure 3.9.
Figure 3.9 is just a compress version of the original design, operation and performance will be the same and results will be as expected.

For this development, three steps were defined:

1. Monitoring Environment Definition.
2. Technologies Installation and Configuration
3. Results

Monitoring Environment Definition

Monitoring an entire functioning system is the ideal scenario when it comes to implementing a SMS, but being this project a prototype and the first step of this implementation, not all aspects will be monitored in this opportunity.

Having decided the technologies that will be used, in this case the monitoring environment will be defined as it follows:

- Using Filebeat
  As is well known, Filebeat is used for collecting and shipping logs and files. To do this, Filebeat uses modules that define the type of log and/or file it is going to collect. These modules support common log formats and is the best way to get started with this kind of monitoring.

Filebeat has a variety of modules, these are listed below [11]:

1. Apache2 module
2. Auditd module
3. Icinga module
4. IIS module
5. Kafka module
6. Logstash module
7. MongoDB module
8. MySQL module
9. Nginx module
10. Osquery module
11. PostgreSQL module
12. Redis module
13. System module
14. Traefik module

For this project only the system module and the logstash module will be used.

- System module
  The System module collects and parses logs created by the system logging service of common Unix/Linux based distributions [14].
Chapter 3. Description

– **Logstash module**
  The Logstash module parse logstash regular logs and the slow log, it will support the plain text format and the JSON format (~log.format json) [12].

- **Using Auditbeat**
  As mentioned before, Auditbeat is used for auditing user and process activity on a Linux server.
  Like Filebeat, it also has modules to define the type of events it will audit.
  Auditbeat has three modules, these are listed below:
  - Auditd
  - File Integrity
  - System

  For this development, *File Integrity module* and *system module* will be used.

– **File Integrity module**
  The File Integrity module sends events when a file is changed (created, updated, or deleted) on disk. The events contain file metadata and hashes [10].

– **System module**
  The System module collects various security related information about a system. All datasets send both periodic state information (e.g. all currently running processes) and real-time changes (e.g. when a new process starts or stops)[15].

- **Using Packetbeat**
  As defined, Packetbeat is a real-time network packet analyzer. Unlike Filebeat and Auditbeat, it doesn’t has modules to define the type of data it will collect. Instead, it uses a similar concept called *flows*, that quoting the official documentation page its defined as follows:

  "A flow is a group of packets sent over the same time period that share common properties, such as the same source and destination address and protocol. You can use this feature to analyze network traffic over specific protocols on your network. For each flow, Packetbeat reports the number of packets and the total number of bytes sent from the source to the destination." [13]

**Technologies Installation and Configuration**

For this project development several requirements were mandatory, including the installation and configuration of multiple technologies.
Although it seems very straightforward, configuration changes depending on the hardware were is going to be installed, and various modifications (not mandatory all the time) had to be made.
Below are the steps to install, configure and run this project’s architecture.

**3.5.1 Hardware**

This project was built on a MacBook Air with a 1.6 GHz Intel Core i5 Processor and 8GB Memory; that served as a host for the virtual environment that was created to develop the system.
3.5.2 Software

As mentioned above, this project is a virtual environment. It is composed of three (3) Virtual Machines that communicate between each other and simulate a real world production environment.

Oracle VM VirtualBox Manager 6.0.8 was the hypervisor used to virtualize the three hosts. The hosts named elk, mosquitto and apache, are running a 64-bit Ubuntu Server 16.04.06 LTS and are configured as follow:

- **elk**
  - Configuration
    1. 4GB Memory
    2. 2 processors
    3. 20GB disk
    4. 2 network adapters
    5. IP address 192.168.59.105
  - Installed Software
    1. Elasticsearch 7.2
    2. Kibana 7.2
    3. MySQL Database

- **mosquitto**
  - Configuration
    1. 1GB Memory
    2. 1 processor
    3. 20GB disk
    4. 2 network adapters
    5. IP address 192.168.59.110
  - Installed Software
    1. Mosquitto 1.6.3
    2. Logstash 7.2
    3. Filebeat 7.2
    4. Auditbeat 7.2
    5. Packetbeat 7.2

- **apache**
  - Configuration
    1. 1GB Memory
    2. 1 processor
    3. 20GB disk
    4. 2 network adapters
  - Installed Software
    1. Apache HTTP Server 2.4.39
    2. Logstash 7.2
    3. Filebeat 7.2
    4. Auditbeat 7.2
    5. Packetbeat 7.2
Chapter 3. Description

Installing and Configuring Technologies

1. Elasticsearch 7.2

- Install Elasticsearch
  (a) Install Java. The Elastic Stack requires Java 8 or Java 11.
  
  ```
  sudo apt-get install default-jre
  ```
  
  (b) Import the Elasticsearch PGP Key
  
  ```
  wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch
  | sudo apt-key add -
  ```
  
  (c) Add Elasticsearch 7.x APT repository
  
  ```
  echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main"
  | sudo tee /etc/apt/sources.list.d/elastic-7.x.list
  ```
  
  (d) Install the apt-transport-https package
  
  ```
  apt-get install apt-transport-https
  ```
  
  (e) Update repositories and Install Elasticsearch
  
  ```
  sudo apt-get update
  sudo apt-get install elasticsearch
  ```

- Configure Elasticsearch
  - Open Elasticsearch configuration file
  
  ```
  vim /etc/elasticsearch/elasticsearch.yml
  ```
  and edit the following:
  
  (a) `node.name: "node-1"`
  The node name, its the way to identify the node, it can be any name.
  
  (b) `network.host: 0.0.0.0`
  To make sure the server listens to all the IPv4 addresses on the local machine.
  
  (c) `http.port: 9200`
  Elasticsearch connection port.
  
  (d) `discovery.seed_hosts: ["127.0.0.1"]`
  IP address to discover the node when started. In this case its localhost because Elasticsearch is running on the same machine.
  
  (e) `cluster.initial_master_nodes: ["node-1"]`
  Because there is only one node, it is configured as a master.

- Running Elasticsearch
  
  ```
  sudo systemctl daemon-reload
  sudo systemctl enable elasticsearch
  sudo systemctl start elasticsearch
  ```

2. Logstash 7.2

   As mentioned before, Logstash requieres Java 8 or Java 11.
   Before installing it, it should be verified that the correct Java version is running.

- Install Logstash
  
  (a) ```
  sudo apt-get install logstash
  ```
3.5. Implementation

- Configure Logstash

Logstash data processing pipeline has three sections:

(a) INPUT: input section is used to ingest data from different endpoints into Logstash.
(b) FILTERS: which processes and transform the data received.
(c) OUTPUT: which stashes processed data into a specified destination, which can be Elasticsearch.

(a) Define and configure Logstash’s input.
Logstash can read sources of events from many plugins. In the configuration file, it must be specified from which plugin the events are going to come from.
In this case, Logstash is configured to read events from the Elastic Beat framework.
Create an input configuration file:

```
sudo vim /etc/logstash/conf.d/beats-input.conf
input {
  beats {
    port => 5044
  }
}
```

(b) Configure Logstash Filters
Create a filters configuration file:

```
sudo vim /etc/logstash/conf.d/ssh-auth-filter.conf
filter {
  grok {
    match => { "message" => "\{SYSLOGTIMESTAMP:timestamp\}\s+
                \{IPORHOST:dst_host\}\s+
                \{WORD:syslog_program\}\[\d+:\]\s+(?<status>\w+\s+password)\s+for\s+
                \{USER:auth_user\}\s+from\s+\{SYSLOGHOST:src_host\}\.*" }
    add_field => { "activity" => "SSH Logins" }
    add_tag => "linux_auth"
  }
}
```

Grok works by combining text patterns into something that matches logs. The syntax for a grok pattern is

```
\%{SYNTAX:SEMANTIC}
```

The SYNTAX is the name of the pattern that will match the text.
In this case, the configuration file was made to match ssh failed login attempts.

- Configure Logstash Output

There are different output plugins that enables Logstash to send event data to particular destinations.
In this case elasticsearch is used, enabling Logstash to send data to Elasticsearch.
Create Logstash output configuration file.

```
sudo vim /etc/logstash/conf.d/elasticsearch-output.conf
```

This configuration sends data to Elasticsearch running on a different Virtual Machine, therefore the IP address on the "hosts" field, is the IP address of the Virtual Machine running Elasticsearch.
output {{
  elasticsearch {
    hosts => ["192.168.59.105:9200"]
    manage_template => false
    index => "ssh_auth-%{+YYYY.MM}"} 
}
}

During this project development, three configuration files were created; one for every Logstash section. The reason of this practice was a better visualization and management. All the sections can be together in one configuration file.

- Running Logstash
  
  ```
sudo systemctl start logstash
sudo systemctl enable logstash
  ```

3. Kibana 7.2

- Install Kibana
  
  ```
sudo apt-get install kibana
  ```

- Configure Kibana
  
  (a) Open Kibana configuration file
  
  ```
sudo vim /etc/kibana/kibana.yml
  ```

  (b) Define Kibana’s connection port and Elasticsearch URL.
  
  ```
server.port: 5601
elasticsearch.url: "http://localhost:9200"
  ```

  Because Elasticsearch and Kibana are running in the same Virtual Machine, the URL is **localhost**.

- Running Kibana
  
  (a) `sudo systemctl start kibana`

  (b) Open a web browser to visualize Kibana

  ```
192.168.59.105:5601
  ```

  In this case, the IP address corresponds to the Virtual Machine where Kibana is running.

4. Beats

Installing Beats is very straightforward, they’re install the exact same way as the other components.

- Configure Beats
  
  Depending of the type of data that is going to be monitored, a different Beat should be used.

  In this case, **Filebeat, Auditbeat and Packectbeat** were installed and configured.

  All Beats are configured the same way.

  (a) Open Filebeat configuration file
3.5. Implementation

(b) Define the output

```
sudo vim /etc/filebeat/filebeat.yml
```

```
#---------------- Logstash output ---------------
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.59.110:5044"]
...
```

In this project, Filebeat is sending Logstash the data, and Logstash is sending it to Elasticsearch. Therefore, Logstash is the defined output and the IP address corresponds to the Virtual Machine running Logstash.

Elasticsearch can also be a configured output.

(c) Enable Filebeat System Module

All beats have multiple modules to collect and parse different logs. Modules are disabled by default and they have to be enabled manually.

In this project, Filebeat system’s module was enabled to parse logs created by the system service.

```
filebeat modules enable system
```

(d) Load the index template in Elasticsearch

```
filebeat setup --index-management -E
  output.logstash.enabled=false -E
  output.elasticsearch.hosts=["192.168.59.105:9200"]
```

(e) Generate the template

```
filebeat export template > filebeat.template.json
```

(f) To install the template on Elastic Stack server, copy and run locally on Elastic Stack server.

```
curl -XPUT -H 'Content-Type: application/json'
  http://localhost:9200/_template/filebeat-7.0.1
  -d@filebeat.template.json
```

- Start Filebeat

```
sudo systemctl start filebeat
```

```
sudo systemctl enable filebeat
```

To install Logstash module and Auditbeat’s modules, the same procedure was made.

5. IoT Sensors Simulator

To simulate the IoT Application defined in the architecture, a sensor simulator made by MSc. Ramón López was used. The source code and all the configuration files can be found in [https://github.com/rlopezv/iot-simulator](https://github.com/rlopezv/iot-simulator).

This program was used to simulate three systems. A heating system, an irrigation system and a lightning system.

To configure them, the steps provided in the README file located in the GitHub repository, were followed.

As defined by the author, the program “allows to simulate systems containing...”
sensors and actuators that will use MQTT for communication.”

The project is formed by:

- “Clients that will process the messages.”

And two controllers:

- “One for MQTT”
- “and another to act as a gateway for messages obtained from a MQTT broker.”

After configuration, the project can be executed using the command:

```
java -jar [path_to_jar] [path_to_config_file]
```

This project has been enriched with the incorporation of a MySQL DataBase where the system’s geolocation is stored.

The steps to configure and run this project are explained with details in the mentioned file. They are not included in this document because of space limitations, but they can be accessed through the repository link [github link].

The following is an example of the .json file where the Gateway is configured to obtain MQTT messages.

```
{
  "implClassName": "com.upm.miot.rlopezv.iotsimulator.control.GatewaySystemController",
  "mqttConfig": {
    "brokerUrl": "tcp://192.168.59.110:1883"
  },
  "additionalProperties": {
    "httpURL": "http://localhost:1880/data"
  },
  "sensors": []
}
```

The IP address corresponds to the IP address of the Virtual Machine where the MQTT Broker is running.

### 3.5.3 Results

After following all the steps described above, the Security Monitoring System developed is capable of monitoring a system and its network, providing several dashboards and graphs that allow administrators to detect anomalies and prevent attacks.

This section will show some of the SMS’s features developed during this project.

**Monitoring IoT Sensors**

This example shows how the SMS monitors IoT sensors. As shown in Figure 3.10 messages are being sent and received successfully.
3.5. Implementation

Figure 3.10: Messages being sent and received

This data is sent to Elasticsearch and displayed so it can be visualized by the user, as shown in Figure 3.11.

Figure 3.11: Sensors logs shown in Elasticsearch

Also, the received data is shown in a line graph, so anomalies are more visible and suspicious events can be detected easier. Figure 3.12 illustrates the action.
Monitoring System Events

1. SSH Logins Attempts

This example shows how the SMS can detect SSH login attempts. While installing and configuring Logstash and Beats, Filebeat was configured to collect SSH authentication events from the Ubuntu Operating System. To do this, a Logstash filter able to process this kind of events and matches the SSH authentication log lines was created (Indicated above in the Configure Logstash Section).

```
Jul 10 18:17:09 mosquitto sshd[7054]: Accepted password for user from 192.168.59.3 port 49872 ssh2

Jul 10 18:22:26 mosquitto sshd[7588]: Failed password for user from 192.168.59.3 port 50092 ssh2
```

Once the system is up and running, an index with the name ssh_auth-2017.04 is created in Elasticsearch.

When a user tries to establish a SSH connection with a host, an event is generated and stored in Elasticsearch’s index ssh_auth-2017.04. As shown in Figure 3.18.
## 3.5. Implementation

![Figure 3.14: New SSH session](image1)

From now on, everything that happens on that host is registered in a log file. These log files can be seen and analyzed in Kibana.

### Figure 3.15: Kibana Logs

Because the `grok` filter was created to match the SSH logs, records indicating a successful or denied authentication are shown in Kibana.

![Figure 3.16: SSH Accepted Password Log](image2)

At the same time, an event indicating the suspicious incident is created in Elastic SIEM and its shown bellow.

![Figure 3.17: SSH Failed Password Log](image3)
2. Auditbeat, Packetbeat and Filebeat Events

This example shows how the SMS can process and analyze the information received from the different beats (explained in Section 3.4).

Figure 3.19 the SIEM overview interface, where Host and Network Events are clearly visible. The Host Events are produced by Filebeat and Auditbeat’s actions, bringing to Kibana all logins, users actions and systems events. The Network Events are produce by Packetbeat, that is bringing its flow to Kibana.

Figure 3.18: Elastic SIEM Events

Figure 3.19: SIEM Overview
3.5. Implementation

Figure 3.20 shows Hosts Events. Indicating how many hosts are active, how many authentications have been made and the sources from where they came from.

![Hosts Events Diagram](image1)

**Figure 3.20: SIEM Host Events**

Figure 3.21 shows Network Events. Indicating how many events had occur, the flow from where they come, IP sources and DNS queries.

![Network Events Diagram](image2)

**Figure 3.21: SIEM Network Events**

Figures 3.22 and 3.23 show dashboards with Auditbeat and Filebeat events.
Figure 3.22: Auditbeat Events

Figure 3.23: Filebeat Events

Figure 3.24 shows a Timeline made in SIEM, where all the events are clearly displayed and ready to be analyzed by the user.
Figure 3.24: Elastic SIEM Timeline
Chapter 4

Conclusions

4.1 Conclusions

Security is organizations’ biggest concern, with technology’s evolution and the Internet’s vast scope, data is not completely safe and has become a target for cyberattacks. This issue can caused irreparable damage and multimillionaire losses, and has to be solved as soon as possible.

This problem can be addressed in various ways, and this project proposed the use of a Security Monitoring System (SMS) to solve it.

SMS is not a new concept, it has been around for many years and it has evolved in a significant way. Since harm can be caused by the different vulnerabilities organizations can have, or by the variety of attacks they can be a victim of, SMS presents two approaches intended to reduce damage and prevent distressful situations.

The first approach, focused on reducing vulnerabilities to prevent attacks has been used for a long time, but has become outdated when it comes to IoT applications. In the IoT era, system’s inputs are constantly changing, making it impossible to center on vulnerabilities, therefore, the second approach focused on threat reduction, is the one applied in this project, that handles an IoT application.

The solution presented was originated in a Big Data Course Laboratory Assignment. The exercise introduced an architecture that used Elasticsearch to monitor IoT sensors and visualized them in real-time in a Kibana dashboard. The Laboratory activity provided a foundation for this project, and was key for decision making.

On the other hand, an analysis and comparison of the most used SIEM solutions was made; in order to decide what technology to use. After studying several platforms, Elasticsearch and Elastic SIEM presented the most positives features that suited this project’s necessities, becoming the chosen solution to be applied during this development.

Although this project was born in a classroom, it gave various contributions to the course, that may be applied in the future.

First, a new architecture (based on the existent) was defined and implemented. All the technology installation was migrated from Elasticsearch 6 to Elasticsearch 7.2, generating configuration issues and hurdles that were handled and overcame. Second, in this project an IoT sensor simulator developed by MSc. Ramón López was used, instead of the one provided by the professors. Finally, a SIEM approach focused on detecting threats and monitoring the system was applied, bringing security into the package.
This last task represented the biggest challenge of them all, since the Elastic SIEM is a beta version launched on June 2019, the lack of documentation and sources was the main defiance situation.
In addition, the limited compute resources available for this development, was another significant hurdle present in this solution.

Despite all the difficult situations confronted, all obstacles were defeated with a successful result. Demonstrating that cyberthreats and attacks are real and must be defied under any circumstance.
Finally, this project showed the importance of data protection in any field, warning companies of risks and attacks that most of the time they’re not even aware of.
Bibliography


