



POLITECNICO DI TORINO

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING (DAUIN)

Master Degree in Computer Engineering

Master Degree Thesis

Threat Hunting driven by Cyber Threat Intelligence

Author: Alessandro BOLLA, Federico TALENTINO

Advisor: Paolo Ernesto PRINETTO

Co-Advisors: Nicolás MAUNERO, Fabio DE ROSA

April, 2022

Abstract

The ever-increasing number of cyber-attacks and cyber-criminal activities necessitates the implementation of countermeasures to protect both public and private businesses. In Italy, the government has decided to take steps to ensure a high level of security for networks, information systems and services especially for the public administrations. The *Perimetro di Sicurezza Nazionale Cibernetica (PSNC)* was established and public and private stakeholders that provide crucial services for the maintenance of civil, social or economic activities that are essential to the state's interests have been included in it. Given the clear demand for models, methodologies and tools to carry out operational tasks efficiently, this thesis focuses on Cyber Threat Intelligence and Cyber Threat Hunting technologies.

The goal of this thesis work is to propose a method for making intelligence data actionable leveraging analysis tools, Machine Learning and SIEMs. Everything was done in an open source style, with strong emphasis on endpoints. The work analyses and signals anomalous operations with the goal to improve the state of the art of open source applications.

The first part focuses mostly on cyber threat intelligence data and progresses through studies of CTI ontologies, taxonomies, and languages before concluding with the choice to focus efforts on the well-known MITRE ATT&CK framework. ATT&CK contains a huge amount of valuable knowledge, however it remains at a high level (focusing on TTPs, meaning Tactics, Techniques and Procedures), whereas it would be beneficial to make it more actionable in a practical manner. This section of the project focuses on detecting a subset of MITRE ATT&CK techniques using the open source Osquery tool as a probe for collecting logs. To capture logs of cyber attacks Atomic Red Team was used: developed by Red Canary, it is a framework for testing ATT&CK techniques by executing atomic attacks against a target system.

The second part of the thesis is focused on Cyber Threat Hunting, Machine Learning, log collection and delivery. Among many popular open source tools, the ELK stack (ElasticSearch, Logstash and Kibana) technology was selected, exploited by OpenSearch, which is an open source tool released by Amazon. The ELK infrastructure is built upon Docker containers, which ensures performance and a plug&play configuration. A new algorithm is proposed based on BERT, a state of the art Machine Learning algorithm initially developed by Google. The algorithm performs Cyber Intrusion Detection using Natural Language Processing to learn a baseline of normal logs and eventually recognize anomalous ones. Detected logs are collected by Filebeat, an agent of the ELK stack, and sent to OpenSearch.

The user endpoint is where the two halves meet together: logs are gathered by the Osquery probes, analysed and sent to OpenSearch. This procedure has two outcomes: testing the effectiveness of queries in recognizing MITRE ATT&CK techniques and verifying the ability of the new algorithm to identify malicious logs. This can be considered as a baseline implementation and future work must be done in order to improve the solution and strengthen the connection of the components.

This thesis is a joint effort of Federico Talentino and Alessandro Bolla where, considering the two major parts of the work identified above, Federico focused on the first half of it while Alessandro on the second one.

Contents

List of Tables	8
List of Figures	9
1 Introduction	11
2 Background - Cyber Threat Intelligence	17
2.1 Cyber Threat Intelligence	17
2.1.1 Definitions	18
2.1.2 Brief history and diffusion	19
2.1.3 CTI types	19
2.1.4 Typical CTI management: the SOC	21
2.1.5 CTI ontologies	22
2.1.6 CTI languages: the case of STIX	28
2.2 TTPs and MITRE ATT&CK	31
2.2.1 Tactics, Techniques and Procedures	31
2.2.2 Adversarial Tactics, Techniques & Common Knowledge: ATT&CK	33
2.3 Osquery	38
2.3.1 Tables schema and organization	38
2.3.2 Osqueryi and Osqueryd modes	39
2.3.3 Osquery configuration file and flags	39
2.3.4 Osquery results	41
2.3.5 Importance in the domain of this work	41
2.4 Atomic Red Team	41
2.4.1 Library organization	41
2.4.2 Atomic test details	42
2.4.3 Invoke-Atomic module	43
2.4.4 Reasons for the choice	44
2.5 Red Canary - 2021 Threat Detection Report	44
2.5.1 Report organization	45
2.5.2 Reasons for the choice	47
3 Background - Threat Hunting	49
3.1 Threat Hunting Overview	49
3.1.1 Introduction and definitions	49

3.1.2	APT and Most Popular Threats	51
3.2	The Unified Kill Chain	54
3.3	SOC, SOAR and SIEM tools	56
3.3.1	Most relevant proprietary tools	58
3.3.2	Most relevant open source tools	60
3.3.3	EDR and SIEM - Consideration	62
3.4	EDR and SIEM Challenges: Analysts Burnout and Predictive Analysis	63
3.4.1	Log flood and Analyst burnout	63
3.4.2	Predictive Analysis	65
3.5	Let the Hunt continue	66
3.5.1	Common Vulnerabilities and Exposures (CVE)	67
3.5.2	National Vulnerability Database (NVD)	68
3.5.3	The problem of unknown	69
3.5.4	Call to weapons	69
4	Perimetro di Sicurezza Nazionale Cibernetica	71
4.1	Introduction: the <i>PSNC</i>	71
4.2	<i>PSNC</i> organization	72
4.3	Legislation	72
4.3.1	DPCM 1	72
4.3.2	DPR	73
4.3.3	DPCM 2	73
4.3.4	DPCM 3	73
4.3.5	DPCM 4	74
4.4	Implementation	74
4.5	Cyber Threat Intelligence and Threat Hunting in the <i>PSNC</i>	74
5	Contribution - Log Collection	77
5.1	Initial attempts	77
5.1.1	Existing repository update: <i>osquery-attck</i>	77
5.1.2	Mapping APT groups to Osquery	79
5.1.3	Combining MITRE CAR with Osquery	80
5.1.4	Mapping MITRE ATT&CK techniques for Linux systems	80
5.2	Query writing	82
5.2.1	Mapping <i>2021 - Threat Detection Report</i> top 10 techniques to Osquery	82
5.2.2	Sources and methodology used to write the queries	85
5.2.3	Configuration files overview	87
5.2.4	Problems faced and drawbacks	91
5.3	Query testing	91
5.3.1	Test environment	91
5.3.2	Clean log collection	92
5.3.3	Dirty log collection with Atomic Red Team	93

6	Contribution - Threat Hunting	95
6.1	The ELK Stack	95
6.1.1	Elasticsearch - OpenSearch	95
6.1.2	Logstash	97
6.1.3	Kibana – OpenSearch Dashboard	98
6.1.4	Why OpenSearch	99
6.2	Filebeat	100
6.3	Docker	102
6.3.1	Docker architecture	103
6.4	Virtual Box	106
6.5	Proposed Architecture	107
6.6	Machine Learning overview	110
6.6.1	Key concepts	112
6.6.2	Introduction to the BERT model	115
6.6.3	Masking Language Model and Next Word Prediction	118
6.7	The Proposed Algorithm	119
6.7.1	Log parsing	121
6.7.2	Tokenization	122
6.7.3	Training	123
6.7.4	Baseline scenario	126
6.7.5	Malicious Log Detection	127
7	Experimental Results	129
7.1	Log Collection results	129
7.1.1	Introduction	129
7.1.2	Clean Log Collection results	129
7.1.3	Dirty Log Collection results	130
7.1.4	Log Collections comparison	131
7.2	Threat Hunting results	132
7.2.1	Parsing result example	133
7.2.2	Tokenization result example	134
7.2.3	Training results	134
7.2.4	Baseline results example	136
7.2.5	Analysis results example	137
7.2.6	OpenSearch Dashboard visualization	141
8	Conclusions and Future Work	143
8.1	Cyber Threat Intelligence conclusions	143
8.1.1	Future Work	144
8.2	Threat Hunting conclusions	145
8.2.1	Future Work	146
A	The OpenSearch project	147
B	Open source initiative and Elastic license issues	151
B.1	The open source initiative	151
B.2	Elastic background and new license	153

List of Tables

7.1 Log Collection Experimental Results.	133
--------------------------------------------------	-----

List of Figures

2.1	Threat management plans and organization [46].	22
2.2	The Detection Maturity Level Model - DML [45].	25
2.3	The Cyber Threat Intelligence Model - CTI [45].	26
2.4	Proposed Ontology by Y. Merah and T.Kenaza [52].	28
2.5	ATT&CK enterprise matrix sliced	34
2.6	ATT&CK: relationships between main components [77].	35
2.7	Sub-techniques list for technique T1059 - Command and Scripting Interpreter	36
2.8	Summarizing panel for technique T1059 - Command and Scripting Interpreter	37
2.9	Osquery kernel_info table	38
2.10	Example of osqueryi usage with Windows Powershell.	39
2.11	Osquery random configuration example.	40
2.12	T1037.001 - Test #1 - Markdown file	42
2.13	T1037.001 - Test #1 - YAML file	43
2.14	Some commands executed on test 1 of sub-technique T1053.005.	44
3.1	Main APT attack steps. [32]	52
3.2	Ransomware lifecycle illustrated by CyberArk, world leader company in Privileges Access Management. [13]	54
3.3	The Unified Kill Chain 18 tactics. [69]	55
3.4	Magic Quadrant for endpoint protection platform, published by IT Gartner in the 2021 report.[29]	59
3.5	Result of the study, showing if EDR detected the threats and if the attacks succeeded.[41]	60
3.6	Organizations experiencing a security breach in 2020, from PaloAlto networks. [100]	64
3.7	Business impact of a security breach for an organization. [100]	65
3.8	Personal impact of a security breach for security analyst. [100]	65
3.9	CVE Record Lifecycle, from discovery and reporting to publish. [19]	67
3.10	CVSS assigned to CVE-2019-18323.	68
3.11	Windows of Exposure.	70
5.1	Example file: <code>windows_scheduled_tasks.conf</code> of <i>osquery-attck</i> [54].	78
5.2	ATT&CK detection information for T1105 - Ingress Tool Transfer	85
6.1	<i>DB-Engines, DBMS popularity ranking accessed in March 2022.</i>	96
6.2	<i>DB-Engines, Search Engine popularity ranking accessed in March 2022.</i>	97
6.3	Logstash configuration file empty template.	98
6.4	OpenSearch sample dashboard provided during the installation as example.	99

6.5	OpenDistro for Elasticsearch, a project to add advanced capabilities to the old open source Elastic stack. [86]	100
6.6	Filebeat working flow.	102
6.7	Docker logo.	103
6.8	Docker architecture example.	104
6.9	Docker Union File System example.	105
6.10	Oracle Virtual Box logo.	106
6.11	Table showing all different Virtual Box network type and which type of communication each type allows to achieve.	106
6.12	Overview of the final infrastructure, highlighting the logical connection between OpenSearch stack, Virtual Box and the guest machine.	107
6.13	Schema of overall system architecture.	109
6.14	Schema of the host machine, running Windows 10.	110
6.15	Schema of the virtual machine, running Ubuntu.	111
6.16	Tokenization example applying a Sub-word approach. [24]	117
6.17	Example of masking and prediction results, with input the previous considered example sentence.	119
6.18	BERT algorithm pipeline, from log collection to output results.	120
6.19	WordLevel model for Tokenization, with important parameters.	122
6.20	Trainer object listing its parameters and the function called to launch the training process.	125
7.1	Results of the best model, coming from the 10th epoch, obtained with training process.	136
7.2	Results of the baseline scenario obtained using the best pre-trained model obtained in the training step.	137
7.3	Template of a JSON object used to store a detected anomaly in an output file.	138
7.4	Example of tokenized query from pack 7, with many unknown tokens which do not allow an effective prediction.	140
7.5	Analysis of 155 dirty logs coming from different packs.	141
7.6	Example of log visualization in OpenSearch Dashboard web application.	141
7.7	Example of log visualization in OpenSearch Dashboard web application.	142
A.1	OpenSearch logo.	147
B.1	Open source initiative logo.	151

Chapter 1

Introduction

The number of cybersecurity attacks nowadays is continuously increasing at a very high rate. The problem is real, and it is possible to notice it without being a cyber-insider. No matter where, if written in the newspaper or broadcasted in TV news, in any case a considerable number of cyber-related novelties are presented to the people every single day. Cyber-attacks, and in general cyber-criminal activities, are performed against victims that ranges from the single person's device to the biggest enterprise or government network. Phenomenon like phishing, or more in general any kind of social engineering scam, happens on a daily basis. A lot of unaware company employees, named *Unintentional Insiders*, may become the cause of big problems for businesses leading to loss of money, disclosure of sensible information or violation of the intellectual property. These episodes are wide spreading quickly, and cyber-criminals are proliferating, improving their skills and becoming a serious threat in today's IT landscape. While bigger companies are usually investing a great amount of money for the protection of their assets (both in terms of technology and in terms of employees security training), small and medium enterprises are often in a difficult situation for a number of reasons. First of all, being their profit lower with respect to large enterprises, company managers are convinced that attackers are not interested in their money, letting their guard down and leaving space for intrusions. Secondly, and this obviously depends on how small the company under question is, being the budget typically limited, many decision-makers are not willing to spend enough money for security. Last point, often companies lack both staff and capabilities to properly train employees, since an advanced knowledge is required. The problem arises when attackers decide to target also little realities, and this is happening today. This issue is often applied also to public administrations and in general government controlled entities. Those may be obsolete in their IT organization and are consequently vulnerable from the security point of view. Nowadays, big nation states are facing themselves in a cyber-war for economic, military or social reasons, just to mention a few.

With this situation, it becomes fundamental to be well protected. This is important for private entities as well as for the public sector. To make an example, UniCredit¹, one of the biggest banks in Italy, has been attacked multiple times between 2017 and 2019

¹<https://www.unicredit.it/it/privati.html>

with millions of user accounts stolen [27]. Public sector experiences the same troubles: following a report created by Sham² organization together with University of Turin [74], during Covid-19 pandemic, 24% of Italian healthcare structures have been attacked. This because the health sector is progressively going under a digital transformation process that has highlighted all the security problems [80] that it may arise.

In Italy, with the decree-law 133 of year 2019, the *Perimetro di Sicurezza Nazionale Cibernetica (PSNC)* [44] has been established and since then has evolved its specification during years. The *PSNC* has been instituted in order to ensure a high level of security for networks, information systems and information services of public administrations, national entities and operators, both public and private, through the institution of a cyber perimeter and, the adoption of measures aimed at guaranteeing the necessary security standards and minimize the risks [38]. To summarize, there is a list of public and private entities (included in the *PSNC*), that are asked to accomplish some cybersecurity requirements. Included entities, are considered fundamental for the maintenance of civil, economic or social functions that are crucial for the State. *PSNC* is still evolving and not completely adopted; this leads to the necessity, in following years, of technical suggestions, guidelines and projects that can strengthen its application.

The purpose of this thesis is to investigate, and make proposal, in some of the aspects covered by the *PSNC*. The work developed focused on *Cyber Threat Intelligence (CTI)* and *Cyber Threat Hunting (CTH)*. Respectively, CTI regards knowledge, skills and information about the cyber threat landscape, while CTH is the action of monitoring IT system for the detection of cyber attacks. Particular attention has been paid to open source solutions because, this can bring benefits in several aspects:

- Open Source solutions allows a more fine grained control, and customization, of what a CTH solution can monitor;
- Open Source solutions allows, to a certain extent, to save money by providing tailored solutions based on needs and constraints.

Moreover, this work aims at improving the state of the art, by discussing following topics:

- Find a way to make threat intelligence high-level information actionable in a practical way;
- Explore the possibilities given by some open source software for threat intelligence and threat hunting;
- Propose a Machine Learning algorithm that may be used as starting point to perform initial analysis on collected logs, in order to reduce the noise around false positive alerts.

Normally CTH solutions performs analysis on both user device and centralized system, but for the scope of this thesis work, the focus has been put on the user endpoint.

²<https://www.sham.com/it/>

The first half of the thesis is mostly focused on *Cyber Threat Intelligence*. This contribution started with the exploration of the CTI world, focusing in particular on semantic models. First research efforts have been directed to the study of CTI and more in general cybersecurity-related ontologies. After some reviews, the attention shifted then from ontologies to the topic of cyber threat information sharing. CTI sharing is a key element for the success of intelligence today, both in public and private sectors. Boosting the information sharing between companies will improve the general posture of each one. Among the vastness of different languages and standards, *Structured Threat Information Expressions* also known as *STIX*³, seems to be the most well-known and widespread. The study of *STIX* language peculiarities, like its organization in objects and relationships, puts *STIX* himself in the world of ontologies. When studying ontologies and languages anyway, the conviction that the work should have been focused on a specific and widespread framework became stronger. Considering that ontologies presented in scientific papers still consist in a sort of niche of CTI, and *STIX* is largely appreciated but still not adopted enough by the community, something more widespread was needed. After some investigations of the state of the art, the choice fell on the famous *MITRE ATT&CK*⁴ framework. *ATT&CK*, created some years ago by MITRE corporation, is a big database of known Tactics, Techniques and Procedures (TTPs). TTP is an essential concept for tactical CTI but also for the other aspects of the subject. The study of TTPs lead experts to the possibility of understanding the enemies' intentions and ways of attack; this is undoubtedly of paramount importance for defense purposes. *MITRE ATT&CK* can be used in a lot of different ways: threat report analysis, vulnerability assessment, threat intelligence activities and so on. There is, however, one issue that has been noted and has lately revealed to be a quite prevalent idea: the problem is that *ATT&CK* stands at a very theoretical high-level of abstraction. The basic block of the framework is the technique and, each of these contains much information in text format. The idea that came up consisted in trying to implement a solution that, leveraging the technique description inside the framework, could be useful for attack/intrusion detection. *Osquery*⁵ was chosen to put in practice the idea. *Osquery* is an open source tool that consists of an SQL-programmable probe that collects customizable logs on the endpoint. Practically, user can write SQL queries that are addressed to the OS: each query produces logs that are collected for analysis. *Osquery* permits to gather only targeted data, limiting invasiveness of the solution. The target was to make detection, taking advantage of rich information offered by *ATT&CK*. After some failed attempts, the decision taken was to create a sample of queries able to detect a sample of techniques contained in the *2021 Threat Detection Report* by Red Canary [4]. *Atomic Red Team*⁶, developed by the aforementioned Red Canary⁷ company, is a library of atomic tests which has a strong bond with *MITRE ATT&CK*. Inside *Atomic Red Team*, tests are divided for *ATT&CK* techniques, and there are tests for a consistent number of techniques. The idea

³<https://oasis-open.github.io/cti-documentation/stix/intro>

⁴<https://attack.mitre.org>

⁵<https://osquery.io>

⁶<https://github.com/redcanaryco/atomic-red-team>

⁷<https://redcanary.com>

in the end, has been to activate the *Osquery* probe and perform atomic tests against the endpoint, with the aim of collecting custom logs and evaluate if the queries were effective in the reconnaissance of the attacks and consequently of the techniques adopted.

At this point the focus shifts to the second contribution of this thesis, which emphasizes the aspect of *Cyber Threat Hunting*. The process here started with a survey on the most popular open source software in the category of Security Information Event Management tools (SIEMs). The *Elastic Stack*⁸ was selected, which framework has grown in popularity over past few years. The tool, also known as ELK Stack, is composed by three main components:

- **ElasticSearch**⁹: the search engine, the real core of the stack; data are stored in a non-relational database and can be searched and queried in a very efficient way;
- **Logstash**¹⁰: the tool that collects data coming from agents installed on the user endpoint, analyzing and delivering them to ElasticSearch;
- **Kibana**¹¹: the visualization engine; it is a web application that allows to create dashboards, graphs and generate reports that a user can leverage to perform analysis. Kibana also allows to set triggers and alert in case given constraints are violated.

Data are collected through agents installed directly on the user machine, named *Beats*. Those agents will send data to the Stack which will display them to the user, with Kibana, as results. The selected software is *Opensearch*¹², a new and fully open source tool, released on October 2021 by Amazon and based on a fork from the ELK stack project. In order to provide a lightweight and plug&play environment, OpenSearch installation was performed leveraging *Docker*¹³ virtualization containers.

Moving the focus to the endpoint, the Endpoint Detection and Response (EDR) systems state of the art is analyzed, considering both open source and enterprise available tools. Their main characteristics and limits are highlighted, with a special emphasis on what should be improved, in the near future, in order to build more resilient systems. A focus on Predictive Analysis is proposed with the discussion of a new Machine Learning algorithm able to leverage AI characteristics. The proposed solution is based on *BERT* [24], a Machine Learning algorithm originally released by Google, which is a Transformer model that performs Bidirectional Deep Learning analysis. Leveraging Transformers capabilities, it is possible to use BERT for work with Natural Language Processing (NPL) with its Masking Language Model (MLM) capabilities. In this way, the model is expected to learn from a baseline of clean logs which are “normal” behaviours. Then, thanks to MLM, the algorithm is expected to detect which logs deviates from the baseline scenario and hence may represent a potential threat.

⁸<https://www.elastic.co/elastic-stack/>

⁹<https://www.elastic.co/elasticsearch/>

¹⁰<https://www.elastic.co/logstash/>

¹¹<https://www.elastic.co/kibana/>

¹²<https://opensearch.org/>

¹³<https://www.docker.com/>

Here is where the two halves are merged together; logs collected with *Osquery* are analyzed leveraging Machine Learning capabilities and suspicious ones are delivered to OpenSearch through Filebeats, the selected delivery agent. On centralized management side, where the ELK stack is actually working, Logstash will receive delivered logs and, through OpenSearch's version of Elasticsearch, they will arrive to the visualization engine; there the analyst will perform further examinations.

The goal of this work is to provide a base environment where future analysis and consideration can be performed, leveraging the open source nature of this effort. Improving considerations made in the first part, optimizing the proposed algorithm and building a reliable infrastructure for log collection and analysis, in a completely open and free solution, can represent a good starting point for all the organizations and public administrations that struggled in implementing security measures.

This work is a joint effort of Federico Talentino and Alessandro Bolla where, considering the two major parts of the work identified above, Federico focused on the first half of it while Alessandro on the second one.

The remainder of the document is organized as follows. In chapter 2, the background of the first contribution about Cyber Threat Intelligence is presented; in chapter 3, the background for the second contribution regarding Cyber Threat Hunting is presented; in chapter 4, a brief explanation of the context in which this thesis has been developed is given with an introduction of *Perimetro di Sicurezza Nazionale Cibernetica*; in chapter 5, the contribution in terms of the work done for CTI log collection is described; in chapter 6, the contribution in terms of work done for the Threat Hunting part is described; chapter 7 shows the experimental results of both the contributions and in the end, chapter 8 draws the conclusions about the thesis developed and presents some possible future improvements. Being the thesis shared, it is important to specify that chapters 2, 4 and 5 have been written by Federico Talentino while introduction, chapters 3 and 6 have been written by Alessandro Bolla. The two chapters regarding experimental results and conclusions have been written as a joint effort, with authors focusing on their respective subjects.

Chapter 2

Background - Cyber Threat Intelligence

2.1 Cyber Threat Intelligence

Cyber Threat Intelligence is one of the most important subjects to concentrate on, in order to prevent cyber attacks or mitigate their effects. Cyber Threat Intelligence is a matter of information that can be gained from different sources, from social media to dedicated feeds, from people words to a simple newspaper. Threat Intelligence is a key element for the survival of many businesses on these days, because it is capable of increasing a lot the ability of Security Teams to prevent cyber incidents or, to remediate better to their damages when too late to avoid those.

Cyber Threat Intelligence has a lot of benefits, both for big companies with a lot of expertise and for SMEs (Small and Medium Enterprises) that are slowly adopting it[39][81][66].

First of all, financial aspect must be considered: according to some studies, the cost of a data breach has been estimated to be a few billion dollars, an amount that directors would surely not be happy to pay. But money is not the only problem: protection of sensitive data and intellectual property are also of great relevance. Cyber intelligence is of paramount importance to anticipate adversaries and protect all these assets. Technical aspects must not be undervalued: intelligence can reduce incident response latency, drive software patching when vulnerabilities arise or help prioritize threats. A good Cyber Threat intelligence solution, is also able to facilitate the job of security teams that, can gain helpful information to improve efficiency. Collaboration and intelligence information sharing between different companies, are able to improve a lot the overall security posture of an enterprise.

In the following sections, focus will be put on some important aspects of Cyber Threat Intelligence with the aim of introducing to this important slice of the Cybersecurity world. From now on, Cyber Threat Intelligence will be often called CTI for the sake of simplicity.

2.1.1 Definitions

The National Institute of Standards and Technologies (NIST¹) defines threat intelligence as “Threat information that has been aggregated, transformed, analyzed, interpreted, or enriched to provide the necessary context for decision-making processes” [59]. One important aspect that may be underlined here, consists in the sequence of operations: aggregate, transform, analyze, interpret, and enrich that are the typical steps under which CTI information goes, in order to be made in some way actionable and avoid to remain just information. The second aspect that is important to underline is the *decision-making*: intelligence activities not always, but often, end up with a decision about something that can be made to protect someone assets, respond to an attack or mitigate an incident.

A less official, but still clear, definition is given by Wikipedia: “CTI is knowledge, skills and experience based information concerning the occurrence and assessment of both cyber and physical threats and threat actors that is intended to help mitigate potential attacks and harmful events occurring in cyberspace” [96]. This second definition is remarkable for other two aspects that are not underlined in the previous one: the skills and the experience-based information. Working in CTI field requires not only technical but also soft skills. These can be useful during the reasoning process that guides the analyst from the cold information, to the construction of a higher level path that is essential to understand what are the patterns behind an attack and consequently how to respond. The experience then comes naturally, helping to reduce the amount of time needed to put in practice some actions after an information arrives.

Kaspersky is a major Security Solution provider and gives a more extended CTI definition, from which a nice passage can be extracted: “Threat intelligence is about sifting through piles of data. It’s examining it contextually to spot real problems, and deploying solutions specific to the problem found” [43]. This definition permits to learn other aspects of threat intelligence: the necessity to examine tons of data and, the ability to discriminate the useful ones, from which solutions can be built, from not useful data (i.e., attack detection false positives).

Another good definition of what Threat intelligence means, is given by the American security company CrowdStrike that tells “Threat intelligence is data that is collected, processed, and analyzed to understand a threat actor’s motives, targets, and attack behaviours. Threat intelligence enables faster, more informed, data-backed security decisions and change their behaviour from reactive to proactive in the fight against threat actors.” [7]. This definition includes some of the aspects already outlined in the previous ones, and adds the idea of studying the threat actor’s behaviour and intentions. This key concept will be described more in depth during a later chapter. Moreover, the concept of the fast response is also important because it’s crucial to respond to security incidents as fast as possible to try to reduce damages. The idea to act proactively instead of reactively is fundamental in the direction of anticipating the movements of an adversary.

¹<https://www.nist.gov/>

2.1.2 Brief history and diffusion

Once presented the formal definitions of Cyber Threat Intelligence, a good way to understand its reason for being is to retrace quickly its history [42][2][21].

Without travelling too much back in time, it is possible to state that the concept of intelligence in Cybersecurity, started to be relevant during the period between the end of the 90's and the begin of year 2000s. In that period, primitive SIEMs and firewalls started to appear: they were based on IP and URL's blacklists and a great amount of the job of signaling possible menaces was done manually by security specialist who directly created reports for customers. Years passed and cyber threats started to spread faster around the world with IOCs (Indicators of compromise) becoming too many to be managed by security software solutions. During those years, Machine Learning (ML) and Artificial Intelligence (AI) started to spread and gave a concrete help in the analysis of the increasing amount of data.

The problem of the work done by AI and ML, was the big number of false positives alerted. Once realized the problem, the role of human analysts and researchers started again, to be far more appreciated. The solution was in fact, to use ML software together with the human ability to distinguish false positives.

The biennium 2018 - 2019 showed again an incredible explosion of interest in CTI. During those years, many companies were born, and many new products were commercialized. Nowadays also smaller enterprises are gradually starting to integrate intelligence solutions or to outsource them to protect their businesses. At the same time, nation states are making huge investments in threat intelligence to survive on the cyber-war that today characterizes the relationship between nations. Moreover, the role of threat intelligence during years has progressively evolved from reactive to proactive: CTI teams are always more responsible of company businesses and, need to be able to identify and stop threats earlier before the effective attack is performed. The focus is now no more reduced to IOCs but also, higher-level concepts like adversary tactics, techniques and procedures (TTPs) are now considered of the same importance (TTPs are described in 2.2). Also the role of CTI teams has assumed now a different shape: CTI is no more just about writing threat reports but must consider every aspect of an organization regardless the fact of being in a commercial company or in a governing institution. Cyber Threat Intelligence market is still increasing a lot on these days: some research state that it could be worth 13 billion of US dollars until 2023 and, not far from 16 billion of US dollars by 2026 [85].

2.1.3 CTI types

Cyber Threat Intelligence can be organized in layers. It is made of a lot of different activities done by different specialists and, it can be useful to categorize it in order to depict a clear schema that may be helpful to fully understand the topic [15][16][17][67][92][87].

CTI is usually divided in three main types: *Strategic*, *Tactical* and *Operational*. Some models are used to add a fourth type that is *Technical* CTI that is used to separate some more low-level details from the tactical layer. In the following paragraphs, each of these four levels of application will be described highlighting their main features.

Strategic CTI Strategic level is a totally non-technical type of CTI. Its target is the

understanding of who is behind an attack campaign, what are its motivations and why a specific objective was chosen. Strategic CTI information is very high-level and it is also consumed by high-level management of an organization: typically board members like CISOs or other executives. The audience is therefore non-technical. This CTI analysis leads to significant business decisions regarding vital aspects of a company; decisions may regard employees, infrastructures, customers or applications. Practically this intelligence is often made explicit in the form of reports or white papers but also policy documents. Strategic information often consists of topics like TTPs' evolution, money impact of attacks, geopolitical cyber situation, statistics on data breaches and malwares, APT tendencies (chapter 3), how industrial sectors may be impacted by attacks and money impacts in general.

Tactical CTI Tactical CTI is placed one level under the strategic. Here the key questions are *how* the cyber attack is performed (so the domain is that of Tactics, Techniques and Procedures) and *where* it is performed (what is the extent of the attack). Tactical information is directed to IT service managers and security professionals that may be skilled in network, architectures or administration. A key concept here is the one of threat hunting (deeply analyzed in chapter 3), which is crucial to identify incidents and to prevent future ones. Tactical intelligence is usually written on technical papers and, the main sources of information is third party feeds, technical reports about malware, campaigns, tools, attack groups and so on. Human intellect plays a big role here: people involved need to be able to correlate many information and act as soon as possible. Tactical intelligence has the twofold function of helping prevent future attacks and gathering forensic evidences.

Technical CTI The key concept of the technical layer is the IOC (indicator of compromise). IOCs can be different types of artifacts such as IPs, URLs, file hashes or domain names. They are used to identify malicious activities on the network or malicious artifacts that arrives at the network boundaries. IOCs are quite volatile and may be valid for a limited time (sometimes hours) so it's important to monitor them early enough. This kind of very technical information is better consumed by tools like firewalls, SPAM filters, IDSs, SIEMs or SOARs. These indicators are usually collected thanks to CTI sharing and honeypots and, technical CTI consumers are SOC staffs.

Operational CTI Operational threat intelligence is also focused on TTPs, precisely on the discovery of new threats and attacks, differentiating itself from tactical intelligence which concentrates primarily on already known threats. Activities performed in operational intelligence, may also include a sort of cyber espionage made against less experienced hacker groups to intercept their intentions and understand who they are and how they work. This activities are usually pursued by security managers together with fraud detection groups, forensics specialists and heads of incident response teams. The main sources of operational information are social media, chats in general and human activities. Furthermore, this kind of job requires a very good attention to legal aspects: normally a private entity can not access enemies' infrastructure and it's easy to have legal issues if some limits are overtaken. For this reason, this kind of intelligence is quite more common in governments than in enterprises.

2.1.4 Typical CTI management: the SOC

The Security Operation Center (SOC) is the place where security related questions of a company are usually managed [94][46][102][47]. Preventing, detecting, analyzing and responding are key activities in a SOC; inside it all the IT infrastructures are constantly monitored and defended. While bigger companies may implement an internal SOC, smaller corporations are used to outsource SOC services buying it from a MSSP (Managed Security Service Provider). SOCs typically provide three main services:

- **Management services:** management of security functionalities of IT infrastructures;
- **Monitoring services:** real-time monitoring of the infrastructure aimed at blocking attacks, intrusion attempts and control eventual misuse;
- **Proactive services:** may include vulnerability and security assessment, in order to improve company's security level.

To get into specifics, it is possible to identify some average services offered by a SOC like security device management, configuration and fault management, reporting services, DDOS mitigation, security alerting, vulnerability assessment or simply technical security assistance. Cyber threat intelligence has a major importance inside a SOC because, thanks to its activity of information gathering and sharing, it plays a role of primary importance in all the activities described before. The technology employed inside a SOC consists of some specific types of software that will be listed and briefly analyzed here:

- **SIEM:** stands for Security Information and Event Management and is useful to aggregate and correlate information, IOCs and other security data from various feeds and sources;
- **SOAR:** Security Orchestration, Automation and Response as the name says, helps security specialists to automate some repetitive triage processes to give a quicker response to incidents avoiding the waste of time;
- **EDR:** Endpoint Detection and Response is a concept of tool that includes some different features that may also characterize the tools previously described: they essentially perform real-time continuous monitoring and rule-based response activities;
- **GRC:** Governance Risk and Compliance systems are higher level tools that align information technology with business objectives, ensuring that regulatory compliance requirements are met;
- **IDP/IPS:** respectively Intrusion Detection and Intrusion Prevention systems that, thanks to static rules or also ML algorithms, may recognize ongoing attacks or better anticipate them;
- **Vulnerability assessment and penetration testing tools:** all the possible tools that may be employed to assess the security of a system or can be used to perform pen-testing.

This is not an exhaustive list of all the software tools you may find in a SOC but just an overview of the main used: the more security tools are included in a SOC, the more its efficiency and effectiveness will improve. It is important to note that, many of the functionalities described above are often enclosed in the same TIP (Threat Intelligence Platform). People working in a SOC are often led by a CISO (Chief Information Security Officer) who coordinates some technical and non-technical managers; analysts are at the base layer and can be organized in different levels (L1, L2, L3), characterizing their increasing responsibilities and mansions.

Without going too much into detail, this chapter concludes showing the typical process organization of a SOC that consists in five steps and is depicted in the figure 2.1.

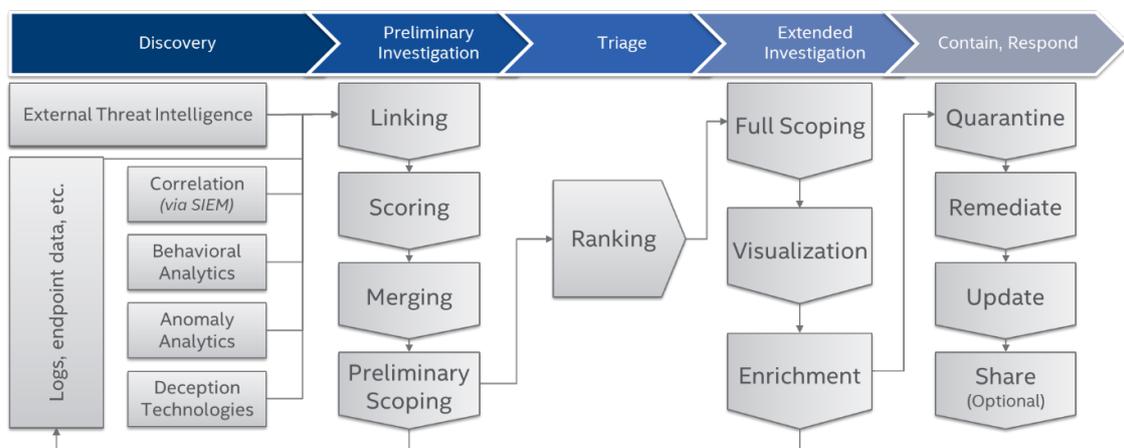


Figure 2.1. Threat management plans and organization [46].

2.1.5 CTI ontologies

Following the most popular and consequently main meaning of the word ontology, it is possible to define it as: “The branch of philosophy that studies concepts such as existence, being, becoming, and reality” [97] or in a different way, “A branch of metaphysics concerned with the nature and relations of being” [23]. But not everyone knows that the ontology has also a strong meaning in the world of computer sciences and this will be briefly analyzed in the following section.

The concept of ontology in Computer sciences

The Wikipedia definition says: “In computer science and information science, an ontology encompasses a representation, formal naming and definition of the categories, properties and relations between the concepts, data and entities that substantiate one, many or all domain of discourse. More simply, an ontology is a way of showing the properties of a subject area and how they are related, by defining a set of concepts and categories that

represent the subject.” [98]. This definition is quite simple to understand and gives a clear picture of the field about which this chapter is going to concentrate. Ontologies in IT are used to correlate concepts in a specific field and, organize them in order to obtain a clearer overall idea of the subject. Ontologies aim at representing objects, ideas and events together with their properties and the relationships.

Ontology concept started to be applied to computer sciences in about mid-1970s, particularly in the field of artificial intelligence; this tendency continued in 80s. In 1995, thanks to a famous paper by Tom Gruber [31], the word started to be used closer to the ideas of semantic networks and taxonomies. Lately, Gruber gave more successive precise definitions of Ontology by himself, trying to distinguish the concept from the earlier one of taxonomy. In 2016 finally, taking in account Gruber’s definitions, other authors stated that: “An ontology is a formal, explicit specification of a shared conceptualization that is characterized by high semantic expressiveness required for increased complexity.” [26].

As anticipated before, ontologies nowadays share a series of key concepts: individuals, classes, attributes, relations, function terms, restrictions, rules, axioms and events and they are commonly defined using an ontology language. Ontologies can be categorized according to three main types:

- **Domain ontologies:** represent concepts which belong to a realm of the world, each domain ontology typically models domain-specific definitions of terms;
- **Upper ontologies:** they are a model of the commonly shared relations and objects that are generally applicable across a wide range of domain ontologies;
- **Hybrid ontologies:** a combination of domain and upper ontologies.

The construction process of an ontology can be named ontology building or ontology engineering and, there exist some ontology editors (software) that are used to help authors in this process. During years, numerous languages have been developed to describe ontologies; a pair of quite famous example can be the couple OIL+DAML² which is nowadays obsolete and the OWL³ language used together with the RDF⁴ standard.

Cyber Threat Intelligence ontologies

Threat intelligence ontologies are a niche of computer science ontologies. Actually, given the small quantity of specific cases, it is hard to talk about “Cyber Threat Intelligence ontologies” and this fact has been clear since the first research done on this field. To present briefly but at least in a quite complete way the concept of ontologies in this field, it is better to broaden the explanation to “Cybersecurity ontologies” in general, of course taking in account examples that may be placed near the intelligence field. An overview of some of the main ontologies described in scientific literature will be presented starting from early 2000s to the modern days.

²<https://www.w3.org/TR/daml+oil-reference/>

³https://it.wikipedia.org/wiki/Web_Ontology_Language

⁴<https://www.w3.org/RDF/>

The presentation of main CTI/Cybersecurity ontologies starts from a famous paper published in 2003 by Undercoffer et al. [88]. The paper suggests the necessity to make a transition from previously used taxonomies to ontologies because, these provide powerful constructs and, machine interpretable definitions of concepts and their relationships within a domain. Ontology terminology is used to provide a formal specification of objects and their relationships within a specific domain.

The same paper, suggests then that CTI information should be collected in a knowledge base which will be queried for evidence of an intrusion. Undercoffer et al., moreover, identify fundamental properties of taxonomies and take a subset of them as essential for ontologies. An ontology must be: mutually exclusive, exhaustive, unambiguous, useful, objective, deterministic, repeatable and specific. To conclude, they state that the power and utility of the ontologies are realized by the fact that those can express the relationships between the collected data, and use them to deduce that particular data represents a particular type of attack. This concept has to be kept in mind because, it is at the base of the work contribution that will be presented later in this thesis.

To find a second important example of ontology to present, it is necessary to make quite a big temporal jump from 2003 to 2016 when Syed et al. [79] presented their work, the *Unified Cybersecurity Ontology*. This ontology is focused on the topic of information sharing and exchange. The UCO ontology is presented as an attempt to shift from syntactic representation of cybersecurity to a more semantic one. UCO wants to unify information from heterogeneous sources to support reasoning and rule-writing. The target of this ontology is in general the unification of many cybersecurity standards like CVE⁵, CWE⁶, CVSS⁷, CAPEC⁸, Cyber Kill Chain⁹ or STIX, just to mention a few. A particular attention may be dedicated to STIX (which will be analyzed deeply in Section 2.1.6): following the paper, UCO has been created as a semantic version of STIX. UCO is organized in eight classes:

- **Means:** represent methods for executing an attack;
- **Consequences:** represent possible outcomes of an attack;
- **Attack:** characterizes a cyber threat attack;
- **Attacker:** identification or characterization of the attacker;
- **Attack Pattern:** provides usual methods for exploiting software, providing attackers perspective and guidance to mitigate effects;
- **Exploit:** an individual exploit;
- **Exploit target:** vulnerabilities or weaknesses in software that may be exploited;

⁵<https://cve.mitre.org/>

⁶<https://cwe.mitre.org/>

⁷<https://www.first.org/cvss/>

⁸<https://capec.mitre.org/>

⁹<https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>

- **Indicator:** a pattern identifying certain observable conditions.

Moreover, there are a number of classes representing other concepts like Network or Processes and all these classes are mapped to an object in STIX; UCO in fact, wants to be like an extended version of STIX.

Another interesting paper published during year 2017, is the one by Mavroeidis et al. [45]. Authors outlines the necessity of common representation, standard formats and protocols for sharing and understanding of relevant concepts. The major problems for this issues are recognized in the confusion that is diffused by vaguely defined terminology, in the lack of a formal standard representation of relevant information and, lack of coherent relationships between different layers of abstraction in ontologies. The authors say that ontologies could solve these problems for threat intelligence but, the existing ones, have been often introduced to address the field of cybersecurity and not specifically the CTI one. The methodology presented in the paper is based on two models:

- The **Detection Maturity Level Model (DML)** shown in figure 2.2: organized in nine levels from DML-0 to DML-9, describes the maturity of an organization regarding the ability to consume and take decision about threat information;
- The **Cyber Threat Intelligence Model (CTI)** shown in figure 2.3: created with the purpose to evaluate taxonomies, ontologies and sharing standards, it is not hierarchical like the DML but represents the information needed for advanced CTI and eventual attack attribution.

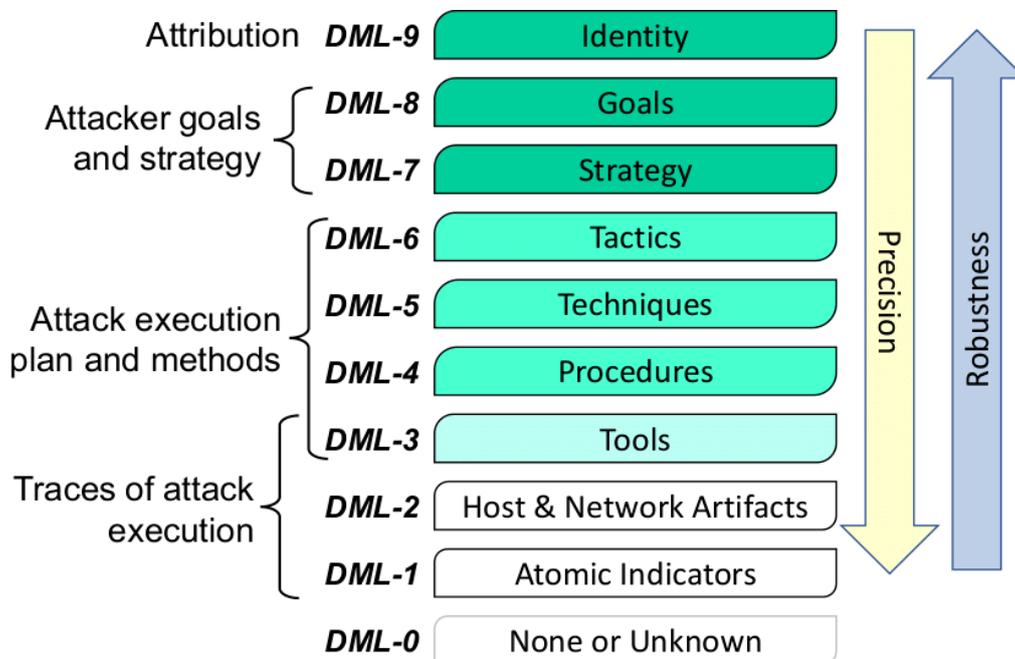


Figure 2.2. The Detection Maturity Level Model - DML [45].

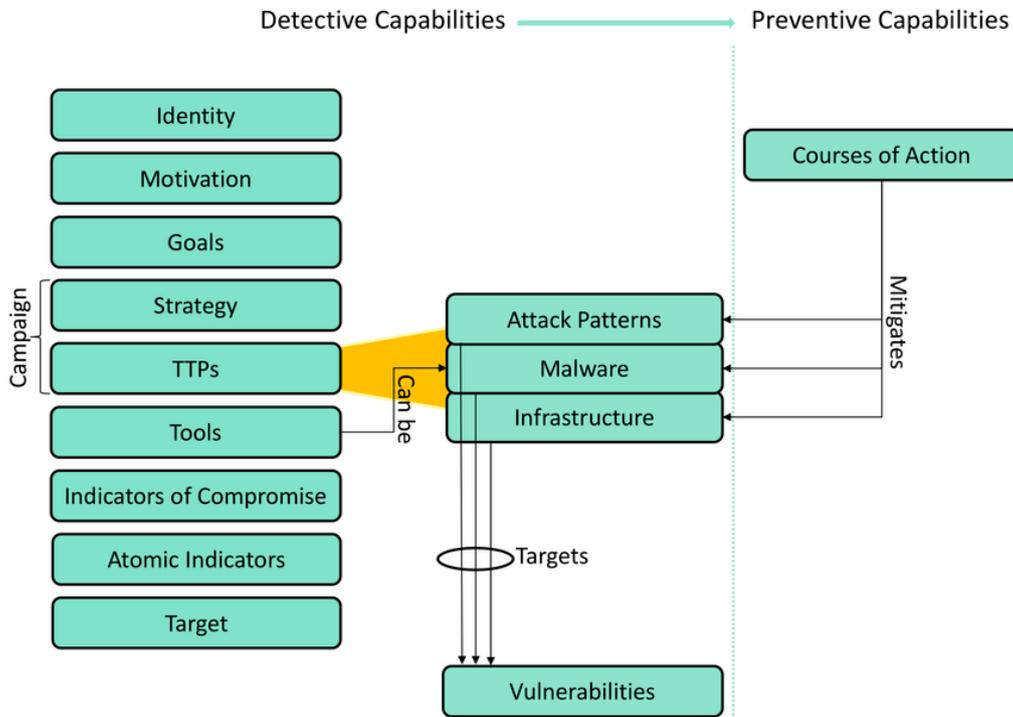


Figure 2.3. The Cyber Threat Intelligence Model - CTI [45].

Besides this, authors make a categorization of taxonomies and standards in three groups.

- Enumerations, such as CVE, CWE, CAPEC or ATT&CK;
- Scoring systems, such as CVSS;
- Sharing standards, STIX is recognized as the most used one.

Some key points about ontologies, are then identified and discussed by authors:

- **Relationships and reasoning:** there's a lack of relationships between taxonomies that should be addressed to create a common ontology that is not only limited to a specific sub-domain of CTI but more general;
- **Knowledge collection:** it should be taken as an opportunity the idea to put the big amount of knowledge together in a common ontology: that would help analysts in whose head only the knowledge resides today;
- **Attribution:** there should be more attention on attack attribution that is here recognized as the most important aspect of threat intelligence, this would help recognize and better deal with attackers;
- **Trust and Uncertainty:** it is needed to give importance to the fact that threat actors are aware of attribution methods and use masquerading techniques; moreover,

there is a lack of standard requirements for quality of evidence so it is needed to take into consideration the level of certainty of all the information.

Authors conclude confirming the absence of an ontology available for CTI and asserting the necessity to build one.

Proceeding in chronological order with the ontology overview, the next interesting stop is proposed by C.Onwubiko [64]. CoCoa is a process ontology which is aimed at shifting from simple event logs to considering five CTI sources, namely: *events and logs*, *network information*, *structured digital feed*, *semi and unstructured feed* and *threat intelligence*. The CSOC (Cyber Security Operations Center) analysis process is outlined: it starts from an internal/external **source** which provides CTI information that is collected and analyzed by a **sensor**; then an eventual attack crosses a **detect** phase and a **monitor** phase where dashboards are analyzed. The last two steps are the **respond** and **recover** steps where respectively, mitigation plans, and recovery actions are put in place. The process set out by CoCoA, corresponds to the one implemented in this thesis: logs are collected from a source(the endpoint), thanks to a sensor(Osquery). After this, detection and monitoring are performed thanks to ML and SIEMs.

A further interesting reading has been published in 2021 by Yeboah-Ofori et al. [99]. This paper is concentrated on two main subjects: Cyber Threat Ontology (CTO) and Adversarial Machine Learning (AML) with the aim to perform threat analysis and prediction. CTO describes, thanks to a formal and structured approach, organizational security concepts, object relationships and interdependencies. The goal of CTI, as stated, is to extract from intelligence, relevant attack information to ensure consistency and accuracy. The AML techniques instead, are used to inject malicious data in the dataset during training and testing phase to distort the classification model. The approach followed in this case consists in modelling CTO for APT attacks and then applying AML attacks to manipulate the model.

One last considerable example is the work by Merah et al. [52] inside which STIX is introduced as the de facto standard language for expressing and sharing cyber threats unambiguously. The Ontology is described as a descriptive logic for representing objects and their relationships within a specific domain. STIX, is the baseline that is chosen to unify information sharing and construct eventual ontologies. According to the authors, STIX is good also because is created taking concepts from frameworks like CAPEC or ATT&CK. The authors confirm the lack of a methodology to create ontologies and the necessity to create an ontology that allows to solve security issues with an inference mechanism. The work results in a quite difficult to comprehend but, absolutely complete example of cyber ontology. Ontology is depicted in figure 2.4 below.

The examples of ontologies and related models are a lot: just some of particular interest have been selected and described. The key aspect that must be taken away from this chapter is the strong need of models and frameworks that are able to describe clearly the world of cyber threat intelligence and the threat landscape in general. These models when mature, will be helpful to recognize and respond to attacks and threats. Next section will focus on STIX, the aforementioned Cyber Threat Intelligence standard language.

OASIS¹² organization. In STIX's history, two main versions can be found: the 1.x and the 2.x. There have been subversions of these two but, the big differences for the moment must be searched between the two main versions. Without going too much into technical detail, the main differences between versions 1 and 2 are:

- The merging of STIX and CybOX¹³ standards: while in version 1 CybOX was a standalone standard for representing some observables in threat intelligence, in STIX v2, OASIS decided to merge the standard into STIX;
- While STIX v1 was based on XML¹⁴ language, STIX v2 is based on JSON¹⁵ format: both the languages have benefits, but JSON has been evaluated as lightweight, simpler to use and expressive enough for the semantics of CTI;
- All the **SDOs** namely, Stix Domain Objects (later explained), are at the top-level in STIX v2, while in STIX v1 they could be embedded inside each other, leading to a bit of confusion and difficulty;
- STIX 2.x introduces Stix Relationship Objects (**SROs**) to create links between other objects;
- There are also other less important differences regarding Data Markings, Streamlined model and Indicator pattern but those are not relevant here.

STIX has now reached its 2.1 version and is becoming more and more widespread both in government entities and in private businesses.

For the sake of completeness, talking about STIX requires to cite also TAXII¹⁶. Trusted Automated Exchange of Intelligence Information (TAXII), is a protocol for CTI exchanges over HTTPS and is at version 2.1. It defines APIs and a set of requirements for servers and clients. Although TAXII has been designed to work well with STIX and vice versa (both are maintained by OASIS), it can be easily used also in combination with other languages and standard, to share CTI. The two protocols remain independent.

Organization and features

STIX is organized in two main types of objects:

- **STIX Domain Objects** (SDOs): they are 18 in STIX 2.1, and each is characterized by some attributes to be filled, these objects represent some main concepts of CTI like Attack Pattern, Course of Action, Campaign, Malware, Vulnerability and so on;
- **STIX Relationship Objects** (SCOs): they are two, namely Relationship and Sighting and are used to create connections between different SDOs.

¹²<https://www.oasis-open.org/>

¹³<https://cyboxproject.github.io/>

¹⁴<https://it.wikipedia.org/wiki/XML>

¹⁵<https://www.json.org/json-it.html>

¹⁶<https://oasis-open.github.io/cti-documentation/taxii/intro.html>

To be complete, also SCOs (STIX Cyber-Observable Objects) must be cited: these objects derive from CybOX standard mentioned before and are now integrated in SDOs and SROs. The complete object list with specific descriptions can be viewed on the website [62].

As already told above, STIX's most recent version is based on JSON format: the fact that JSON is used, makes STIX easily machine-readable, but there are also instruments such as the *cti-stix-visualization* library which is freely available on Github and, can create a graphic representation of a STIX document. This is great for analysts because they can reason on STIX documents having a graphical picture of the situation.

STIX objects are useful because can be linked together to represent all the information related to a security event. In this way, TTPs may be recognized and in some cases, the chaining of objects might lead to the recognition of an APT attack too. STIX objects can be collected in a single JSON file thanks to STIX bundles. In a possible real scenario, two companies may exchange CTI information using bundles. For example, a company may detect a security episode in its network, collect and organize detailed information about it and then push this information to a TAXII server. Other companies can retrieve this intelligence from server and put in place appropriate countermeasures to avoid risks. Examples of STIX bundles and also some intelligence reports written thanks to STIX can be found on the official website [63]. A lot of other resources can be found on STIX website [61], where the complete documentation is available, together with the documents and links to a list of Github repositories which contain useful things like python APIs for STIX, JSON validators, converters from v1 to v2 and so on.

The concept of threat sharing with STIX

During the previous study of ontologies, what emerged as a key concept in the modern days is the information sharing of Cyber Threat Intelligence. For sharing intelligence, a common language is clearly needed and among the various OpenIOC¹⁷ and Incident Object Description Exchange Format (IODEF)¹⁸, STIX is perceived as the de-facto standard. Also TAXII is preferred in general to other competitors like RID¹⁹ (usually coupled with IODEF). This chapter is strongly related with the one about ontologies, because basically all these languages (STIX, IODEF and so on) are built following the typical organization of a semantic model like the ontology.

Burger et al. [12] tried to classify main existing languages in a taxonomy, following what they define an agnostic framework. They saw the frequent overlaps between language ontologies and tried to propose this framework to compare them. Their taxonomy is divided in five levels: **transport, session, indicators, intelligence, 5W's** (who, what, when...). This architecture is used then to analyze from each of the 5 points of view, mainly the couples STIX/TAXII and RID/IODEF, but also other technologies that are taken into account. Details can be found inside the paper but, the important conclusion is that also if the research has been published in year 2014, when STIX was still a primitive version of the actual one, at that time it was already considered providing a broader scope of terms

¹⁷<https://www.mandiant.com/resources/openioc-basics>

¹⁸https://en.wikipedia.org/wiki/Incident_Object_Description_Exchange_Format

¹⁹<https://datatracker.ietf.org/doc/html/rfc6545>

and, also the gaps filled by IODEF, where considered as easy to be integrated, also with more details on STIX’s major objects.

Two years later another paper by Asgarli et al. [3], recognized again to STIX the role of the most extensive standards, having definitions for a lot of objects and leveraging information from many other standards such as MAEC²⁰ or CAPEC. The authors tried then to unify STIX and IODEF choosing among others, the strategy of merging the second inside the first in a single RDF/OWL ontology which implementation is out of scope here. What is interesting is again the fact that many experts in cybersecurity have confirmed during years that although more standards exist and everyone has its advantages, overall STIX is the preferred.

Another example of STIX validity corroboration is given in 2018 in the work by Zhao et al. [103] where the ontology model proposed is said to be created taking STIX objects as a strong reference.

Importance in the domain of this work

To sum up, STIX will not be taken as a central argument in the contribution of this thesis. Despite this, its role has been important for the practical comprehension of ontologies first. Moreover, it has been a key passage that led to the framework exploited for this thesis which is the famous MITRE ATT&CK described in section 2.2. In the end, STIX remains a meaningful way to express and share intelligence and there are a lot of possible roads that may be followed for its integration in the future.

2.2 TTPs and MITRE ATT&CK

In this section, the primary framework that has been leveraged in this contribution, the MITRE ATT&CK, will be introduced together with the basis above which it is built: the Tactics, Techniques and Procedures concept, easily callable with the term TTP.

2.2.1 Tactics, Techniques and Procedures

The triplet in question, that will be almost ever called TTP for the sake of brevity, can be comprehended simply with this definition: “The behaviour of an actor. A **tactic** is the highest-level description of this behaviour, while **techniques** give a more detailed description of behaviour in the context of a tactic, and **procedures** an even lower level, highly detailed description in the context of a technique.” [59]. Some more details about the triplet is given in the following paragraphs.

Tactics

As mentioned above, *tactics* are the highest level concept. A tactic represents what the adversary wants to achieve with his attacks and how it behaves to pursue his intentions. A simple example is given by the case in which a malicious employee decides he wants to

²⁰<https://maecproject.github.io/>

get access to the system of his superior to stole data regarding salaries in the office. After an exploration of the possibilities, not having enough hacking skills he decides to steal his manager username/password, performing a sort of *credential access*. The bad guy knows when the credentials are obtained, he needs a way to use them without being discovered, he needs a way to perform *defense evasion*. Last but not least, the employee needs a way to steal effectively the information about salaries he desires, so he wants to do *exfiltration* of data.

Techniques

The *technique* is a lower level concept and consists in how practically a tactic of the attack is put in place. A bunch of techniques that may be applied to the example presented above will be listed here. Each of the techniques listed, is related to one of the tactics previously described (indicated in brackets).

1. **Social engineering technique:** simulate phone call pretending to be someone other to get private information (*credential access*);
2. **Hidden access to stolen account:** avoid leaving traces using someone other computer, without being physically seen (*defense evasion*);
3. **Private information stealing:** taking photos of the screen showing files with employees list with their salaries (*exfiltration*).

It is important to specify that a tactic, may be composed of more techniques also if in the example there is a one-to-one correspondence. Next paragraph, about procedures will clear any doubt.

Procedures

The *procedure* is the way in which every technique, corresponding to a specific tactic, is effectively applied in order to achieve the target. Following the example presented before, it is possible to think that the bad employee decides to apply social engineering simulating a phone call to the manager in which he pretends to be the CISO of the company. It is supposed the bad employee is smart enough, simulating a fake voice and calling from a public number at a reasonable time (maybe eleven o'clock in the morning), avoiding to be recognized. During the phone call, the bad actor tells the Manager that there have been problems with managers credentials after an attack with consequent data breach: he is calling each manager to ask for usernames and passwords in order to verify if they were stolen or not. Supposing the manager is not aware of cybersecurity frauds and gives the credentials to the fake CISO, first step is gone. The bad guy thanks the manager and promise to call the day after with news about credentials. Suppose now that the employee waits for the perfect moment, maybe at the end of working hours, and manages to remain alone. He cheats colleagues asking why he still there, with the excuse of having troubles with some contracts and proposing to close the office himself. When everyone is away, the employee leverages one of his colleagues PC (to avoid leaving traces) to log in the manager account and access the desired files. Second step completed. At that point, the employee takes his smartphone and takes some photos of the file with the salaries of each people in

the office, then disconnect and goes home. The day after, he calls the manager ensuring him that his credentials were not part of the breach and he takes no risk. The attack is completed.

The example described gives quite a clear idea of what a TTP consists of. Next section will introduce MITRE ATT&CK, a big database of TTPs.

2.2.2 Adversarial Tactics, Techniques & Common Knowledge: ATT&CK

ATT&CK is a knowledge base of tactics, techniques and procedures created by MITRE corporation. It is open, free, and globally accessible by everyone. The framework consists of a well-designed collection of adversary behaviours and attack's lifecycle. ATT&CK can be used for many disparate operations: it can be used for offensive activities like adversary emulation and red teaming, for SOC maturity or defensive gap assessments, for the development of behavioural analytics or for Cyber Threat Intelligence purposes. In particular, it can be used in manual or automated ways to map data aggregated from SIEMs to the techniques in the framework. Moreover, it can be leveraged to map events detected on EDR agents, prioritize risks and speed up responses. Worthy of mention is the fact that ATT&CK comes together with a lot of resources: a *getting started* guide, training, conferences, related projects and the possibility to explore elder versions of the framework if needed. The framework can be considered as a behavioural model and is used both by adversary emulation teams and defensive teams. Next paragraphs will highlight some features of the framework [53][48][77].

Brief history

ATT&CK was born as a project to document and categorize post-compromise adversary behaviours during some MITRE's systematic adversary emulation exercises that started back in 2010. First version of it was created in 2013 and was mainly focused on Windows OS: after refinements it was publicly released in 2015 with its first 96 techniques organized in 9 tactics. Since then, thanks to its community, ATT&CK has gone through an exponential growth until in 2017 Linux and MacOS systems were included. 2017 was the year in which the Enterprise and Mobile versions of it were divided and the PRE-ATT&CK model was born. This last model was a complementary one, focusing on the activities performed mainly before the effective attack execution. Following history, the versions for Cloud and ICS systems were born respectively in 2019 and 2020. Starting from year 2018, two or three versions per year were published until now. The framework has now reached its version 10 published in October 2021 which is also the one on which this thesis is based. Old versions are still accessible from the website. Major changes in recent years have been the introduction of sub-techniques since v7 and the merge of PRE-ATT&CK with the main matrix since v8.

ATT&CK matrices

ATT&CK tactics, techniques and sub-techniques are well organized in matrices. More precisely, the actual version contains three matrices: *enterprise*, *mobile* and *ICS*. All these

matrices are complete, but inside the website they may be modified to narrow down on specific use cases. This means that for mobile matrix, it's possible to select just Android or just iOS techniques. The same thing is possible with enterprise matrix where PRE, Windows, MacOS, Linux, Cloud, Network, Containers are selectable alone if just one of these domains is needed. ATT&CK matrices are organized with Tactics on columns and Techniques associated to each tactic. Techniques may repeat under one or more tactics. Each technique then, is extensible in order to show its sub-techniques when there are some. A detailed representation of a piece of the enterprise matrix with a technique showing also its sub-techniques is in figure 2.5. The complete enterprise matrix is very big and it is not possible to display it here but, ATT&CK website is a good place to explore it.

Resource Development (6 techniques)	Initial Access (9 techniques)	Execution (12 techniques)	Persistence (19 techniques)	Privilege Escalation (13 techniques)	Defense Evasion (40 techniques)	Credential Access (15 techniques)	Discovery (29 techniques)		
Acquire Infrastructure (6)	Drive-by Compromise	<ul style="list-style-type: none"> PowerShell AppleScript Windows Command Shell Unix Shell Visual Basic Python JavaScript Network Device CLI 	Account Manipulation (4)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)	Adversary-in-the-Middle (2)	Account Discovery		
Compromise Accounts (2)	Exploit Public-Facing Application		BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Brute Force (4)	Credentials from Password Stores (5)	Application Discovery	
Compromise Infrastructure (6)	External Remote Services		Command and Scripting Interpreter (9)	Boot or Logon Autostart Execution (15)	Boot or Logon Autostart Execution (15)	Build Image on Host	Exploitation for Credential Access	Browser Discovery	
Develop Capabilities (4)	Hardware Additions		<ul style="list-style-type: none"> Container Administration Command Deploy Container Exploitation for Client Execution Inter-Process Communication (2) Native API Scheduled Task/Job (6) Shared Modules Software Deployment Tools System Services (7) 	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)	Deobfuscate/Decode Files or Information	Forced Authentication	Cloud Infrastructure Discovery	
Publish Accounts (2)	Phishing (3)			Browser Extensions	Create or Modify System Binary	Create or Modify System Process (4)	Direct Volume Access	Forge Web Credentials (2)	Cloud Service Dashboard
Gain Capabilities (6)	Replication Through Removable Media			Compromise Client Software Binary	Create Account (3)	Domain Policy Modification (2)	Domain Policy Modification (2)	Input Capture (4)	Cloud Service Discovery
Manage Capabilities (5)	Supply Chain Compromise (3)			Exploitation for Client Execution	Create or Modify System Process (4)	Escape to Host	Execution Guardrails (1)	Modify Authentication Process (4)	Cloud Storage Resource
	Trusted Relationship			Inter-Process Communication (2)	Event Triggered Execution (15)	Event Triggered Execution (15)	Exploitation for Defense Evasion	Network Sniffing	Domain Discovery
	Valid Accounts (4)			Native API	External Remote Services	Exploitation for Privilege Escalation	File and Directory Permissions Modification (2)	OS Credential Dumping (8)	File and Directory Discovery
				Scheduled Task/Job (6)	Hijack Execution Flow (11)	Hijack Execution Flow (11)	Hide Artifacts (9)	Steal Application Access Token	Group Policy Discovery
				Shared Modules	Implant Internal Image	Process Injection (11)	Hijack Execution Flow (11)	Steal or Forge Kerberos Tickets (7)	Network Scanning
				Software Deployment Tools		Scheduled	Impair Defenses (9)		Network Discovery
		System Services (7)				Indicator Removal on Host (6)		Network	

Figure 2.5. ATT&CK enterprise matrix sliced²¹.

Work on matrices can be done thanks to ATT&CK Navigator. It is an instrument on which some more words may be spent because it has been useful for reasoning during the thesis development. It is a web-based tool that permits to explore all the matrices and select/deselect operative systems, show/hide sub-techniques, create layers, customize them and so on. Navigator permits to highlight different techniques with different colors, improving reasoning on the different APT groups or procedures in general. Layers then can be exported in many formats like JSON, MS Excel or svg images. Navigator web app is easy and intuitive to use and this confirms it as a powerful tool.

The whole resources included in ATT&CK are:

²¹<https://attack.mitre.org/matrices/enterprise/>

- **Tactics:** it's the adversary goal, the reason to perform an action (indicated with an ID like TAXXXX where XXXX are four numbers, e.g., TA0003 - Persistence);
- **Techniques:** how an adversary achieves a tactic goal by performing an action (indicated with an ID like TXXXX where XXXX are four numbers, e.g., T1087 - Account Discovery);
- **Data Sources:** represents the various information that may be collected by sensors collecting logs, they can include data components identifying specific properties of a source (indicated with an ID like DSXXXX where XXXX are four numbers, e.g., DS0026 - Active Directory);
- **Mitigations:** a list of security concepts that can be used to prevent execution of a technique (indicated with an ID like MXXXX where XXXX are four numbers, e.g., M1049 - Antivirus/Antimalware);
- **Groups:** they are sets of related intrusion activities with common name in the community; groups are mapped thanks to publicly available reported techniques use but, no one can say that a group will never use another technique not included in its ATT&CK list (indicated with an ID like GXXXX where XXXX are four numbers, e.g., G0005 - APT12);
- **Software:** a generic term that may indicate any kind of tool, from open source to commercial, from OS utilities to malware, that may be used to perform a behaviour modeled in ATT&CK (indicated with an ID like SXXXX where XXXX are four numbers, e.g., S0002 - Mimikatz).

Figure 2.6 shows the relationships between ATT&CK components. The work of this thesis is mainly focused on techniques of ATT&CK: next paragraph will analyze them in depth.

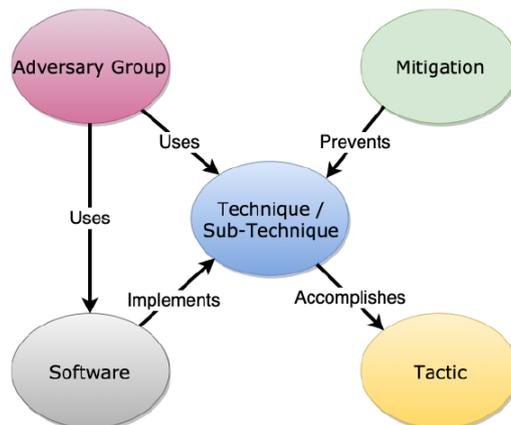


Figure 2.6. ATT&CK: relationships between main components [77].

Technique details

Each MITRE ATT&CK technique, has its personal description. For each technique, a list of possible procedures may be present including software or adversarial APT groups involving that technique; additionally, there may be lists of mitigation or detection mechanisms related to that technique. If a technique has sub-techniques, they are indicated adding a dot and three digits representing a progressive number to the technique name. To give an example, T1059 has eight sub-techniques, the first one is called Powershell²² and has the ID T1059.001. Sub-techniques when present, are listed together with their major technique (figure 2.7). Each technique, is completed with a series of basic information

Command and Scripting Interpreter

Sub-techniques (8) ^	
ID	Name
T1059.001	PowerShell
T1059.002	AppleScript
T1059.003	Windows Command Shell
T1059.004	Unix Shell
T1059.005	Visual Basic
T1059.006	Python
T1059.007	JavaScript
T1059.008	Network Device CLI

Figure 2.7. Sub-techniques list for technique T1059 - Command and Scripting Interpreter²³.

about itself like ID, applicable platforms, permissions required, impact type and so on. Not all the techniques contain the same information, it's possible that different techniques have slightly different fields. This information is easily readable in a small panel at top-right position of the technique's webpage (an example of panel for T1059 is shown in figure 2.8). The complete list of fields that may be present can be found in the philosophy paper about MITRE ATT&CK, cited at the top of this section. Sub-techniques are organized in the same way of techniques, but they are often more specific; this means descriptions and lists may become longer. Sub-techniques were introduced in 2020 to address some abstraction level issues appeared during years: these were in fact, leading to some confusion with the growth of the framework. It is important to underline that, differently from the relationship

²²<https://docs.microsoft.com/it-it/powershell/>

²³<https://attack.mitre.org/techniques/T1059/>

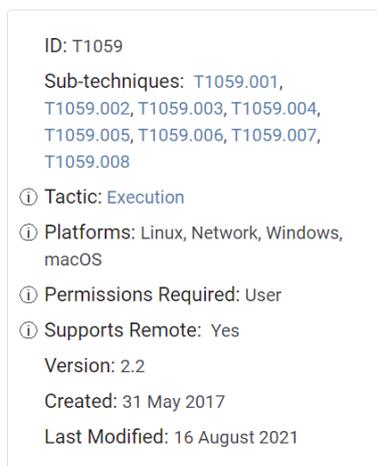


Figure 2.8. Summarizing panel for technique T1059 - Command and Scripting Interpreter²⁴.

tactic-technique, sub-techniques have just a single parent technique.

MITRE ATT&CK literature review

Like for Cyber Threat Intelligence ontologies or sharing languages like STIX, scientific papers and documents are largely available. Among many, some papers have been selected for a brief discussion about. A first contact with ATT&CK has been achieved thanks to the work by Husari et al. [37], which contains concepts that are in the end, the same of this thesis: the idea to put in practice Cyber Threat Intelligence information for attack detection with the aid of MITRE ATT&CK (by the way, also STIX is considered in this work). V. N. Son et al. [73], leaving aside the implementation of their idea, also base their work on MITRE ATT&CK knowledge base, with the aim of detecting abnormal malware behaviour, thanks to the building of some behaviour profiles. The paper by R. Stoleriu et al. [76], is closest to the Threat Hunting concept that will be examined in the second part of the thesis (see Section 3), but is another example of ATT&CK use, in combination with other architectures, aimed at detecting attacks acting proactively.

Herwaman et al. [33] propose a solution which has much in common with the one proposed here. A Threat Hunting platform is developed and above the others, MITRE ATT&CK's basic blocks, the techniques, are used and other instruments like Atomic Red Team library (analyzed in depth in Section 2.4) are involved too for the pen-testing.

Last two documents that deserve to be cited are two official MITRE documents [78][22]. The first one puts the focus on the presentation of a methodology for ATT&CK used to identify sensors, build tests and refine behavioural analytics; this has been of great inspiration when thinking about how to apply ATT&CK techniques into intrusion detection.

²⁴<https://attack.mitre.org/techniques/T1059/>

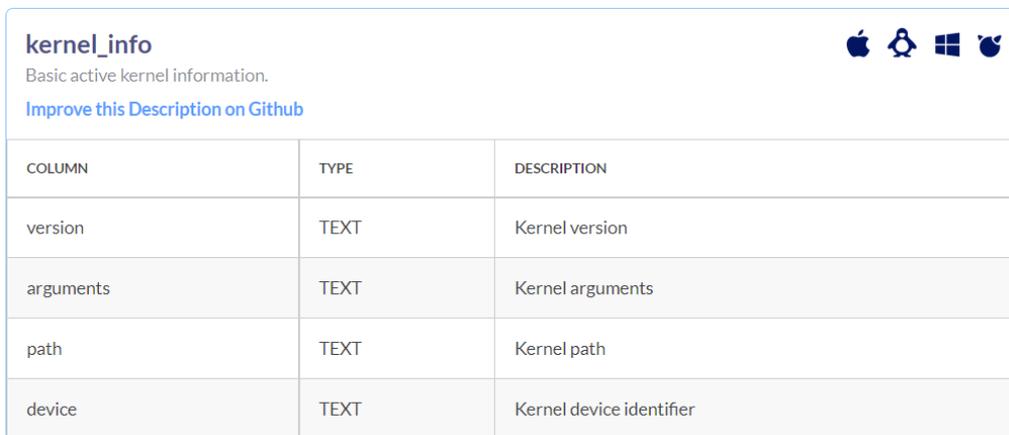
An ATT&CK-based analytics development method is presented and has been useful to develop thesis's analytics in the form of OS queries. The second MITRE paper has been also useful, giving interesting information on topics like anomaly-based detection and hunting.

2.3 Osquery

Osquery is an open source endpoint agent, created in 2014 by Facebook²⁵(today Meta). Osquery exposes the operating system as if it was a relational database: this is useful because, the OS becomes easy to be queried like every other database. Osquery interrogations are normal SQL queries that can be exploited to gain information about the endpoint. Osquery is a fast and tested tool and is available for Windows, MacOS and almost every Linux distribution released since 2011. There are also many different open source projects regarding Osquery on Github.

2.3.1 Tables schema and organization

Osquery structure is quite simple. After installation, a series of tables are created thanks to data offered by the host OS. These tables are just the same of a traditional relational database and are a total of 279 in version 5.1.0, which is the one employed in the thesis. Some tables are in common between all the OSs, some others are specific for Windows or Linux for example. Tables are accessible from *Schema* section on the website for a graphical view and for each field, the column name, the data type and a description are explicated. An example of table, as it's shown on the website, is in figure 2.9.



COLUMN	TYPE	DESCRIPTION
version	TEXT	Kernel version
arguments	TEXT	Kernel arguments
path	TEXT	Kernel path
device	TEXT	Kernel device identifier

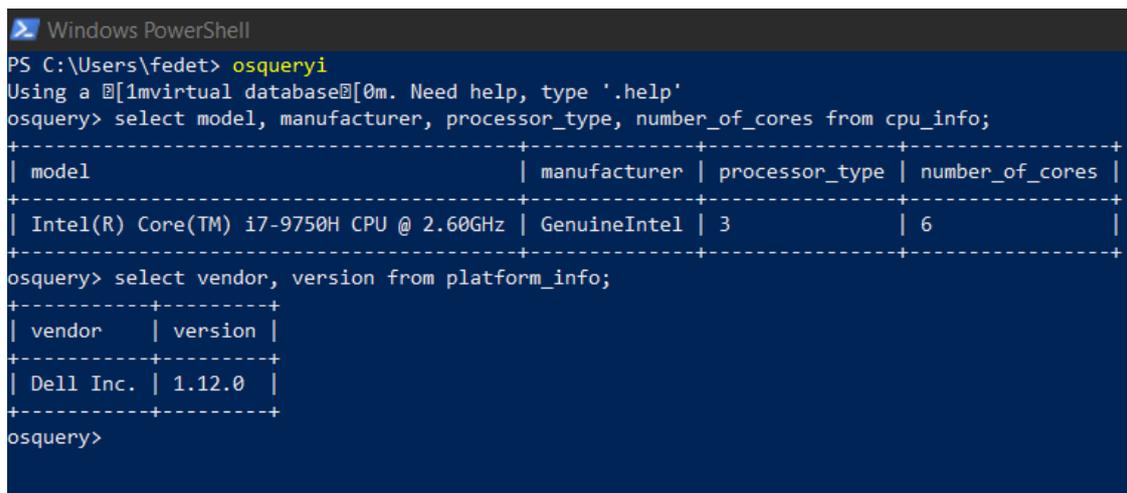
Figure 2.9. Osquery **kernel_info** table²⁶.

²⁵<https://about.facebook.com/meta>

²⁶<https://osquery.io/schema/5.1.0>

2.3.2 Osqueryi and Osqueryd modes

Osquery has two main working modes: an interactive command line mode which is called *osqueryi* and a daemon mode called *osqueryd* which runs in background after being configured. With *osqueryi*, queries are written in a CLI and, the results are shown immediately like in figure 2.10 where two simple queries are shown. A second mode for Osquery is repre-



```

Windows PowerShell
PS C:\Users\fedet> osqueryi
Using a [1]mvirtual database[0]m. Need help, type '.help'
osquery> select model, manufacturer, processor_type, number_of_cores from cpu_info;
+-----+-----+-----+-----+
| model                | manufacturer | processor_type | number_of_cores |
+-----+-----+-----+-----+
| Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz | GenuineIntel | 3              | 6              |
+-----+-----+-----+-----+
osquery> select vendor, version from platform_info;
+-----+-----+
| vendor  | version |
+-----+-----+
| Dell Inc. | 1.12.0 |
+-----+-----+
osquery>

```

Figure 2.10. Example of *osqueryi* usage with Windows Powershell.

sented by the daemon mode which, once configured properly, runs in background collecting logs related to the queries that have been set. *Osqueryd* executable works following instructions that are given by the user in a configuration file: this file is called *osquery.conf* by default, and is put in the *Osquery* installation directory. Some configurations might need the additional setting of a flag file called *osquery.flags*. This file contains eventual flags that are used for many disparate reasons, from the activation of a specific function to the specification of some parameters. *Osqueryd* is the working mode adopted for this thesis. Details about configuration and flags will be provided in the next paragraph.

2.3.3 Osquery configuration file and flags

Osquery configuration file is JSON-based therefore, each component is put inside curly braces. This paragraph will illustrate briefly the main components of the configuration. First part of the configuration file consists in the *options*. Options are not mandatory, but there are quite a lot of them available and detailed on the website documentation. It is possible to configure in *Osquery*, a query *schedule*. Inside it, each query is identified by a specific name, and is composed by two main parts: the SQL query itself and the *interval*, which is the time interval between two periodical executions of that query. Other properties, like a *description*, might be added. More than this, it is possible to set *decorators*, that are special queries used to append some common data, like general information about the system, to each log produced. Last important feature of *Osquery*, are the query *packs*.

Packs are simply other configuration files which contain queries. They are like libraries of queries and, become useful to group them for reuse in different configurations. To use packs, it is sufficient to reference them in the configuration. Default packs already exist and custom ones may be created. A random example is shown in 2.11.

```
{
  // Configure the daemon below:
  "options": {
    // Select the osquery config plugin.
    "config_plugin": "filesystem",

    // Select the osquery logging plugin.
    "logger_plugin": "filesystem",

    // Splay the scheduled interval for queries.
    // This is very helpful to prevent system performance impact when scheduling
    // large numbers of queries that run a smaller or similar intervals.
    //"schedule_splay_percent": "10",

    "utc": "true"
  },

  // Define a schedule of queries:
  "schedule": {
    // This is a simple example query that outputs basic system information.
    "system_info": {
      // The exact query to run.
      "query": "SELECT hostname, cpu_brand, physical_memory FROM system_info;",
      // The interval in seconds to run this query, not an exact interval.
      "interval": 3600
    }
  },

  // Decorators are normal queries that append data to every query.
  "decorators": {
    "load": [
      "SELECT uuid AS host_uuid FROM system_info;",
      "SELECT user AS username FROM logged_in_users ORDER BY time DESC LIMIT 1;"
    ]
  },

  "packs": {
    "osquery-monitoring": "/opt/osquery/share/osquery/packs/osquery-monitoring.conf",
    "incident-response": "/opt/osquery/share/osquery/packs/incident-response.conf",
  },
}
```

Figure 2.11. Osquery random configuration example.

For what concerns the flags, the `osquery.flag` file is empty by default. In this thesis work, not a lot of flags will be used but someone is needed. Complete list of flags can be retrieved easily with `osqueryi`. Flags may be used to enable or disable something specific, but just flags needed for this thesis will be described in the contribution (chapter 5).

2.3.4 Osquery results

Once launched, the daemon works in background with the settings the user has done. Meanwhile, results are progressively produced and stored. More specifically, in the osquery working directory, there is a folder called `log` which, empty by default, starts to be populated when daemon is launched and logs are collected. More precisely, if there are no errors in the configuration and no problems with the OS during query execution, two files are created automatically: `osquery.results.log` that contains the logs and an information file. During the daemon execution, logs are collected in differential mode, so each query is executed when the interval expires and if new records are registered, new logs are added, otherwise no new results are saved. User can also choose to save logs in `snapshot` mode: in this case, at each interval, queries are saved also if already present in the results file. Stored results are saved in a simple RocksDB²⁷ instance on disk. Results are put in JSON format.

2.3.5 Importance in the domain of this work

Osquery has a key role in the implementation proposed in this thesis. It is the probe that collects logs, which are the basic block above which it is possible to identify different attack techniques applied. Osquery is used here in a very specific way to search actions that malicious actors may have performed on the endpoint. Osquery is of course, much more compared to what is described here, the whole documentation [65] is rich and well-organized and details can be found there.

2.4 Atomic Red Team

Atomic Red Team²⁸ is an open source library of security tests that everyone can execute. Atomic Red Team tests are all mapped to the already discussed MITRE ATT&CK framework. The library was first released in 2017 and since then, has evolved increasing the number of tests and, improving their quality constantly. It has been created by Red Canary²⁹, a famous American security company.

2.4.1 Library organization

Atomic Red Team [5] is available freely on Github. The repository is full of material but, what is interesting for this thesis, is the `atomics` folder. Inside this folder, many other folders are contained: each one is related to a specific ATT&CK technique that is mapped. It is important to underline that still not all the techniques are mapped in the library but, the number is quite large. Moreover, the Repository contains a useful Wiki section with the documentation explained.

²⁷<http://rocksdb.org/>

²⁸<https://atomicredteam.io/>

²⁹<https://redcanary.com/atomic-red-team/>

2.4.2 Atomic test details

As said in the previous paragraph, each folder is coherently named with the ID of the technique or sub-technique to which it refers and always contains two files:

- A **Markdown (.md)** file: is a typical Github markdown file, that contains the ATT&CK description of the technique, an initial list of all the tests available for that technique (the number is variable), and a detailed human-readable description of each test (example in figure 2.12);
- A **YAML (.yaml)** file: contains the same information of the previous (description, attack commands, cleanup commands and so on) but in YAML³⁰ format, so less easily readable by users, but perfectly machine-readable (example in figure 2.13).

Atomic Test #1 - Logon Scripts

Adds a registry value to run batch script created in the %temp% directory. Upon execution, there will be a new environment variable in the HKCU\Environment key that can be viewed in the Registry Editor.

Supported Platforms: Windows

auto_generated_guid: d6042746-07d4-4c92-9ad8-e644c114a231

Inputs:

Name	Description	Type	Default Value
script_path	Path to .bat file	String	%temp%\art.bat
script_command	Command To Execute	String	echo Art "Logon Script" atomic test was successful. >> %USERPROFILE%\desktop\T1037.001-log.txt

Attack Commands: Run with `command_prompt!`

```
echo "#{script_command}" > #{script_path}
REG.exe ADD HKCU\Environment /v UserInitMprLogonScript /t REG_SZ /d "#{script_path}" /f
```

Cleanup Commands:

```
REG.exe DELETE HKCU\Environment /v UserInitMprLogonScript /f >nul 2>&1
del #{script_path} >nul 2>&1
del "%USERPROFILE%\desktop\T1037.001-log.txt" >nul 2>&1
```

Figure 2.12. T1037.001 - Test #1 - Markdown file³¹.

Some folders then, may contain additional scripts or executables. Each test taken individually, is composed of a series of commands to be executed on specific terminals: in

³⁰<https://yaml.org/>

³¹<https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1037.001/T1037.001.md>

```

attack_technique: T1037.001
display_name: "Boot or Logon Initialization Scripts: Logon Script (Windows)"
atomic_tests:
- name: Logon Scripts
  auto_generated_guid: d6042746-07d4-4c92-9ad8-e644c114a231
  description: |
    Adds a registry value to run batch script created in the %temp% directory. Upon execution, there will be a new environment variable in the HKCU\Environment key
    that can be viewed in the Registry Editor.
  supported_platforms:
  - windows
  input_arguments:
    script_path:
      description: Path to .bat file
      type: String
      default: '%temp%\art.bat'
    script_command:
      description: Command To Execute
      type: String
      default: echo Art "Logon Script" atomic test was successful. >> %USERPROFILE%\desktop\T1037.001-log.txt
  executor:
    command: |
      echo "#{script_command}" > #{script_path}
      REG.exe ADD HKCU\Environment /v UserInitMprLogonScript /t REG_SZ /d "#{script_path}" /f
    cleanup_command: |
      REG.exe DELETE HKCU\Environment /v UserInitMprLogonScript /f >nul 2>&1
      del #{script_path} >nul 2>&1
      del "%USERPROFILE%\desktop\T1037.001-log.txt" >nul 2>&1
  name: command_prompt

```

Figure 2.13. T1037.001 - Test #1 - YAML file³².

Windows, some tests are engineered for Command Prompt, others for Powershell. Focus here has been put on Windows more than Linux. Testing techniques may leave the endpoint in an undesired state: files can be created or settings modified as example, and this is quite normal because, tests are trying to simulate a malicious behaviour. To solve this issue, when necessary, atomic tests are completed with some cleanup commands. After execution, these commands reset the changes just made and put the system in its previous state.

When executing tests, it is important to be aware that potentially malicious payloads are executed in some cases; users must have the permissions to perform those actions on the desired device. This said, the best way to execute tests is not the manual one but, involves a specific Powershell module called *Invoke-Atomic* that will be described in the next paragraph.

2.4.3 Invoke-Atomic module

Also more completely named *Invoke-atomicredteam*, this module is contained in a dedicated repository together with installation and usage instructions. Invoke-Atomic is a utility that works with Powershell making it easily usable with Windows Systems. It permits to execute tests just indicating the desired technique and the test number or name, without writing commands manually on the terminal. Also cleanup commands are available for automatic execution. In practice, Invoke-Atomic eases the job of text execution avoiding errors too. The typical command of this module starts with `Invoke-AtomicTest TXXXX`

³²<https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1037.001/T1037.001.yaml>

where TXXXX is the ID of the technique. It becomes TXXXX.YYY if a sub-technique is taken under consideration. The command may be completed with some CLI parameters to specify which test name or number the user wants to execute, details about the test may be showed in more or less detailed versions, prerequisites can be checked and eventually got when not already satisfied. Also the aforementioned cleanup commands are executed thanks to a command line flag. Figure 2.14 below, shows execution of some of the commands outlined. There is also a GUI version of Invoke-Atomic, but it has been not used in this case. Whole information is available on the Github page [6].

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tutti i diritti sono riservati.

PS C:\Windows\system32> Import-Module "C:\AtomicRedTeam\invoke-atomicredteam\Invoke-AtomicRedTeam.psd1" -Force
PS C:\Windows\system32> Invoke-AtomicTest T1053.005 -TestNumbers 1 -ShowDetailsBrief
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

T1053.005-1 Scheduled Task Startup Script
PS C:\Windows\system32> Invoke-AtomicTest T1053.005 -TestNumbers 1 -CheckPrereqs
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

CheckPrereq's for: T1053.005-1 Scheduled Task Startup Script
Prerequisites met: T1053.005-1 Scheduled Task Startup Script
PS C:\Windows\system32> Invoke-AtomicTest T1053.005 -TestNumbers 1
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1053.005-1 Scheduled Task Startup Script
Done executing test: T1053.005-1 Scheduled Task Startup Script
OPERAZIONE RIUSCITA: l'attivit... pianificata "T1053_005_OnLogon" § stata creata.
OPERAZIONE RIUSCITA: l'attivit... pianificata "T1053_005_OnStartup" § stata creata.
PS C:\Windows\system32> Invoke-AtomicTest T1053.005 -TestNumbers 1 -Cleanup
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing cleanup for test: T1053.005-1 Scheduled Task Startup Script
Done executing cleanup for test: T1053.005-1 Scheduled Task Startup Script
PS C:\Windows\system32>
```

Figure 2.14. Some commands executed on test 1 of sub-technique T1053.005.

2.4.4 Reasons for the choice

Atomic Red Team has been selected for the purpose of this thesis because it is easy to install and use, and gives the chance to perform targeted attacks without making mess in the system. It is widely adopted and appreciated in the community and, it was considered a perfect tool when the idea to work singularly on specific ATT&CK techniques came out. This because it permits to focus the testing on one technique or sub-technique each time, avoiding to create confusion.

2.5 Red Canary - 2021 Threat Detection Report

Red Canary is a Security Company founded back in year 2013, by some experts in offensive security and intelligence that at that time were already supporting the community with research. Since then, the company has grown up, offering an increasing number of security

services, developing tools like Atomic Red Team and hiring people until they have now more than 400 employees [83]. Since 2019, the company is publishing each year a threat detection report, with many statistics and information about major threats regarding the year before. The interesting aspect, is that the report is written taking as basis the MITRE ATT&CK framework and, this makes it another perfect brick in this work. In March 2021, the *2021 - Threat Detection Report*³³ has been published and it is at this moment, the most recent one.

2.5.1 Report organization

2021 - Threat Detection Report is 122 pages long and is freely downloadable from the website. The report presents three main sections: the top 10 ATT&CK techniques/sub-techniques associated with confirmed threats across company's customers, the top 10 threats in the sense of APT groups, software solutions, most widespread trojans and last, a description of less widespread threats that deserve attention in today's landscape. Methodology about how report has been written is explained inside it. The top 10 techniques section is the one that is interesting here. The rest of the report has not been used in this work but it may be useful for other purposes. The 10 techniques in the ranking may be considered in the form of the main technique or sometimes, a specific sub-technique can be discussed. This happens when that sub-technique is the one primarily contributing to diffusion above the others. Brief descriptions of the 10 techniques selected by the report, directly taken from MITRE ATT&CK³⁴, are presented below. When the technique considered in the report is the main technique, its description is provided, when a particular sub-technique is taken into account, the description is provided for that sub-technique avoiding to consider the general case. This is done because in the end, if sub-techniques are considered, the main technique is not analyzed by the report that instead, focuses just on the sub-techniques.

1. T1059 - Command and Scripting Interpreter

- **T1059.001 - PowerShell:** Adversaries may abuse PowerShell commands and scripts for execution. PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system. Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the `Start-Process` cmdlet which can be used to run an executable and the `Invoke-Command` cmdlet which runs a command locally or on a remote computer (though administrator permissions are required to use PowerShell to connect to remote systems).
- **T1059.003 - Windows Command Shell:** Adversaries may abuse the Windows command shell for execution. The Windows command shell (`cmd`) is the primary command prompt on Windows systems. The Windows command prompt can be used to control almost any aspect of a system, with various permission levels

³³<https://redcanary.com/threat-detection-report/>

³⁴<https://attack.mitre.org/techniques/enterprise/>

required for different subsets of commands. The command prompt can be invoked remotely via Remote Services such as SSH.

2. T1218 - Signed Binary Proxy Execution

- **T1218.011 - Rundll32:** Adversaries may abuse rundll32.exe to proxy execution of malicious code. Using rundll32.exe, vice executing directly (i.e. Shared Modules), may avoid triggering security tools that may not monitor execution of the rundll32.exe process because of allowlists or false positives from normal operations. Rundll32.exe is commonly associated with executing DLL payloads (ex: rundll32.exe DLLname, DLLfunction).
- **T1218.005 - Mshta:** Adversaries may abuse mshta.exe to proxy execution of malicious .hta files and Javascript or VBScript through a trusted Windows utility. There are several examples of different types of threats leveraging mshta.exe during initial compromise and for execution of code.

3. T1543 - Create or Modify System Process

- **T1543.003 - Windows Service:** Adversaries may create or modify Windows services to repeatedly execute malicious payloads as part of persistence. When Windows boots up, it starts programs or applications called services that perform background system functions. Windows service configuration information, including the file path to the service's executable or recovery programs/commands, is stored in the Windows Registry. Service configurations can be modified using utilities such as sc.exe and Reg.

4. T1053 - Scheduled Task/Job

- **T1053.005 - Scheduled Task:** Adversaries may abuse the Windows Task Scheduler to perform task scheduling for initial or recurring execution of malicious code. There are multiple ways to access the Task Scheduler in Windows. The `schtasks` can be run directly on the command line, or the Task Scheduler can be opened through the GUI within the Administrator Tools section of the Control Panel. In some cases, adversaries have used a .NET wrapper for the Windows Task Scheduler, and alternatively, adversaries have used the Windows netapi32 library to create a scheduled task.

5. T1003 - OS Credential Dumping

- **T1003.001 - LSASS Memory:** Adversaries may attempt to access credential material stored in the process memory of the Local Security Authority Subsystem Service (LSASS). After a user logs on, the system generates and stores a variety of credential materials in LSASS process memory. These credential materials can be harvested by an administrative user or SYSTEM and used to conduct Lateral Movement using Use Alternate Authentication Material.

6. T1055 - Process Injection: Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process.

7. **T1027 - Obfuscated Files or Information:** Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit. This is common behaviour that can be used across different platforms and the network to evade defenses.
8. **T1105 - Ingress Tool Transfer:** Adversaries may transfer tools or other files from an external system into a compromised environment. Files may be copied from an external adversary controlled system through the command and control channel to bring tools into the victim network or through alternate protocols with another tool such as FTP. Files can also be copied over on Mac and Linux with native tools like scp, rsync, and sftp.
9. **T1569 - System Services**
 - **T1569.002 - Service Execution:** Adversaries may abuse the Windows service control manager to execute malicious commands or payloads. The Windows service control manager (**services.exe**) is an interface to manage and manipulate services. The service control manager is accessible to users via GUI components as well as system utilities such as **sc.exe** and **Net**.
10. **T1036 - Masquerading**
 - **T1036.003 - Rename System Utilities:** Adversaries may rename legitimate system utilities to try to evade security mechanisms concerning the usage of those utilities. Security monitoring and control mechanisms may be in place for system utilities adversaries are capable of abusing. It may be possible to bypass those security mechanisms by renaming the utility prior to utilization (ex: rename **rundll32.exe**). An alternative case occurs when a legitimate utility is copied or moved to a different directory and renamed to avoid detections based on system utilities executing from non-standard paths.

2.5.2 Reasons for the choice

During the progress of the work, it has been clear at a certain point, the necessity to choose a sample of techniques to experiment on. MITRE ATT&CK v10 in fact, is made up of 188 techniques and 379 sub-techniques and, put in practice the job on such a big number of them, would have probably been too long and difficult. This was enforced by the fact that not a lot of previous inherent works were available to help. This report seemed to be recent, complete and relevant. Moreover, the fact that it was published by the same company of Atomic Red Team, was convincing. In the end, the report has been used both to define the subset of techniques to consider and, to take information for the writing of SQL queries with Osquery.

Chapter 3

Background - Threat Hunting

3.1 Threat Hunting Overview

Threat Hunting is the activity of searching for something that may cause trouble, harm or in general leading to some kind of damage¹. Cybernetic threats are continuously evolving, changing and bringing new ways to break computer systems in order to perform frauds and create damages to any kind of organization or user.

It is impossible to create a computer system invulnerable from a security point of view. Updates, patch and new versions of a software may introduce some malfunctioning that can be exploited by a malicious user to perform an attack. The process of releasing patches by defender and attempts to create new exploits by attackers generates an infinite loop where a team tries to secure systems from the fraudulent intent of the other one. Many organizations joined their efforts to build open source knowledge of the most common threats and attacks used by attackers, but this is not enough. Indeed Threat Hunting not only focuses on identifying existing attack patterns, but also on detecting unknown ones.

In this section is provided an overview of Threat Hunting. Firstly is given a brief introduction about the subject, what it does and its goals. Then, are introduced the main tools and procedures used to perform Threat Hunting and a focus on one of the major global effort to build an open knowledge base that helps cybersecurity expert in defending their systems against attackers.

3.1.1 Introduction and definitions

To start, is provided a more technical definition of threat:

- NIST² 800-53 [28]: “Any circumstance or event with the potential to adversely impact organizational operations, organizational assets, individuals, other organizations, or the Nation through a system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.”

¹<https://www.merriam-webster.com/thesaurus/threat>

²<https://www.nist.gov/>

- RFC-4949 [72]: “Any circumstance or event with the potential to adversely affect a system through unauthorized access, destruction, disclosure, or modification of data, or denial of service.”

Basically a threat is a general term to include a very huge space of potentially dangerous actions, that very often results in an attack exploiting one or more Vulnerabilities, which are flaws or weaknesses in a system that may lead to a security breach.

It is theoretical impossible to create a system without any kind of vulnerability. Notice that not every vulnerability leads to a successful attack, but there are many factors to consider when discussing the security of a software: if the software is maintained, the technology evolution, new attacks discovered and so on. There is need of an approach that extends traditional ones, such Firewalls, IDS and IPS.

Cyber Threat Hunting is the set of actions that **proactively** search for advanced threats and malicious behaviours inside of systems, organizations, endpoints and networks. The goal is to discover a potential threat in early stage, avoiding that more complex and effective attacks are put in place leading to a huge damage from different points of view.

Types of Cyber Threat Hunting

When some unexpected event occurs, the Threat Hunting system raises a trigger. Starting from there is possible to perform further analysis in order to verify the possibility of a possible incident. Those in-depth analysis are distinguished in three types [93]:

1. **Structured Hunting:** it is based on Indicator of Attacks (IoA) and TTPs (see Section 2.2). IoA and TTPs are operations performed by malicious users in order to success in an attack. Those indicates typical actions, specific pattern and operations that must be followed in order to obtain a given result. By following them, it is possible for an analyst to identify dangerous behaviour and block the attack before it gains access to confidential data or causes damages.
A structured hunting strategy follows the MITRE ATT&CK framework, which is described in Section 2.2.
2. **Unstructured Hunting:** also this type of hunt start with triggers and alerts, which are named Indicator of Compromise (IoC). The analyst searches the network for malicious pattern before and after the trigger. This means that even past data can be analysed, as much as is needed. By looking backward, this analysis may cause the discovery of new threats that may represent a risk or that have already penetrated the perimeter and are waiting for instruction from the CnC or some event to become active.
3. **Situational or Entity-Driven Hunting:** it is focused on analysing most sensible data and critical resources. Following this approach, threat hunting activities are prioritized to focus on critical assets, that most of the time are the attackers' target. Critical assets are IT administrators, Domain Controllers, accounts with high privileges, Databases and so on.

Joining efforts of Cyber Threat Intelligence and Threat Hunting is possible to create a custom or situational environment that focuses on most critical assets, also based on customer capabilities and requirements.

3.1.2 APT and Most Popular Threats

In this section will be analysed two of the main threats that grown in popularity over last few years. Both types are considered *advanced attacks* because of the ability of breaching traditional security systems, such Firewalls and Antivirus.

APT - Advanced Persistent Threats

As stated in the previous section, Threat Hunting focuses mostly on advanced threats which are difficult to detect and defence against. In particular, there is a specific category of malicious users that is really dangerous especially for large organizations and public administrations; this group is named **APT – Advanced Persistent Threat**. Accordingly to the NIST definition, [28], APT are adversaries that have sophisticated levels of experience and significant resources which allow to achieve its objectives using multiple attacks vectors, including cyber, physical and deception. Their target are frequently IT infrastructures of big companies and public administrations, attempting to corrupt critical aspects of a program, a mission or a service.

APTs are hacking groups typically referring or even sponsored by National Governments. One of their main characteristics is the ability of address attacks to be long-term, by intruding into target system, or into a system associated to the target one, and establishing a presence that remains stealth even for months. During this period, adversaries try to expand their influence by deep analysis of IT infrastructures and by moving laterally and vertically in preparation of the real attack.

To try to detect APT attacks, many advanced analysis and monitoring tools are available, such SIEM and EDR, which will be analysed later on. Another key factor is the need of properly formed personal, which drives those tools, but also employees able to avoid potentially dangerous behaviour that may lead to data breaches. Many companies and administrations struggle in achieving this last point.

A typical APT attack follows some steps [1]:

1. Attackers gain entry to the system through an email, the network, a file, a malware or some vulnerability. Here, the network is considered compromised but not breached.
2. The malware tries to gain additional access to the network, exploits vulnerabilities or communicate with the Command and Control (CnC) server. This is a dedicated attacker's machine which is able to send updates, code and instructions to the malware in order to successfully spread the infection.
3. Lateral movement and infection spreading, to ensure that if one node is blocked the attack continues.
4. Once a reliable network access has been established, the malware starts to collect and gather data about network infrastructure, endpoints, servers, user information, account names and passwords. The goal is to gain access to confidential data and understand which machine and services may become the target of a future attack.
5. The malware continues in collecting data, saving them in a staging server. Results are exfiltrated and sent off the network, to a server under the attacker control. Now the network is considered breached.

6. Evidence of the APT attack is removed, but the network remains compromised. The enemy may use the gained entry point to continue analysis and perform actions.

How are APTs executed?

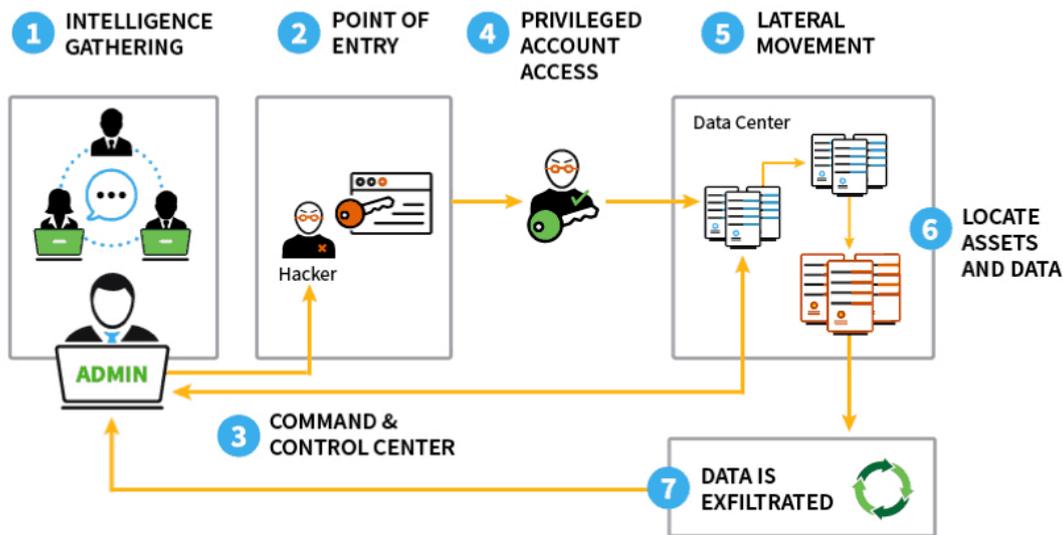


Figure 3.1. Main APT attack steps. [32]

Traditional cybersecurity measures are not enough to protect against APT. Firewall, Packets filter and in some cases even IDS & IPS are not successful and leave the network vulnerable.

Since APT target mostly organization and public services, users should not be involved in such attacks. Unfortunately they may suffer from undirected consequences; for example, a compromised web site of a company may cause the user to download or execute some scripts, that leads in blocking the user computer or downloading spyware, ransomware and other malware. This phenomenon is known as **collateral risks**.

Ransomware

Ransoms are another type of malware that grown in popularity over last few years. The goal of these attacks is to encrypt files on a device, making them inaccessible and unavailable. To achieve that, a Ransomware may take some time to analyse for the infected node and scan file system, hard drives and networks to figure out which files to encrypt.

As any other type of attacks, even Ransomware follows TTPs. In particular, they scan network and devices to find most critical node, such a Domain Controller, performing lateral movement and privileges escalation in order to spread the infection.

The encryption may affect the entire file system, some critical directory only and backup devices attached to the infected node. Usually it is performed with a double encryption

key option: one different for every tampered host or with a default key. If the Ransomware succeed in communicating with the CnC server, then the unique key is used to encrypt the file system leading to a situation where every infected node is encrypted with different secrets. If the outbound communication fails, the encryption proceeds using a default key.

Once the attack is performed, the user is notified and usually a message asking for a ransom is popped. If the fee is not paid, the threat is to publish sensible information or even deleting the whole file system. Being able to be protected against this kind of threats is crucial for an organization, for obvious reasons. Notice not all Ransomware attacks succeed; indeed an organization with strong policies and a proper incident detection and response strategy is able to detect the threat before the network is compromised or succeed in blocking the connection from the infected node to the CnC server. This last scenario is fundamental: blocking the connection between the infected node and the attacker’s server is crucial. Even if several nodes are affected and blocked by the malware, if the default key has been used it means that all machines were encrypted with the same secret. Having one, single, secret can mean:

- Higher probability of breaking the encryption, if the key is simple.
- Purchasing just one key to unlock all the infected machines.

According to the Cybersecurity and Infrastructure Security Agency (CISA) ³ [55], it is possible to identify some common infection vector:

- **Internet-facing Vulnerabilities and misconfigurations:** conduct regular scanning analysis, patch and update software, ensure proper devices and services configuration, use Multi-Factor Authentication, perform strong monitoring, audit and access control to crucial network machines.
- **Phishing:** make sure employees are properly trained and able to recognize suspicious email messages, implement gateway filters based on most common phishing techniques, disabling macro scripts for Microsoft Office files sent by email.
- **Precursor Malware Infection:** ensure antivirus and anti-malware are up to date, possibly using centralized management solutions. Blocking the execution of programs by processes that does not have the rights, implementing IDS and IPS.
- **Third parties and Managed Service Providers:** do not assume as always secure third party Service Providers or services; they may be vulnerable to infections and become vectors as well

CISA gives also some guidelines and best practice to enforce Ransomware protection:

- It is critical to maintain offline and encrypted backups, audit and validate them in regular time intervals. Maintain “golden images” of most important services that can be used as recovery method to avoid huge losses.

³<https://www.cisa.gov>

- Adopt a proper incident response and analysis strategy, including disaster recovery policies.
- Enforce Multi-Factor Authentication to all accounts and assets.
- Apply the “Least Privilege Principle”, giving the minimum rights to users that they require to perform needed tasks. Restrict the usage of PowerShell or other high privilege processes to only people that really have rights.
- Ensure strong access monitoring, audit and privilege access management to all sensible resources, adopting proper Remote Desktop Protocols and policies management.
- Secure most sensible network nodes, such as Firewalls, Domain Controllers, DNS and Databases.

In the 3.2 is shown the typical steps a malicious user has to perform when executing a Ransomware attack. All those steps follows MITRE ATT&CK TTPs and are aimed to achieve Privilege Escalation and data exfiltration.

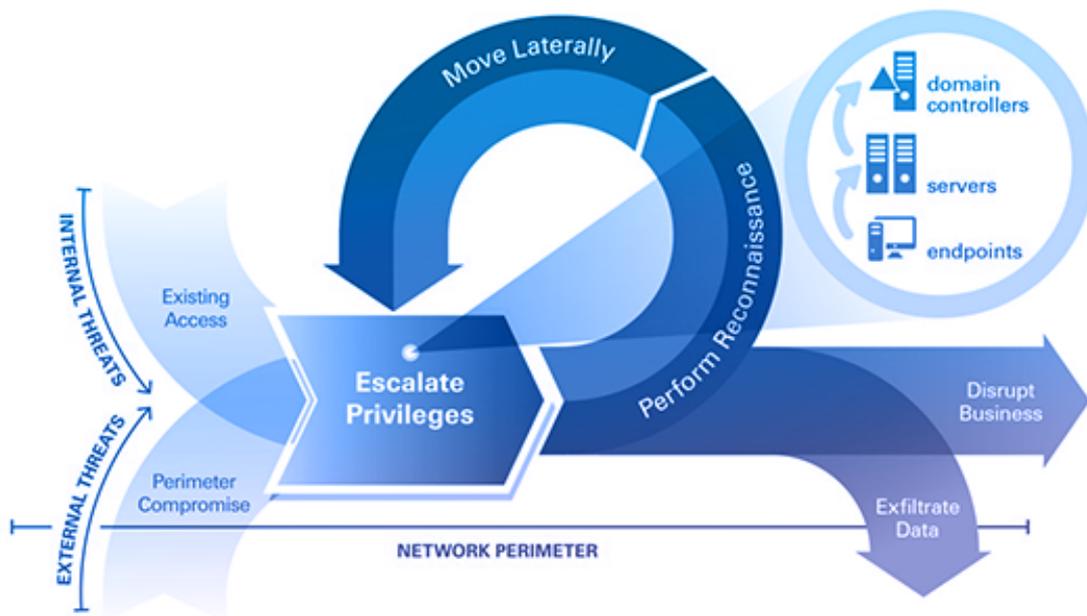


Figure 3.2. Ransomware lifecycle illustrated by CyberArk, world leader company in Privileges Access Management. [13]

3.2 The Unified Kill Chain

To being able to recognize in an effective way possible attack pattern, it is important to mention “The Unified Kill Chain” [69]. This project was created aiming at improve resilience against advanced cyber-attacks.

As stated in the previous section, APT must follow some steps in order to success in an attack. Indeed, the document ensure that: “The UKC give insights into the arrangement of phases in attacks from their beginning to their completion, by uniting and extending existing models.”

As mentioned, there are other Cyber Kill Chain (CKC) models which focus on target malware rather than building a general model. The paper makes another important point: traditional detection chains does not assume the possibility an attacker may bypass some phases. The assumption may cause the detection chain to fail in building an attack path, hence leading to a higher success probability for an adversary. In this field, the UKC can be used to prevent and detect attacks in early stages, then allows to build an effective response strategy.

Generally speaking, the attacks need to be understood and split in discrete phases in order to rationalize what steps an attacker may perform and which are its goals. UKC can be used to build defence mechanisms focused on contrasting certain phases, prioritize detection and to build model of past/potential attacks.

The UKC was created by analysing and joining some of the major models, in particular CKC and MITRE ATT&CK framework; 18 tactics were obtained which can be used to describe almost any kind of modern attack. From 3.3 is possible to distinguish three main

The Unified Kill Chain		
1	Reconnaissance	<i>Researching, identifying and selecting targets using active or passive reconnaissance.</i>
2	Weaponization	<i>Preparatory activities aimed at setting up the infrastructure required for the attack.</i>
3	Delivery	<i>Techniques resulting in the transmission of a weaponized object to the targeted environment.</i>
4	Social Engineering	<i>Techniques aimed at the manipulation of people to perform unsafe actions.</i>
5	Exploitation	<i>Techniques to exploit vulnerabilities in systems that may, amongst others, result in code execution.</i>
6	Persistence	<i>Any access, action or change to a system that gives an attacker persistent presence on the system.</i>
7	Defense Evasion	<i>Techniques an attacker may specifically use for evading detection or avoiding other defenses.</i>
8	Command & Control	<i>Techniques that allow attackers to communicate with controlled systems within a target network.</i>
9	Pivoting	<i>Tunneling traffic through a controlled system to other systems that are not directly accessible.</i>
10	Discovery	<i>Techniques that allow an attacker to gain knowledge about a system and its network environment.</i>
11	Privilege Escalation	<i>The result of techniques that provide an attacker with higher permissions on a system or network.</i>
12	Execution	<i>Techniques that result in execution of attacker-controlled code on a local or remote system.</i>
13	Credential Access	<i>Techniques resulting in the access of, or control over, system, service or domain credentials.</i>
14	Lateral Movement	<i>Techniques that enable an adversary to horizontally access and control other remote systems.</i>
15	Collection	<i>Techniques used to identify and gather data from a target network prior to exfiltration.</i>
16	Exfiltration	<i>Techniques that result or aid in an attacker removing data from a target network.</i>
17	Impact	<i>Techniques aimed at manipulating, interrupting or destroying the target system or data.</i>
18	Objectives	<i>Socio-technical objectives of an attack that are intended to achieve a strategic goal.</i>

Figure 3.3. The Unified Kill Chain 18 tactics. [69]

phases:

- **Initial foothold:** the green phases; the attacker tries to compromise the network by gaining an initial access. If the enemy fails in one phase, the attack may fail as

well; the attacker may change tactic, approach or methodology to recover. On the other hand, if the attacker gains access to the system and notices some vulnerability or misconfiguration, it may decide to jump to some advanced phase in order to speed up the attack.

- **Network propagation:** the orange phases; they cover all operations that an attacker may perform in order to spread the infection. The final goal is to study the system network and configuration, moving laterally and escalate privileges until highly privileged accounts or critical network assets are breached. Here is where enemies face Privileges Access Management systems, which may help in detecting and stop the attack.
- **Action on Objectives:** the red phases; here the attacker has gained access to some resource. It starts the procedure of sending out sensible information to a server outside the system. In these phases, the actual attack is in place.

Thanks to this categorization, companies can leverage the Unified Kill Chain to focus their defence strategies on more frequent or crucial phases, where the attack may likely succeed. Another important point is the ability of defending the whole network; for a large network, indeed, is not suitable to protect any resource. Therefore it may be necessary to build a strategy aiming to defend critical assets, such Domain Controllers and accounts with high privileges, which are targets of attacks.

Since the UKC was published, the MITRE ATT&CK has introduced some tactics firstly identified in the Paul Pols' document, such the *Impact* tactic. This is an insight of the important contribution given by Paul Pols's job in all stages of Cyber defence.

The Unified Kill Chain is published under GNU General Public License v2, which is recognized as an open source license. This allows the framework to be used, studied and extended in total freedom respecting open source foundation guidelines.

In the next section, we will analyse some of the most popular tools for the detection of modern threats, both APT and Ransomware, from Enterprise and open source point of view.

3.3 SOC, SOAR and SIEM tools

In Section 2.1.4 has been mentioned those terms; they are referred to all systems, tools and personnel that joined together create all the security measures and mechanisms that a company needs to protect against cyber-threats. In this section are analysed in details these tools.

Security Operation Center - SOC

From [91], **SOC** is a function that collects tools, processes, employees and resources to perform monitoring, analysis, detection and response to cyber-attacks. Working in the SOC there is a team, led by a SOC manager and including SOC analyst and threat hunters. The SOC team reports to the Chief Information Officer CISO that reports directly to the CEO or CIO; by involving the most important rank in a company, it is possible to understand how much important SOC activities are.

Here are listed the main SOC activities:

- **Collect available resource**, in order to have a knowledge of the system, security tools available and which assets to protect.
- **Proactive monitoring**, performed at every time and every hour, to be able to detect threats and react in minimum time; tools such SIEM and EDR are used to enhance this activity.
- **Alert Ranking and Management**, monitoring highlights incoming from tools, allowing SOC analyst to discard false positive alerts and reacting to true positive, which are signals of a threats.
- **Threat response**, performing some first reaction such shutting down a system or disconnecting it from the network.
- **Recovery and Remediation**, allowing the SOC team to investigate about the incident and restarting the system from a backup or by solving the issue.
- **Log Management**, the task to collect, maintain and review logs collected all across the network. Those logs will be used to perform analysis, inspection and detection.
- **Investigation and internal Auditing**, by inspecting collected logs, both from incidents and normal situations. SOC team should ensure that the system is compliant to security requirements, performing improvements when needed.

Security Information and Event Management – SIEM

A **SIEM** is a tool that gathers many other tools and resources across the network. It collects, aggregates, filters logs and displays relevant data in dashboards or report in a human readable way. SIEM must be properly configured in order to recognize malicious path, leveraging AI and Machine Learning capabilities and applying rules to detect TTPs.

Security Orchestration, Automation and Response – SOAR

A **SOAR** is a collection of tools intended to speed up and improve the detection. SIEMs require a huge manual effort, being thought for alerting and visualization; SOAR, instead, are a collection of automatic tools and procedures to gather, analyse, filter and create evidences of raised alerts. They may be employed in post incident response analysis and incident investigation.

Many SIEMs solution try to aggregate into a single framework both SIEM and SOAR capabilities in order to have a more efficient and unique environment.

Endpoint Detection and Response – EDR

EDR are agents that run on endpoint machine. Their goal is to continuously query the operating system, collecting logs and send them to the centralized security solution, hence SIEM-SOAR solutions. Being installed on the machine, EDR are one of the first obstacle that an attacker must face. They are considered as an evolution of Antivirus, which often

are simpler and work on malware signatures. EDRs can also be an aggregation of software that searches for suspicious behaviours and attack patterns.

As mentioned, collected logs can be sent to a SIEM solution. In order to avoid an ingestion of data, which may lead to false positive alerts, advanced EDRs may include some first analysis mechanism. These operations are helpful to search and perform filtering of logs, optimizing the overall detection pipeline.

Almost any vendor of SIEM solution offers also an EDR agent. The problem is that such systems and architecture may become really expensive and not always a transparent documentation is provided. This issue will be treated in a next section.

3.3.1 Most relevant proprietary tools

EDR plays a central role in the defence against cyber advanced threats in these days and will do the same in the future as well. As mentioned, they are defence mechanism more advance with respect to firewall, but what are the actual capabilities?

A study conducted by George Karantzas et al., [41], chose eleven of the major enterprise EDR solutions selected from Gartner Magic Quadrant ⁴, with the goal to provide a significant assessment about the efficacy of real-world tools.

The study simulates the behaviour of an APT and tries to understand which are the weaknesses of selected EDRs, what info they report and their meaning. To perform relevant tests, it was used the famous CnC framework, CobaltStrike, leveraging and expired domain and issuing a certificate from *Let's Encrypt*, a Certification Authority that allows to issue and renew X.509 certificates automatically and free of charge.

In order to simulate a real scenario, selected *Attack Vectors* was a spear phishing email containing a malicious attachment. The study case [41] takes into consideration four attack methods:

1. ***.cpl file:** a DDL file which can execute malicious code in its context once started. A shellcode storage is used, with Memory-mapped files storage and then is triggered with delegates;
2. **Legitimate Microsoft Teams installation that loads a malicious DDL:** a DDL side-loading leads to a self-injection allowing the malware to live under a signed binary;
3. **Unsigned PE file:** an executable that performs process injection using *Early Bird* technique which executes malicious code in a very early stage of threat initialization, before many control mechanisms start their checks. The malware is protected against EDR spoofing the parent of “explore.exe” process.
4. **HTA file:** a harmless HTML page that redirects to another HTML page tampered with executable code that loads .NET code, that performs self-injection under the context of “mshta.exe” process.

⁴<https://www.gartner.com/en/documents/4001307/magic-quadrant-for-endpoint-protection-platforms>

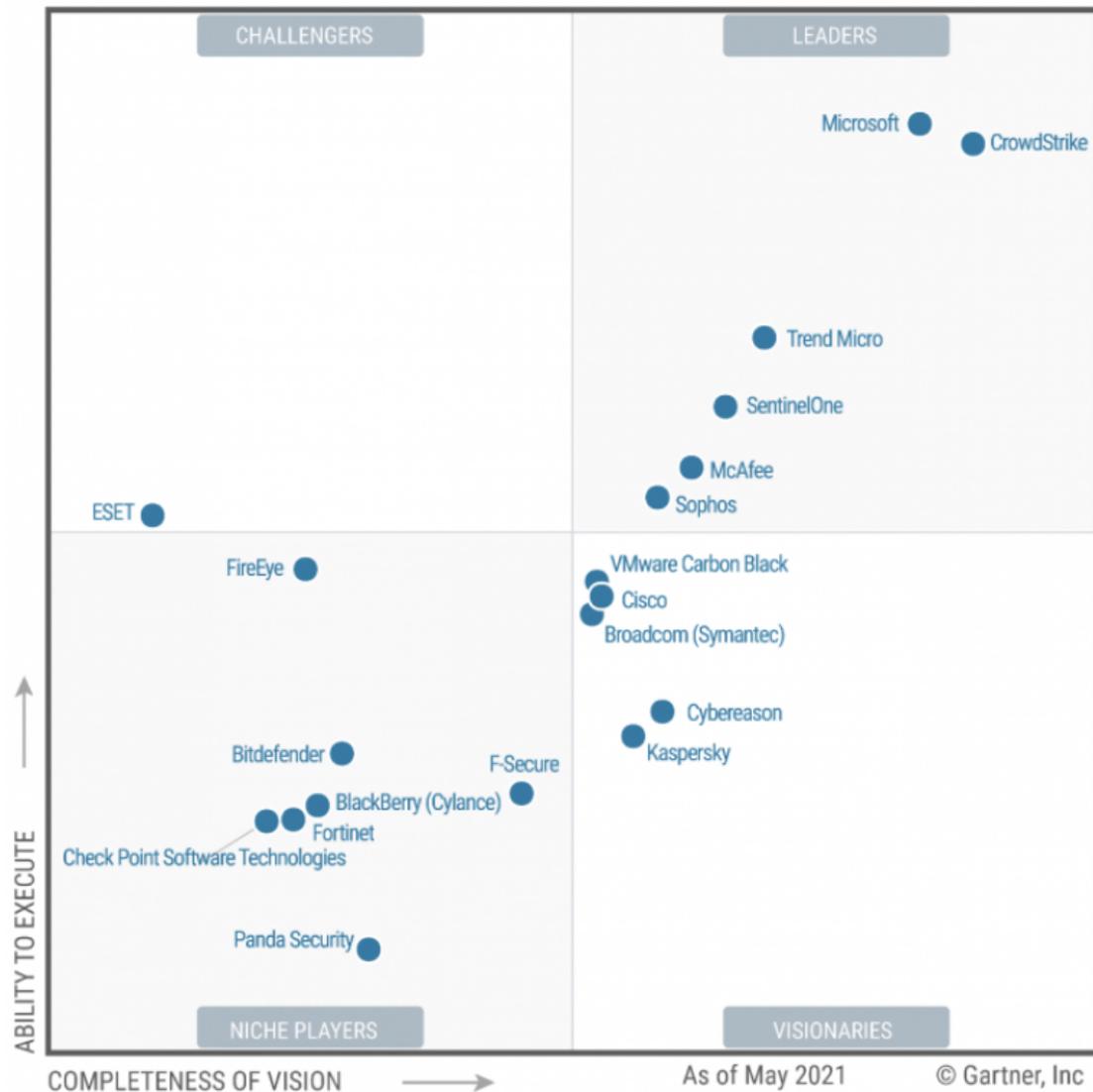


Figure 3.4. Magic Quadrant for endpoint protection platform, published by IT Gartner in the 2021 report.[29]

Study's results

Results from Figure 3.5 are alarming: neither one enterprise level EDR was able to detect and block all the attack vectors. Although sensors' behaviours can significantly change based on their configuration, the lesson is that does not exist a security mechanism that protects in an effective way from all threats and type of attacks.

It can be noted that the infection vector was a spear phishing email, which is a very common malware diffusion technique. This means that a human downloaded the attachment,

Table 1. Aggregated results of the attacks for each EDR. Notation: ✓: Successful attack, ●: Successful attack, raised minor alert, ✱: Successful attack, alert was raised ○: Unsuccessful attack, no alert raised, ✕: failed attack, alerts were raised.

EDR	CPL	HTA	EXE	DLL
Carbon Black	●	✕	✓	✓
CrowdStrike Falcon	✓	✓	●	✓
ESET PROTECT Enterprise	✕	✕	✓	✓
F-Secure Elements Endpoint Detection and Response	✓	✓	✓	✓
Kaspersky Endpoint Detection and Response	✕	✕	✕	✓
McAfee Endpoint Protection	✕	✕	✓	✓
Sentinel One	✓	✓	✓	✕
Sophos Intercept X with EDR	✕	✕	✓	-
Symantec Endpoint Protection	✓	✕	✓	✓
Trend micro Apex One	✓	○	✓	✓
Windows Defender for Endpoints	✱	✕	✕	✓

Figure 3.5. Result of the study, showing if EDR detected the threats and if the attacks succeeded.[41]

or followed a tainted link, leading to a first breach. In many cases such bad behaviours may be prevented by investing more resources in employee formation and awareness, teaching them how to recognize a suspicious email, how to signal them and what must be done when there is the risk of being compromised.

Also, every enterprise platform that provides an EDR also provide a visualization platform, which is basically a SIEM.

3.3.2 Most relevant open source tools

The previous session discussed about enterprise level EDR; but what about open source tools? The short answer is that, in the open source market, **does not exist** a complete EDR and SIEM system. There are some effort and notable projects, but almost every company offers a base-detection capability environment and/or an enterprise version including much more features. Most popular open source EDR solution are:

- **OSSEC**: the software offers Host IDS and IPS, log analysis, real-time monitoring and other EDR features; these are only basic functionalities, to have a full version there is need of an upgrade to the enterprise solution⁵. Furthermore OSSEC is addressed

⁵<https://www.ossec.net/products/>

to large companies and government agencies.

- **TheHive project**: a free Security Incident Response Platform designed for SOC and cybersecurity analyst to dealing with incidents that needs to be investigated. The goal is to speed up investigation operations. The project is fully free, but it is more focused on incident analysis rather than detection and prevention.
- **Osquery**: described in Section 2.3 and briefly mentioned here; it permits to view an OS as a relational database, allowing to query it in order to obtain many kind of information. The project only provides the infrastructure, while the configuration is totally up on the security staff. Osquery customization degree potentially allows to create a monitoring system based on final user needs and capabilities.

From the SIEM point of view, hence, considering log collection, visualization and analysis centralized platforms, there are also some meaningful project. Some of them offer the possibility to upgrade the solution to an enterprise level. As mentioned, there is not a complete SIEM framework at enterprise level completely free and ready to use. Anyway, the open source landscape offers many tools able to perform some specific task and that can be joined together. In this way more complex systems can be built, respecting open source foundation principles and hence remaining publicly available.

A list of the most popular open source solution identified in the landscape [11]:

- **Alien Vault OSSIM⁶**: it is provided as open source tool and enterprise level. The free version, with lower capabilities, includes key SIEM components. This tool joins together many other open source project, such Suricata, Snort, OSSEC and others. The free solution works fine with small environment, but with increasing resource demand clients are “encouraged” in buying the commercial version.
- **ELK Stack⁷**: The ELK stack is by far the most popular building block for SIEM solutions and in general for data collection and visualization. The stack will be discussed in another chapter in details, here are highlighted the main components:
 - Elasticsearch: a Search Engine database, a non-relational database optimized to perform queries on document often stored in JSON format.
 - Logstash: it performs log collection from endpoint and a first elaboration of them; it delivers data to and from Elasticsearch and Kibana for data management and visualization. It can also process, filter correlate collected data; Logstash, if correctly configured, can receive data from a huge landscape of sources leveraging a big number of available plugins.
 - Kibana: a web dashboard with SIEM visualization capabilities; it allows to create dashboards, alerts, visualize and collect data in a way that security operators can easily understand.

⁶<https://cybersecurity.att.com/products/ossim>

⁷<https://www.elastic.co/>

- **Beats:** they are a collection of agents that are installed on the end point and that collect different kind of logs, depending on their target. Beats deliver data to Logstash that will carries them.

Initially thought to be open source, in the 2021 January, Elastic company releasing a new version of the stack changing the licence to a proprietary one. The previous version are still available but are no longer updated and may lack of functionalities.

Being ELK a great solution, Amazon forked the last open source release of the stack and started creating its own open source version, named OpenSearch, implementing new functionalities and improving many aspects.

Due to its flexible nature, the stack was used to build many other solution, both enterprise and open source. Here are briefly describe two of the most popular freely available:

- **Wazuh:** a SIEM solution heavily based on ELK. It provides many ready-made dashboard for data visualization as well as many searching queries heavily Rule-based that can be customized, extended and integrated with MITRE ATT&CK framework. The project also provide an agent that can be installed on the end point, with the task to collect log and deliver them to Logstash centralized process. Most analysis operations, hence, are executed on server side leaving to the agent the task of collection and delivering.
- **SIEMonster:** it integrates ELK functionalities with other open source project, such RabbitMQ for queueing, SearchGuard for encryption and authentication and ElastAlert for alerting. As additional feature, this solution can be implemented both on cloud, through Docker containers, and on bare metal.
- **OSSEC:** the previously mentioned agent has also a platform to perform log collection and visualization. It uses an external visualization which may be Kibana or Grafana. This solution is able to run analysis on network services, endpoint and servers offering the main capabilities that SIEM should provide; anyway it does not perform really deep analysis and lack of some core log management components.

3.3.3 EDR and SIEM - Consideration

In this section was provided an overview of the most important features and products of EDR and SIEM landscape, both from an enterprise and open source point of view.

Attackers and their strategies continuously evolve; hence a continuous research is needed in order to provide an effective defence. As pointed out by previously analysed documents, neither enterprise solution are able to detect all kind of threats, so a great effort must be done for the overall community.

Companies of any dimension must be aware of how much employee training and formation matter, investing more resources in awareness. No matter how much a system is secure and protected with the most advanced protocols, the weakest ring chain is, almost, always the human factor. Indeed, without proper behaviours, any asset and network is vulnerable.

Another important limit in the current landscape is the cost of SIEM and EDR solutions; small companies and public administration often does not have enough resources to maintain those security measures, both speaking from a personnel and economic point of view. On the other hand, open source solution often are not suitable for organization-wide deployment and lack of some core features. The development of a real open source solution with, almost, **all** features is as much required as difficult to achieve. Indeed building a such complete infrastructure is so expensive (in terms of difficulty, resources and time) that companies simply offer their product under a fee or a subscription. The result is the lack of a real economic solution for all needs and requirements.

3.4 EDR and SIEM Challenges: Analysts Burnout and Predictive Analysis

The previous section highlighted how state-of-the-art EDR and SIEM platforms are not enough in providing a 360 degrees protection to a system. For this reason it is necessary to implement a layered protection, joining different tools and security measures.

It is important to signal that improving a detection and response system may not be the only evolution requirement. In this section is described one of the greatest challenge companies are fighting nowadays and some enhancement of next generation EDR systems.

3.4.1 Log flood and Analyst burnout

In 2020, Paloalto networks conducted a study [100] about Analyst burnout. This is a phenomenon that happens “When analyst spend their days chasing false positives, they miss the real threats and success metrics.”. In other words, companies receive a huge amount of log every day that makes impossible a one-by-one hand analysis. Those logs are for the greatest portion false positives; due to this noise of false alarms, real malicious logs may bypass the security controls performed by Analyst. Indeed when a human has to manually check such an amount of data will soon or later suffer a loss of attention, leaving the system more vulnerable. The study takes into consideration a team that receives 11.000 alerts per day. This is clearly an insane amount of data to analyse, including a lot of false positive alerts that create a huge amount of noise leading to a more difficult, and hence less effective, analysis. Of alerts pool considered, on average are estimated:

- Only 18% manually reviewed, highlighting how much a security team struggle in keeping up with all coming alerts;
- 17% pass through some automation, showing how much automation in triaging alerts is not performed enough;
- 32% are false positives, meaning that more that one out of three alerts should be discarded before being analysed. In an ideal system, able to detect all false positives, 3.520 alerts would be excluded, and most important no noise would exist;
- 28% of alerts are completely ignored, with the obvious consequence of leaving potential malware to perform their malicious actions.

The consequences of these percentage are that almost every organization has experienced a security breach only within last few years. There is need to insert automation, both AI and Machine Learning capabilities, in security analysis systems in order to support SOC teams to effectively perform their job, possibly reducing to 0 the percentage of alerts ignored. From Figure 3.6, it is possible to see that almost every company faced a security breach over past few years; notice that almost half of considered enterprises was affected in the last year, data that highlights the importance of a good security strategy. Security breach

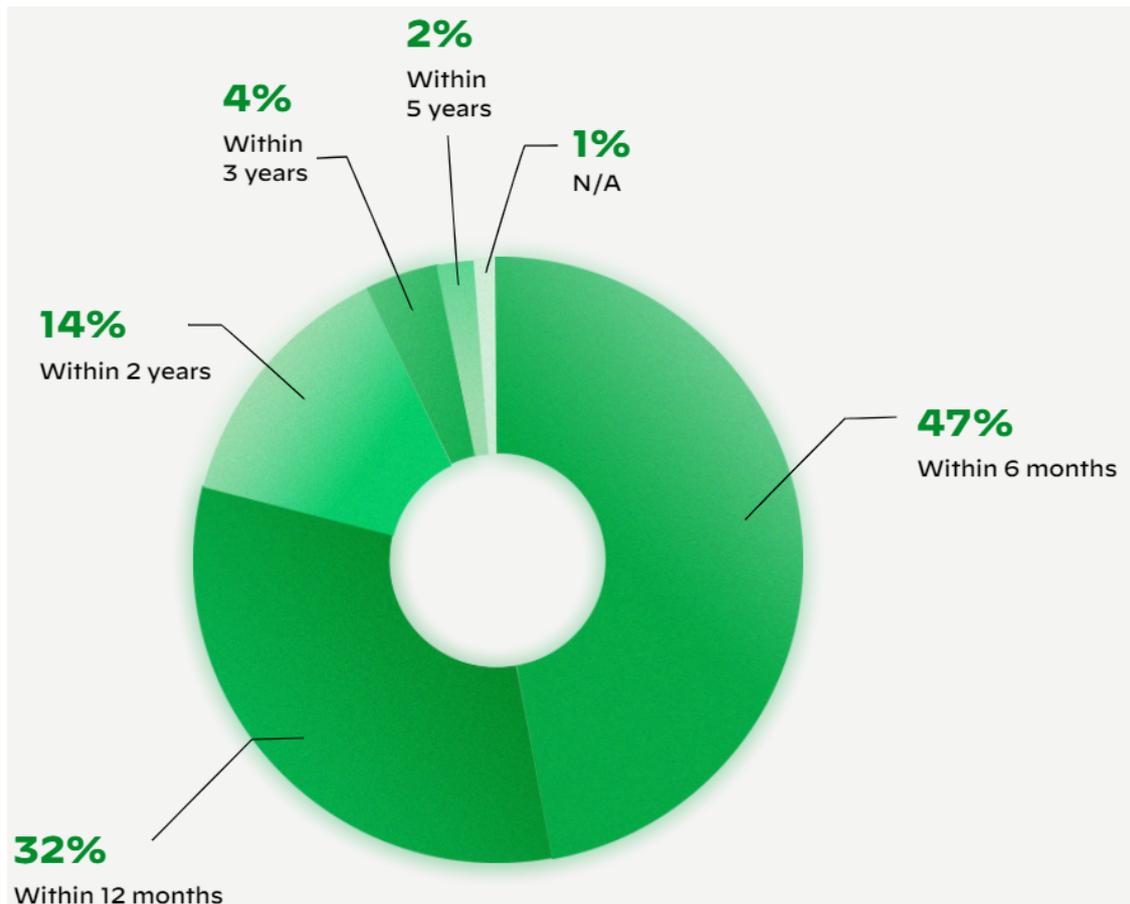


Figure 3.6. Organizations experiencing a security breach in 2020, from PaloAlto networks. [100]

may significantly affect business. From Figure 3.7, it is estimated that, from a data breach, in only 5% of cases no data was compromised; on the other hand many organization experienced a loss of employee, customer and company sensible data, a disruption of some service or business and a financial loss. Other indirect consequence is the reputational damage that a company suffer from a successful attack, with the risk of losing clients or difficulties in expanding the business.

Another clue argument that must not be ignored is the human factor. Indeed analyst employees facing a security breach suffer personal consequences, that may vary from lack

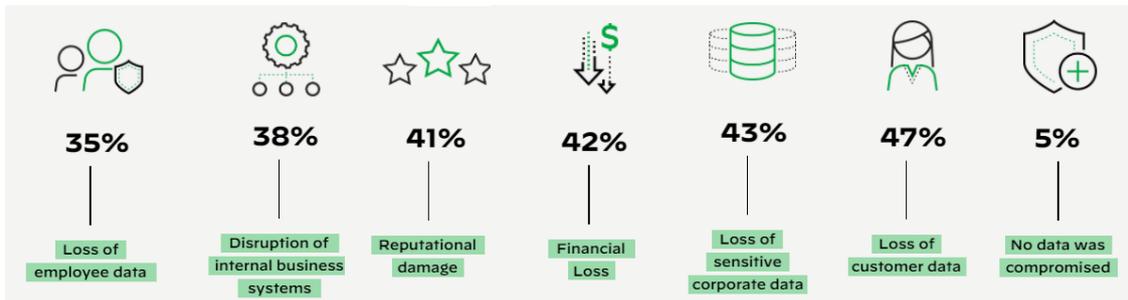


Figure 3.7. Business impact of a security breach for an organization. [100]

of sleep and anxiety to additional pressure, as shown in Figure 3.8. Those factors cause a worse work environment and a less effective security team.

Nowadays, companies started focusing in including automation in their security procedure

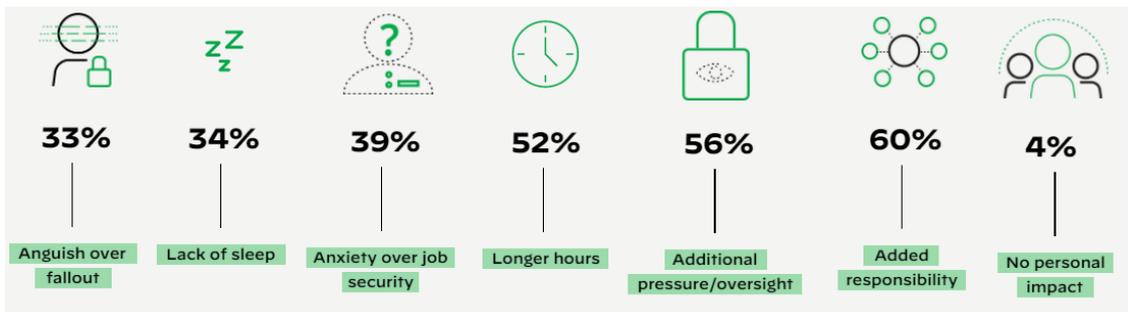


Figure 3.8. Personal impact of a security breach for security analyst. [100]

and systems. At the moment, organizations still reside in an early stage, but it is clear the need of a **Predictive Analysis** to improve SOC effectiveness and security teams members wellness.

3.4.2 Predictive Analysis

In its paper [30], *Dave Gruber* introduces the concept of Predictive Analytics. Humans depends, understand and learn from past experiences. The human abilities to understand how to behave, to identify which patterns to follow in a given situation and how to react allow to take the correct decisions in everyday life. “Transfer” these abilities to a security system would be a crucial evolution in threat detection. Predictive Analytics would be exploited in this sense: recognize good patterns and react or alerts for a malicious one.

The paper cites which are the most common analysis approach for nowadays EDR:

- **Signature-based Prevention:** agents which are heavily rule based, they look for known threats filtering logs in order to detect malicious actions based on specified

criteria. They may be continuously updated by interacting with online shared repositories where threat information are disclosed. This approach covers the majority of classic EDRs but is also the less effective; being rule based, some change in the malware actions may lead to bypass security controls.

- **Machine Learning:** the approach leverages AI and ML capabilities in order to discover malicious actions; not being based on rules, ML may be able to detect unknown threats looking for suspicious operations rather than matching fixed criteria. Deep learning may be used to understand and identify behavioural patterns, applying ML to many resources. ML may be a very effective approach, but it comes along with other issues, such computation capabilities and the ability to build a suitable dataset in order to perform a proper training.
- **Behavioural Analytics:** the goal of this approach is to identify fileless attack techniques, which often tamper good software with malicious code. The result is an execution of a legitimate process, with the ability of performing dangerous actions. To detect such behaviour, it is necessary to monitor activities looking for performed operations. Since those attack are more complex to be identified, fileless attack are expected to become more popular.
- **Multi-technique Prevention:** building a system with different layer of analysis is obviously the best choice from a security point of view. These techniques are able to detect the majority of attacks, but it must be remembered that the human factor is the weakest link; detection and response system is always required, since an employee may do some wrong action that leads to a security breach.

To summarize, a Predictive Analytics allows to:

1. Detect malicious behaviours, that otherwise would bypass security control or be ignored
2. Speeding up triage time and improving investigation procedures
3. Improving SOC teams effectiveness and security analyst wellness

3.5 Let the Hunt continue

Cyber defence is not only a matter of SIEM and EDR tools, but it should be considered as a joined global effort. Exploits continue their evolution, along with software updates: for every new release, new bugs and vulnerabilities may be introduced and companies must be aware to react in a proper way.

One of the best and most popular defence method is to build a common knowledge of known exploits. Some of the greatest repositories are open source and are strongly suggested in order to be up to date with latest threats in worldwide landscape. Every company should be compliant with these directory, ensuring to implement in their systems all security mechanism to defend against the more advanced threats.

Other than MITRE ATT&CK project, discussed in Section 2.2, other shared repositories are available, such as CVE⁸ and NVD⁹.

3.5.1 Common Vulnerabilities and Exposures (CVE)

The **Common Vulnerabilities and Exposures - CVE** is a system launched in September 1999. It is maintained by MITRE Corporation and collects all known information security vulnerabilities and exposures [95].

A CVE can be submitted by anyone discovered a vulnerability. The request will be analysed by a CVE Numbering Authority (CNA) and will be eventually published. An ID is uniquely assigned to a request so that is possible to have an identifier for every entry. A CVE Record is published both in human and machine readable format, making possible the interaction with security systems. A CVE has three states [18]:

- Reserved: initial state of a CVE Record, a CNA associated a unique ID
- Published: when a CNA the CVE Record, which must contain a description, the ID and at least one public reference
- Rejected: when the ID and the associated Record should not be used anymore. A rejected Record can still be accessed by users

Organization should remain up to date with CVE, in order to be ready to implement security measures in case a new disclosed vulnerability affects its security systems. In the Figure 3.9, it is possible to see the lifecycle of a vulnerability, from its discovery to publish on CVE website.



Figure 3.9. CVE Record Lifecycle, from discovery and reporting to publish. [19]

⁸<https://www.cve.org/>

⁹<https://nvd.nist.gov>

3.5.2 National Vulnerability Database (NVD)

The **National Vulnerability Database - NVD**¹⁰ is the US government repository of standards-based vulnerability management. [60] This project includes dataases of security checklist references, security software flaws and misconfigurations, product names and impact metrics.

The NVD performs analysis on CVE that has been published. NVD builds upon CVE Records analysis, containing a description, advisories on solution and tools and a severity score compliant to the Common Vulnerability Scoring System (CVSS). This scoring system assings a numerical value reflecting the severity of a vulnerability: a higher score represents a more dangerous vulnerability and it depends on the damage caused if it is exploited and other criteria.

This project can be used to have a baseline on how much an exploit may affect a system

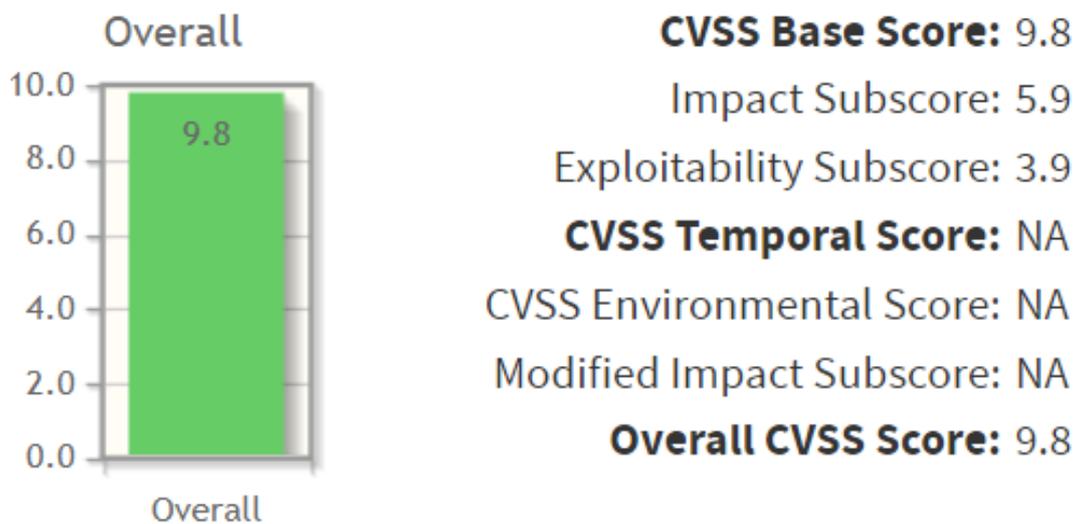


Figure 3.10. CVSS assigned to CVE-2019-18323.

and eventually prioritize which security measures implement based on the CVSS score. In 3.10 is shown a CVSS associated to CVE-2019-18330¹¹. The vulnerability signals that an attacker with network access to the MS3000 Server¹² can cause Denial of Service, and potentially gain remote code execution by sending ad hoc crafted packets to port 5010 over TCP.

The score associated is 9.8 (Critical) because of the high impact the attacker can cause to the system. Indeed, DOS attacks are one of the most disrupting type, with the additional possibility of malicious code execution.

¹⁰<https://nvd.nist.gov/>

¹¹<https://nvd.nist.gov/vuln/detail/CVE-2019-18323>

¹²MS3000 is a migration server

3.5.3 The problem of unknown

Even if such project are crucial for cybersecurity defence of all companies in the world, they are focused on identifying and publishing **known vulnerabilities**. Therefore they cannot be considered as a prevention system against APT and unknown vulnerabilities; despite of the great contribution, business may be still vulnerable to Zero-days attacks.

Zero-Day Attacks

A Zero-Day attack is a software vulnerability exploited before vendors are aware of it [20]. Since their existence is unknown, attackers can easily exploit it without facing security measures. Depending on the severity of a vulnerability, the attacker can achieve different results representing a huge threats for the whole environment.

Once an attacker gained access to the victim system, it can start the exploit. Such attack has higher success possibilities since standard security mechanisms are bypassed. Furthermore, once a Zero-Day is discovered, a vendor may take some time to develop a patch and then other time will be required before the update is installed on all interested systems. This is called Windows of Exposure and it can be shown in Figure 3.11¹³. Notice that the risk associated to a vulnerability may not expire completely over the time, because not all systems may install the security patch associated. Being associated with a high risk, Zero-Day Vulnerabilities are subject of great attention by attackers. Many vendors offer a bounty to analysts that discover a vulnerability related to their software. Unfortunately, such info are also exchanged between attackers; an actual market exists and vulnerabilities and exploits information are bought by malicious users to build new attacks.

Zero-Day Vulnerabilities are and will be always a hot topic in cybersecurity. They are one of the main reason for the development of EDR and SIEM systems, and in general of Predictive Analysis, that has the goal to detect unknown malware, based on behavioural analysis and ML capabilities.

3.5.4 Call to weapons

To highlight the need of evolving cybersecurity tools, the USA President himself, Mr Joe Biden, ordered in May 2021 to improve National cybersecurity launching the *EDR Initiative* [40]. The President claimed *“faces persistent and increasingly sophisticated malicious cyber campaigns that threaten the public and the private sector”*. The executive order was released after the APT attack to SolarWinds Supply Chain company, named by FireEye SUNBURST, which campaign compromised nine US agencies, including Treasury, State and Energy along with many private companies.

In February 2021, Senator Ron Wyden stated: *“\$6 billion cyber shield failed to stop or detect the hacks.”* The referred tools was EINSTEIN, an IDS used by the government to protect federal networks. CISA ¹⁴ declared that EINSTEIN performs detection based only on signatures; this means that it was able to detect only known malware.

¹³https://commons.wikimedia.org/wiki/File:Window_of_Exposure.png

¹⁴Cybersecurity and Infrastructure Agency

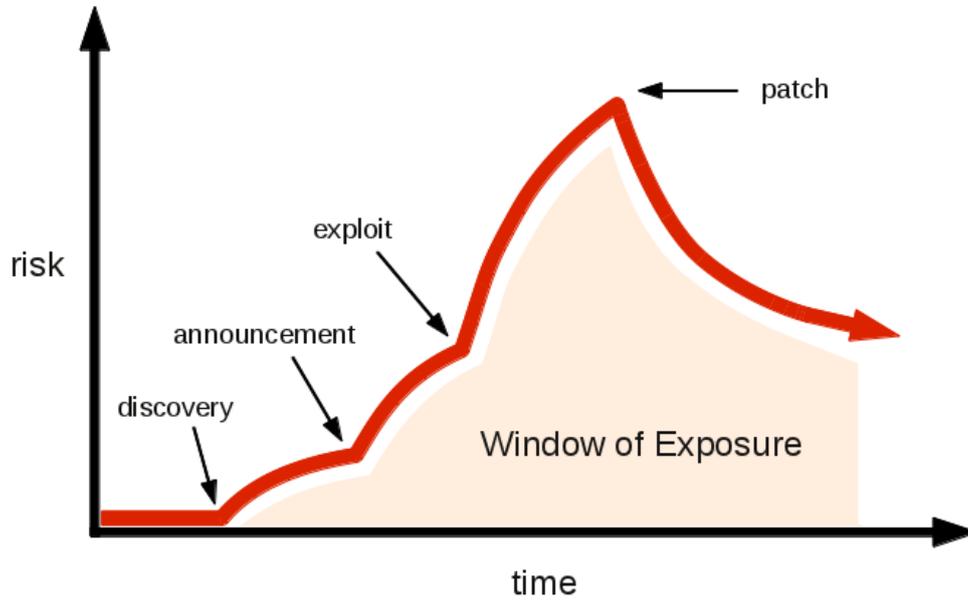


Figure 3.11. Windows of Exposure.

SUNBURST acted as a line delimiter that highlighted how even USA government was not enough protected and the importance of a 360 degrees protection, since hundreds of businesses were involved both from private and public sectors. The whole supply chain must be protected in order to achieve security. The only way to do so is to invest in EDR and advanced prevention and detection systems, in personnel and its formation, in building a real architecture using the most advanced security technologies.

Chapter 4

Perimetro di Sicurezza Nazionale Cibernetica

4.1 Introduction: the *PSNC*

The *Perimetro di Sicurezza Nazionale Cibernetica* has been progressively built by the Italian Government during the past two years [56][70][51][8][14]. *PSNC* plays the important role of safeguarding for the State, with a considerable legislative body. Taking a step back, as a part of the European Cybersecurity strategy¹, the European *NIS*² directive (Network and Information Security) was, in year 2016, the first brick for an European Union wide cybersecurity legislation. The directive was aimed at enhancing the security of EU member states, in order to improve the common security level across the union. *NIS* directive, was implemented in Italy with the decree-law number 65³ of May 18th, 2018. That decree was followed by decree-law 133⁴ of year 2019 when the perimeter has been defined and, since that moment, it has evolved a lot. Concretely, an entity included in the regulation, has to consider its actual situation, make a list of its ICT assets and communicate them to the authority. Moreover, entities are obliged to notify incidents to authorities, with a specific procedure. This is done in order to give intervention possibilities to the State, just in case national security issues arise. In general, who is included in the *PSNC* has to follow a continuous implementation process to maintain and improve its security posture. The *PSNC* can be located in the field of cyber resilience, and has been created with the aim of taking Italy to a new level of security. Also, it is remarkable to say that features of the *PSNC* are mainly based on NIST's Cybersecurity Framework⁵.

¹https://ec.europa.eu/commission/presscorner/detail/en/IP_13_94

²<https://www.enisa.europa.eu/topics/nis-directive>

³<https://www.gazzettaufficiale.it/eli/id/2018/06/09/18G00092/sg>

⁴<https://www.gazzettaufficiale.it/eli/id/2019/11/20/19G00140/sg>

⁵<https://www.nist.gov/cyberframework>

4.2 PSNC organization

Entities are included in the perimeter when they are located on National territory and, an essential State function depends on them. This function must consist in a fundamental one, regarding the maintenance of social, civic or economic activities. Other two central actors are the *CISR*⁶ and the *CVCN*⁷. The first is the *Comitato Interministeriale per la Sicurezza della Repubblica* and presents proposals from the adopted decrees. The second stands for *Centro di Valutazione e Certificazione Nazionale* and as a target, has to guarantee the security and vulnerability absence, in hardware and software, that will be employed by the entities included in the perimeter. When included entities are intended to trust someone for the provision of ICT goods, CVCN makes the necessary security tests. CVCN has been instituted inside the recently born *Agenzia per la Cybersicurezza Nazionale*⁸. In general, it is important to think about the requirements management and the structure of the *PSNC*, as something that evolves during time and should be modified dynamically.

4.3 Legislation

The legislation body of the *PSNC* consists of four DPCMs⁹ and one DPR¹⁰, that contribute to define rules and requirements, for public and private entities included in the *PSNC*. The contents of the various decrees are briefly described in the next sections.

4.3.1 DPCM 1

First DPCM, precisely number 131¹¹ from July 30th, 2020, delineates the parameters with which, entities to be included in the perimeter, are identified. Three criteria are selected for the definition of the subjects: *essential function*, *essential service*, *graduality*. Moreover, it defines the essential sectors that, in addition to the Government one, are: interior, defense, space and aerospace, energy, telecommunications, economy and finance, transportation, digital services and critical technologies. Entities operating in these sectors are included in the perimeter. In addition, this DPCM obliges involved entities to arrange and update, at least annually, lists of ICT goods with indication of networks, information systems and information services that compose them. Last, this decree establishes an inter-minister board for the implementation of the *PSNC*, that will support the aforementioned *CISR*. This first DPCM, is surely an important step, but the following are probably more effective on the operational layer.

⁶<https://www.sicurezzanazionale.gov.it/sisr.nsf/chi-siamo/organizzazione/comitato-interministeriale-per-la-sicurezza-della-repubblica-cisr.html>

⁷<https://atc.mise.gov.it/index.php/sicurezza/cvcn>

⁸<https://www.acn.gov.it/>

⁹https://www.treccani.it/vocabolario/dpcm_%28Neologismi%29/

¹⁰<https://www.garzantilinguistica.it/ricerca/?q=D.P.R.>

¹¹<https://www.gazzettaufficiale.it/eli/id/2020/10/21/20G00150/sg>

4.3.2 DPR

This decree is a different one, because it was proposed by the President of the Republic, instead of Prime Minister like other DPCMs. The DPR is the number 54¹², published on February the fifth, 2021. This time, procedures and modes for ICT assets provisioning are defined along with modes and deadlines in which competent authorities put in place verification activities are outlined. The DPR 54, defines articulated relationships between stakeholders and authorities included in the perimeter: before concluding agreements for assets supplies, entities must communicate the decisions to the CVCN or to a CV (which stands for *Centro di Valutazione*). At that moment, contractual activities must be stopped to allow the verification process. This process, is divided in three main phases:

- A preliminary one, 45 days long, extensible only once for additional 15 days;
- A test preparation phase;
- A final hardware and software test execution phase.

The last two phases can last at most 60 days. This decree, is a necessary element for the applicability of the law.

4.3.3 DPCM 2

The second DPCM is the number 81¹³, April 14th 2021. Thanks to some tables, it defines the categories of incidents that have impact on ICT goods. The classification is useful for the identification of the notification time to the CSIRT, hence when an incident occurs. Depending on the category, the notification may be mandatory from one hour, for most serious incidents, to six hours for less severe episodes. This obligation starts from January the first, 2022.

4.3.4 DPCM 3

June 15th 2021 is the date for the third DPCM¹⁴. This one, defines ICT assets categories to be employed in the *PSNC*. These categories, must be updated at least annually, coherently with the improvements and technology evolution. The categories defined are four and, all the goods, systems or services, are divided and included in them. The categories are summarized here (the one depicted is not the precise name of the category but a description):

1. Telecommunication hardware and software;
2. Hardware and software for the security of networks and data;

¹²<https://www.gazzettaufficiale.it/eli/id/2021/04/23/21G00060/sg>

¹³<https://www.gazzettaufficiale.it/eli/id/2021/06/11/21G00089/sg>

¹⁴<https://www.gazzettaufficiale.it/eli/id/2021/08/19/21A05087/sg>

3. Components (both HW and SW) for data acquisition, monitoring, control and automation of telecommunication networks;
4. Softwares for security mechanisms implementation.

4.3.5 DPCM 4

The fourth DPCM has still not been issued at the time of this thesis. In any case, it should be about technology screening and laboratory accreditation.

4.4 Implementation

The legislative components that have been outlined above, are the foundation of the real implementation of the *PSNC*. Each entity has six months, from the communication that establish its entrance in the perimeter list, to communicate the list of ICT assets that are employed for functions and services, that are considered essential for the State. Notifications are mandatory, for the involved subjects, from January the first 2022 but, the six months that preceded that date were taken as an experimental period. During that period, and in any case, since the date of ICT goods transmission until December 31st 2021, to each included subject was requested to notify any incident occurred. This was just made to break-in and, no penalties were given if something went wrong. The operations of the *PSNC* started officially from 24th June 2021, with this experimental period. Another practical aspect that, although not mentioned in the first documents, seems to have been considered recently, is the probable decision to predict the incident notifications, just in case these episodes do not involve specific investigative secrecy needs. In comparison with the original drafts, another requisite that has been relaxed, is the one that mandates to locate in Italy the digital data processed. This fact has been, and is still today, a problem especially for private stakeholders because it can create economic damages and it is not updated with the nowadays evolution of technological services. To sum up, the application of the *Perimetro di Sicurezza Nazionale Cibernetica*, marks an important step for the future of Italian cyber resilience and national security.

4.5 Cyber Threat Intelligence and Threat Hunting in the *PSNC*

Cyber Threat Intelligence and Cyber Threat Hunting, together with the activities of detection and response to cyber attacks, are key aspects for an effective and efficient cyber defense. Public administrations above all, need to be defended from arising threats and, strong defense measures are needed to protect crucial assets. The contribution of this thesis, is aimed at highlighting some specific aspects of endpoint detection, giving great emphasis to the use of widespread knowledge bases like MITRE ATT&CK and exploiting Machine Learning and AI capabilities for anomaly detection. Taking into account the decrees described in Section 4.3, this job can be placed in the second DPCM (Section 4.3.3), as it is explained in the section about *Security measures*. This section identifies five different categories in which functions are organized to guarantee high security levels: *Identify*,

Protect, Detect, Respond, Recover. The fourth, the *Detect* phase, concentrates on three main sub-categories:

- The detection of anomalous events and the analysis of their potential impact;
- The continuous monitoring, to identify events and verify effectiveness of the employed defensive solutions;
- The monitoring procedure aimed at ensuring comprehension of anomalous events.

Complete details about what is specified by the DPCM, for what concerns the detection, can be found from page 28 to 30 of the *Gazzetta Ufficiale della Repubblica Italiana*¹⁵ number 138, year 162.

All these themes are, in some way, related to the work carried out within this thesis. The thesis, wants to be an open source contribution for the development of future implementations of the necessary detection measures, which are imposed by the *PSNC* requirements.

¹⁵<https://www.gazzettaufficiale.it/eli/gu/2021/06/11/138/sg/pdf>

Chapter 5

Contribution - Log Collection

5.1 Initial attempts

The following sections will outline in detail, the main attempts done in the direction of mapping MITRE ATT&CK techniques into SQL with Osquery. The aforementioned mapping, consists in the writing of SQL queries aimed at searching for certain malicious behaviours, executed in the endpoint. Specifically, given that each technique in ATT&CK can be applied in many ways, system is queried for evidences of that ways to perform the technique. Some of the described attempts have been useful for the final work, while some others have been failures for different reasons. All of them, in any case, contributed to the achievement of the final implementation.

5.1.1 Existing repository update: *osquery-attck*

After some research on what open source offered, a concrete example that was coherent with this thesis' idea has been found. This consisted in a repository called *osquery-attck*¹. The repository is in line with the idea of mapping, in some way, the intelligence information included inside MITRE ATT&CK with a probe instrument like Osquery. The repository under question in fact, proposes itself as an attempt to cover some elements of ATT&CK with some Osquery packs. This project is a good example of how to do threat hunting taking advantage of intelligence. The *osquery-attck* project, contains Osquery packs both for Windows and Linux systems, a brief explanation of the philosophy behind it and, a not really well ordered documentation. This said, there is quite a relevant problem that appeared almost immediately: the repository has not been updated for approximately 3 years. The project has probably been left out, because it is clear that there were still lots of possible bug fixing and improvement activities to be done. This defect is pretty important because, MITRE ATT&CK has evolved quickly each year and substantial updates have interested techniques. Some names or IDs are changed, some techniques have been added, other deleted, sub-techniques have been introduced with the consequence that some techniques are now sub-techniques and more. This situation makes the repository obsolete and

¹<https://github.com/teoseller/osquery-attck>

not very useful. However, being the project interesting the same, the decision has been to try to update it to ATT&CK v10 as an initial step.

Update procedure

The idea of the update, consisted essentially in a general refresh of all the query descriptions. The queries in fact, were not obsolete, but their links with specific techniques were. As already explained in 2.3, osqueryd mode works thanks to a configuration file, where the queries to be executed are specified together with the interval between their executions. To get an idea of how configuration files are organized, figure 5.1 shows a typical example of one of them. The platform is specified together with a description including all the techniques covered in the file and then, a group of queries with intervals and descriptions always present, is listed.

```
{
  "platform": "windows",
  "description": "ATT&CK: T1053",
  "queries": {
    "Scheduled_Task": {
      "query": "select name,action,path,enabled,datetime(next_run_time,'unixepoch','UTC') from scheduled_tasks;",
      "interval": 3600,
      "description": "Lists all of the tasks in the Windows task scheduler - ATT&CK T1053",
      "platform": "windows",
      "removed": false
    },
    "Snapshost_Scheduled_Task": {
      "query": "select name,action,path,enabled,datetime(next_run_time,'unixepoch','UTC') from scheduled_tasks;",
      "interval": 28800,
      "description": "Lists all of the tasks in the Windows task scheduler",
      "platform": "windows",
      "snapshot": true
    }
  }
}
```

Figure 5.1. Example file: `windows_scheduled_tasks.conf` of `osquery-attck` [54].

After some checks, it has been established that the version employed in the repository was probably ATT&CK v4. This is coincident with the time of the last commit because, version 4 was the most recent one at that time. For the sake of completeness, when proceeding with the update, also spot checks on version 3 have been performed, in case the author did not use the last possible version. Anyway, these checks always confirmed that probably v4 was the chosen one. The repository update regarded all the files: `.conf` files where examined once at a time and a complete list of the techniques employed for each was written.

Five categories of update have been identified:

1. Techniques that have not changed the name nor the ID, just some descriptive details;

2. Techniques that have maintained the same ID but changed the name;
3. Techniques that have changed ID or have been subdivided in sub-techniques or have shifted from being a technique to a sub-technique;
4. Techniques deprecated;
5. Techniques deprecated but with suggestions for other techniques that play the same role;

Each technique that appeared in the repository's files has been categorized in one of these five. First two categories do not require updates. In the query descriptions, just the ID is reported, then if the substance is the same, no modifications are required. Third type of techniques are the ones that necessitate more work: for each of these, the corresponding technique in ATT&CK v10 has been searched, comparing descriptions and other features and then, the ID in the `.conf` has been changed, deleting the v4 version and putting the v10 technique. The fourth type of technique, being deprecated and then eliminated from the framework, needs just to be deleted from the configuration files where present. Fifth type's techniques have been substituted with the suggested techniques thanks to the update details present in the website². An example for each type is shown here:

1. T1112 – `Modify registry` has remained the same;
2. T1219 – `Remote Access Tools` has now the same ID but different name T1219 – `Remote Access Software`;
3. T1086 – `Powershell` has become a sub-technique, precisely T1059.001 – `Powershell (with T1059 being Command and Scripting Interpreter)`;
4. T1153 – `Source` has been deprecated and then eliminated for the scarce use;
5. T1108 – `Redundant Access` has been deprecated since ATT&CK v7 and then eliminated but, according to the official instructions, it is covered by techniques T1078 – `Valid Accounts` and T1133 – `External Remote Services`;

In the end, the whole repository has been updated following this procedure and some experiments were done both on Linux and on Windows endpoints. The packages worked well logging interesting things but, the results weren't still satisfactory because, the link with ATT&CK in this way is not strong enough. Moreover, dividing techniques in this way led to a bit of confusion. This work anyway, has been interesting to take confidence with Osquery and has provided a way to map techniques to queries.

5.1.2 Mapping APT groups to Osquery

Once concluded the updating work just described, a new ambitious idea came out. Considering that Advanced Persistent Threat groups (APTs, described better in 3.1.2) are today the biggest menace for governments and businesses, with the long time interval during

²<https://attack.mitre.org/resources/updates/>

which they are executed (maybe weeks or also months) and, the human skills and powerful technology behind them, something in that direction has been tried. MITRE ATT&CK models, among other things, also these APT groups. Inside the framework, between all the information, for each of these groups the list of techniques that has been already employed by the group itself is present. Ideally, this permits to design a sensor that is able to recognize when an APT has been executed or better is still being executed, thanks to detection of the techniques that characterize that group. A random sample of 10 APT groups was selected from ATT&CK, but a series of problems emerged quickly. The first problem is that not all the techniques in MITRE ATT&CK are easily detectable just with system logs obtained through sensors like Osquery. This may be a problem when, in the list of the techniques employed by a specific APT group, the majority of them are not detectable with system logs. In those cases, the risk is that recognition of a specific APT becomes too hard and false positives eventually arise. Another major problem, consists in the fact that APTs are usually developed in a long time span. This makes the work of recognition very hard, because a technique recognized today may be related to another that results applied three weeks later, and another one applied two months later. In short, it is very hard to recognize different steps of an attack that occur in a span time of months: too much data to store, very difficult to do meaningful experiments, high risk of false positives and false negatives.

For the reasons explained this idea, although ambitious, has been left apart for the time being.

5.1.3 Combining MITRE CAR with Osquery

Among the many instruments offered by MITRE, the Cyber Analytic Repository (CAR³) is a knowledge base containing analytics developed once again taking the ATT&CK adversary model as a base. Analytics are developed to detect adversary behaviours and each is related to one or more ATT&CK techniques. An analytic, consists of a description about the behaviour wanted, some references to related ATT&CK techniques (and sometimes other frameworks), data model references and sometimes a pseudocode for implementations. The pseudocode was the interesting part. Again the idea was to take this pseudocode, translate it in an SQL query to be used with Osquery and then collect logs that were able to recognize one or more techniques. CAR analytics include a good number of techniques but for reasoning, just Linux analytics were selected (because they are a small number). Almost immediately, the work stopped. This time, the problem was that pseudocodes were often not available and this increases a lot the work's difficulties. To make matter worse, when available, pseudocodes are often not easy to translate and first attempts were absolutely not satisfactory. Another idea was scrapped.

5.1.4 Mapping MITRE ATT&CK techniques for Linux systems

After not successful attempts with APTs and MITRE CAR which both consisted in the combination of more techniques to obtain a meaningful result, the subsequent work, was

³<https://car.mitre.org/>

targeted at making things simpler. Not having great open source basics to develop the work above, except from the *osquery-attck* repository described in 5.1.1, an approach starting from simple things was needed. The focus shifted again on the basic brick of ATT&CK: the technique. A translation of single ATT&CK techniques was targeted.

This time, it has been decided to reduce the significant number of *Enterprise* ATT&CK techniques before start. The reduction has been performed in a simple way: delimiting the techniques to the one dedicated to Linux platforms. In this way the number was reduced, although still very large.

Methodology applied

A long list was written with all the desired techniques and respective sub-techniques, organized in tactics. Then tactic by tactic, each technique was studied and, considerations were done about whether or not it was possible to write SQL queries for detection of that technique. Considerations were mainly based on two sources:

- the detection suggestions given by MITRE ATT&CK for the specific technique;
- the techniques already mapped in at least a file in the *osquery-attck*.

Each technique was then labeled as positive or negative: positive has meant that the technique was evaluated with the possibility to be translated in a query; if negative, translation was evaluated as not possible. The reasons for a technique to be evaluated as unsuitable for mapping in Osquery were multiple, but substantially consisted in the fact that detection for that technique was not possible with endpoint's system logs. In those cases, other instruments and data are needed for detection, like network traffic analyzers or application logs as examples. A technique may also have been considered not mappable when the information given about it was too scarce to perform an adequate SQL mapping.

Results and decisions

As suggested in 2.2.2, an effective way to visualize the quantity of techniques that were considered mappable, is to color them in an ATT&CK Navigator matrix. This has given an idea about how many of them were eligible for the work. Mappable techniques were selected and they were quite a good number.

This led to some conclusions:

- the mapping job is potentially possible because, the majority of the ATT&CK techniques for Linux is mappable: a successful mapping of all of them would probably create a powerful open source probe, for intrusion detection;
- it must be considered that, although the majority of the techniques has been evaluated as mappable, not all of them are: this means that the work, would probably not consist in a probe comprehensive of the whole ATT&CK matrix considered;
- to achieve a good quality mapping, a great amount of time and a lot of expertise on intrusion detection would probably be necessary, making the work difficult to be completed for this thesis.

To sum up, the work was of great interest, but very difficult to achieve in limited time. In any case, the idea seemed good and, since each project must start little and then eventually become big, the decision taken was to apply the methodology of mapping MITRE ATT&CK to Osquery, starting from a sample of techniques. This gave the possibility to analyze them deeply.

5.2 Query writing

At this point, considered what was decided after the previous attempts, the primary thing to do was to identify a reduced set of techniques to translate to SQL queries. The set should have been small, meaningful and well translatable. In this section, the chosen subset is presented and, details about how the detection queries have been written and tuned are outlined.

5.2.1 Mapping *2021 - Threat Detection Report* top 10 techniques to Osquery

The top 10 of the most used techniques inside the Red Canary *2021 - Threat Detection Report* described in section 2.5, appeared to fulfill all the previously outlined requirements. It is small enough, being composed of a 10 techniques ranking and it is meaningful too, because the techniques on probation are relevant for diffusion in the real world. Moreover, a brief check showed that probably all the techniques were included in the subset of translatable techniques. Decision taken, an initial organization was needed. The idea was to write one or more queries for each technique in the rank. Considering the possibilities given by Osquery, the building of a query pack in the form of a configuration file for each of the techniques, was probably the best way to divide and organize queries. The process then developed with the writing of queries drafts that were progressively refined.

The 10 techniques selected have different platform compatibilities: 6 are Windows-only applicable, while 4 are multi-platform. Given this, it has been decided to focus on Windows OS. Techniques' descriptions are outlined in 2.5.1. The list below, will give a brief description of which behaviours have been searched with Osquery to make detection of those techniques:

1. T1059 - Command and Scripting Interpreter

- T1059.001 - PowerShell: to detect the application of this sub-technique, main efforts have been concentrated on the detection of `powershell.exe` process instances, showing particular behaviours. Precisely, `powershell` malicious instances masquerading as legitimate Windows processes, strange network connections, `powershell` instances including encoding or obfuscation commands in their command line and `powershell`-related files in specific Windows directories. Moreover, malicious script blocks execution is checked.
- T1059.003 - Windows Command Shell: this sub-technique is detected thanks to checks on obfuscation commands inside `cmd.exe`, excessive use of specific words like `set` or `call` on command line of `cmd.exe` and, output redirection to

local admin's shared folders. Moreover, checks on Windows event logs for *Process Creation* may be useful.

2. T1218 – Signed Binary Proxy Execution

- T1218.011 – *Rundll32*: this sub-technique can be detected mainly searching for process instances of `rundll32.exe` with particular behaviours. This means instances without command line arguments while spawning childs or making network connections, instances executed from strange parent processes or, instances loading or writing to some specific Windows folders. Also `powershell` scripts containing `rundll32` word are checked.
- T1218.005 – *Mshta*: `mshta.exe` malicious use, may be recognized checking for parent or grandparent relationships of the executable instances. Suspicious network connections and file modifications in the working directory are also useful. Some protocol handlers (e.g., `javascript`, `vbscript` and so on) in conjunction with the `mshta` executable, may be an indication of attack: this behaviour can be checked also looking for executed scripts, with `powershell_events` table. Also, `mshta.exe` spawned by unusual software like MS Word, may be a sign of compromise. Spawning of the mentioned executable outside from Windows folder is strange too.

3. T1543 – Create or Modify System Process

- T1543.003 – *Windows Service*: to spot the malicious use of Windows services, a check of installed ones, especially if they are set to start automatically, is the first thing that has been performed. Together with it, an investigation of the various possible uses of `sc.exe` and `reg.exe` processes have been done. After that, there are specific paths to monitor because, it is unusual that services are loaded from there (also `ntfs_journal_events` can be used for this purpose). Focusing on Windows event logs, the one for *Service Installation* may be checked. Last, also service registries can give interesting information about this technique.

4. T1053 – Scheduled Task/Job

- T1053.005 – *Scheduled Task*: scheduled tasks, may be detected checking for specific directories' file modifications made by `schtasks` or `taskeng`. Also, `scheduled_tasks` table has been queried for executables that are usually exploited for bad purposes. In the end, checks for known binaries commands performed on `powershell.exe` or, modification of specific folders has been performed. Windows event logs have been queried for scheduled tasks related events too.

5. T1003 – OS Credential Dumping

- T1003.001 – *LSASS Memory*: detection queries for this sub-technique, have been concentrated on `lsass.exe`: instances of this executable with suspicious child processes like `powershell`, making network connections over specific ports are searched. Moreover, Windows event logs are checked for some specific event IDs.

The `powershell_events` table is queried to look for suspicious script keywords like `lsass`.

6. T1055 - **Process Injection**: to detect *Process Injection* technique, even if it may be a bit noisy, a general process monitoring is done. After this, queries have been written to check different behaviours: network connections opening over specific ports performed by `svchost.exe`, connections opened by unusual executables such as `notepad.exe`, presence of some specific binaries' instances without command line arguments, presence of Microsoft's `vbc.exe` instances paired with known malicious arguments. Different combinations of binaries together with specific CLI's arguments are also checked for presence on `powershell_events` table.
7. T1027 - **Obfuscated Files or Information**: two of the queries in this techniques' pack are dedicated to the research of `base64` and other encoding commands, performed in `powershell.exe` or in `cmd.exe`. These behaviours are searched also between `powershell` script blocks. In the end, `windows_eventlog` table is queried for *Process Creation*.
8. T1105 - **Ingress Tool Transfer**: this technique offers a lot of tips for detection: external network connections and listening ports are checked (e.g., also with keywords like `curl` or `wget`), different queries about `certutil.exe`'s behaviour are present, command lines related to *download* behaviours paired with specific binaries (e.g., `powershell.exe` or `bitsadmin.exe`) are checked, `powershell_events` is queried for binaries and commands that may be related to this technique.
9. T1569 - **System Services**
 - T1569.002 - **Service Execution**: queries checking for `PSexec` tools have been written, together with queries checking for `sc.exe` and `services.exe` executions in specific situations (e.g., `services.exe` spawned by `cmd.exe` with `/c` in the command line). DLLs loading is checked thanks to `powershell_events` and, `PSTools` directory modifications are checked thanks to `ntfs_journal_events`.
10. T1036 - **Masquerading**
 - T1036.003 - **Rename System Utilities**: queries used to detect this technique, must focus on the detection of command line arguments that may have been used to change system utilities name; another sign that may confirm utilities have been renamed is the fact that, file names on disk are mismatched if compared with binary's PE metadata.

Information contained in the previous list is not exhaustive, but gives an introductory idea of what kind of queries have been written. Information about what to check for each specific technique, have been taken from the three sources that are described in the next section (5.2.2), together with brief explanations on how information has been translated to queries. The whole queries writing process and methodology are outlined below.

5.2.2 Sources and methodology used to write the queries

The queries have been written taking inspiration from three main sources of information. The three sources are the *osquery-attck* repository, the MITRE ATT&CK framework and the information inside the *2021 - Threat Detection Report* (mainly the one contained in the *Detection* section of each technique). Details about how sources have been respectively used are now presented.

The *osquery-attck* repository on Github

This is the moment in which the updating work did initially with the aforementioned repository, turned to be more useful than expected. Given the fact that, after some trials, the queries present inside the repository were considered useful and well written, the decision taken was to take them (when available for the subset of techniques considered), eventually adapt and correct errors and put them inside the *in the making* list. Queries in fact were ready to use and, just some minor bug fixing work was done. Queries descriptions were modified and put in the style of the new project; each query was then inserted in the section relative to the technique it represented. This was possible thanks to the open source nature of the repository.

The MITRE ATT&CK framework

The second source that has been used for query writing, is the MITRE ATT&CK framework. Each technique comprehends much information but, the more useful one in this case, is the *Detection* section (an example is shown in figure 5.2).

Detection

ID	Data Source	Data Component
DS0022	File	File Creation
DS0029	Network Traffic	Network Connection Creation
		Network Traffic Content
		Network Traffic Flow

Monitor for file creation and files transferred into the network. Unusual processes with external network connections creating files on-system may be suspicious. Use of utilities, such as FTP, that does not normally occur may also be suspicious.

Analyze network data for uncommon data flows (e.g., a client sending significantly more data than it receives from a server). Processes utilizing the network that do not normally have network communication or have never been seen before are suspicious. Analyze packet contents to detect communications that do not follow the expected protocol behavior for the port that is being used.^[394]

Figure 5.2. ATT&CK detection information for T1105 – Ingress Tool Transfer⁴.

⁴<https://attack.mitre.org/techniques/T1105/>

As it's possible to see in the figure above, detection tips consist in a list of MITRE ATT&CK *Data Sources* identified by a code that starts with DS followed by 4 digits. Data sources are 38 in total (version 10 of the framework) and are useful to gain information about attacks. Some examples of data sources are: DS0015 - *Application Log*, DS0017 - *Command* or DS0022 - *File*.

Not all the data sources are useful for the purpose of this thesis that is limited to system logs. The data sources specified are accompanied, for each technique, by a textual explanation. Sometimes, also the general description of a technique can be useful to gain information to be transformed in a query.

The question now is: how has this information been used practically? The answer is explained with a simple example. Technique T1053 - *Scheduled Task/Job* occupies the fourth position in the top 10. The sub-technique considered in the report is T1053.005 - *Scheduled Task* and, its textual detection suggestions contain the following sentence: "*Monitor process execution from the `svchost.exe` in Windows 10 and the Windows Task Scheduler `taskeng.exe` for older versions of Windows.*". As an example, the following query has been written for the T1053.005 - *Scheduled Task* technique:

```
SELECT pid, name
FROM processes
WHERE name = 'svchost.exe'
OR name = 'taskeng.exe';
```

This query is very simple and selects pid and name fields from the `processes` table presenting the name of the desired executables. The procedure followed for writing queries of this type is basically the one depicted. Sometimes the process has been quite easy, sometimes a bit more challenging.

Red Canary report

The third and last source of information capitalized in this work, has been the one contained in the aforementioned report by Red Canary. The report in question, thus acquires a double role: it served both to delineate the subset of techniques and, to effectively write the queries for each of them. The second aspect, which is the one important here, is realizable thanks to the good amount of detection indications supplied inside it. Each technique in fact, presents two main sections in the report: *Analysis* and *Detection*. Both the sections are useful to write queries but especially the second. Detection section gives hints about files to check, command line arguments or keywords that may be a sign of application of that technique, suspicious processes or connections that may indicate the technique's application and still strings that, when appearing, are proof of attack or yet folders to monitor. Moreover, suggestions for weeding out false positives are given. Well, a lot of information is available and may be useful to write detection Osquery files. The procedure is quite the same applied for MITRE ATT&CK where, textual information is translated to SQL. Despite this, an example is provided. The query shown below, has been written for T1105 - *Ingress Tool Transfer*. Following the report, in the section *Suspicious commands* it is suggested to check for the execution of `powershell.exe` with command lines containing the following keywords: `downloadstring`, `downloaddata` and `downloadfile`. This query executes these simple passages:

```
SELECT pid, name, cmdline
FROM processes
WHERE name = 'powershell.exe'
AND (cmd LIKE '%downloadstring%'
OR cmd LIKE '%downloaddata%'
OR cmd LIKE '%downloadfile%')
```

5.2.3 Configuration files overview

Now that the methodology employed has been outlined for every of the three sources of knowledge employed, it's possible to describe how the queries written have been organized. As already anticipated in 5.1.1, the queries written have been divided by techniques. The detection queries of each of the 10 techniques, have been brought together in 10 different Osquery configuration files that played the role of Osquery packs. The correspondence is one pack for one technique. As already anticipated in the initial part of this section, this work has been executed for Windows platform. For each technique, three files are needed to perform a log collection:

- `N-TXXXX.conf`: is the query pack, where N is a number between 1 and 10 which reflects the ranking position of the technique, and XXXX are the four digits identifying the technique;
- `osquery.conf`: is a generic configuration file for Osquery, customized to be used with the specific query pack;
- `osquery.flags`: is an Osquery flag file that contains the necessary flags for the proper functioning of the query pack.

As just presented, the project contains substantially three types of files for each technique. Two of these are configuration files doing different things: the first works as a library of queries (a query pack), while the second is the real configuration for the daemon. Third type of file contains flags necessary for the correct functioning of Osquery. Ten of these triplets of files have been created for Windows platform. Properties of these files are explained in the following sections.

Query packs configuration files

These files constitute the real contribution. Each of these files is dedicated to one of the 10 techniques selected. The file contains a variable number of queries which are used for detection of specific technique application.

Each file has two general attributes that are valid for the whole file: the indication of the `platform` (always Windows in this case) and a brief `description` that introduces the technique's ID and eventual sub-techniques' ID. Although the files are called with the name of their major techniques in fact, not always that technique is the one taken into consideration: the report specifies often the sub-technique that has the major impact.

Sometimes two sub-techniques are responsible and in these cases, the file contains queries for both (each description specifies the targeted sub-technique). To be clearer, it is possible to distinguish three cases:

1. The technique specified in the file name is also the one taken into account and for which queries are effectively written (e.g., file `7-1027.conf` contains queries for T1027);
2. The technique specified in the file name is not really the one taken into account because, one sub-technique has a much bigger impact than the others in determining the ranking (e.g., `4-T1053.conf` contains queries for T1053.005);
3. Similar to the second case but, sometimes there is not just one sub-technique majorly contributing to the presence in the ranking but two of them (e.g., `1-T1059.conf` contains queries both for T1059.001 and T1059.003).

This method may appear a bit strange but, it's important to consider that the 2021 report by Red Canary, is the first one written since sub-techniques were introduced in ATT&CK. The imperfection of the solution is acknowledged by Red Canary authors and, specific methodology employed is explained in the document.

The following pseudocode, shows the typical organization of the created query packs.

```
{
  "platform": "windows",
  "description": "Red Canary Threat Detection Report 2021 Top 10 Techniques N: TXXXX (
    TXXXX.YYY).",
  "queries": {
    "TXXXX.YYY_query_A": {
      "query": "...QUERY A...",
      "interval": 1,
      "description": "ATT&CK TXXXX.YYY - ...QUERY A DESCRIPTION...."
    },
    "TXXXX.YYY_query_B": {
      "query": "...QUERY B...",
      "interval": 1,
      "description": "ATT&CK TXXXX.YYY - ...QUERY B DESCRIPTION...."
    },
    ...
    ...
    "TXXXX.YYY_query_W": {
      "query": "...QUERY W...",
      "interval": 1,
      "description": "ATT&CK TXXXX.YYY - ...QUERY W DESCRIPTION...."
    }
  }
}
```

As it's possible to see, after `platform` and `description`, the third and last major field is `queries`. Inside it, as expected, the list of queries is presented. Each query is put inside a JSON object which has three fields and is identified by a name. The three fields are:

- **query**: contains the SQL query;

- **interval:** contains the interval that is applied between two consecutive query executions;
- **description:** contains a brief description of the query, what it searches, why some tables are queried and some other details.

Each query has a simple but specific naming convention: the name of each query is `TXXXX.YYY_query_W` where `XXXX` identifies the four digits indicating the represented technique, `YYY` identifies the three digits indicating the sub-technique (if no sub-technique, `YYY` is simply absent) and `W` is a progressive letter that univocally identifies the query inside the file. Letter `N` above is the number of the technique in the top 10 and, must be imagined as a number that identifies the position of the technique.

Last thing to say about these files is that, for testing purposes, intervals have been set to one second, so each query is repeated every second. Reasons for this choice are outlined in [5.3.2](#).

osquery.conf files

The `osquery.conf` files that are employed here, are very simple if compared to the general description presented in [2.3](#). Each configuration file has quite the same structure. Options set are the same for each technique and are not a lot: `config_plugin` and `logger_plugin` are set to `filesystem` which is the default value. The first of the two is the method that permits to read the configuration while, the second specifies the fact that the output logs are written in JSON. There was no need of different and more sophisticated settings but, in any case other configurations are possible. After the plugins, `schedule_splay_percent` is set to the value of 10. This value indicates that the execution interval of the queries, may variate of a percentage of 10 from the interval value set. This option is used because, when in the same configuration many queries are executed with the same time interval (like in this case), the user might not want them to be executed all precisely at the same time for performance reasons. The last option set is `utc` that is set to `true` but this, is just a timing convention option and does not affect a lot the daemon execution.

After the options, a second block may be present or not. This block is called `file_paths` and contains a simple comma-separated list of folders. This option is present just in case, queries to the table `ntfs_journal_events` are present in the query pack. The aforementioned table, is an evented table that checks if any modification happens to the filesystem (file creation, deletion and so on) for the specified folders. The list of folders specified is then fundamental to permit those queries to work and produce logs. When that specific table is not queried, `file_paths` is omitted in the configuration.

The third block that is instead always present, is the `packs` block that specifies each time a query pack from which the queries are read. In this case, it is sufficient to specify the name of the pack and the path where the pack file is located in the filesystem. By default, packs are put inside a folder called `packs` that is inside the `osquery` main folder.

Last thing that is present in the configurations created and that is the same for each of the techniques, is the `decorators` block. In this case, decorators have been limited to a couple of queries that, are respectively directed to `system_info` and `logged_in_users` and both simply select a field. First decorator selects `uid` of the host while, second selects `username`. This information is always added to the logs collected.

osquery.flags files

The last file that is needed to run correctly the Osquery daemon is the file containing the flags. Osquery flags can be used with both the `osqueryi` mode, specifying them in the command line, or with `osqueryd`, thanks to the *flagfile*. Some flags can also be specified in the *configuration* options. There are many flags and, all of them are specified well in the Osquery docs. The focus here, will be put just on the flags used for these configurations.

Firstly, it is important to say that the flags used for this project, are all related to the Osquery evented tables. Evented tables are different between the various platforms; in Windows they are five: `example`, `ntfs_journal_events`, `powershell_events`, `windows_events` and `yara_events`. Without going into detail, it is possible to say that the two that are interesting for the purpose are, the ones related with `ntfs` and `powershell`. Evented tables are useful for auditing some specific actions that happen on the system. Normally, osquery tables are generated each time the query has to be executed and, this permits to retrieve information available in that precise time. Instead, evented tables are different because, when enabled, they collect data from the OS asynchronously with respect of the query execution. To understand the difference between normal tables and evented ones, an example can be done. Processes executions are taken into account. If the interval set for a process monitoring query is something like 10 seconds, and the query has just been executed, an hypothetical process that starts after two seconds from the query execution and is alive for 5 seconds, will not reach the following query execution. It will die in fact, 3 seconds before that moment. When query executes, it will not know anything about that process existence. This makes the data retrieval, in some specific situations, lossy and ineffective. If instead, the query for process monitoring is addressed to an evented table for process monitoring, each process execution is saved asynchronously in Osquery backing store and each time the query gets executed, it will retrieve information about processes from that storage. The difference is that while normal Osquery's tables are generated when data are requested synchronously, evented tables progressively collect and save data and, once queried, data is retrieved from there. The process mentioned before, in this case will be easily retrieved from the backing store.

The `powershell_events` evented table, when enabled, registers all the commands that are executed in Powershell instances. Script blocks are reconstructed to their full script content. The `ntfs_journal_events` evented table instead, is useful to track changes (like creation, modification, deletion of files or folders) of the filesystem, limited to specified paths. The `file_paths` specified in the configuration file, are used to specify which directories have to be monitored. In this case all the changes in those directories are put in the table and logged when related queries execute.

Now that evented tables have been introduced, the core of this section can be explained easily. A total of five flags are used in the various configurations, the first four listed here are always present while, the last is present just when `ntfs_journal_events` needs to be queried:

- `-disable_events=false`: used to enable events, giving the possibility to query evented tables;
- `-enable_windows_events_publisher`: this is used to enable Windows events that in this case, are useful to publish Powershell events;

- `-enable_powershell_events_subscriber`: this definitely enables events for Powershell;
- `-windows_event_channels=Microsoft-Windows-PowerShell/Operational`: this flag is used to subscribe to windows published events for Powershell logging;
- `-enable_ntfs_event_publisher`: enables the events for the ntfs;

To sum up, first flag is used generally to enable events, the three in the middle for Powershell logging while the last for ntfs logging. This was the last ingredient necessary for the correct working of the osqueryd daemon. All these files are read when daemon is executed.

5.2.4 Problems faced and drawbacks

Despite the limited list of techniques that have been interested, query writing process has been quite long and difficult. Written queries have been all tested and, the majority of them give results in the tests that will be explained in 5.3. Some queries do not log results because the aspects they are searching for, are simply not touched by those tests. In any case, in general they find the expected results. For some techniques, information given by the sources was more complete and accurate, so it has been easier to write a larger number of them. Moreover, when better information is given, more powerful and effective queries can be written. Techniques that come with less detection suggestions, may have been equipped with less queries, but the results are in general satisfactory.

5.3 Query testing

The testing activities have been performed to evaluate the work carried out with the combination of MITRE ATT&CK and Osquery. The best way to test the query packs, was trying to apply the techniques included in the report and see what happened with the detection. Atomic Red Team fitted perfectly this job. The library in fact, gives the possibility to test each technique atomically, isolating it from the rest of the actions. The atomic testing was performed in a Windows environment and, different log collections were created. The following chapters will explain the whole process.

5.3.1 Test environment

The test environment inside which the tests have been performed, is a *VirtualBox*⁵ Windows 10 virtual machine. More precisely, *Oracle VM VirtualBox* was installed in its version 6.1.16. VirtualBox is a free and open source software that supports the execution of virtual machines. In this case tests were done on a Windows 10 guest, with a Windows 10 host. The VM on which the work was done was a ready to use one, directly supplied by Microsoft⁶ as a browser testing environment. Apart from the last versions of the browsers installed in fact, the VM was a normal Windows 10 one. This gave the possibility to install

⁵<https://www.virtualbox.org/>

⁶<https://developer.microsoft.com/it-it/microsoft-edge/tools/vms/>

the software needed and to perform the tests. The Invoke-Atomic module was installed together with the Atomic Red Team library folders containing tests. Osquery was installed from the Chocolatey⁷ packet manager because, this was the recommended way.

5.3.2 Clean log collection

The first log collection activity performed, has been labeled with *clean* adjective. This because, being the VM on which the work was being executed quite new and not used for dangerous activities, it has been supposed to be clean. Clean is intended as not affected by viruses or any other kind of malware. The clean log collection, was necessary to have a picture of the normal activity of the system. This would have been useful then, to make comparisons between logs collected during normal system activities and logs produced when under attack. The clean collection was executed in two steps. Firstly, a technique targeted collection was performed for each technique from the first to the tenth. This meant that Osquery was set each time with the three configuration files explained in 5.2.3. Starting from the first technique, configuration files were set in the `osquery` folder and then, Osquery itself was launched from Powershell (this was the choice but, also Command Prompt launching was absolutely working).

An important aspect to underline, is the one about the interval set between two executions of the same query. Osquery leaves the user, the possibility to set a specific interval in seconds, for each query. In this tests, interval has been set to one second for each query, meaning each of them is executed one time every second. This may be a problem if a great number of queries must be executed in a not very powerful environment but in this case, maintaining a short interval permitted to collect a good number of logs in a not excessive amount of time, without giving performance problems.

Log collection for each technique, lasted the time necessary to harvest a number of logs that was considered sufficient for the following analysis. Logs collected would have been used both for manual comparisons between clean and dirty logs, both for training of the ML algorithm discussed in the second part of the thesis. An example of innocuous log collected with pack 1 is presented here:

```
{ "name": "pack_1-T1059_T1059.001_query_C", "hostIdentifier": "MSEdgeWin10",
  "calendarTime": "Sun Feb 13 22:03:15 2022 UTC", "unixTime": 1644789795,
  "epoch": 0, "counter": 0, "numerics": false,
  "decorations": { "host_uuid": "ACB3DB84-69C8-A346-A98E-977175C2273F",
    "username": "IEUser" },
  "columns": { "name": "svchost.exe", "path": "C:\\Windows\\System32\\svchost.exe",
    "remote_address": "20.54.36.229", "remote_port": "443" }, "action": "added" }
```

During probe working period, the system has been used in a way targeted at simulating everyday actions of a generalist user. More precisely, things like visiting different websites through browsers, writing and saving documents, changing system settings, opening and using installed applications were done. These are just examples of action but give back an idea of what is intended for normal use. This work has been achieved once for each of the techniques and at last, the same job has been executed also with a custom configuration

⁷<https://chocolatey.org/install>

that included all the query packs written and options suitable for all techniques. This second step, the comprehensive collection of logs generated from the total of the query packs, was done in order to try to perform a single training phase and then test techniques singularly. This however, is better explained in the second half of the thesis. Results were as usual, contained in the `osquery.results.log` file. Last thing to say: these collections were all performed with Windows security *Virus & threat protection* enabled, like an average user should have set on its system.

5.3.3 Dirty log collection with Atomic Red Team

The very last piece of contribution related to the first part of this thesis, consists in the already anticipated execution of the Atomic Red Team tests. These were taken as a final experimental trial to validate the quality of the written queries. For each technique or sub-technique employed, the whole tests available were performed. Before the definitive execution, some random trials were performed and it appeared clear that, the largest part of the tests were not working with Windows Security antivirus enabled.

To get meaningful results in the end, it has been necessary to deactivate the *Virus & threat protection* of Windows. This left the possibility to execute a greater number of tests successfully, contributing significantly to the completeness of the work.

Once the clean log collection was finished, the same job of log collection was performed again, technique by technique, with the only difference that now the Windows threat protection was disabled.

Thanks to Powershell, Osquery daemon was launched, and logs started to be collected one after the other. During the accumulation, thanks to two other Powershell instances, the various tests were performed. Two shells were used, one with administrator privileges, one without them. This happened because, some tests can be easily executed with a normal Powershell while, other tests require the privileges of a super user, otherwise they are not working properly. All the tests were performed at least for one time, but sometimes also more than one. To be more precise, it's sufficient to take an example. Let's take technique number 5, T1003: in this case, the sub-technique T1003.001 is taken into account and Atomic Red Team makes available thirteen tests for it. Tests are executed one after the other, and each time a test does require a cleanup, this is performed before execution of the following test. In this manner, all the possible ways to use various queries, searching for anomalous behaviours in the system are triggered. A suspicious log collected during tests for technique ranked fifth in the report, is shown here as example:

```
{"name":"pack_5-T1003_T1003.001_query_E","hostIdentifier":"MSEdgeWin10",
"calendarTime":"Mon Feb 14 16:36:29 2022 UTC","unixTime":1644856589,
"epoch":0,"counter":5,"numerics":false,
"decorations":{"host_uuid":"ACB3DB84-69C8-A346-A98E-977175C2273F",
"username":"IEUser"},
"columns":{"cosine_similarity":"0.0","datetime":"2022-02-14T16:36:25.665751600Z",
"script_block_count":"1","script_block_id":"78257b1e-2ef7-4561-87ef-120dd3b90d78",
"script_name":"","script_path":"","
"script_text":"& {IEX (New-Object Net.WebClient).DownloadString
('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/
f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/Invoke-Mimikatz.ps1');
Invoke-Mimikatz -DumpCreds}","time":"1644856588"},"action":"added"}
```

The log just presented is suspicious because, this query does not log in standard situation, but just under testing. Moreover, it is possible to see that the field `script_text` contains invocation of the famous *Mimikatz* tool for credential dumping, that is likely to be used by adversaries. This stuff, executed in a Powershell instance is a clear sign that something wrong is going on. Logs like this are consequently saved and, dirty results are ready to be compared with the clean logs. This simple procedure has been repeated for each case.

This concludes the contribution explanation and creates conditions for final evaluations of the work that will be presented in section [7.1](#).

Chapter 6

Contribution - Threat Hunting

In this section will be analysed the OpenSearch¹ tool, which is the framework selected as centralized logs management system. In other words, is the SIEM platform chosen for this work. An overview of the stack composing OpenSearch is given, and why it was preferred with respect to the platform from which it was forked. Finally, the main characteristics and components of OpenSearch are presented, in order to provide an introduction to the actual tool.

In order to go into details about OpenSearch project history and the background split from Elasticsearch Appendix A. Also an overview of the lawyerly dispute involving Elastic company and Amazon is offered, with an explanation behind license change decision, in Appendix B.

6.1 The ELK Stack

This section analyses the ELK stack in detail. Notice again that OpenSearch and OpenSearch Dashboard were born from a fork of Elasticsearch and Kibana, respectively, both at version 7.10. It is, hence, necessary provide a focus on the stack main components.

6.1.1 Elasticsearch - OpenSearch

*Elasticsearch - OpenSearch*² is the Database Management System (DBMS) of the stack. In particular, it is based on Apache Lucene, which is an open source search engine software library based on Java; indeed Elasticsearch is categorized as a “Search Engine”. This technology consists of a non-relational, non-SQL structure that is optimized for search operations, including full text search, ranking and grouping of search results and high scalability [71].

¹<https://opensearch.org/>

²<https://opensearch.org/docs/latest/opensearch/index/>

DB-Engines³ is an initiative that collects and monthly ranks DBMS by popularity and usage. Also, this website collects principal characteristics of many database management systems. In Figure 6.1, DB-Engines ranks the ten most popular DBMS in the entire world, updated in March 2022. As it can be seen, Relational databases still are the most used technology by far, for both historical and organizational meanings. Anyway, Elasticsearch is placed on 8th position highlighting that is the most popular Search Engine, with a wide usage and able to enter in top-10 rank of DBMS. As a confirmation to the general rank-

388 systems in ranking, March 2022

Rank			DBMS	Database Model	Score		
Mar 2022	Feb 2022	Mar 2021			Mar 2022	Feb 2022	Mar 2021
1.	1.	1.	Oracle	Relational, Multi-model	1251.32	-5.51	-70.42
2.	2.	2.	MySQL	Relational, Multi-model	1198.23	-16.45	-56.59
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	933.78	-15.27	-81.52
4.	4.	4.	PostgreSQL	Relational, Multi-model	616.93	+7.54	+67.64
5.	5.	5.	MongoDB	Document, Multi-model	485.66	-2.98	+23.27
6.	6.	7.	Redis	Key-value, Multi-model	176.76	+0.96	+22.61
7.	7.	6.	IBM Db2	Relational, Multi-model	162.15	-0.73	+6.14
8.	8.	8.	Elasticsearch	Search engine, Multi-model	159.95	-2.35	+7.61
9.	9.	10.	Microsoft Access	Relational	135.43	+4.17	+17.29
10.	10.	9.	SQLite	Relational	132.18	+3.81	+9.54

Figure 6.1. DB-Engines, DBMS popularity ranking accessed in March 2022.⁴

ing, the Figure 6.2 shows the popularity by considering only the Search Engine category, collected in a logarithmic chart. As it can be seen, Elasticsearch is the leader in its landscape, since 2016. On the other hand, OpenSearch was registered in DB-Engines recently and appears in the chart for the first time in March 2022 update. Despite being barely inserted in the scoring system, OpenSearch is actually placed on 10th position. This work and their authors believe in OpenSearch project, and they bet on it, hoping the software will escalate positions in near future.

Since OpenSearch was introduced recently, and it was forked by a quite recent version of Elasticsearch, meaning that they have the same features, the name E/OSearch will be used to address to both software.

E/OSearch can be used to perform full-text searches, to query specific fields, sort and aggregate results. Its storing and indexing engine is very efficient and thought to properly scale when needed, according to system requirements. E/OSearch was designed to work compatibly to a distributed environment, indeed is able to work in cluster, which is a collection of nodes for storing data and process requests. There are multiple advantages, for a deployment environment, to leveraging cluster capabilities; one of the main reason is the ability to perform *Load Balancing*.

Being a non-relational DBMS, in E/OSearch, the base entity used to organize the storage

³<https://db-engines.com/en/>

⁴<https://db-engines.com/en/ranking>

⁵https://db-engines.com/en/ranking_trend/search+engine

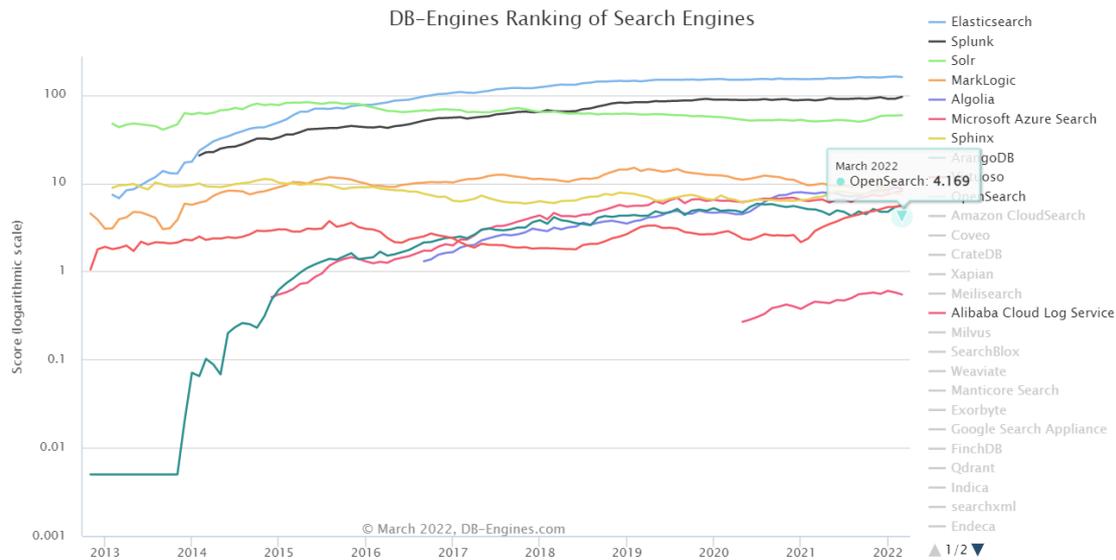


Figure 6.2. *DB-Engines, Search Engine popularity ranking accessed in March 2022.*⁵

are emphindices. Indices allow the engine to efficiently perform look-up operations in order to satisfy incoming queries; they are composed by collections of *JSON documents*. When indices become too big, E/OSearch splits them into *shards*, dividing the workload from one to the needed number of nodes. Furthermore, by default, the framework creates a replica for each primary shard. Even if more space is required, replicas can act as a backup, in case a node fails, and speed up search processes. The communication with E/OSearch can be achieved with REST API, offering great flexibility.

6.1.2 Logstash

Logstash⁶ is a real-time event processing engine. In other words, it can receive data from different sources, process them eventually applying some filter or by adding metadata and then delivers them to one or more destinations. It can be used to process any type of events, in particular JSON and XML format.

Logstash basically has the same behaviour of a pipeline. There are three stages, where data are received in input, filtered and sent in output. These stages work with plugins, which are scripts designed to work with Logstash and with the corresponding software they are referring to. E. g., in this work will be used the OpenSearch Dashboard plugin that allows to send received data in output converting them into JSON format, if compliant with the standard.

In the Figure 6.3, it is shown a sample template with the three stages of the pipeline. In particular:

⁶<https://opensearch.org/docs/latest/clients/logstash/index/>

- *input*: section to specify input plugins; there are many integration available, working both on TCP and UDP protocols
- *filter*: data are parsed, leveraging plugins to filter, adding or removing fields to collected data e.g. adding a label or a data to a log
- *output*: where are specified the destination of the collected data; as the input section, also here are supported both TCP and UDP, depending on the wanted plugin

```
input {
  input_plugin => {}
}

filter {
  filter_plugin => {}
}

output {
  output_plugin => {}
}
```

Figure 6.3. Logstash configuration file empty template.

In this work, Logstash is used to aggregate data coming from user endpoints, eventually filtering them and performing delivery to the visualization engine. This also means that user endpoints are charged-off the task of indexing their data, hence they just need to send to Logstash all meaningful information.

6.1.3 Kibana – OpenSearch Dashboard

OpenSearch Dashboard was forked by a Kibana distribution; indeed they share many features. For this section, the name K/D refers to both software.

K/D offers a user interface through the server hosting the service, at port 5601. Other than offering the possibility of creating custom dashboard to visualize collected data, K/D can be used to set alerts, install plugins, create reports and supports a language to query E/OSearch engine.

K/D tool is able to graphically show indices data, searching from some JSON field and modifying the time window of visualization allowing log analysis. These are only some of the powerful capabilities of K/D framework, which may be used as an application with SIEM capabilities. In the Figure 6.4 it is shown a sample dashboard provided by OpenSearch to present its features. In this work, OpenSearch Dashboard receives collected logs by Logstash and shows them in graphs. A custom Dashboard, joining different graphic components, can be created. If the stack is able to deliver correctly formatted data, the K/D

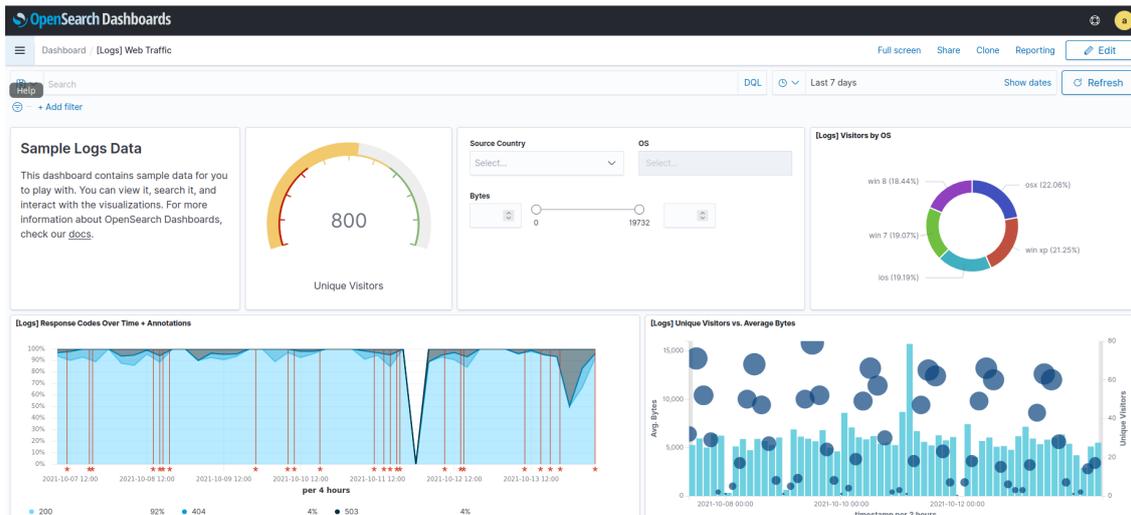


Figure 6.4. OpenSearch sample dashboard provided during the installation as example.

is able to automatically parse JSON documents and aggregating them for a ready-made visualization.

6.1.4 Why OpenSearch

The open source license of OpenSearch was not the only reason that influenced the choice of the tool. Indeed, Amazon released its product including many features that previously were not available in Elasticsearch free version but were provided with a subscription. These functionalities were derived by an already existing project, named *OpenDistro for Elasticsearch*, which gathered a set of plugins with the goal of offering advanced features in open source solutions. In Figure 6.5 are shown some of the implemented plugins, which now the main ones will be presented, as done by Chunny Yun in an article in early days of OpenSearch [101]:

- *Advanced security*: to provide encryption, authentication, authorization and audit. This plugin allows to support different type of authentication, such Active Directory, LDAP, Kerberos, JSON web token and others; furthermore, it allows role-based access control in order to managing user permissions at indices, document and even JSON field level.
- *SQL Query Syntax*: to expose the database engine and make it available through SQL syntax. Documents are read ensuring the great flexibility of SQL, making easier to query the system leveraging a well-known language.
- *Reporting*: to schedule, export, and share reports from dashboards, saved searches, alerts, and visualizations.
- *Anomaly Detection*: to perform anomaly detection using Machine Learning capabilities based on Random Cut Forest, an algorithm to automatically detect anomalies

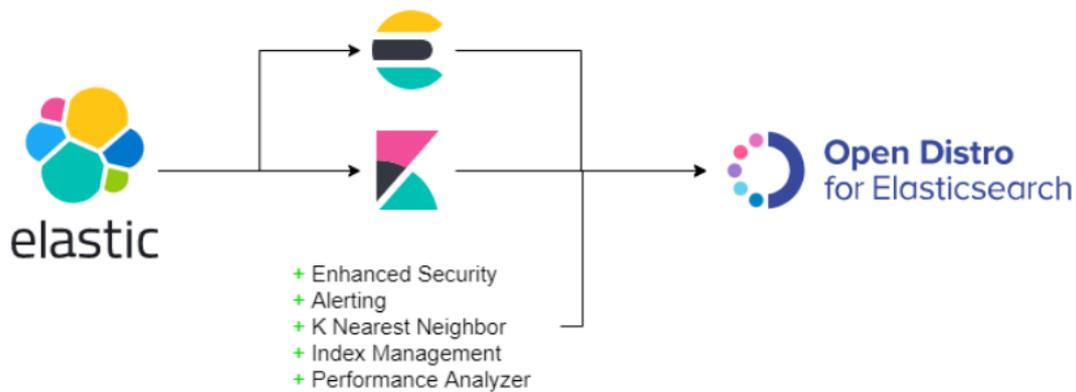


Figure 6.5. OpenDistro for Elasticsearch, a project to add advanced capabilities to the old open source Elastic stack. [86]

in ingested data. Despite this great functionality, this work focus on the user system hence capabilities of this plugin are not exploited; anyway it can be a solid base for future works.

- *Index Management*: to define policies to facilitate indices management introducing routines.
- *Alerting*: to automatically monitor ingested data and to raise alert in case some boundaries are not respected. This plugin offers an interface, on OpenSearch Dashboard, that can be used to set custom values that OpenSearch engine will ensure to check and, eventually, signal.
- *Dashboard Notebooks*: a set of tools useful for data visualization, that allows to enhance data analysis, reporting and results explanation.

The majority these features, previously the license change, were provided in Elasticsearch through its X-Pack component; it was a built-in module under Elastic License, hence a functionality that “stained” the open source project. With OpenSearch, these functionalities are already included, and X-Pack was removed after a long process of code cleaning.

All these additions through open source plugins and the license change lead this work on choosing OpenSearch. The community using Elasticsearch has now to decide if Elv2 license is suitable for their business or migrate to the competitor. There is not a right choice but granting full freedom to customer and having Amazon as main contributors is an insight of the success that OpenSearch may gain in next few years.

6.2 Filebeat

Generally speaking, Beats are a family of lightweight log shippers, designed to collect different kind of logs. They are installed on the end system and act as agents, collecting

data and shipping them to the centralized system, represented by Logstash, Elasticsearch or to a broker such Apache Kafka.

They are developed and maintained by Elastic company; the license change affected them as well, hence they are not covered anymore by Apache License since version 7.10. OpenSearch is fully compatible with them, since its Elasticsearch core nature, until the specified version. For further versions, Amazon released a compatibility matrix ⁷ that can be used as a guideline to understand which version is suitable to download. The compatibility and correct Beat configuration is crucial and can be one of the most time consuming operations, when facing with troubleshooting. So, it is necessary to pay particular attention during the setup.

There are two kind of Beats that suit for this work purpose:

- **Winlogbeat**⁸: created to ship Windows events to ELK stack and can be installed as a Windows service. This Beat is able to capture data from any Windows event log, it can filter and send them in output. A configuration file specifies which event log are required to be captured and if there are some filter operations to perform.
- **Filebeat**⁹: created to monitor log file or locations specified in configuration file and deliver data to the ELK stack. Filebeat is able to collect info from any type of log file, not regarding its format. It is, hence, the Beat chosen every time there is not a specialized Beat, such Winlogbeat for Windows event logs, and often it is used to collect results of another process and shipping them in output to ELK.

Despite Winlogbeat is able to collect any kind of windows log, a proper configuration is not an easy task. Furthermore, data collected are often rich of metadata that not useful to monitoring goals. This work is focussed on collecting log with a very high precision and not only limiting the monitoring to Windows events. For this reason the Beat chosen is **Filebeat**, which collects output results coming from Osquery (see Chapter 5) and ships them to the Logstash plugin for OpenSearch in near-real time.

The Osquery output is collected into an output file, that is monitored by **Filebeat**. To avoid shipping many time the same logs, the agent keeps track of the file offset. In Figure 6.6 is shown the functionality of Filebeat, obtained by the original documentation [36]. Essentially, it has two component: **inputs** and **harvesters**. **Inputs** are declared in the configuration file, and they contain the necessary information to find files required for monitoring and their type. **Harvesters** are driven and managed by inputs; they are the processes that read each file line by line. In particular one Harvester is instantiated for each log line.

Harvesters keep the file descriptor open for all the time they are running. This means that if the file is deleted or renamed, the memory is not freed until the last harvester ends. On the other hand, a renamed file continues to be read without interruption.

⁷<https://opensearch.org/docs/latest/clients/agents-and-ingestion-tools/index/>

⁸https://www.elastic.co/guide/en/beats/winlogbeat/7.10/_winlogbeat_overview.html

⁹<https://www.elastic.co/guide/en/beats/filebeat/7.10/filebeat-overview.html>

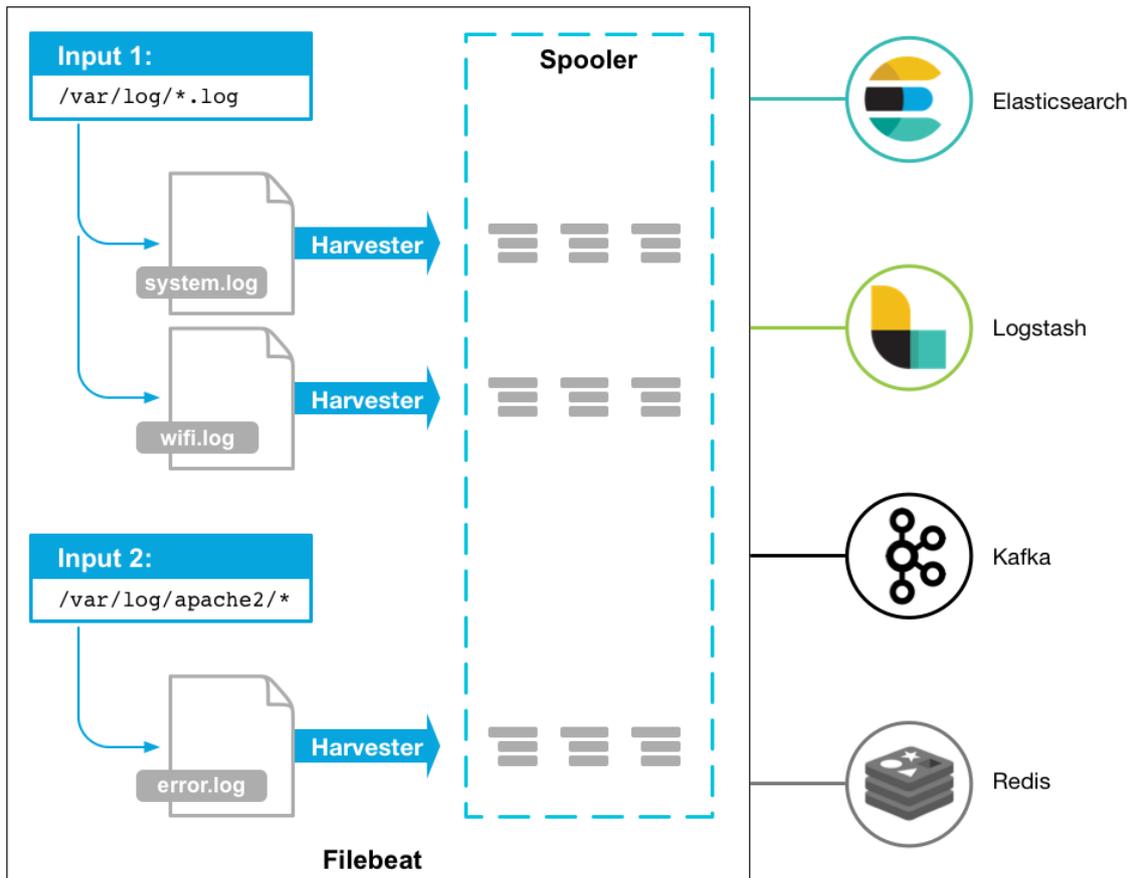


Figure 6.6. Filebeat working flow.

For each file, a status structure is stored containing the offset of the last harvester. If the destination program, Logstash in this work, is not reachable, Filebeat tracks also the last line shipped and when the recipient returns online all logs collected in the meanwhile are sent. Using these strategies, it is possible to ensure that logs are sent only once, without repetitions or ignored lines.

6.3 Docker

Docker¹⁰ is an open platform for developing and launching software. It runs upon every operating system, from natively on Linux through a CLI to Windows through Docker Desktop application. The client, both desktop application and command line interface, allows to manage the whole lifecycle of the unit component of Docker infrastructure, which is a *Container*.

¹⁰<https://www.docker.com/>



Figure 6.7. Docker logo.

A Container is a cluster running an application and contains everything is needed to run that software; being self-contained, a container can run upon any operating system, regardless Windows or Linux, and ensures the same functionalities. This is possible thanks to *Sandboxes*, which are environment providing security and isolation to the single application that runs transparently enclosed in its virtual box; this behaviour is achieve leveraging existing primitives, such cgroups and namespaces.

A Container is a lightweight environment that allows a highly portable development and deployment, being able to run on almost any hardware without compatibility issues; from a developer laptop to a physical or virtual machine in a data centre and even on cloud providers. The lightweight nature of a Container allows a dynamic resource management and workload distribution, allowing efficient scaling up and down the infrastructure in almost real time without consuming extra resources. Docker permits to build infrastructure of applications in a way much more convenient, in terms of performance and required resources, with respect to traditional full-virtualization solutions such VMware and Virtual Box; indeed many containers can run on the same physical host, without significant overhead. Saving resources is synonymous of cheaper deployment and running environment, reducing costs with respect to traditional development framework.

6.3.1 Docker architecture

The section will analyse the Docker architecture, giving an introduction on the main components.

The Docker framework, [25], is managed through a client-server architecture. The Docker client communicates with Docker daemon, the server, through REST API interfaces; the daemon is in charge of instantiating and managing container lifecycles. Client and server may be on the same machine, or on different one and communicate remotely; in this case, may be needed an orchestrator to manage a pool of containers, such Docker Swarm or Kubernetes.

As shown in Figure 6.8, are now presented the main components:

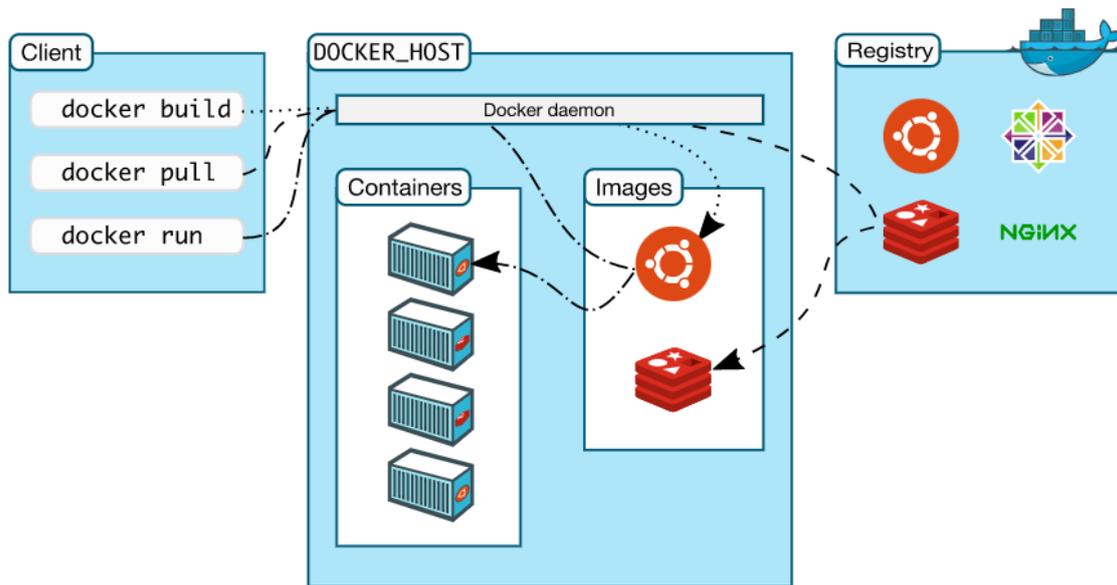


Figure 6.8. Docker architecture example.

- *Docker Daemon*: it listens for requests coming from the client in order to manage images, containers, networks and volumes
- *Docker Client*: it is the user interface for Docker. Commands are launched from the client, which interacts with one or more Daemons to perform operations
- *Docker Desktop*: a desktop application that enables Docker on Windows and MAC operating systems. It contains all component and engines to build, develop and share Containers and microservices
- *Docker Registries*: a registry is a remote repository that allows to manage Containers in a way similar to git version control. Images can be pulled, to store them locally, and pushed, to update them. By default, the registry is Docker Hub, but other registries can be used
- *Images*: it is a read-only template with instruction for creating Docker Containers; when a container is run, its corresponding image is used to build and launch the application. An image is built on top of others, creating a layered infrastructure where every layer contributes with some feature. For example, an application requiring a Linux file system may be built using as base an Alpine distribution. Being read-only an image cannot be modified but can be extended adding components with the result of creating a new one. When a new image is added, only new features are considered so that common functionalities are not considered multiple times; this approach is shown in Figure 6.9 and is named *Union File System* and allows to keep as low as possible the disk space required by an image. Notice that the Kernel must be shared among all other layers and the top one is the only writable one, where the actual container runs.

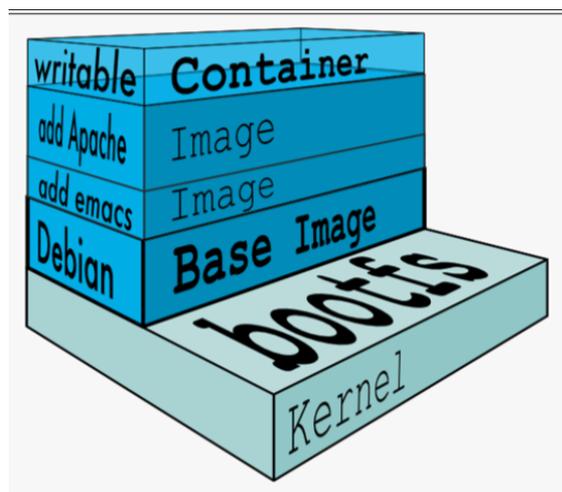


Figure 6.9. Docker Union File System example.

- *Containers*: they are runnable instances of an image; they can be run, stopped, paused, moved and deleted. Containers are, with a fair amount of approximation, like lightweight virtual machines, meaning that each instance is relatively isolated, can be connected to different networks, has its own file system and allocated resources. A Container is defined by its associated image configuration. A container does not persist any data or information when it ends, unless explicitly allowed and changes are stored in the persistent storage. Containers are automatically built through particular files, named “*Docker File*”, which contain all images, commands, volumes and in general configuration that the application requires to run. Another option is to use *Docker Compose*, which is a tool for running multi-container cluster application to run all services from the same configuration file in a single command. Docker Compose uses a YAML file containing all required containers
- *Docker networks*: by default, Docker connects all Containers to a bridged network, named docker0 bridge, automatically managed with its IP address space and routing tables. It is possible to create ad-hoc network for multi-container applications, to provide a better isolation. NAT is used to provide outbound connection, while internally are used the “classic” networking rules with virtualized devices such virtual switches.

In this work, Docker is the core component; indeed it is used both on server side and end user device. On the server, OpenSearch infrastructure is built using Docker Compose tool that launches all container and creates a dedicated network. In this way, Containers can be managed as a cluster in simple way leveraging Docker APIs, allowing troubleshooting and an easy management of the whole application. On the endpoint, instead, Docker client is installed, and it is used to manage the Filebeat Container lifecycle.

6.4 Virtual Box



Figure 6.10. Oracle Virtual Box logo.

The last piece of the architecture is *Virtual Box*¹¹. It is a tool developed by Oracle that provides a virtualization management environment; with it is possible to run virtual machines in a full-virtualized environment. Hence, it allows to run any kind of operating system in a virtual and isolated environment giving access to the machine as it was a physical one.

Thanks to Virtual Box capabilities, it is possible to create a virtual environment where OpenSearch framework is installed on a remote virtual Ubuntu machine, with Docker running on it, and interacts with the physical Windows operative system of the host. To achieve this, Virtual Box enables the creation of a virtual network between the virtual machine and the physical one; the application offers different network configuration, as shown in Figure 6.11 In this work, the Ubuntu virtual machine has two configured network

Network type	Access Guest -> other Guests	Access Host -> Guest	Access Guest -> external Network
Not attached	-	-	-
Network Address Translation (NAT)	-	-	✓
Network Address Translation Service	✓	-	✓
Bridged networking	✓	✓	✓
Internal networking	✓	-	-
Host-only networking	✓	✓	-

Figure 6.11. Table showing all different Virtual Box network type and which type of communication each type allows to achieve.

interfaces:

¹¹<https://www.virtualbox.org/>

1. *NAT-type* allowing outbound internet connections mainly for downloading images and installing all required applications. Accordingly to Figure 6.11, this configuration is not suitable for communication between VMs and the guest system.
2. *Host-only* networking allowing bidirectional communication between the virtual machine and the Windows guest operating system. This interface is used to ship all log collected, from the physical machine to OpenSearch stack.

The double network interface was preferred with respect to the single *Bridged* one in order to provide isolation to the VM by inbound communications. In this way, the internal network cannot be reached directly by the internet since only outbound connections are available through NAT protocol.

Before discussing the proposed architecture, in Figure 6.12 is summarized the environment.

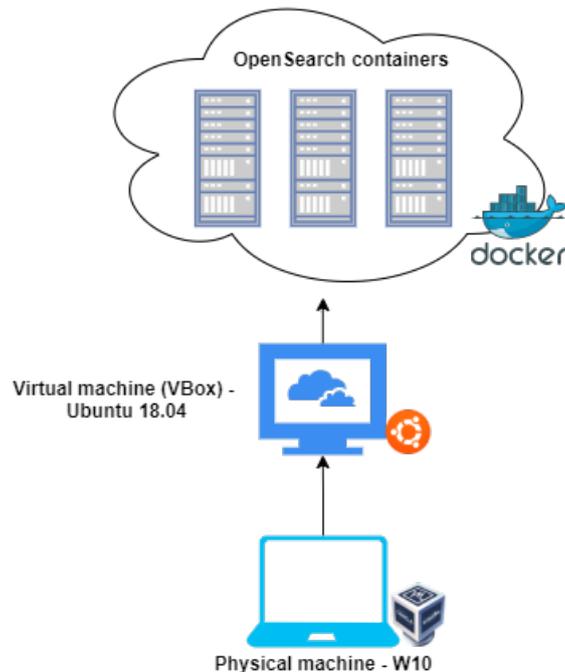


Figure 6.12. Overview of the final infrastructure, highlighting the logical connection between OpenSearch stack, Virtual Box and the guest machine.

6.5 Proposed Architecture

In this section is analysed the architecture in detail, starting from a general overview to client and server side.

In order to simulate a client-server architecture in a local environment, it was chosen a virtualization engine, Virtual Box, installed on the physical machine, where resides Windows 10. Through Virtual Box it is created an Ubuntu 18.04 virtual machine, where all the software required to run the OpenSearch stack will be installed.

On the virtual machine:

- Docker engine was installed.
- OpenSearch containers are downloaded.
- A Docker-Compose file, that allows to launch the cluster at the same time, has been created to simplify the management of the infrastructure.

On the host machine, it is installed Osquery daemon that runs as a Windows service, collecting all logs that match rules specified in the Osquery configuration file and stores them in a text file.

Installed into the Windows machine, along with Osquery agent, can be also found the Machine Learning algorithm proposed by this service, which can run as a Windows service or installed in a Docker container. The algorithm process takes in input the text file generated by Osquery, containing all collected logs, and, by analysing it, will produce another text file containing all events that are evaluated as malicious.

The final piece of this pipeline is performed by Filebeat. It takes all files specified in the input plugin section, inside its configuration file, and ships them to the virtual machine.

The connection schema is summarized in Figure 6.13. Also, notice the two network interfaces virtualized by Virtual Box: the NAT Network allows outbound communication, providing internet connection to the VM for component installation, while the Host-Only Network makes possible the bidirectional communication between the host Windows OS and Ubuntu. This double network allows to have the VM unreachable by external connections, thanks to NAT protocol, and allows communication between the client-server actors.

Considering the physical machine, the architecture is illustrated in Figure 6.14.

Osquery reads its configuration and queries the operating system to obtain logs. All results are stored in `osqueryd.results.log` file, which resides in **C:**

Program Files

osquery

`log` directory. BERT algorithm should be installed as a service or inside a Docker container and run, analysing the logs published by Osquery service. If logs considered malicious are detected, the algorithm generates a file containing a summary of the possible threats, which is suitable to be shipped to OpenSearch.

Finally, Filebeat contains the name of file that must be monitored, which can be the BERT output file, or some given path containing malicious collection of logs. Filebeat can be installed both as a Docker container and as a Windows service. The configuration chosen for this testbed is by Docker, that simplifies the installation of the service by instantiating a container that can be easily managed; furthermore, if some heavy troubleshooting is required, Docker simply allows to destroy the container and creating a new, and clean, one. Filebeat ships the monitored files to the VM using the Logstash plug-in.

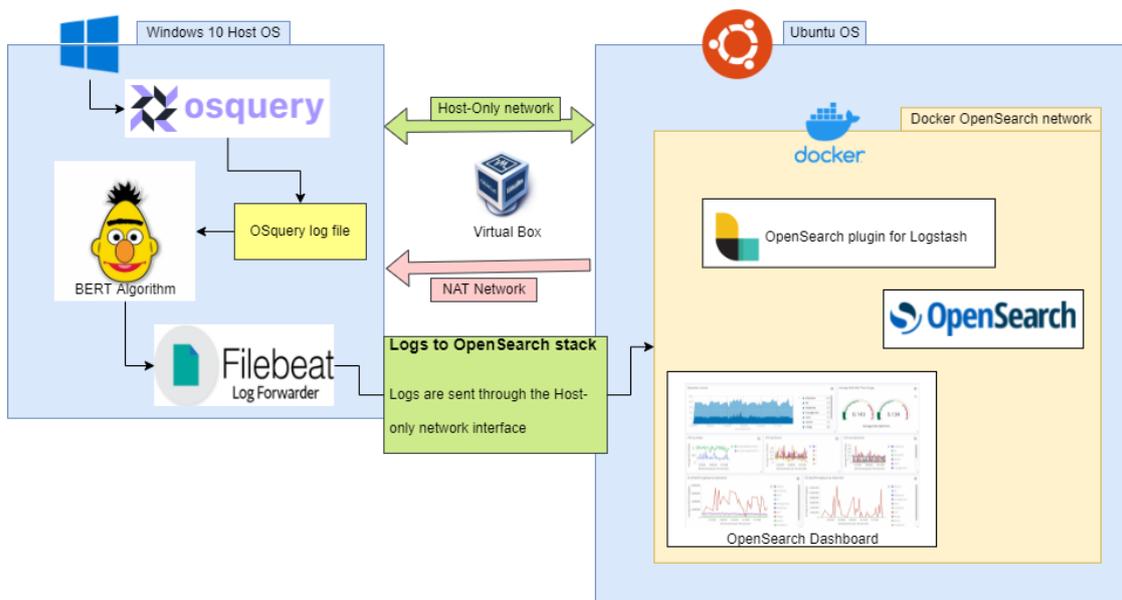


Figure 6.13. Schema of overall system architecture.

It is important to notice that also OpenSearch Dashboard can be remotely accessed using, by default port, 5061. By opening a browser and navigating to the address shown below, it is possible to remotely manage and visualize the OpenSearch Dashboard.

URL example -> `http://your-UbuntuVM-IP-address:5061`

Moving to the VM, Figure 6.15 summarize the software installed on it. Docker is the core of the architecture, offering the possibility to easily manage the whole lifecycle of OpenSearch stack. A Docker-Compose file is used to setup the environment with a single command, allowing the creation of a cluster for the application.

The three containers are connected through a custom user bridged network that, accordingly to Docker documentation ¹², offers additional features with respect to the default Docker bridge such:

- *Automatic DNS resolution between containers*, that using their aliases are able to communicate
- *Better isolation*, allowing only connected container to communicate within the network
- *Flexible connection*, allowing disconnection and attaching of containers to custom network while they are running, without need to stop them

OpenSearch Dashboard and Logstash need each one to expose a port, in order to be reachable from outside the bridged network. Docker achieve this by exposing directly on

¹²<https://docs.docker.com/network/bridge/>

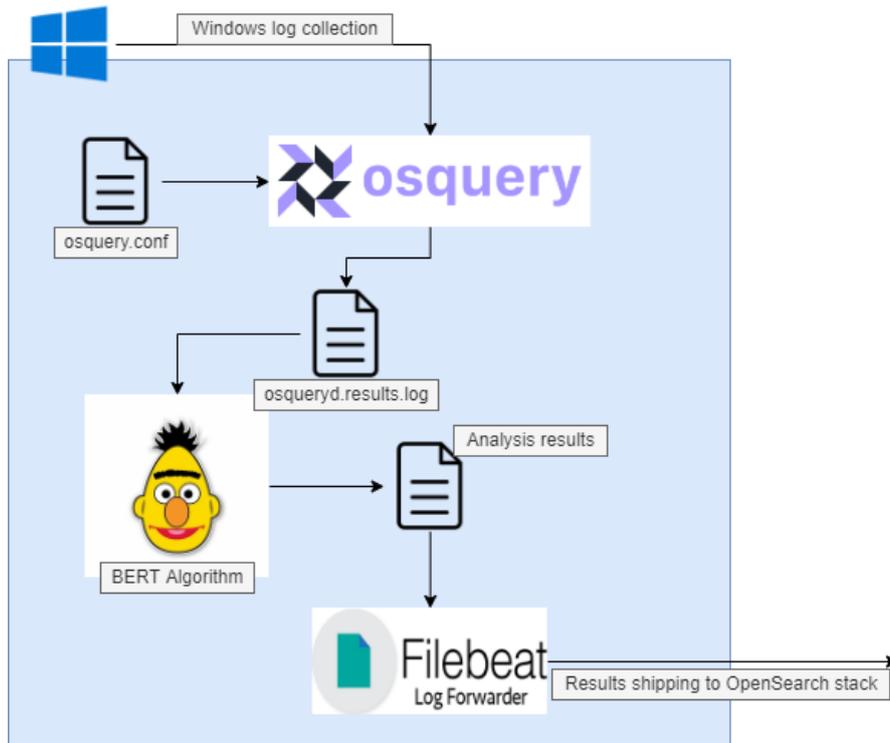


Figure 6.14. Schema of the host machine, running Windows 10.

the VM port 5044 for Logstash and 5601 for OpenSearch Dashboard. Containers will be reachable from outside through the Host-Only network provided by Virtual Box. All three container can be downloaded by Docker Hub¹³ registry and are releases, updated and maintained by OpenSearch project. Docker it is the more convenient and plug&play way to install OpenSearch environment and Amazon created a brief step-through guide to make easier the environment deployment¹⁴.

6.6 Machine Learning overview

The presented architecture is able to search with high precision for very specific events that happen client side, by exposing the operating system as a relational database thanks to Osquery capabilities. Despite some query focusing on identifying some known threat, such looking for a specific name or path, other monitor general purposes events. These tables are populated for the greatest part of legitimate events, that simply are generated by the operating system in its normal lifecycle; such events, anyway, may be tampered

¹³<https://hub.docker.com/u/opensearchproject>

¹⁴<https://opensearch.org/downloads.html>

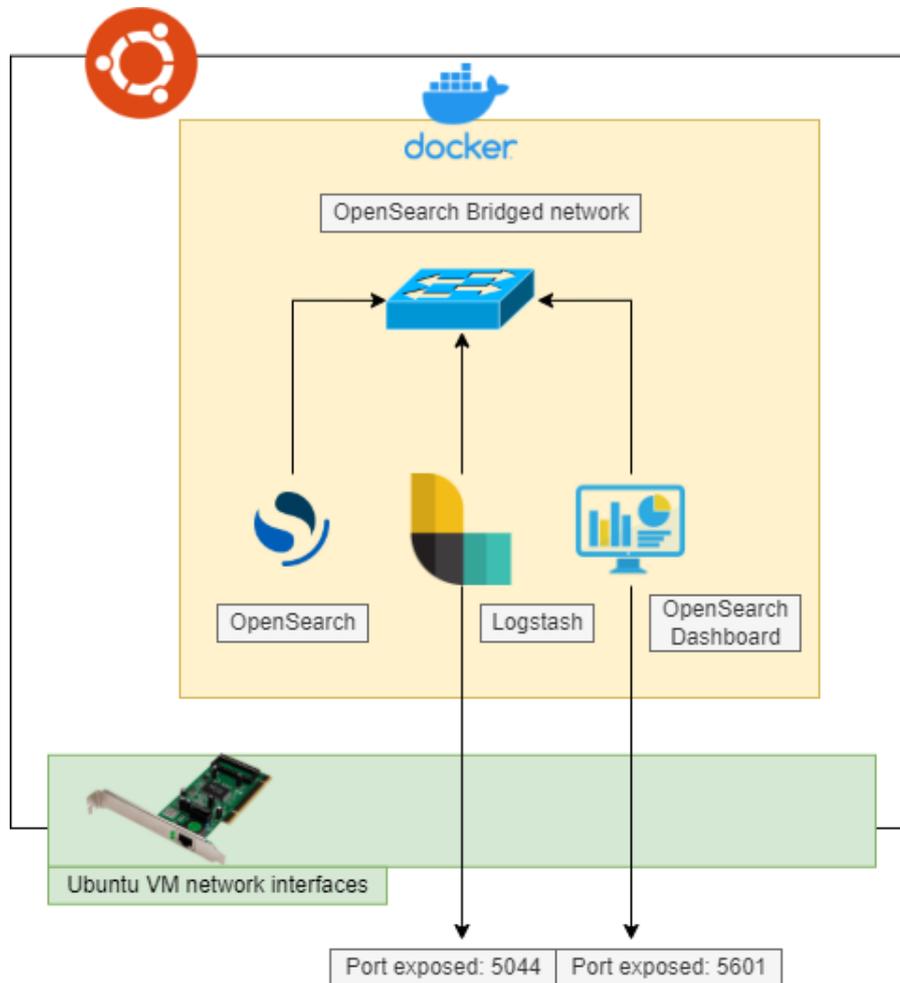


Figure 6.15. Schema of the virtual machine, running Ubuntu.

with malicious one, such the ones generated by malicious processes.

Without leveraging Artificial Intelligence features and Machine Learning capabilities, each log collected by Osquery would be sent to the SIEM application. Since the research of malicious events may lead to face also legitimate one, the overall log collection will include both normal and anomalous logs. This situation leads in having a huge quantity of data to analyse on SIEM side, where many of them, being legitimate events, are identified as false positive alerts. This fake malicious events "*storm*" results in a huge noise around really malicious events, which may not be correctly identified. The introduction of Predictive Analysis techniques should mitigate the false positives problem; by performing automatic analysis aiming on detecting, it would be possible to take into consideration all collected logs and identify which one are likely the most dangerous one. In this thesis is proposed Predictive Analysis through a Machine Learning algorithm; the algorithm is taught to learn what are normal behaviours of a system, by training on clean and non-malicious logs, and

then perform analysis generating results based on "how much an events is distant from a normal one". In other words, if an event is detected as different from a normal one it is considered as anomalous. In this way, the goal is to learning a baseline of normal logs able to detect events that are "too much" different from it.

Before introducing the algorithm, it is necessary to give an overview of most significant Machine Learning concepts for this work.

6.6.1 Key concepts

In order to properly illustrate the algorithm proposed in this thesis, it is required to give an introduction about key concepts of Machine Learning, especially those that directly interest this work. In this way a context is given, allowing a better understanding of decision taken.

ML introduction

Machine Learning (ML) is a branch of Artificial Intelligence; thanks to the technology evolution and to the development of more efficient algorithms over past few years, ML grown in popularity and nowadays is applied in many sectors, from home-tasks to business use cases.

Some of the main common tasks[68] for which ML is used are:

- *Prediction*: classify data in groups in order to produce a report that predicts the behaviour of a system
- *Image Recognition*: the ability of recognizing images, by identifying common pattern, shape and colours
- *Speech Recognition*: used in word recognition, both coming from an oral speech and written document. It can be used to create Chatbots, used in AI assistant such Google and Alexa and in many other use cases
- *Medical diagnoses*: to help doctors in recognizing pattern in order to provide a diagnosis of diseases, such cancer or rare sicknesses
- *Financial and trading*: applied in fraud investigations, credit checks and to predict the ongoing of some financial asset

Accordingly to Arthur Samuel, a pioneer in ML and AI, "*Machine Learning algorithms enable the computers to learn from data, and even improve themselves, without being explicitly programmed*". The goal, therefore, is to create algorithm that, receiving a proper amount of data both in quantity and quality, is able to take decision and create reports autonomously.

Deep Learning

Deep Learning is an approach that tries to emulate the human brain. It uses a waterfall of nonlinear processing units layers in order to identify, extract or transform some feature of the data. Each output of a layer is used as input to the next one, like a pipeline of learning

stratifications.

Deep Learning algorithms can be **Supervised**, hence they have insights of what are possible prediction results, or **Unsupervised**, and used in pattern recognition. Those algorithms are really powerful and can be programmed to perform a specific task, improving their model for that specific task and even outperforming human being in that field; when an AI technology is specialized in a very specific task is named *Narrow AI*, also known as “weak AI” because it is only focus on that specific task, not in others.

The possibility of focus on a specific task, Deep Learning requires huge amount of data and its processes of training and prediction require long time and resources. This approach, indeed, is really computationally intensive, especially if applied on use cases involving big quantity of data.

This approach has been chosen in the implementation of the proposed algorithm because, despite being an intensive task, it is the best solution for complex tasks such log prediction and behavioural analysis.

Natural Language Processing

Natural Language Processing (NLP) [57] is a branch of AI that aims of giving the ability to a computer of understanding text and spoken words in a way similar to human beings. Almost any person interacted with NLP; e.g. it is used by any voice assistant, chatbot, text correction and word predictions.

Indeed some of the most common use cases are:

- Speech recognition, word sense disambiguation and speech tagging, which can be applied to spam detection, machine translation application such Google Translator, virtual agents and chatbots
- Sentiment analysis, widely applied on social media and social networks, to understand the sentiment of a user behind a post, a comment or a review. This allows businesses to understand how a user feels, and eventually use it as feedback, or perform addressed advertising campaign
- Natural language generation, as opposite of speech recognition. It may be applied in text summarization, where from a long document only some key sentence is extracted, or for identifying crucial records in a database

In this work NLP is used to read and try to understand the logs collected by Osquery agent. Each log represents a sentence and the algorithm chosen will be able to understand the context from a semantic point of view, even if collected information are not human understandable, or do not follow human orthography rules. Indeed there are studies such [75], from which the proposed algorithm takes inspiration, that uses NLP applied to Windows event logs to perform anomalies detections.

Attention

The *Attention* mechanism was introduced in 2017 by Ashish Vaswani et al.[90] during the presentation of a new architecture, the Transformer. In that document the attention mechanisms is described in highly detailed terms, while in another paper[58], instead, is given

an overview of the concept.

The problems of a Recurrent Neural Network (RNN) algorithm, a deep learning approach, are that it is unidirectional and that it understands the context of a sentences as it advances with it. This means that, to make better predictions, there is need of more and more context to be analysed. Having a huge context to be analysed mean more memory with the risk of an obfuscation of the context understanding in case of longer sentences. In other words, RNN algorithms focus too much on closed words of a sentence and too much effort is put into a single stream direction (e.g. from left to right) of the context, despite of the other one (e.g. from right to left). Attention is the mechanism Google proposed as a solution to these issues.

In few words, Attention mechanism takes two sentences and creates a matrix using one as columns and the other as rows. From there it creates concatenation and matches, identifying relevant context. Such an approach would be great in language translations; it is also possible leverage the matrix by putting on both rows and columns the same sentence, creating a process where the algorithm understands how each word is related to the others. In this way, for a human language, would be possible to understand the subject or the noun of the sentence. This approach is named *self-Attention*.

The described mechanism allows Attention to examine the whole sentence, from both directions. Furthermore, the approach is capable of relating words that are far from each other but that are connected by a semantic relationship.

Transformer

A *Transformer* is a neural network architecture based on self-attention; this model was introduced by Vaswani et al in [90]. A Transformer [89] performs small and constant number of steps, chosen empirically. In each step is applied self-Attention, allowing the model to understand the relationship between all words in a sentence.

Each word is compared to every other word in the sentence, resulting in an *attention score* for any of it. The score determines how much the target word is related to others: a higher score means a closer relationship, logically speaking. Each score is then used as weight to a weighted average of all words' representation and given in input to a fully connected network to generate a new representation for the target word, returning as result the context around it.

Transformer model has higher performance and accuracy with respect to RNN, accordingly to [89]. Furthermore, it can relate other parts of a sentence when processing a given word, hence giving insights of which info are exchanged.

In this work it has been chosen, as base implementation, a Transformer algorithm that applies the self-attention mechanism, providing a state-of-the-art model. The algorithm will be analysed in the next section.

6.6.2 Introduction to the BERT model

BERT was presented in a paper published by Google, in May 2019 [24]. The acronym stays for *Bidirectional Encoder Representation for Transformers*. It was designed to pre-train deep bidirectional representation from unlabelled text by jointly conditioning on both left and right context in all layers. Its structure allows to fine-tune a model using a pre-trained one, hence adding just one computational layer in order to obtain a state-of-the-art algorithm able to solve different tasks without requiring huge modification to the overall architecture.

The fine-tuning operation is the task of introducing minimal task-specific parameters, and training on downstream tasks by fine-tuning all pretrained parameters. This means that is possible to take a pre-trained model and make it focus on another one, where final tasks will be executed, without huge workload overheads.

The paper claims that models previous to BERT restrict the potential of pre-training and fine-tuning, especially due to the limit of unidirectional streams, such RNN cases. BERT improves these approaches, by adopting a different approach named *Masked Language Model* (MLM); it consists in masking random tokens from the input and then predict the word by using only its context. By using both left and right context to the masked word, it is created a bidirectional approach allowing to build a deep bidirectional Transformer model.

BERT Structure and how does it work

In the paper[90], it was introduced a model with two main parts:

- *Encoder*, used to process the input text and search for important part, building a relationship between a word and all other words;
- *Decoder*, takes the output of Encoder and transforms it into a text output

BERT does not loop over the input several times, instead it processes the input through multiple attention layers. Also known as *attention heads*, each attention layer can be interpreted as a way to understanding better the input sentence, introducing bidirectionality and addressing some problems of RNN approaches.

The concept of bidirectionality is a key factor on BERT architecture, as stated by [35]. The Transformer attention mechanism allows parallel processing, meaning that the model is able to work with any word in a sentence; to do so, anyway, there is need of knowing each position of each word, otherwise the parallel execution would lead to loss of the words order. The Transformer model marks the position of each word, so that when analysing the sentence in parallel it would be possible to know the order of each token at each time. Furthermore, this approach allows to lookup for any word at any moment, at any step in the attention layer. Final words of a sentence can be accessed without overhead in early stages, meaning that both direction are suitable.

Even if BERT is a Transformer model, it present some differences with those:

- BERT uses the encoder to create a model suitable to work with NLP, meaning that it can be trained on multiple languages and be used for several tasks. The encoder,

basically, converts the input sentences into a format readable by the algorithm and that can be adapted in order to perform different operations.

- BERT does not use the decoder, by default. This means that the BERT output is an embedding, not an actual text; BERT is optimized to work with the embedding in tasks that leverages its output, such in the computation of similarity scores. Anyway, the decoder can be explicitly used to obtain some text output without requiring additional operations
- BERT trains with two main tasks: MLM and Next sentence prediction. The first task is actually applied in this work, while the second one is not fundamental here, since it consists in trying to understand if two consecutive sentences are related, which is not the case when considering logs.

The BERT Encoder receives in input a sentence made of text words. The sentence passes through a component named Tokenizer; it basically process the sentence based on some rules and it assigns to each word a token, hence a numeric value. In this way, the model is able to associate any word, in any language, with an integer number meaning that can receive any kind of text input. The couple word-number is stored in a dictionary built by the Tokenizer. There are different approaches to perform this task, such:

- *White spaces based*: the tokenizer split a sentence based on white spaces; the main problem here is that it would be required a huge vocabulary. Since BERT reads through numeric tokens, it is not able to recognize similar words, such plurals (e.g. “dog” would not be related to “dogs”), or same words separated by punctuation (e.g. “Dog!” would be a different token with respect to “Dog”). In human language would not be feasible to build an exhaustive dictionary with this approach;
- *Punctuation based*: the tokenizer split sentences based on white spaces and on punctuation; e.g. the word “Dog!” would be split into “Dog” and “!”. The problem of huge dictionary still persist, since each word that differs for just one character from another is considered as completely unrelated;
- *Character based*: the tokenizer split words in sequence of characters, assigning to each of them a different token. A really small dictionary would be able to represent any word, but it is added an overhead due to the multiplication of the number of tokens that a model should track. The word “Dog!”, previously split in one or two token depending on the approach, now would be split into four token: one for each letter and one for the esclamation mark;
- *Sub-words split*: the tokenizer algorithm breaks each word in pieces, similar to a syllables division. Each syllable, made of few characters, receives a token. This approach has different advantages for a human language since it allows to build almost any kind of word by joining syllable tokens; i.e. the word “eating” would be split into “eat” + “ing”. The two main problem of the white space approach are solved: a dictionary with a suitable size is built and the model would be able to relate similar words (“dog” and “dogs” are related since the second case is split as “dog” + “s”).

In this work a White space approach is taken. The reason is because the algorithm works with NLP generated by Windows operating system and collected by Osquery; the number of possible words is somehow limited and there is no need to relate similar tokens. The problem of this approach should, hence, be mitigated by the special NLP language chosen, which using a machine-readable format does not require to be split in sub-words.

A tokenizer does not only create a dictionary of obtained tokens; to add additional information to the BERT model, the tokenizer uses some **special tokens** which are inserted in the dictionary and used for specific purposes:

1. [UNK]: usually mapped with 0 in the dictionary, it is used to map unknown tokens or words that are not enough frequent to be inserted in the vocabulary;
2. [CLS]: usually mapped with 1 in the dictionary, it is inserted at the beginning of a sentence in order to signal to BERT the starting of a new phrase;
3. [SEP]: usually mapped with 2 in the dictionary, it is used to separator between two consecutive and related sentences. Thanks to this token, it is stored an embedding recording to which sentence each word belongs to;
4. [PAD]: usually mapped with 3 in the dictionary, it is used to insert padding tokens to a sentence in order to have the whole text at the same length. The amount of padding can be derived from the maximum length parameter, explicitly fixed, or dynamically, taking as maximum length the longer sentence;
5. [MASK]: usually mapped with 4 in the dictionary, it is used to mask the original token of a word in order to perform future prediction tasks.

In the Figure 6.16 is shown an example of tokenization; the sentence “*my dog is cute, he likes playing*”. A Sub-word approach is taken, as it can be seen from the tokenization of “*playing*” word. Special characters are added to signal the start and the separation of two sentences. Notice the position embedding which records the order of each word and the segment embedding object, that has the task of mapping each word to the proper piece of sentence.

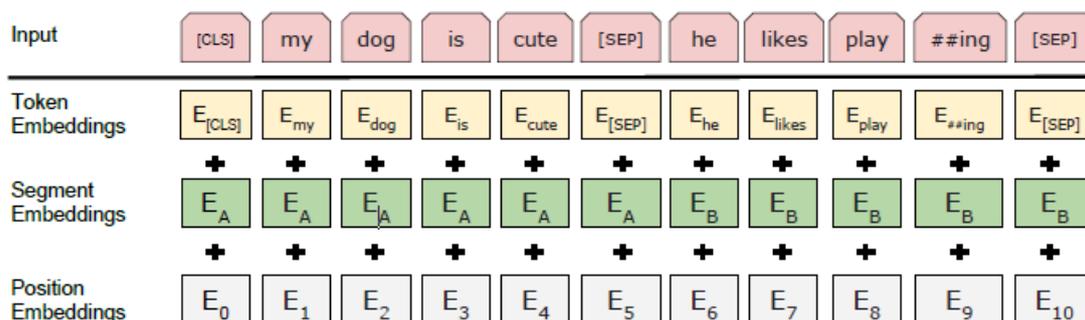


Figure 6.16. Tokenization example applying a Sub-word approach. [24]

In this work, special tokens are assigned to the first five mappings of the dictionary, allowing the real tokens to be assigned starting from the next position.

6.6.3 Masking Language Model and Next Word Prediction

It is now analysed the core task of the model that is proposed in this thesis: the *Masking Language Model*.

Remembering that BERT model is able to look any word in a sentence at a given moment, it would be trivial for such an algorithm to predict each token by simply leveraging its bidirectionality ability. In order to train such a model, Devlin et al. [24] propose to mask only some percentage of the input tokens, randomly chosen, and predict them. This approach forces BERT to stay “focus” on the text, avoiding it to “cheat” by analysing the context using bidirectionality.

Only masked tokens are predicted; when a [MASK] word is faced, the prediction task passes through each hidden layer of BERT model and the embedding provides an output expressed by a Softmax probability, assigning values between 0 and 1 over the whole vocabulary. These values are then returned by the embedding model as logits values, hence mapping each values on Real numbers.

As mentioned, only some token are masked. In particular only the 15% of them, with some specific rules:

- 80% of selected masked words are actually masked with [MASK] special token;
- 10% of selected words are masked with a random token, mapped into the dictionary;
- 10% of selected words is not masked and the original token is left.

There is a specific reason for this guideline mapping, explained by Devlin et al: *"The advantage of this procedure is that the Transformer encoder does not know which words it will be asked to predict, or which have been replaced by random words, so it is forced to keep a distributional contextual representation of every input token."* Even if the random token replacement leads to wrong predictions, it does not make performance worse, since such mapping is applied in only 1.5% of cases.

Once the masking is done, it will be used to predict the original token applying a distance measure, which is the *Cross Entropy Loss*.

In Figure 6.17 is considered the same sentence as the previous example; two word was selected and substituted with the mask token. The model gives in output the embeddings of the most probable words for each masked token. An example of output is represented, including the actual word that was previously masked, meaning that the model has correctly performed the prediction.

The Next Word Prediction task was not defined in Devlin et al paper; indeed it is a special application of MLM task which consists in masking the last word of a sentence, or the last word of a piece of it, in order to predict it. Some studies, such [75] uses this approach in order to perform malicious log detection based on the results of their model predictions.

In this work it is used a classic MLM task for training, while for the actual logs analysis

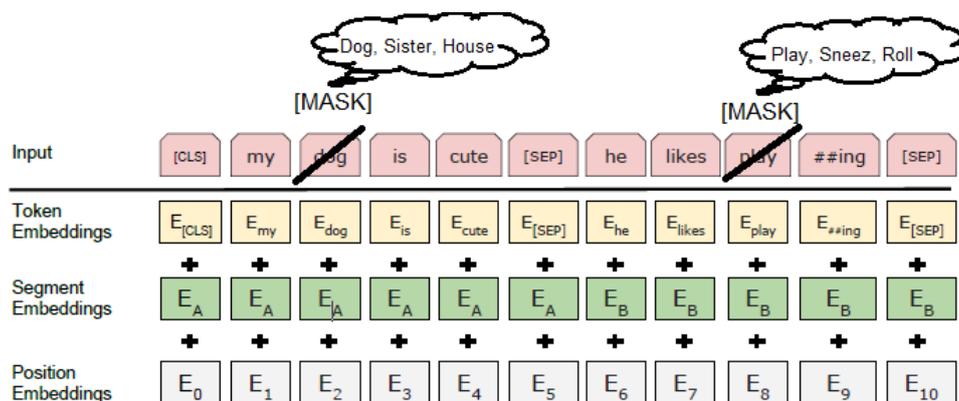


Figure 6.17. Example of masking and prediction results, with input the previous considered example sentence.

the MLM is applied with some modification of the approach, which will be explained in 6.7.4.

6.7 The Proposed Algorithm

This thesis propose an algorithm that aims to achieve a similar goal: the detection of threats through log analysis. Until now, it was chosen an open source-oriented approach and this algorithm follows this principle as well. There are no similar solution in open source landscape, that tries to offer such a level of analysis and authors of this work hope that this will be a starting point for future improvements.

The selected algorithm model is BERT, a Bidirectional Transformer for Deep Learning of NLP. The algorithm is entirely written in Python, leveraging the *Hugging Face*¹⁵ library. This initiative offers an open source Transformer library developed with PyTorch, TensorFlow and JAX.

Hugging Face also offers a wide choice of pre-trained models that can be downloaded and fine-tuned, but what this thesis tries to achieve is the creation of a brand new model, specialized in log analysis and malicious events detection. For this reason a dataset research was performed, in order to detect a collection of logs suitable for the purpose. Unfortunately, this research does not succeeded.

The effort of creating a brand new dataset of logs was taken, leveraging all great capabilities of Osquery, such its ability to perform specific lookup. Furthermore, the open source nature of the project allows in any moment to have control over what is being collected, in order to mitigate possible privacy issues that a lot of EDRs and in general agents may introduce due to the lack of documentation or license restrictions.

The dataset is built by running the Osquery service on a Windows 10 machine, which is

¹⁵huggingface.co

described in Section 6.5. In particular, at least three batches of logs are required to be collected in order to reproduce the algorithm “loop”:

1. Training logs: a batch of logs useful for training; they must be clean without any tampering or potential threat infection;
2. Baseline logs: used to test and define a baseline scenario of clean logs, which results will be used as comparison with the real logs under analysis;
3. Actual analysis: where the malicious log detection actually happens; the baseline scenario results are used to evaluate events in this batch and in case of suspicious logs, an alert will be generated by storing the output in a proper file.

In a real environment, the algorithm should run continuously but in this section is considered just one interaction.

The obtained model is presented as a pipeline of events, that can be shown in Figure 6.18.

In the next sections will be given an overview of operations performed in each step.

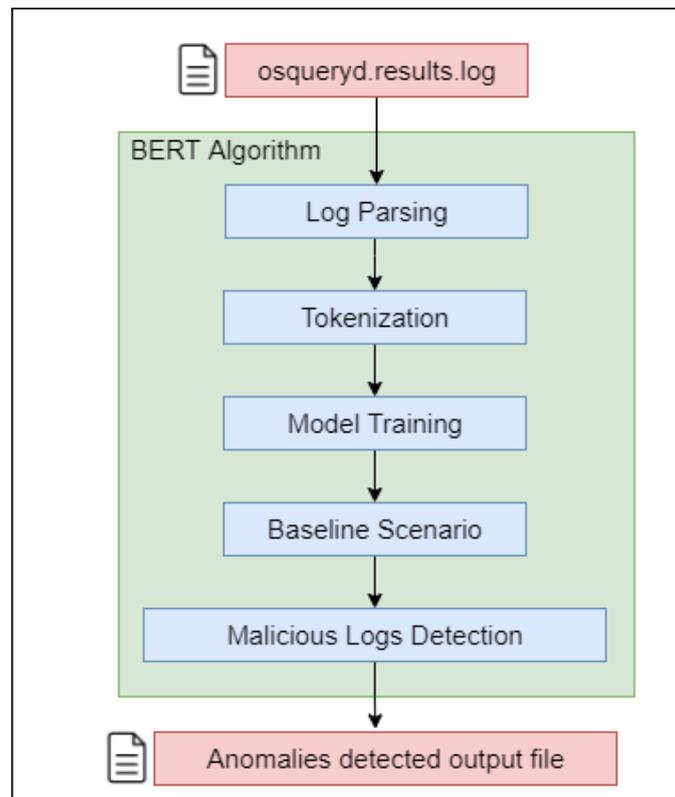


Figure 6.18. BERT algorithm pipeline, from log collection to output results.

6.7.1 Log parsing

Log parsing operation is the first step in the algorithm pipeline. It must be applied to any log, row per row. The goal of this task is to take unstructured and machine-unreadable data and processing them in order to obtain a format that can be taken as input by BERT model.

Osquery output consist in a row representing the result of a query; this row is in JSON format and contains both metadata and actual meaningful information. Below is given an example of a log snippet that requires parsing:

```
{ "name": "pack_1-T1059_T1059.001_query_A", ..., "calendarTime": "Mon Feb 14
  08:39:10 2022 UTC", ..., "columns": { "atime": "1644827934", "attributes": "AI
  ", ..., "directory": "C:\\Windows\\Prefetch\\", ..., "path": "C:\\Windows\\
  Prefetch\\POWERSHELL.EXE-59FC8F3D.pf", "product_version": "", "size
  ": "39749", ..., "action": "added" }
```

The JSON object must be manipulated in order to filter the amount of information given in input to BERT, allowing an effective training. On the other hand, meaning full data must be maintained.

Metadata fields such "calendarTime, unixTime, epoch, counter, numerics, action" are immediately discarded; they are fixed for any query and do not carry sensitive information. A recursive process starts, in order to detect any JSON object that may be stored inside others. Logs are not so long, typically, so recursive approach does not add too much overhead at this stage.

Some other text manipulation is applied:

- Dates object in format such YYYY-MM-DD and eventually hours and seconds are substituted with Unix Timestamp;
- IP addresses are split, and each octet will receive a different token;
- Each path is kept untouched, expect for the drive letter, i.e. "C:\\...", which is separated and receives a different token;
- Each white space in a path JSON object is replaces with an underscore, e.g. "...\\Program Files\\..." becomes "...\\Program_Files\\...";
- Each equal sign, column, new line, and ticks is removed from the row and substituted with a white space;
- Each JSON object, nested or not, is tracked by leaving in the log line the corresponding curly parenthesis.

Following these rules, the log is processed, and a new string is generated. The output, for a the example log considered before, would be something like:

```
{ name pack_1-T1059_T1059.001_query_A ... columns { atime 1644827934
  attributes AI ... directory C \\Windows\\Prefetch\\ ... path C \\Windows\\
  Prefetch\\POWERSHELL.EXE-59FC8F3D.pf product_version size 39749 ... } }
```

This new representation can be given in input to the Tokenizer embedding, which will be able to process it and it will assign to each word a specific integer token.

As final note, all parsed logs are saved in output into an intermediate file that will be used as input by the tokenization script.

6.7.2 Tokenization

Tokenization is the process of assigning to each word an integer value in order to have a data format suitable for the Transformer input. Basically, it consists in defining a Tokenizer object, reading the file save the output into a JSON file.

The goal of this step is to build a dictionary made of word-value pairs; the dictionary will be used during each other further step in order to translate from text word to tokens every log.

First of all are identified the two main components of this script: a Tokenizer object and a Trainer embedding. First considering the Tokenizer object, it requires three sub-component that must be declared:

1. A model: the chosen one is `WordLevel`, which is a words tokenization, hence it keeps each word as a whole without splitting them
2. A pre-tokenizer: used to perform initial operations, before proceeding with the actual tokenization; in this case the chosen algorithm is `Whitespace` which spit the text based on white spaces and on punctuation
3. A post-processor: a component with the goal to show to the Tokenizer object how two sentences should be joined, hence, how to use sentences special tokens such `[CLS]` and `SEP`.

The order of these operation is important, because once the Tokenizer object is defined, step one, then the other two points are assigned leveraging properties of the main object. The second component is the Trainer embedding object; it requires a model to run, specifying some important parameter, such in Figure 6.19; note the `vocab_size`, which is the maximum dictionary dimension, `special_tokens`, which are a list of all special tokens and finally `min_frequency`, that is the minimum number of times that a word must appear in order to be stored in the dictionary. This last parameter is crucial, and it has two main

```
23 trainer = WordLevelTrainer(  
24     show_progress=True,  
25     min_frequency=2,  
26     vocab_size=30522,  
27     special_tokens=["[UNK]", "[CLS]", "[SEP]", "[PAD]", "[MASK]"]  
28 )
```

Figure 6.19. WordLevel model for Tokenization, with important parameters.

consequences during log analysis:

- When BERT finds a word that is not contained in the dictionary it assigns the [UNK] token; this happens also to words already processed in Tokenization step, but that appear only once
- By setting a minimum frequency parameter, leading to use the [UNK] token also during training, the model is taught in facing unknown words and it learns how to behave in such cases, which may be common in analysis environment

This step is performed only once and only at model build time, meaning that it is performed during pre-training, or fine-tuning, on server side. Indeed the goal is not to tokenize each file to analyse, which is a step internal to the anomaly detection one, but to create a dictionary of all token meaningful to the proposed algorithm model.

This step is performed using the Tokenizer library¹⁶ published and maintained by Hugging Face organization.

6.7.3 Training

The *Training* step is the core of the whole algorithm. It is often the most resource intensive step, both in terms of required data and time.

Firstly, are listed all most important components:

- Tokenizer: object created by loading the dictionary created in the previous step. This object contains also all metadata required;
- Model: the object representing the BERT model, along with its configuration;
- Trainer: the object that performs the actual training, exploiting the Model and the Tokenizer capabilities;
- Data: input file used for training, properly parsed in previous step and then tokenized here.

In details, the *Tokenizer* is basically the same object of the previous step loaded with the created dictionary. It is used to tokenize input, parsed logs and returns the embeddings object containing two main contributions:

- `[input_ids]`: a tensor containing all tokenized log rows; it has one dimension for every row and another for every word composing a log.
- `[attention_mask]`: a binary tensor with the same dimension of the previous point, that represents if a word must be considered or not; in other terms, it allows the model to ignore padding tokens.

Below its usage:

```
tokens = tokenizer(lines, max_length=100, padding='max_length',truncation=True, return_tensors='pt')
```

¹⁶<https://github.com/huggingface/tokenizers>

The `lines` object are the log lines, the maximum log length is 100 and if shorter the log is padded, if longer the log is truncated. This means that all lines will have a length of 100 words, since are padded or truncated to that value. The return object is a PyTorch one, that allows to manage the object as a tensor, hence a sort of multidimensional array.

Moving to the *Model* object, it is first necessary to introduce an intermediate object that represents its configuration. The values are taken from BERT guidelines and/or by Steverson et al [75] paper: hidden size equal to 768 and 8 attention heads. Now the model object can be created, using the `BertForMaskedLM` API and receives as input the configuration object mentioned before. Along with the BERT model, also the optimization algorithm must be chosen, and it was selected the AdamW optimization one, with default parameters (learning rate 10e-4, betas 0.9 and 0.999).

Since the training process requires many computational resources, the model is passed to the CUDA device, hence the GPU, if available, otherwise it remains assigned to the CPU. *Input data* must pass through some step before being processed by the training engine. After the tokenization, they need to be masked; a custom function is written to achieve this task, and following rules for MLM listed in Section 6.6.3, a couple of tensors are retrieved:

- `input_ids`: same object as before, containing the original and masked tokens; it will be used by the model to perform predictions;
- `labels`: same `input_ids` dimensions; the tensor is made of -100 tokens, which are ignored by the BERT model, and of the original values of masked tokens; this object is used as reference by the algorithm in order to verify the goodness of its predictions.

An additional object is created; it is basically a support object that keeps together `input_ids`, `attention_mask` and `labels` in a single structure. Every element of this object is shuffled to ensure that the algorithm cannot rely on some connection between consecutive logs. Notice that the shuffle process involves each element of the support object, while internally each triplets is not decomposed; this means that every tensor of a triplet, before the shuffle, is associated with same other tensors after the internal order is changed. The mentioned additional object will represent the *training dataset*

Finally, accordingly to [75] paper, a set of 100 random rows are chosen among the overall training log file. The selection represents the *validation dataset*; it is processed accordingly to all previously described steps, and it is used to validate the training model in order to determine if performance are satisfying or not.

The final component of this step is the Trainer object. In Figure 6.20 are listed the required parameters; briefly mentioning them, the model is the MLM Bert model, the tokenizer, the training and validation dataset, the `args` which is a configuration object for the trainer containing info such the output directory and the number of epochs, the `compute_metrics` which is a function written to evaluate the goodness of predictions against the validation dataset and the `callbacks`, a function to stop the training process.

In Figure 6.20 are shown which are the main components requested by the *Trainer API*, provided by Hugging Face library. The APIs allows to run the run the training automatically just defining a proper Training object and by calling a function. Another great advantage of this approach is the `callbacks` parameter. It consists in defining a technique called *Early stopping* which allows to stop the training execution before it arrives to the end. This allows the algorithm to autonomously establish when to stop the execution,

```
trainer = Trainer(  
    model=model,  
    tokenizer=tokenizer,  
    args=args,  
    train_dataset=train_dataset,  
    eval_dataset=eval_dataset,  
    compute_metrics=compute_metrics,  
    callbacks=[EarlyStoppingCallback(early_stopping_patience=5)]  
)  
trainer.train()
```

Figure 6.20. Trainer object listing its parameters and the function called to launch the training process.

avoiding overfitting on the dataset.

`compute_metrics` is a function that the Trainer API exploits in order to measure the goodness of its predictions and which result is used to establish when to stop the training. The function takes in input the `labels` object and `logits`, which is an object containing all probabilities of performed predictions mapped on real axe. From `logits` are extracted `predictions`, an array containing the most probable value for each word, hence the actual prediction performed by the model.

For all couples label reference – prediction, it is computed a metric evaluation. The chosen one is the **F-1 score**, which is a measure of accuracy computed from *precision* and *recall*. In few words, the more the score is high the more predictions on the validation set are accurate. The model is stopped in its training execution when it notices that the F-1 score does not improve anymore.

The output of this step is a directory containing all steps and intermediate results achieved during training process, for each epoch. The main component is a pre-trained model file that can be loaded and used to perform predictions, such in the next two steps. This step must be performed only once, by the server on powerful machines, not by the user. Eventually, in order to keep updated the generated model, it is possible to build a process of continuous training by performing periodic training processes, always on server side. On end user machine would be enough to load the pre-trained model and run the analysis algorithm as a service.

To create the current Training script, the Baseline and Malicious log detection steps, the Transformers library¹⁷ was exploited, always offered by Hugging Face organization.

¹⁷<https://github.com/huggingface/transformers>

6.7.4 Baseline scenario

This section aims in performing a first analysis of **clean** log in order to establish a *normality score*, also named *baseline score*. Additionally, a *standard deviation* value is computed that is a measure of how much clean log anomaly score changes from each other.

First of all, the Tokenizer object obtained in Step 6.7.2 is loaded, and also the pre-trained model obtained in Step 6.7.3 as well. If available, the MLM model is passed to the GPU. The Tokenizer object returns the same tensors as before: `input_ids` and `attention_mask`. Also the `labels` tensor is required, with the same scopes as before: it is used as reference to store the actual value of original words, and `-100` values for all other placeholders.

A new approach is used to perform predictions; by considering a single log line, are performed the following operations:

1. In `input_ids`, only one token per line is masked, the other are left with their original values;
2. `labels` is fulfilled with ignore tokens, hence `-100` values, except for the masked one, which contains the original value, and it will be used as reference;
3. The model performs the prediction, returning as an output a tensor of `logits` representing the probability of each token of being the original one, and a **loss score**;
4. The loss score is stored in an object;
5. Iterate on previous steps until the end of the line is reached; **notice** that predictions are performed for all tokens, except for special ones (e.g. the padding tokens are not considered);
6. At the end of the log line, the algorithm has stored an array of losses; a mean operation is performed among these loss scores and its value is saved.

These steps are repeated over each log line of the baseline data set. The innovation of this approach is the way MLM is applied; it differs both from standard masking guidelines provided by BERT during training and by [75] approach, where MLM is used as a Next Word Prediction task. Indeed, here one word per time is also masked, but all the log line are given to the model, which can perform the prediction having the whole context under consideration.

Once all logs have been processed, the maximum score is selected among all the average losses computed during the analysis. This value is stored as `max_baseline_score` and represents a measure of the maximum score a clean log can obtain to be considered legitimate.

Furthermore, it is computed also the standard deviation of the averages score of every line, value that will be named `baseline_standard_deviation`.

The last consideration is about the loss score. It is computed by the model exploiting the `CrossEntropyLoss`¹⁸ function, that receives as input the `logits` and the reference `labels`. Essentially, the loss is a measure of distance between predicted logits tensor and the actual

¹⁸<https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>

label references. A higher loss means a worse prediction or better a measure of how much a given log is far away from the predicted and ideal one, from model point of view. This script must be executed only once, on server side. Usually the baseline selected dataset represents only a small quantity of logs, with respect to the amount required for the training phase. The output are the mentioned `max_baseline_score` and `baseline_standard_deviation`, which values will be used as input in the next section.

6.7.5 Malicious Log Detection

This is the final step of the algorithm. The goal is to identify potential malicious logs, generate a report, or a file containing meaningful information, that can be shipped to OpenSearch by Filebeat agent.

Many performed operations are similar to the previous step 6.7.4, with the addition in the scoring system generation. Here are summarized all performed steps:

1. In `input_ids`, only one token per line is masked, the other are left with their original values;
2. `labels` is fulfilled with ignore tokens, hence `-100` values, except for the masked one, which contains the original value, and it will be used as reference;
3. The model performs the prediction, returning in output a tensor of `logits` representing the probability of each token of being the original one, and a **loss score**;
4. The loss score is stored in an object;
5. Iterate on previous steps until the end of the line is reached; **notice** that predictions are performed for all tokens, except for special ones (e.g. the padding tokens are not considered);
6. The average score is computed among all losses of the log line, also named **anomaly score**;
7. The value from the previous point is used to compute the **alert score**, which represents measure of the potential threat caused by a log.

The **alert score** is computed following the formula published by Steverson et al study [75]. The formula is showed in Equation 6.1; the **alert score** is basically a distance from the maximum score measured in baseline scenario. Indeed, the more a log receives a higher score the more its alert score is high and, hence, the more is considered dangerous.

$$Alertscore = \frac{Score - MaxBaselineScore}{BaselineStandardDeviation} \quad (6.1)$$

The key term of the formula is the numerator: if the subtraction is positive, also the result will be positive, and the log will be considered as malicious.

The output of this step is a file containing a report of all logs considered as malicious. These steps must be performed on client side and for, almost, all line collected by Osquery service.

By shipping them to OpenSearch stack, will allow a deeper analysis looking for real threats or filtering remaining false positives. Notice that with such an approach, it would be possible to detect potentially anomalous logs already on end point machine, with the possibility of filtering may of the false alerts, represented by legitimate operations, that Osquery collects.

Chapter 7

Experimental Results

7.1 Log Collection results

7.1.1 Introduction

Once completed the tests described in Section 5.3, it's possible to make some considerations about the results obtained. Before starting, it's necessary to make a premise: considering how this thesis has been organized, the main statistical and experimental results, are obtained after the Machine Learning analysis performed in the second half of the thesis. This because, the first half has been concentrated more on the preparation of a significative procedure, to collect logs to be analyzed by the ML algorithm. Moreover, the nature of Machine Learning, makes it better to be evaluated with specific metrics. This said, it's important to evaluate in a mainly manual way, the results obtained with the log collection implementation outlined in chapter 5. Next three sections will concentrate respectively on the *clean* log collection, on the *dirty* one and, on a comparison between these two.

7.1.2 Clean Log Collection results

The *clean* log collection that has been described in Section 5.3.2, has been executed for different reasons:

1. To test the functionality, in standard conditions, of the Osquery probes configured with the MITRE ATT&CK customized query packs;
2. To create a useful baseline of logs to be compared later, with the logs collected under attack, and check if adversary techniques are effectively recognized;
3. To accumulate a sufficient number of logs for the training phase of the BERT-based ML algorithm described in 6.7.

Going in order, it is possible to say that in all the 10 cases, after a physiological phase of little bugs fixing, the probe did not give any kind of errors or operational problems. All the trials went smooth and, after Osquery probe activation, not **ERROR** nor **WARNING** files were created, together with the `osquery.results.log` file.

The second target was easily reached. Sometimes query packs gave more results, sometimes less but, in any case there were enough clean and dirty results to be manually compared, in a following evaluation.

The third point listed above, has been a bit more challenging. This because, even if second and third targets may seem similar, to compare logs manually, it is not necessary to have a high number of clean logs. Potentially, if clean collection does not produce logs and, dirty one produces some, it is already an evidence of attack. Machine learning algorithm's training instead, needs quite a big number of logs to obtain a satisfying result. From this point of view, the collections divided by technique, have not always been a success. Although some packs had no problems in producing a lot of logs in a short time interval (minutes) of normal system's use, other ones just produced less than a hundred of logs. This is due to the fact that some queries, easily produce logs and, it is necessary to examine its selected fields' content, to distinguish malicious from benign activities. Other queries instead, are more targeted to search known specific malicious actions and produce logs just in case, those actions are put in place. If a query pack contains a big amount of this second type of query, it is more probable that a clean collection will not output many logs. To overcome this problem, a general log collection was done: a common configuration including all the query packs was set and many logs were gathered. This was done in order to try a single training phase, to be used for different techniques' testing phases.

7.1.3 Dirty Log Collection results

The *dirty* log collection, meaning the gathering of malicious actions' logs, has been the second step in the experimental work. Details about how it has been carried out can be found in Section 5.3.3. This collection has been performed leaving the Osquery probe gathering logs in background, while Atomic Red Team tests were executed thanks to the Invoke-Atomic module, from a Powershell instance. Also in this case, activities have been limited to a single technique each time, to better concentrate on one case at a time, avoiding to lose the focus on the target. When under attack, logs collected by Osquery were always interesting and for each of the nine techniques tried, it was clearly possible to recognize the presence of anomalous activities. Logs collected while performing attack tests, similarly to the ones collected in the previous section, were also important for the testing phase of the ML algorithm developed in the second half of the thesis. In general, the majority of the queries written produced logs at least when under attack. Some queries did not produce logs, but there is a simple explanation for these. Query packs, whose creation process is explained in Section 5.2, have been written thanks to the intelligence information contained inside three main sources: the *osquery-attck* repository, the MITRE ATT&CK framework and the Red Canary report. The information provided by these sources, is not necessarily coherent with Atomic Red Team tests. This means that sometimes, queries may be prepared to search for certain specific adversary behaviours, but tests from Atomic Red Team are not exhaustive enough to simulate those behaviours. Consequently, queries that search for those specific behaviours will not be triggered and will not produce logs. Atomic tests are overall quite complete, but there are differences between the various techniques: some of them possess a considerable number of tests, others just a few. It is a consequence that, if there are many tests, the probability to obtain results from a greater number of queries increases a lot. On the other hand, if the number of tests is very limited, the

evaluation will be less complete. To sum up, even if some of the queries have not logged any result, this can not be considered a defect because, here they have been tested just under specific behaviours that are not all the possible ones, regarding that technique.

7.1.4 Log Collections comparison

Now that both have been presented, it is possible to make some examples regarding the differences between the two distinct collections. A premise is necessary: this section will not be an exhaustive and detailed discussion about the totality of the query packs created, rather it will be a brief explanation about the differences found in the two above presented types of log collections, limited to a few query packs.

First query pack analyzed, is the one related to the fifth technique in the Red Canary ranking, precisely it is the sub-technique T1003.001. Five queries, named from A to E, have been written for its detection, not a large quantity but a significative sample of behaviours is searched. Clean logs gathered while Osquery was up and running, configured for this pack, are really few. To be more precise, despite the many user standard activities performed during the time in which the probe was active, just one of the five queries logged some results. This query, is a query that searches for execution of specific binaries: these binaries are `lsass.exe`¹, `powershell.exe` and `mimikatz.exe`². The results showed some instances of the first two binaries (it is difficult to see a Mimikatz instance if the system is not under some sort of attack). No other results were logged. Atomic Red Team, fills the T1003.001 folder with thirteen tests and, twelve of them have been executed successfully during experiments (one was not executed because of installation issues). The dirty logs were a larger number and this time, in addition to query A, queries C and E logged results. Query C searches for process instances of `rundll32.exe`, paired with command lines containing the `minidump` keyword. The responsible of these logs is, without a doubt, the test called *Dump LSASS.exe Memory using comsvcs.dll*³; this because, it is the only test that combines the two requisites just listed in the previous sentence. Query E, involves an evented table, precisely `powershell_events` and looks for some suspicious keywords in the `script_text` field. Given that these keywords are present in many of the tests executed, logs are produced and collected. No logging records are saved for queries B and D because, as anticipated before, the behaviours researched by these queries are not simulated by atomic tests.

One concrete example of the case in which results are useful for the first half of the thesis, but become difficult to employ for ML training, is the one related to query pack 7-T1027. This time, no sub-techniques are considered, the main technique with its eight atomic tests, is the one chosen in the report. One of these tests has been excluded because made for Linux and MacOS systems, two of them have given some technical problems and the remaining five have been conducted correctly. There are no clean logs produced for this pack. This makes it impossible to perform a concrete training, but this is not

¹<https://www.glasswire.com/process/lsass.exe.html>

²<https://www.sentinelone.com/cybersecurity-101/mimikatz/>

³<https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1003.001/T1003.001.md>

important in this phase. The dirty logs collected for this query pack, are of different types: four over a total of five queries, produce results. This fact, is a confirmation that the ATT&CK techniques employed have been detected. A simple example is represented by the query A: it searches for evidences of *base64*⁴ encoding/decoding commands, executed inside Powershell and Command Prompt instances. Atomic tests⁵ called *Execute base64-encoded PowerShell* and *Execute base64-encoded PowerShell from Windows Registry* have this characteristic and in fact, their execution is detected and, results are collected.

As a last example, needed to cover all the possible cases, pack 6-T1055 can be considered. This pack, just like the previous, regards the main technique T1055, and no sub-techniques are considered. The pack contains seven queries. Only one of seven is able to produce logs, both in clean and dirty situations. Even if, at first sight, this may seem a sort of unsuccess, a deeper review reveals a different situation. If behaviours modelled by the tests available are compared with the one being searched by the queries, it is easy to see that there are evident discrepancies. This means that tests produce behaviours that, are simply not the cases modelled by the queries. This is not a sign of bad quality queries, they must just be tested with different benchmarks (maybe new atomics that will be introduced in the future, maybe other sort of tests).

All the other packs, precisely number 1, 2, 3, 4, 8, 9 and 10 produced a log collection that follows the same idea of the cases described. These packs collections, are more similar to the one related to pack 5-T1003 but sometimes, queries not triggered like the examples presented for 6-T1055 appear. Given that behaviours of these other packs are similar to the three examples described, they will not be analyzed deeply because, it will be a redundant repetition of the same concepts. Table (7.1) shows a summary of the results obtained. From the table it is possible to observe, how in the tests performed, 8 times over 10, the queries that logged under attack, were more than the ones that logged in standard conditions. This means that they are effective in recognizing techniques. In the remaining two cases, the same queries logged, but this just means, that evidences of attacks must be searched inside the logs, distinguishing data collected before and after the application of the tests. This means that these logs must be investigated, to learn if the selected SQL fields contain suggestions of malicious activity execution or signs of a compromised system. Many instruments can be used to better analyze the results, including visualization tools and scripts to discard not useful data from the logs. In any case, it is definitely possible to say, that tests gave promising results.

7.2 Threat Hunting results

In this section are given results insight regarding the Machine Learning proposed algorithm and the output provided to the OpenSearch stack, with an example of visualization through OpenSearch Dashboard.

⁴<https://en.wikipedia.org/wiki/Base64>

⁵<https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1027/T1027.md>

Table 7.1. Log Collection Experimental Results.

Query pack	Total number of queries inside	Number of queries that logged (clean)	Number of queries that logged (dirty)
1-T1059	11	5	8
2-T1218	18	2	12
3-T1543	12	5	8
4-T1053	9	5	5
5-T1003	5	1	3
6-T1055	7	1	1
7-T1027	5	0	4
8-T1105	17	2	9
9-T1569	6	1	3
10-T1036	4	2	3

7.2.1 Parsing result example

Parsing is the first operation that must be performed when any log file is taken in consideration. This process must, hence, be done on both server and client side.

The file containing all logs collected by Osquery agent is used as input; each row represents a log that matches one of the rules inserted in its configuration file, therefore each line requires to be analysed. Following all rules presented in Section 6.7.1, a log is processed, and the transformed output is produced, in a suitable format compatible to BERT Transformer model.

Below is showed a log example, actually collected by a rule in Osquery configuration:

```
{
  "name": "pack_2-T1218_T1218.011_query_A",
  "hostIdentifier": "MSEDGEWIN10",
  "calendarTime": "Mon Feb 14 08:39:15 2022 UTC",
  "unixTime": 1644827955,
  "epoch": 0,
  "counter": 0,
  "numerics": false,
  "decorations": {
    "host_uuid": "ACB3DB84-69C8-A346-A98E-977175C2273F",
    "username": "IEUser"
  },
  "columns": {
    "cmdline": "C:\\Windows\\system32\\lsass.exe",
    "name": "lsass.exe",
    "path": "C:\\Windows\\System32\\lsass.exe",
    "pid": "616",
    "username": "SYSTEM"
  },
  "action": "added"
}
```

Given this real-case log, some pre-processing is required: punctuation is removed, metadata fields are fixed, dates are reformatted, and path are parsed. Every other rule must be compliant to Section 6.7.1. The final output file will have the same number of rows, with different formats and containing only meaningful information. Notice how the nested object `columns`, which is the result of the actual query and hence is a metadata field of each log, maintains a way to signal its beginning and its end through curly brackets. This allows the algorithm to perform a prediction on when a nested JSON object starts and ends.

The result of the parsing process would be the following one:

```
{ name pack_2-T1218_T1218.011_query_A hostIdentifier MSEDGEWIN10
  decorations { host_uuid ACB3DB84-69C8-A346-A98E-977175C2273F username
    IEUser }columns { cmdline C \Windows\system32\lsass.exe name lsass.exe
  path C \Windows\System32\lsass.exe pid 616 username SYSTEM } }
```

Notice the new length of the row: it is significantly shorter; this allows the algorithm on training and processing only meaningful data. Since the truncation is performed, in the next step, is crucial for longer logs to be filtered to their minimum length.

It is also important to note how paths are processed: the character representing the drive is detached from the rest, while the relative path is untouched including the name of the executable. Notice, also, that all the punctuation is removed, except for curly brackets and extension dots. Indeed the last one are maintained since are considered as part of the name of the detected program.

7.2.2 Tokenization result example

As presented in Section 6.7.2, tokenization is a process of building a dictionary of words and numeric values pairs, that will be used to translate each log line in a sequence of token that can be ingested by BERT Transformer algorithm. Additionally to the entry created by processing log lines, also some special tokens have been created. Usually they cover the initial entry of the dictionary. Below is showed a snippet about the dictionary.

```
"vocab":{" [UNK] ":0, " [CLS] ":1, " [SEP] ":2, " [PAD] ":3, " [MASK] ":4, "{" :5, "}" :6, "
  name":7, "C":8, "username":9, "IEUser":10
```

These are the first ten entry of the dictionary, expressed in JSON format, used as example in this thesis.

It is important for performance of the algorithm that the dictionary is not huge; the obtained vocabulary results in 4007 entries, leveraging a file which contains logs collected by all rules matching the top 10 techniques mentioned in Red Canary report.

Notice that the built of an effective dictionary is fundamental and must be performed accordingly to requirements, use cases and task of the appliance filed. In real case scenario, where tens or hundreds of thousands of logs are collected, consider using other tokenization techniques in order to reduce the dictionary vocabulary.

It is also important to set a minimum frequency to allow the creation of an entry, when facing a new word. Rare words should not enter the dictionary, and indeed in a training or analysis environment should receive the [UNK] token. In this thesis, the minimum frequency boundary was set to two, hence any word faced only once in the dictionary file is not stored in the vocabulary.

7.2.3 Training results

This is the core task of any Machine Learning algorithm; it is also the most time and resource consuming process. Using the Trainer APIs provided by Hugging Face Transformers library, the train is launched simply calling the specialized function:

```
trainer.train()
```

The training will run automatically for all the necessary time. Exploiting the `callbacks` feature, a huge number of epochs was set; the actual number of iteration is automatically decided by the `Early Stopping Technique` callback function which establish when the training should stop, before the overfitting of the chosen dataset.

A training process needs a validation dataset. 100 random logs were selected from the training dataset in order to perform validation, exploiting the F1-score. Once the training results does not improve anymore, the process is stopped.

As explained in Section 6.7.3, logs are masked with MLM technique following listed rules. The training process happens by try to predict masked tokens against original one, stored in a separate object. The algorithm learns how to improve its predictions across any step and any epoch, resulting at the end in a model that built a knowledge about the ingested data.

To evaluate the goodness of the model, at the end of every epoch a validation process take place, by comparing predictions on the validation dataset against its original tokens. A F1-score is computed and outputted along with the validation loss score. For an algorithm that does not overfitted the dataset, a higher F1-score and a lower loss value represents a more accurate model.

After the validation step, every epochs saves the models obtained as checkpoint; these intermediate results represent the pre-trained model that can be loaded to perform malicious detection logs or, in general, fine tuning of a Transformer model.

At the end of the training process, the best checkpoint model should be noted and used to load the pre-trained Transformer. The Trainer APIs, and the Early stopping technique, have chosen the model coming from the 10th epoch, hence the *checkpoint-51870*.

Inside of every checkpoint directory there are a number of automatically created files. The one containing the most significant results is `trainer_state.json`; it contains a resume of all epochs results, including each step and validation results. The Figure 7.1 shows the results of the best model obtained by the training process.

The first JSON object represents the result of the trainer on the training dataset. In particular, notice the loss value which is a representation of how well the model is able to predict the masked logs; it is the value used to improve its performance.

The second object shown in Figure 7.1 are the validation results. The most relevant values here are the F1-score and the loss associated to the validation dataset. Once performance on it does not improve anymore, the training stops.

The training was performed taking as input a file built from the collection of 5186 lines, from a Osquery configuration containing the top ten techniques identified by Red Canary report. The input file was composed exclusively by clean log, without possible threats or attacks in place. This assumption is fundamental since having a not tampered bases allows the model to understand a baseline of normal behaviour, which will be compared with real-case scenarios in order to perform malicious log detection.

Notice, one more time, that this step is performed only once on server side. In order to having an updated model is possible to build mechanism of *continuous training*, so that the Transformer can continue to learn normal behaviours with respect to the changing of the technology.

```
{
  "epoch": 9.93,
  "learning_rate": 4.0071332176595336e-05,
  "loss": 0.1969,
  "step": 51500
},
{
  "epoch": 10.0,
  "eval_f1": 0.8832630098452882,
  "eval_loss": 0.6176068782806396,
  "eval_runtime": 42.6078,
  "eval_samples_per_second": 2.347,
  "eval_steps_per_second": 2.347,
  "step": 51870
}
```

Figure 7.1. Results of the best model, coming from the 10th epoch, obtained with training process.

7.2.4 Baseline results example

This step consist in obtain some results that are used as reference when analysing real-case scenario logs. Essentially, the pre-trained model obtained in the training is loaded and used to perform predictions. This phase has been explained in Section 6.7.4; performing all steps, two results are obtained:

1. *Baseline score*
2. *Baseline standard deviation*

These results are obtained by taking into consideration a small quantity of clean logs; the idea is that by considering only legitimate logs, a baseline score is obtained representing the distance between a real, clean, log and the log line predicted.

Results are computed by applying the MLM task, with an innovative approach. Indeed, as explained in Sectionsec:baseline, only one token per time is masked giving in input the rest of the sentence. The goal is to allow the model to use the whole context except for the masked word. In this way, it may be possible to have better results and prediction on every single token composing a log.

In particular for the considered baseline dataset, made of 91 clean logs collected with different queries, the results are shown in Figure 7.2. Finally, both results will be loaded by the algorithm agent running on user machine and used to perform a comparison with scores obtained during the malicious logs detection analysis.

Notice that the **baseline score** has a great impact on the detection of anomaly logs. It

```
1 {  
2   "max_baseline_score": 4.262022362047548,  
3   "standard_deviation": 0.9665868459871625  
4 }
```

Figure 7.2. Results of the baseline scenario obtained using the best pre-trained model obtained in the training step.

is therefore necessary to obtain a value not too high. There are consequences in that case: if the score is too high, more log will be considered as legitimate, hence there is the risk of not identify borderline malicious logs. On the other hand, a score too low increases the number of false positives detected.

7.2.5 Analysis results example

This operation is the main one executed on the user machine endpoint. The algorithm should run embedded into a Windows service or in a Docker container; depending on the choice the running engine must be created.

The dataset used for the training was built by collecting legitimate logs with all the implemented Osquery rules, generated on a clean Windows operating system. Every collected log, without any process of Predictive Analysis, would be sent to the SIEM application, resulting on a big amount of data server side that need to be analysed by both automatic criteria and analysts; this is the False Positive problem. The algorithm proposed performs a screening operating that would allow to send only really suspicious logs to the SIEM, in particular those one that are far away from the *normal* shape of a log.

Logs generated by attack techniques are detected and they receive a higher **alert score** with respect to normal one. Such a score can be visualized on the final OpenSearch Dashboard which allows also to create indices on collected data, dashboards, filter them by fields and data. In Figure 7.3 is shown the structure of an anomaly log; the template is used to store in a JSON format every detected anomaly that, if respected, allows automatic field recognition and parsing on the visualization web application.

Since the collection of log used for training was the results of many queries, the model should be able to perform analysis on any of the analysed technique and pattern. Experimental results, hence, are tested against a mixture of logs generated through tests performed by Atomic Red Team tool.

Since the prediction principle is always the same, hence logs receiving a higher score are considered as malicious, in this section are presented some results with a discussion of them and possible improvements. A selection of 156 logs coming from all packs of query is selected, in order to give insights of what actual results are. This selection will be referenced as *dirty*.

Notice that in this section are considered **alert score** computed on that log, before they are compared with the baseline values; in this way, for understanding purposes, it would be possible to focus on actual score, without the influence of the standard deviation.

```

with open('outputs-v_2_0/T2-ALL/identified_threats.json', 'w') as fjson:
    for index in anomaly_indices:
        anom_json = {
            'words': tokenizer.decode(tokens['input_ids'][index]),
            'words_score': all_anomaly_scores[index],
            'alert_score': alert_scores[index],
            'matched_query': tokenizer.decode(tokens['input_ids'][index][3])
        }
        json.dump(anom_json, fp=fjson)

```

Figure 7.3. Template of a JSON object used to store a detected anomaly in an output file.

Case of study – Pack 1

In the training dataset were collected a total of 1340 logs from Pack 1, where the greatest population was divided among 530 query A and 806 query C; the remaining are split as two query B and two query F. In this scenario is possible to think that there was a pretty good training around query A and C of the first pack, and hence that malicious one would be detected in a proper way.

The dirty log selection counts 35 logs, composed by different queries coming from the first pack. Notice that since the training set only processed queries A, B, C and F different queries will receive a [UNK] token, but since the model is able to understand the context it would be able to address predictions even in case some token are unknown. Following, are resumed meaningful results:

- Query A and C receive a small score, all under the unit point, meaning that the model correctly predicts those log and recognize them as legitimate; probably they are just normal logs collected in the meanwhile one or more technique was run
- Query F, despite being present in the training set, received a high score, about 5, meaning that all F queries selected would be considered as possible threats and hence they must be sent to OpenSearch for further analysis
- Other queries received a score higher than A and C, since their absence in the training set and hence their distance to the normal baseline log mode

This result is useful to highlights how common log, such query A and C, are recognized and, if predicted correctly, receive a smaller score; also known log type, but not following the prediction pattern, such query F, are identified as anomalous. Finally, query absent in training dataset, such other mentioned types, will have more likely other syntax and value that the algorithm is not able to predict and hence will record them as dangerous.

Case of study – Pack 4

The fourth pack of queries also represents a good population of logs; in the training set 1672 logs were collected, divided as 367 coming from query I, 1202 from query E and

93 from query F. The dirty selection, instead, summarize them with a 15 as population including also other queries, such D and A. Meaningful results are:

- Unknown type of queries, such D, received a score among 3.83 and 4.65; these are boundaries values that may be considered malicious or not depending on the baseline score obtained but, generally speaking, they received a higher score with respect to a medium score
- Well known query types, such I, E and F received a lower score, from about 0.5 and below 2 meaning that they respect the pattern of the model
- There is one log, from query I, particularly meaningful: it has a score of 4.42 which is much higher with respect to other logs from the same query; this is an insight of a malicious log

This case of study shows that the model is able to assign higher score not only to logs not analysed deeply during training, but also to well-known logs with malicious pattern, as explained in the last point of the above list.

Case of study – Pack 5 and Pack 7

Those pack of query are represented from a minimum population of the dataset. In particular, among 5187 log lines 0 pack 7 and only four pack 5 were collected. This may be the case of a rare event, that happens only in certain case and may be most of the time malicious. Also, it may be a consequence of specific queries that are triggered only when some event occurs.

The dirty dataset considered ten logs for both cases. Briefly, discussing those results:

- Concerning pack 5, all associated log received a pretty high score with six out of ten with a score higher than 5.1 and three of them with a score higher then 6.5; thanks to the few samples collected in the training dataset, it was possible to assign an elevate score to more likely malicious log for this pack
- Pack 7, instead, was not trained at all. In such case, the model split the behaviour in two: logs with common pattern, referring to log similar considered in the training, receive a score around 2.6 while more ambiguous sentences receive higher scores, from 4.7 up to 5.24.

This Packs analysis highlights the need of an omni comprehensive training, which receives as reference a dataset properly structured in order to assign scores in a coherent way. Indeed, a problem that Pack 7 suffers is the lack of logs which falls into a bad tokenization and then in a bad analysis. As shown in Figure 7.4, a log coming from pack 7, and identified as anomalous, is actually tainted with many [UNK] tokens. This low understanding of some logs may lead to not accurate score, hence in a less effective algorithm.

Limits and considerations

The final point to be considered for the Transformer algorithm are the limitation of the chosen approach; this algorithm has huge possibilities and may lead to more study and investigation to optimize the implementation, but it has presented some critical points.


```

*** Results ***
Max line anomaly score = 7.181748494749729
Max Alert score = 3.020655769136091
Number of anomalies = 50

```

Figure 7.5. Analysis of 155 dirty logs coming from different packs.

by considering all points explained in the list of this section, the algorithm was able to detect anomalies in every pack query, meaning that, on a real environment, the model should be able to detect anomalies coming from most of the written Threat Intelligence Osquery rules.

7.2.6 OpenSearch Dashboard visualization

The last result that is valuable to be showed is the results visualization on server side, on OpenSearch Dashboard.

The output file generated by the Transformer algorithm in the previous step contains all log events detected as malicious. They are stored in JSON format, following rules identified in Section 7.2.5. The selected format allows the OpenSearch stack to automatically parse them, permitting OpenSearch Dashboard to recognize each field and, hence, visualizing it. In Figure 7.6 is shown an example of malicious logs visualization obtained by the *Dirty* use case presented in Section 7.2.5. In Figure 7.7 is show how logs are visualized in OpenSearch

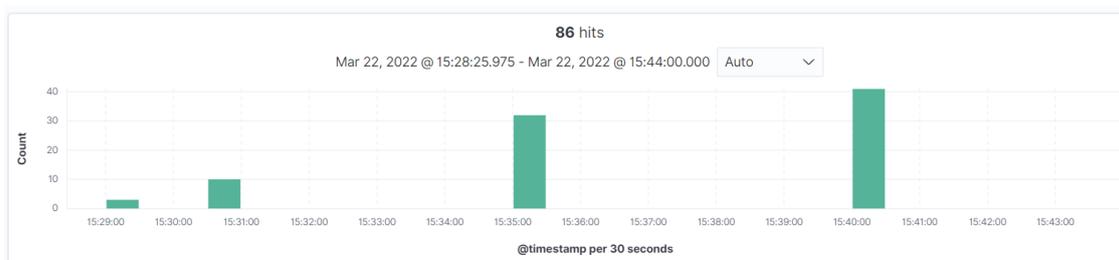


Figure 7.6. Example of log visualization in OpenSearch Dashboard web application.

Dashboard; in Figure A it is shown a preview of two logs and the main fields, both data and metadata automatically added. In Figure B, instead, it is the expanded view of a log: the couples showed is actually written in JSON format, automatically read and visualized by OpenSearch Dashboard.

These graphic capabilities can be further exploited by the creation of custom dashboard, for a better visualization. Also, many further analysis can be performed leveraging all feature of the SIEM application, such for example:

- Creation of alerts that watch for automatic parsed JSON fields

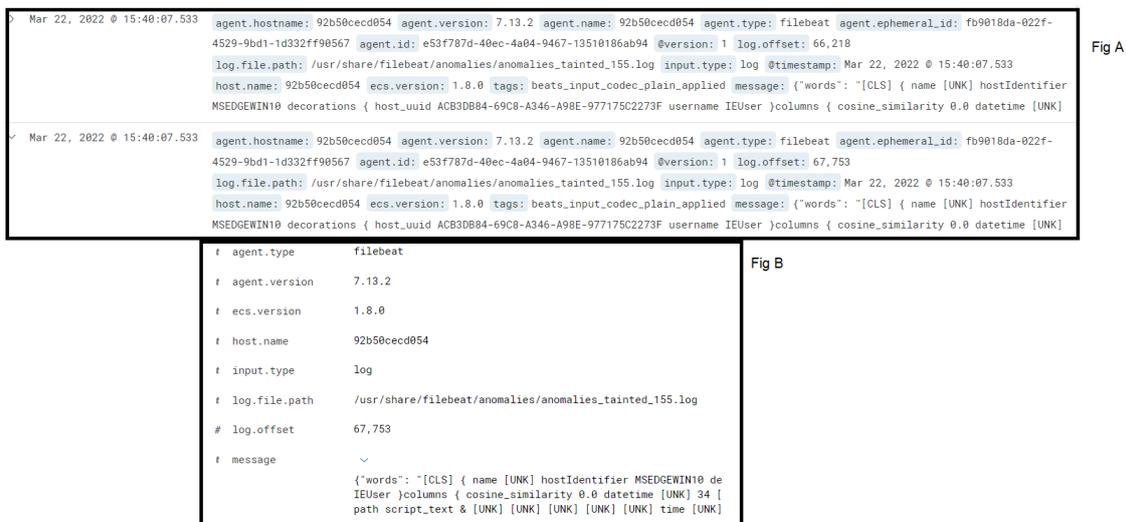


Figure 7.7. Example of log visualization in OpenSearch Dashboard web application.

- Creation of triggers in response to raised alerts, along with automatic email or notification signals
- Creation of indices to automatically parse and create collection of input data
- Exploiting Machine Learning plugins of OpenSearch, in order to perform events correlations and deeper analysis

Concluding, OpenSearch is a young and alive project, which is developed and released by Amazon, one of the most important enterprise in the whole world. Despite being released by less than two years, OpenSearch offers many features that can be exploited to build a fully free and open Threat Hunting application, also thanking the solid base provided by Elasticsearch project.

It would be a pity not exploiting such tools and possibilities to try levelling up the security infrastructure of public and private stakeholders, that would cause, even indirectly, a general services improvement to every Italian citizen.

Chapter 8

Conclusions and Future Work

8.1 Cyber Threat Intelligence conclusions

Both research and experiments, have underlined interesting aspects about Cyber Threat Intelligence and Intrusion Detection.

The theoretical research conducted in the beginning, has been useful to get a wide-open idea about Cyber Threat Intelligence. In particular, the importance of a common representation of the information and a common sharing language, that together are fundamental to improve security posture of each actor, have been underlined. Moreover, it has been clear that frameworks like MITRE ATT&CK are key elements to organize knowledge and make it available for proactive detection activities. The maintenance and development of this type of framework, is crucial for future the future of cybersecurity.

Another theme, above which this thesis is based, is the open source software. Open source has a lot of potential advantages and, as demonstrated in the contribution work of this thesis, powerful tools like Osquery or Atomic Red Team are available also without licenses costs, and give many possibilities to develop new ideas and, to eventually share them.

The contribution, consisting of a series of queries for ATT&CK techniques' detection, introduces a very schematic way to recognize adversary's moves and, could be an help to speed up mitigation procedures. The idea in fact, is that if the defender is able to understand quicker what kind of technique has been applied to perform an attack, he can consequently respond faster to this attack. This because known countermeasures for that technique, given that the technique is known right away after the attack, can be applied immediately. More than this, assuming that the idea will be extended to create a bigger framework, information about attacks can be stored and analyzed, statistics about techniques' usage can be prepared and new countermeasures can be implemented. The number of advantages, if the proposed framework will be further developed and successively used in practice, is consistent.

Tests already executed on this first implementation, are encouraging. Although prepared detection queries have not always been tested in their totality, because some of them are, in some sense, incompatible with Atomic Red Team, the greater part of them has logged the expected results and thus demonstrated to be useful. Logs subdivided by technique, given that they contain a richer information, may also result better to be used for

Machine Learning training: this is the idea that led to the decision to take outputs from this first half of the thesis, and use them as the input for the second.

Last crucial aspect, is related to the importance of updating the queries during time: threat's landscape evolves constantly and detection instruments must evolve in the same direction, also anticipating adversaries when possible. Cyber Threat Intelligence information sharing, plays a key role for this purpose.

This work can be imagined as a brick inside a bigger cybersecurity framework where, together with other fundamental components, contributes to the general defence of an infrastructure.

8.1.1 Future Work

Considering the contribution presented as a first implementation, the possibility to improve and add functionalities to the current version, comes by itself.

Two basic activities that could be done are: improving the detection capabilities of the already existent query packs, and testing of their effectiveness with more than one benchmark. Improvements can be done adding more sources of information, and consequently writing new queries while refining old ones. This would broaden the range of detectable behaviours and strengthen the detection capabilities. The second aspect regarding the tests, means that other red teaming framework, similar to Atomic Red Team, may be employed to see if they cover a broader spectre of behaviours. This will make it possible to test a greater number of queries.

The second main work that could be achieved, is the inclusion of a larger number of techniques inside the framework. The contribution of this thesis, is limited to a subset of MITRE ATT&CK techniques contained inside a threat detection report. This is a considerable limit and, with a view of making the idea behind this work concretely applicable in the real world, an increased set of techniques, also if not the whole available list, is surely needed. Going in this direction, it is impossible to leave apart the fact that some of the techniques are not easily mappable to SQL queries and probably, some of them would remain out of the framework. Once enlarged the framework, testing it on production environments would surely be a good idea.

An additional functionality that may be added, probably better if at a more mature stage, is the combination of system logs, on which the job has already been started, with other data sources, like application logs or network traffic information. This would probably increase dramatically the detection performances even if a tough effort may be required for the implementation.

Last idea for future developments, resumes a concept deeply discussed in the background: the threat intelligence information sharing. STIX's integration in the framework, probably in the form of logs information organization and representation after detection, will be a significant improvement. This integration, would allow sharing of organized and commonly understandable information about respective known threats or suffered attacks, between different actors.

8.2 Threat Hunting conclusions

Nowadays, defenders and attackers evolved with technology. They chase each other in a constant race, one trying to build effective security mechanisms while the other attempt to breach them. Attackers, over past few years, evolved in strength and resources, sometimes also due to government sponsorship; this is the case of *APT* – (*Advanced Persistent Threat*) which are criminal groups of hackers, with significant resources and experience, that target IT infrastructure and public administration, attempting to breach network, steal data and damage systems. Indeed they do not target the end user directly, which instead may suffer some collateral damage. The particularity of these groups are the advanced type of attacks, which are usually performed in a wide window of time and, once the network or the supply chain of a target are compromised, the attacker remain stealth. This allows the enemy to collect data and network information, that can be used later for privileges escalation. The attacker may remain stealth in the victim network for long time, spreading the attack and enhancing its position, until the actual attack take place, by executing operations to damage the service or exfiltrate sensible data.

Another popular malware scenario is represented by Ransomware, which are dangerous malware that encrypt data on the victim’s system with the promise of unlocking them under the payment of a ransom. For large enterprise and public administration it is crucial to protect their assets and customer information from being encrypted and stolen, so a protection against those attack is mandatory.

The contribution consists in analysing which are the main security measures available to security defence teams, with a focus on *SIEM* – (*Security Information and Event Management* systems and *EDR* – *Endpoint Detection and Response* agents. The discussion addresses both the enterprise and open source landscape, by analysing performance and capabilities of both *EDR* and *SIEM* tools and pointing out their limitation. In particular, results highlight that no tool, nor open source or enterprise level, is able to provide a complete and effective protection against all kinds of attacks. Especially new advanced threats, in particular *APT* attacks and Ransomware, are able to breach common security defence techniques such anti-viruses and firewalls.

Standard security mechanisms, exploiting traditional approaches, are typically heavily rule based, meaning that they apply a static strategy to detect malware and threats. Approaches based on specific path, executable or hashes are fundamental, but their effectiveness is heavily limited when new and unknown threats are faced.

Standard security tools are simply not enough anymore; new *EDR* and *SIEM* systems must be adopted, improving them with *Predictive Analysis* concepts that would allow to discover and detect malicious behaviour even on unknown threats. *Predictive Analysis* consists in leveraging Artificial Intelligence and Machine Learning capabilities, behavioural analysis and, in general, implementing multi-techniques prevention systems. Also the adoption of shared knowledge framework, such MITRE ATT&CK, CVE and NVD, are key points in the protection against *APTs*, allowing stakeholders to maintain their IT perimeter up to date with the most advanced cybersecurity defence mechanisms.

Given all previous considerations and analysis, this work proposes an architecture consisting in a pipeline of operation and software that goes from user end point to a centralised events analysis and management system.

The pipeline begins with a focus on the end point. Event logs collected exploiting Osquery

capabilities and are processed by an algorithm, proposed in this work, leveraging Predictive Analysis through Machine Learning technology. The proposed algorithm is based on *BERT*, a Transformer Bidirectional Encoder designed for the pre-training of Machine Learning models, released by Google; this model is able to receive in input any kind of data, which text was previously mapped to numeric tokens, and learn the context of each word by bidirectionally encoding them. The main task adopted is the *MLM – Masking Language Model* which consist in masking some word and let the model predict the hidden token, by analysing both directions. This approach allows *BERT* to earn knowledge of the context of each word; the output generated is an embedding of values that can be exploited to determine if the predicted token was the actual one. In this way, the proposed algorithm analyses event logs and tries to detect which are the anomalous one.

The next step is the shipping of detected anomalous events to the centralized management system, represented by OpenSearch stack, where they can be further analysed and inspected for possible threats. OpenSearch is a novel open source project, released and maintained by Amazon, that offers a search engine management system, for indexing and storing of logs, and a visualization tool named OpenSearch Dashboard accessed via web browser interface, offering many feature of a *SIEM* environment.

8.2.1 Future Work

The proposed architecture represents a first step in building an open source solution for Threat Intelligence and Detection, in order to create a transparent and free environment, for both public and private stakeholder.

Future works may be applied on the two side of the proposed architecture: end point and server side, where OpenSearch stack runs.

OpenSearch is a novel framework with great offering; in order to improve visualization of collected events, it is possible to create dashboards exploiting different graphic features. Logs can be indexed and analysed in detail in order to continue threat analysis. OpenSearch Dashboard offers a graphical interface where is possible to set automatic policies and alerts, for automatic scan and reaction in case of boundaries and constraints are violated. Finally, the tool offers many advanced plugins that can be exploited in order to build a complete and secure environment for log events shipping and analysis.

On the end point the main improvements can be performed on the proposed algorithm. Firstly, a general improvement of the algorithm performance and optimization can be performed. The analysis procedure can be enhanced by deeply studying the structure of collected logs, introducing, for example, weights on tokens and improving the scoring system. The algorithm can be embedded in a service or in a Docker container that should be run on target system for continuous analysis. Finally, the most important and crucial step, would be the creation of an enterprise level dataset; being able to perform the training of the Transformer model on a complete dataset would allow the algorithm to learn a baseline of normal behaviours which is then exploited to perform threat detection processes. Actually, the creation of such base of knowledge is not a one-time action; indeed, to remain up to date to the most advanced threat techniques, it is necessary to create a continuous training process that would allow the model to be constantly updated.

Appendix A

The OpenSearch project



Figure A.1. OpenSearch logo.

Before entering in detail about OpenSearch architecture, it is necessary to give an introduction about its project and how it was released. Amazon firstly announced the OpenSearch project in January 2021 [50], after the statement of Elastic company to move the license of their products to a proprietary version. This choice was taken in order to continue supporting the open source community, with a tool that obtained a great support and a wide usage in many fields. In April 2021, Amazon introduced officially OpenSearch in an article [49] describing which product will be included in the framework:

- *OpenSearch*: a fork of Elasticsearch version 7.10.2, the latest open source distribution; the core of the whole framework, being both a database and search engine;
- *OpenSearch Dashboards*: a fork of Kibana version 7.10.2, the latest open source distribution; the visualization engine that permits to users to perform analysis, create reports and triggers.

Furthermore, OpenSearch inherited many features from another project derived from Elasticsearch, named Open Distro for Elasticsearch, such enterprise security, alerting, Machine Learning capabilities and other features. Leveraging all these new capabilities, and Elasticsearch's natives one, the tool can be used to *easily ingest, secure, aggregate, view and analyse data*. Possible use cases are *search of specific data leveraging the non-relational database model, log analytics, log visualization and many other*.¹

Amazon's project collected a wide support provided by the community, also including many important organizations [49], which took an explicit position in this license change of Elasticsearch, such:

¹<https://opensearch.org/>

- *Red Hat*, declared “we believe in the power of open source, and that community collaboration is the best way to build software.”;
- *SAP*, one of the most important companies in developing enterprise software solutions stated “Our observability strategy uses Elasticsearch as a major enabler. OpenSearch provides a true open source path and community-driven approach to move this forward.”;
- *Capital One*, an American bank holding, underlying the importance of an open source license such as the one provided by Apache: “When our teams chose to use Elasticsearch, the freedoms provided by the Apache-v2.0 license was central to that choice. [...]”;
- *Logz.io*, a cloud native observability platform based on Elastic stack, saying, through its CEO Tomer Levy, “We have made a commitment to work with AWS and other members of the community to innovate and enable every organization around the world to enjoy the benefits of these critical open source projects.”

Along with these big companies, many other support the project, along with a wide community of users. AWS, joined with all contributors, can be an insight of the great effort spent on this project; even if OpenSearch forked from a “more ready” solution for enterprise usage, such Elasticsearch, the hope is that, thanks to the overall experience, Amazon will be able to provide an open source platform that can be freely used, modified, extended and monetized by anyone.

Amazon, in OpenSearch main webpage [1](#), published a list of guidelines that are followed during the tool development, which sounds like a promise to the community. The most relevant points are:

- *Open source like we mean it*: no Contributor License Agreement, the whole project is covered by Apache 2.0
- *Used everywhere, with same capabilities*: OpenSearch will not be optimized to work better for any vendor and the goal is to be used by as many people as possible
- *Open to contributors*: anyone can get involved in the project. The goal is to build the software leveraging “the diversity”, hence the different point of views and skills that only a heterogeneous community can lead to

After the releasing of a Beta-version of OpenSearch, the project was released officially on the 5th of October 2021 [[34](#)] and its download was made available for anyone for free.

As an open source project, Amazon also published the code through GitHub repositories, for both OpenSearch² engine and OpenSearch Dashboard³. Other than these shared repositories, the framework can be quickly run with Docker images⁴⁵, allowing a plug&play

²<https://github.com/opensearch-project/OpenSearch>

³<https://github.com/opensearch-project/OpenSearch-Dashboards>

⁴<https://hub.docker.com/r/opensearchproject/opensearch>

⁵<https://hub.docker.com/r/opensearchproject/opensearch-dashboards>

setup. Currently the project is at its version 1.2, and it is constantly maintained and updated as AWS promised.

Appendix B

Open source initiative and Elastic license issues

Since the beginning of this chapter, it is mentioned a change of license from Elastic company and, from there, the decision of Amazon to create a new project on top of the existing one. In this section is introduced the open source foundation, their principle and why Amazon decided to take a step back with respect to Elastic choices, by releasing its own framework.

B.1 The open source initiative



Figure B.1. Open source initiative logo.

The open source initiative (OSI) was born in 1998, in Palo Alto, California¹. OSI was conceived as a general educational and advocacy organization. Since the early stages, a draft of Open Source Definition (OSD) was defined, and it was modified until the actual one. OSD is composed by ten points, which also are the guidelines that every application and software must follow in order to use an open source license. Here are summarized [82]:

1. *Free redistribution*: the license must not restrict any party from selling or providing away a software as a cluster of application containing programs coming from different sources. Furthermore, the license cannot ask for royalties or other fees.
2. *Source Code*: the source code must be fully accessible and public. If the source code is not provided along with the software, then it must be declared a way to obtain it, possibly through Internet and without any charge. In the code are not allowed any obfuscation.
3. *Derived Works*: the license must allow modifications and the creation of other works based on the current one, allowing the distribution of the new product under the same license conditions.
4. *Integrity of the Author's source code*: the license may restrict the source code from being distributed in modified form **only** if the license allows the distribution of patch files with the source code with the goal of modifying the program at build time. The license must allow the distribution of the software with modified code and can also require running derived works to run a different version from the original software.
5. *No discrimination against people or groups*
6. *No discrimination against fields of endeavour*: the license must not restrict anyone from using the software in any field and possible use case.
7. *Distribution of license*: all the rights attached to a program must apply all recipient of the program, without need of additional execution of other licenses
8. *License must not be specific to a product*: rights granted to a program must not depend on the particular software distribution. If the program is extracted from that distribution and used or shared, all parties receiving the software must have applied the same rights.
9. *License must not restrict other software*: a license must not restrict other software distributed along with the targeted application.
10. *License must be technology-neutral*: No provision of the license may be predicated on any individual technology or style of interface.

Just to highlight a key point: *open source is not just about disclosing the source code*. Indeed the OSD states many points about what an open source license must and must not allow. These points lead to the separation in software between Elastic company and Amazon.

¹<https://opensource.org/history>

B.2 Elastic background and new license

OpenSearch is based on the ELK stack, which main components are Elasticsearch, Logstash and Kibana. Those three were originally released under Apache License version 2.0 (ALv2)², that is a recognized open source one. Elasticsearch, in particular, rose in popularity due to its great characteristics and efficiency; for this reason it was used as a base engine in order to develop a lot of software, used in different fields, from simple search and data visualization to log analysis. On top of it, many companies built their product releasing solution for cloud and enterprise application. Just for example, some companies are the already mentioned Amazon³, Logz.io⁴ and Microsoft⁵.

These huge company providers of cloud services and SaaS software released their software leveraging products such Elasticsearch. The Elastic company suffered this concurrency, from a business point of view, and, in the article where they declared the license change [10], they even accused Amazon is making profit on top of open source software without contributing back. Amazon provided, as mentioned before, a SaaS cloud distribution of Elasticsearch, earning profit as “accused” by Elastic and as allowed by OSD. Amazon was injured of other infractions and bad behaviours, but it would be out of the scope a discussion in this work; anyway, an article is referenced [9] for sake of completeness.

On the other end, the Elastic decision was completely legitimate, with the goal of increasing their business and, at the same time, reducing incomes of cloud provider services. Indeed, it was just a matter of the Elastic business model to be incompatible with the open source one, and it is fine. They chose to allow developers to select among two licenses, the Server Side Public License (SSLP)⁶ or the Elastic License 2.0 (ELv2)⁷. Those two licenses are not approved by open source initiative, with the request of SSLP being rejected by OSI; indeed they add some constraints to the software which are in contrast with OSD. In particular, briefly analysing them:

- *Elv2*: it is a non-copyleft license, which allows to use, copy, distribute, make available and prepare derivative works of the software, but with three restrictions:
 1. Distribute the software to other parties as a managed service, hence is not possible provide SaaS solutions as before, being in contrast with OSDs 1 and 6;
 2. Circumvent the license key functionality or remove/obscure features protected by license keys;
 3. Remove or obscure any licensing, copyright or other notices.

²<https://opensource.org/licenses/Apache-2.0>

³now renamed to OpenSearch, <https://aws.amazon.com/it/opensearch-service/>

⁴<https://logz.io/>

⁵<https://azure.microsoft.com/it-it/overview/linux-on-azure/elastic/>

⁶<https://www.mongodb.com/licensing/server-side-public-license>

⁷<https://www.elastic.co/licensing/elastic-license>

- *SSPL*: released by MongoDB Inc organization associated with the homonymous database engine. This license is copyleft and source-available; actually it is similar to an open source one, with one major modification, in particular on article 13 which states: “*the Corresponding Source for the Program or the modified version, and the Corresponding Source for all programs that you use to make the Program or modified version available as a service, including, without limitation, management software, user interfaces, application program interfaces, automation software, monitoring software, backup software, storage software and hosting software, all such that a user could run an instance of the service using the Service Source Code you make available.*”

To summarize, Elv2 mandates the need of an agreement between Elastic and the company that wants to provide a SaaS software based on their products, while SSPL requires that all the source code of the program and of **all** other software, backups, associated or support software and so on are publicly disclosed; this constraints is unacceptable for an enterprise. Notice that Elastic change does not affect the majority of users; indeed, they can continue to use their product freely ensuring to respect licenses constraints. The problem is that, as stated by OSI in an article [84], a company cannot claim or imply to provide open software when are applied licenses not compliant to OSD.

This technical overview was provided to explain why, for this work, was selected a truly open source software, such OpenSearch, in place of the ELK stack, after its license changes.

Bibliography

- [1] *Anatomy of Advanced Persistent Threats*. FireEye. URL: <https://www.fireeye.com/current-threats/anatomy-of-a-cyber-attack.html>.
- [2] Alon Arvatz. *Evolution of the Cyber Threat Intelligence Practice*. [Online; accessed 15-March-2022]. 2021. URL: [https://www.rapid7.com/blog/post/2021/08/25/revolution-of-the-cyber-threat-intelligence-practice/#:~:text=The%20cyber%20threat%20intelligence%20\(CTI, but%20also%20methodologies%20and%20concepts.%20&text=This%20change%20requires%20a%20shift%20in%20both%20mindset%20and%20methodology..](https://www.rapid7.com/blog/post/2021/08/25/revolution-of-the-cyber-threat-intelligence-practice/#:~:text=The%20cyber%20threat%20intelligence%20(CTI, but%20also%20methodologies%20and%20concepts.%20&text=This%20change%20requires%20a%20shift%20in%20both%20mindset%20and%20methodology..)
- [3] Elchin Asgarli and Eric Burger. «Semantic ontologies for cyber threat sharing standards». In: *2016 IEEE Symposium on Technologies for Homeland Security (HST)*. 2016, pp. 1–6. DOI: [10.1109/THS.2016.7568896](https://doi.org/10.1109/THS.2016.7568896).
- [4] Red Canary authors. [Online; accessed 15-March-2022]. 2021. URL: <https://redcanary.com/threat-detection-report/>.
- [5] Various authors. *Atomic Red Team*. [Online; accessed 15-March-2022]. URL: <https://github.com/redcanaryco/atomic-red-team>.
- [6] Various authors. *invoke-atomicredteam*. [Online; accessed 15-March-2022]. URL: <https://github.com/redcanaryco/invoke-atomicredteam>.
- [7] Kurt Baker. *WHAT IS CYBER THREAT INTELLIGENCE?* [Online; accessed 15-March-2022]. 2021. URL: <https://www.crowdstrike.com/cybersecurity-101/threat-intelligence/>.
- [8] Roberto Baldoni, Paolo Ernesto Prinetto, and Raffaele Angius. *Roberto Baldoni / Il punto sul Perimetro Nazionale / ITASEC21*. [Online; accessed 16-March-2022]. 2021. URL: <https://www.youtube.com/watch?v=y50Tew4I8zc&list=WL&index=20>.
- [9] Shay Banon. *Amazon: NOT OK - why we had to change Elastic licensing*. <https://www.elastic.co/blog/why-license-change-aws>. [Online, accessed 10-March-2022].
- [10] Shay Banon. *Doubling down on open, Part II*. <https://www.elastic.co/blog/licensing-change>. [Online, accessed 10-March-2022].
- [11] Daniel Berman. *11 Open Source SIEM Tools*. Logz.io. 2019. URL: <https://logz.io/blog/open-source-siem-tools/>.

- [12] Eric W. Burger, Michael D. Goodman, Panos Kampanakis, and Kevin A. Zhu. «Taxonomy Model for Cyber Threat Intelligence Information Exchange Technologies». In: *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security*. WISCS '14. Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 51–60. ISBN: 9781450331517. DOI: [10.1145/2663876.2663883](https://doi.org/10.1145/2663876.2663883). URL: <https://doi.org/10.1145/2663876.2663883>.
- [13] Amy Burnis. *The Cyber Attack Lifecycle*. <https://www.cyberark.com/resources/blog/video-the-cyber-attack-lifecycle>. [Online, accessed 12-March-2022].
- [14] Gianluca Cattani, Fabio Mulazzani, and Massimo Vegni. *L'importanza del Perimetro di Sicurezza Nazionale Cibernetico per la competitività delle aziende italiane*. [Online; accessed 16-March-2022]. 2021. URL: <https://www.riskcompliance.it/news/l-importanza-del-perimetro-di-sicurezza-nazionale-cibernetico-per-la-competitivita-delle-aziende-italiane/>.
- [15] Andre Correa CEO. *Three Types of Cyber Threat Intelligence*. [Online; accessed 15-March-2022]. URL: <https://www.malwarepatrol.net/three-types-of-cyber-threat-intelligence/>.
- [16] *Che cos'è la Threat Intelligence?* [Online; accessed 15-March-2022]. URL: <https://www.anomali.com/it/resources/what-is-threat-intelligence>.
- [17] Staff Contributor. *What Is Threat Intelligence? Definition and Types*. [Online; accessed 15-March-2022]. 2020. URL: <https://www.dnsstuff.com/what-is-threat-intelligence>.
- [18] CVE. CVE. URL: <https://www.cve.org/ResourcesSupport/Glossary?activeTerm=glossaryBoard#>.
- [19] *CVE Record Lifecycle*. <https://www.cve.org/About/Process>. [Online, accessed 10-March-2022].
- [20] Cynet. Cynet. URL: <https://www.cynet.com/zero-day-attacks/zero-day-vulnerabilities-exploits-and-attacks-a-complete-glossary/#what-is-a-zero-day-attack>.
- [21] Cyware. *The Evolution of Threat Intelligence*. [Online; accessed 15-March-2022]. 2021. URL: <https://cyware.com/educational-guides/cyber-threat-intelligence/the-evolution-of-threat-intelligence-fdba>.
- [22] Roman Daszczyzak, Dan Ellis, Steve Luke, and Sean Whitley. *TTP-Based Hunting*. Tech. rep. MITRE CORP MCLEAN VA, 2019.
- [23] *Definition of ontology*. [Online; accessed 15-March-2022]. URL: <https://www.merriam-webster.com/dictionary/ontology>.
- [24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «Bert: Pre-training of deep bidirectional transformers for language understanding». In: *arXiv preprint arXiv:1810.04805* (2018). DOI: <https://doi.org/10.48550/arXiv.1810.04805>.
- [25] *Docker overview*. <https://docs.docker.com/get-started/overview/>. [Online, accessed 13-March-2022].

- [26] Christina Feilmayr and Wolfram Wöß. «An analysis of ontologies and their success factors for application to business». In: *Data & Knowledge Engineering* 101 (2016), pp. 1–23.
- [27] Alessio Fodedri. *Tutti gli attacchi hacker alle banche degli ultimi anni*. [Online; accessed 15-March-2022]. 2021. URL: <https://dealflower.it/tutti-gli-attacchi-hacker-alle-banche-degli-ultimi-anni/>.
- [28] Joint Task Force. «Security and Privacy Controls for Information Systems and Organizations». In: *NIST CSRC* 5 (2020). DOI: <https://doi.org/10.6028/NIST.SP.800-53r5>.
- [29] *Gartner names Microsoft a Leader in the 2021 Endpoint Protection Platforms Magic Quadrant*. <https://www.microsoft.com/security/blog/2021/05/11/gartner-names-microsoft-a-leader-in-the-2021-endpoint-protection-platforms-magic-quadrant/>. [Online, accessed 14-March-2022].
- [30] Dave Gruber. «The Next Big Advancement in Detection and Response: Predictive Analytics». In: *Enterprise Strategy Group* (2019).
- [31] Thomas R Gruber. «Toward principles for the design of ontologies used for knowledge sharing». In: *International journal of human-computer studies* 43.5-6 (1995), pp. 907–928.
- [32] *Has the Notorious Carbanak Cybercrime Spree Reached its Finale? What You Need to Know About Stopping the Next APT Attack*. Exabeam. URL: <https://www.exabeam.com/information-security/has-the-notorious-carbanak-cybercrime-spree-reached-its-finale-what-you-need-to-know-about-stopping-the-next-apt-attack/>.
- [33] Denny Hermawan, Nugroho Ganda Novianto, and Digit Octavianto. «Development of Open Source-based Threat Hunting Platform». In: *2021 2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS)*. 2021, pp. 1–6. DOI: [10.1109/AiDAS53897.2021.9574308](https://doi.org/10.1109/AiDAS53897.2021.9574308).
- [34] Andrew Hopp, Charlotte Henkle, Eli Fisher, and Kyle Davis. *OpenSearch 1.1.0 is here!* <https://opensearch.org/blog/releases/2021/10/Launch-Announcement-1-1/>. [Online, accessed 10-March-2022].
- [35] Cathal Horan. *10 Things You Need to Know About BERT and the Transformer Architecture That Are Reshaping the AI Landscape*. <https://neptune.ai/blog/bert-and-the-transformer-architecture-reshaping-the-ai-landscape>. [Online, accessed 15-March-2022].
- [36] *How Filebeat works*. <https://www.elastic.co/guide/en/beats/filebeat/7.10/how-filebeat-works.html>. [Online, accessed 12-March-2022].
- [37] Ghaith Husari, Ehab Al-Shaer, Bill Chu, and Ruhani Faiheem Rahman. «Learning APT Chains from Cyber Threat Intelligence». In: *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*. HotSoS '19. Nashville, Tennessee, USA: Association for Computing Machinery, 2019. ISBN: 9781450371476. DOI: [10.1145/3314058.3317728](https://doi.org/10.1145/3314058.3317728). URL: <https://doi.org/10.1145/3314058.3317728>.

- [38] Michele Iaselli. *Sicurezza nazionale cibernetica: il decreto-legge coordinato in Gazzetta*. [Online; accessed 15-March-2022]. 2019. URL: <https://www.altalex.com/documents/leggi/2019/11/25/sicurezza-nazionale-cibernetica-decreto-legge-coordinato-gazzetta>.
- [39] Luke Irwin. *The Benefits of Cyber Threat Intelligence and Information Sharing*. [Online; accessed 15-March-2022]. 2018. URL: <https://www.itgovernance.co.uk/blog/5-reasons-why-sharing-threat-intelligence-makes-you-more-cyber-secure>.
- [40] Scott Jasper. «Biden Orders Endpoint Detection and Response (EDR) Initiative». In: *United States Cybersecurity Magazine* (2021). DOI: <http://hdl.handle.net/10945/68005>.
- [41] George Karantzas and Constantinos Patsakis. «An empirical assessment of endpoint detection and response systems against advanced persistent threats attack vectors». In: *Journal of Cybersecurity and Privacy* 1.3 (2021), pp. 387–421. DOI: <https://doi.org/10.3390/jcp1030021>.
- [42] Artem Karasev. *How threat intelligence evolved, and where it will go next*. [Online; accessed 15-March-2022]. 2020. URL: <https://www.kaspersky.com/blog/secure-futures-magazine/threat-intelligence-trends/35109/>.
- [43] Kaspersky. *Threat Intelligence Definition. Why Threat Intelligence Is Important for Your Business and How to Evaluate a Threat Intelligence Program*. [Online; accessed 15-March-2022]. URL: <https://www.kaspersky.com/resource-center/definitions/threat-intelligence>.
- [44] *LEGGE 18 novembre 2019, n. 133*. [Online; accessed 10-March-2022]. 2019. URL: <https://www.gazzettaufficiale.it/eli/id/2019/11/20/19G00140/sg>.
- [45] Vasileios Mavroeidis and Siri Bromander. «Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence». In: *2017 European Intelligence and Security Informatics Conference (EISIC)*. 2017, pp. 91–98. DOI: [10.1109/EISIC.2017.20](https://doi.org/10.1109/EISIC.2017.20).
- [46] McAfee. *What Is a Security Operations Center (SOC)?* [Online; accessed 15-March-2022]. URL: [https://www.mcafee.com/enterprise/en-us/security-awareness/operations/what-is-soc.html#:~:text=A%20Security%20Operation%20Center%20\(SOC, and%20responding%20to%20cybersecurity%20incidents..](https://www.mcafee.com/enterprise/en-us/security-awareness/operations/what-is-soc.html#:~:text=A%20Security%20Operation%20Center%20(SOC, and%20responding%20to%20cybersecurity%20incidents..)
- [47] McAfee. *What Is Endpoint Detection and Response (EDR)?* [Online; accessed 15-March-2022]. URL: <https://www.mcafee.com/enterprise/en-us/security-awareness/endpoint/what-is-endpoint-detection-and-response.html>.
- [48] McAfee. *What Is the MITRE ATT&CK Framework?* [Online; accessed 15-March-2022]. URL: <https://www.mcafee.com/enterprise/it-it/security-awareness/cybersecurity/what-is-mitre-attack-framework.html>.
- [49] Carl Meadows, Jules Graybill, Kyle Davis, and Mehul Shah. *Introducing OpenSearch*. <https://aws.amazon.com/it/blogs/opensource/introducing-opensearch/>. [Online, accessed 10-March-2022].

- [50] Carl Meadows, Jules Graybill, Kyle Davis, and Mehul Shah. *Stepping up for a truly open source Elasticsearch*. <https://aws.amazon.com/it/blogs/opensource/stepping-up-for-a-truly-open-source-elasticsearch/>. [Online, accessed 10-March-2022].
- [51] Stefano Mele. *Cosa c'è nel nuovo Dpcm del perimetro cyber. L'analisi di Mele*. [Online; accessed 16-March-2022]. 2021. URL: <https://formiche.net/2021/06/cosa-ce-nel-nuovo-dpcm-del-perimetro-cyber-lanalisi-di-mele/>.
- [52] Yazid Merah and Tayeb Kenaza. «Ontology-Based Cyber Risk Monitoring Using Cyber Threat Intelligence». In: *The 16th International Conference on Availability, Reliability and Security*. ARES 2021. Vienna, Austria: Association for Computing Machinery, 2021. ISBN: 9781450390514. DOI: [10.1145/3465481.3470024](https://doi.org/10.1145/3465481.3470024). URL: <https://doi.org/10.1145/3465481.3470024>.
- [53] *MITRE ATT&CK*. [Online; accessed 15-March-2022]. URL: <https://attack.mitre.org/>.
- [54] Filippo Mottini. *osquery-attck*. [Online; accessed 15-March-2022]. URL: <https://github.com/teoseller/osquery-attck>.
- [55] MS-ISAC. «Ransomware Guide, September 2020». In: *MS-ISAC* (2020). DOI: <https://www.cisa.gov/stopransomware/ransomware-guide>.
- [56] Giampiero Muccioli. *Perimetro di sicurezza nazionale, una delicata questione legislativa*. [Online; accessed 16-March-2022]. 2022. URL: <https://www.cybersecurity360.it/outlook/perimetro-di-sicurezza-nazionale-una-delicata-questione-legislativa/>.
- [57] *Natural Language Processing (NLP)*. <https://www.ibm.com/cloud/learn/natural-language-processing>. [Online, accessed 14-March-2022].
- [58] Chris Nicholson. *A Beginner's Guide to Attention Mechanisms and Memory Networks*. <https://wiki.pathmind.com/attention-mechanism-memory-network>. [Online, accessed 14-March-2022].
- [59] NIST. *Guide to Cyber Threat Information Sharing*. [Online; accessed 15-March-2022]. 2016. URL: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-150.pdf>.
- [60] NVD Nist. NVD. URL: <https://nvd.nist.gov/general>.
- [61] Oasis-CTI-TC. [Online; accessed 15-March-2022]. URL: <https://oasis-open.github.io/cti-documentation/>.
- [62] Oasis-CTI-TC. [Online; accessed 15-March-2022]. URL: <https://oasis-open.github.io/cti-documentation/stix/intro>.
- [63] Oasis-CTI-TC. [Online; accessed 15-March-2022]. URL: <https://oasis-open.github.io/cti-documentation/stix/examples>.
- [64] Cyril Onwubiko. «CoCoo: An Ontology for Cybersecurity Operations Centre Analysis Process». In: *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. 2018, pp. 1–8. DOI: [10.1109/CyberSA.2018.8551486](https://doi.org/10.1109/CyberSA.2018.8551486).

-
- [65] *Osquery documentation*. [Online; accessed 15-March-2022]. URL: <https://osquery.readthedocs.io/en/stable/>.
- [66] Tushar Panhalkar. *Benefits of Cyber Threat Intelligence*. [Online; accessed 15-March-2022]. URL: <https://info-savvy.com/benefits-of-cyber-threat-intelligence/>.
- [67] Tushar Panhalkar. *Types of Threat Intelligence*. [Online; accessed 15-March-2022]. URL: <https://info-savvy.com/types-of-threat-intelligence/>.
- [68] Ayush Pant. *Introduction to Machine Learning for Beginners*. <https://towardsdatascience.com/introduction-to-machine-learning-for-beginners-eed6024fdb08>. [Online, accessed 14-March-2022].
- [69] Paul Pols. «The Unified Kill Chain». In: (2017).
- [70] Davide Lo Prete. *Perimetro di sicurezza nazionale cibernetica: regole e criteri di attuazione*. [Online; accessed 16-March-2022]. 2020. URL: <https://www.cybersecurity360.it/cybersecurity-nazionale/perimetro-di-sicurezza-nazionale-cibernetica-regole-e-criteri-di-attuazione/>.
- [71] *Search Engines*. <https://db-engines.com/en/article/Search+Engines>. [Online, accessed 11-March-2022].
- [72] R. Shirey. «Internet Security Glossary, Version 2». In: *RFC* (2007). DOI: [10.17487/RFC4949](https://doi.org/10.17487/RFC4949).
- [73] VN Son, LDA Tuan, NT Lam, PT Thuong, DX Anh, and TV Nikolaevich. «Detecting behavior of malware using mitre att&ck». In: *International Journal of Advanced Trends in Computer Science and Engineering* 9.5 (2020), pp. 8285–8294.
- [74] Enrico Sorano, Anna Guerrieri, and Alberto Sardi. «Capire il rischio cyber il nuovo orizzonte in sanità». In: (2021). [Online; accessed 15-March-2022].
- [75] Kai Steverson, Caleb Carlin, Jonathan Mullin, and Metin Ahiskali. «Cyber Intrusion Detection using Natural Language Processing on Windows Event Logs». In: *2021 International Conference on Military Communication and Information Systems (ICMCIS)*. IEEE. 2021, pp. 1–7. DOI: [10.1109/ICMCIS52405.2021.9486307](https://doi.org/10.1109/ICMCIS52405.2021.9486307).
- [76] Răzvan Stoleriu, Alin Puncioiu, and Ion Bica. «Cyber Attacks Detection Using Open Source ELK Stack». In: *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. 2021, pp. 1–6. DOI: [10.1109/ECAI52376.2021.9515120](https://doi.org/10.1109/ECAI52376.2021.9515120).
- [77] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. «Mitre att&ck: Design and philosophy». In: *Technical report* (2018).
- [78] Blake E Strom, Joseph A Battaglia, Michael S Kemmerer, William Kupersanin, Douglas P Miller, Craig Wampler, Sean M Whitley, and Ross D Wolf. «Finding cyber threats with ATT&CK-based analytics». In: *The MITRE Corporation, Bedford, MA, Technical Report No. MTR170202* (2017).
- [79] Zareen Syed, Ankur Padia, Tim Finin, Lisa Mathews, and Anupam Joshi. «UCO: A Unified Cybersecurity Ontology». In: Feb. 2016.

- [80] Claudio Telmon. *Sanità italiana sotto attacco cyber: ecco come reagire all'emergenza*. [Online; accessed 15-March-2022]. 2021. URL: <https://www.cybersecurity360.it/nuove-minacce/sanita-italiana-sotto-attacco-cyber-ecco-come-reagire-allemergenza/>.
- [81] *The benefits of Cyber Threat Intelligence for your organization*. [Online; accessed 15-March-2022]. 2021. URL: <https://www.infoguardsecurity.com/the-benefits-of-cyber-threat-intelligence-for-your-organization/>.
- [82] *The Open Source Definition*. <https://opensource.org/docs/osd>. [Online, accessed 10-March-2022].
- [83] *The Red Canary Origin Story*. [Online; accessed 15-March-2022]. URL: <https://redcanary.com/company/origin-story/>.
- [84] *The SSPL is Not an Open Source License*. <https://opensource.org/node/1099>. [Online, accessed 11-March-2022].
- [85] *Threat Intelligence Market with COVID-19 Analysis, by Component (Solutions, Services), Application, Deployment Mode, Organization Size, Vertical (BFSI, IT and ITeS, Retail, and Healthcare and Life Sciences) and Region - Global Forecast to 2026*. [Online; accessed 15-March-2022]. URL: <https://www.marketsandmarkets.com/PressReleases/threat-intelligence-security.asp>.
- [86] David Tippet. *The Difference Between Elasticsearch, Open Distro, and OpenSearch*. <https://aws.plainenglish.io/the-difference-between-elasticsearch-open-distro-and-opensearch-d43c9a2c31b1>. [Online, accessed 12-March-2022].
- [87] Wiem Tounsi and Helmi Rais. «A survey on technical threat intelligence in the age of sophisticated cyber attacks». In: *Computers & security* 72 (2018), pp. 212–233.
- [88] Jeffrey Undercoffer, Anupam Joshi, and John Pinkston. «Modeling computer attacks: An ontology for intrusion detection». In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2003, pp. 113–135.
- [89] Jakob Uszkoreit. *Transformer: A Novel Neural Network Architecture for Language Understanding*. <https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>. [Online, accessed 14-March-2022].
- [90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems* 30 (2017). DOI: <https://doi.org/10.48550/arXiv.1706.03762>.
- [91] *What Is a Security Operations Center (SOC)?* McAfee. URL: <https://www.mcafee.com/enterprise/en-us/security-awareness/operations/what-is-soc.html>.
- [92] *What is Operational Cyber Threat Intelligence?* [Online; accessed 15-March-2022]. 2021. URL: <https://socradar.io/what-is-operational-cyber-threat-intelligence/>.
- [93] *Why threat hunting is important*. IBM. URL: <https://www.ibm.com/topics/threat-hunting>.

- [94] Wikipedia. *Security Operation Center* — *Wikipedia, L'enciclopedia libera*. [Online; in data 15-marzo-2022]. 2021. URL: http://it.wikipedia.org/w/index.php?title=Security_Operation_Center&oldid=124690792.
- [95] Wikipedia contributors. *Common Vulnerabilities and Exposures* — *Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Common_Vulnerabilities_and_Exposures&oldid=1071721389.
- [96] Wikipedia contributors. *Cyber threat intelligence* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 15-March-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Cyber_threat_intelligence&oldid=1077029141.
- [97] Wikipedia contributors. *Ontology* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 15-March-2022]. 2022. URL: <https://en.wikipedia.org/w/index.php?title=Ontology&oldid=1076014887>.
- [98] Wikipedia contributors. *Ontology (information science)* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 15-March-2022]. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Ontology_\(information_science\)&oldid=1076419095](https://en.wikipedia.org/w/index.php?title=Ontology_(information_science)&oldid=1076419095).
- [99] Abel Yeboah-Ofori, Umar Mukhtar Ismail, Tymoteusz Swidurski, and Francisca Opoku-Boateng. «Cyber Threat Ontology and Adversarial Machine Learning Attacks: Analysis and Prediction Perturbance». In: *2021 International Conference on Computing, Computational Modelling and Applications (ICCM)*. 2021, pp. 71–77. DOI: [10.1109/ICCM53594.2021.00020](https://doi.org/10.1109/ICCM53594.2021.00020).
- [100] *Your organization's biggest security threat? Analyst burnout*. Paloalto Networks. URL: <https://www.paloaltonetworks.com/resources/infographics/cortex-forrester-2020.html>.
- [101] Channy Yun. *Amazon Elasticsearch Service Is Now Amazon OpenSearch Service and Supports OpenSearch 1.0*. <https://aws.amazon.com/it/blogs/aws/amazon-elasticsearch-service-is-now-amazon-opensearch-service-and-supports-opensearch-10/>. [Online, accessed 12-March-2022].
- [102] Laura Zanotti. *SOAR: gli strumenti che aiutano a colmare le lacune alla sicurezza*. [Online; accessed 15-March-2022]. 2019. URL: <https://www.zerounoweb.it/techtarget/searchsecurity/soar-gli-strumenti-che-aiutano-a-colmare-le-lacune-alla-sicurezza/>.
- [103] Yishuai Zhao, Bo Lang, and Ming Liu. «Ontology-based unified model for heterogeneous threat intelligence integration and sharing». In: *2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*. 2017, pp. 11–15. DOI: [10.1109/ICASID.2017.8285734](https://doi.org/10.1109/ICASID.2017.8285734).