



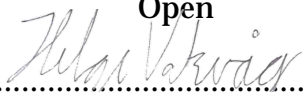
FACULTY OF SCIENCE AND TECHNOLOGY

## BACHELOR THESIS

Study programme / specialisation:  
Computer science

The spring semester, 2022

Author:  
Helge Vatsvåg

Open  
  
.....  
(signature author)

Course coordinator:

Supervisor(s):  
Ferhat Özgür Catak

Thesis title:  
STIX and TAXII based Cyber Threat Intelligence Sharing Parser and Intrusion  
Detection System Rules Generator

Credits (ECTS):  
15

Keywords:  
STIX, CTI, IDS, Snort

Pages: 62

+ appendix: 1

Stavanger, 13.05/2022  
.....  
date/year

---

## 0. Preface

There is a high focus in cyber threats, and cyber crime avoidance in the IT industry. Structured Threat Information Expression (STIX) is a standardized language and file format that is used to store and share safety treat information. A lot of work has been conducted making machine learning models and data learning models to collect and share STIX files. The number of STIX files generated in real-time is high, and it is time consuming for computer safety advisers to read through these files. In this project the goal is to create a software that gather STIX files use Natural Language Processing (NLP) to gather relevant information, and generate alerts in a intrusion prevention system.

# Contents

0.	Preface . . . . .	2
<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Acknowledgements . . . . .	5
1.2	Cyber Threat . . . . .	5
1.3	Recent attacks . . . . .	5
1.4	Incidence Response . . . . .	6
1.5	Training . . . . .	8
1.6	Network Security . . . . .	8
1.7	Detecting attack . . . . .	8
1.8	Safety Threat Intelligence . . . . .	8
1.9	Natural language processing . . . . .	9
1.10	Project Task . . . . .	9
1.11	Related work . . . . .	10
<b>2</b>	<b>Theory</b>	<b>11</b>
2.1	Cyber Threats . . . . .	11
2.1.1	Information gathering . . . . .	11
2.1.2	Phishing . . . . .	12
2.1.3	DOS (Denial of Service) . . . . .	13
2.1.4	Password hacking . . . . .	13
2.1.5	Hacker goals . . . . .	14
2.1.6	Why protect a business from hackers . . . . .	14
2.2	STIX files . . . . .	15
2.2.1	Origin . . . . .	15
2.2.2	Content of STIX files . . . . .	16
2.2.3	Trusted Automated eXchange of Indicator Information (TAXII) . . . . .	18
2.3	Natural Language Processing with SpaCy . . . . .	19
2.4	Intrusion Prevention . . . . .	20
2.5	Scapy . . . . .	21
<b>3</b>	<b>Methods</b>	<b>23</b>
3.1	Problem and its Investigation Methods . . . . .	23
3.2	Collecting CTI from servers . . . . .	24
3.3	Storing STIX files . . . . .	25
3.4	Running SpaCy . . . . .	26

3.5	Shodan . . . . .	28
3.6	Snort . . . . .	29
3.7	Database structure . . . . .	30
3.8	Verifying with ScaPy . . . . .	31
3.9	Experiments . . . . .	31
3.10	Overall program run . . . . .	32
<b>4</b>	<b>Results</b>	<b>35</b>
4.1	vxvault . . . . .	36
4.2	user_AlienVault . . . . .	37
4.3	guest.CyberCrime_Tracker . . . . .	37
4.4	guest.EmerginThreats_rules and guest.EmergingThreats_rules . . . . .	38
4.5	guest.MalwareDomainList_Hostlist . . . . .	39
4.6	guest.Abuse_ch . . . . .	40
4.7	guest.Lehigh_edu . . . . .	40
4.8	guest.blutmagie_de_torExits . . . . .	41
4.9	guest.dataForLast_7daysOnly and system.Default . . . . .	42
4.10	guest.phishtank_com . . . . .	43
4.11	Snort and ScaPy . . . . .	44
<b>5</b>	<b>Recourse's</b>	<b>47</b>
5.1	Financial overview . . . . .	47
5.2	Environmental accounts . . . . .	47
<b>6</b>	<b>Discussion</b>	<b>49</b>
6.1	Data . . . . .	49
6.2	Spacy . . . . .	49
6.3	Shodan . . . . .	50
6.4	Snort . . . . .	50
6.5	ScaPy . . . . .	51
<b>7</b>	<b>Conclusion</b>	<b>53</b>
<b>8</b>	<b>Discussion and future work</b>	<b>55</b>
<b>VI</b>		<b>57</b>
VI.	Appendix A . . . . .	57
VI.	Acronyms . . . . .	57
<b>VIBibliography</b>		<b>59</b>

# Chapter 1

## Introduction

### 1.1 Acknowledgements

I'm grateful for guidance and feedback give from my thesis advisor Ferhat Özgür Catak. The effort and time for discussions in our regular meetings has been a driving force for the work in this project. I would like to thank my colleagues (Steffi Anne Barreto and Seyed Mehdi Sheikholeslami) at Moreld Apply for giving me guidance on a general level. Also, I would like to give a thank to Moreld Apply for letting me take this opportunity an continue developing my personal skills in combination with work.

### 1.2 Cyber Threat

Cyber security is a significant concern in today's world, as threats and quantity of threats are increasing. There are a wide range of cyber threats that can affect a host and/or network. It is relatively easy for anyone with some computer knowledge to set up and use software for malicious purposes. Some known types of attacks are DOS (Denial of Service), password hacking, phishing, Bait and switch, key logger, clickjacking, viruses and trojans. An important part of cyber security is to share information related to potential threats. This is also called Cyber Threat Intelligence Sharing.

### 1.3 Recent attacks

During the last years, many companies have been forced to use remote or hybrid office solutions. This change in infrastructure has given hackers new opportunities related to the business's gaps in security [18]. The trend is that smaller to medium companies are targeted. Small and medium companies tend to be less able to protect themselves, because of lack of resources. Phishing/Social Engineering, compromised/stolen devices, and credential theft are the most common type of attacks.

In UK, the health sector has an increase in cyber security attacks [51]. Health care is a perfect target since these services are highly dependent of their IT systems to be in operation. personal data can be compromised or lost, appointments can be altered, and equipment can fail. This

can compromise integrity, availability and confidentiality, and in worst cases human lives will be jeopardized.

During the pandemic (COVID19), healthcare and hospitals has had an increased pressure. Resources and effort has been focused on handling the increase in health services needed. During this demanding time, the amount of ransomware against hospitals and healthcare facilities has been increasing by approximately 51% since 2021 [21]. One of the main issues seems to be a high amount of IoT devices that can be hard to have fully control over.

A couple of days before Ukraine were invaded by Russia, several Ukrainian government websites and banks were targeted by a massive Distributed Denial of Service (DDoS) attack [20]. In such an attack the owners of the websites will appear as they do not have control in the current situation and seems to be one piece of a sophisticated attack. This is supported by STIX files in the User\_alienvault server in the time period close to the attack. The campaign related to the attack was believed to go back at least to March 2021 [41][42].

## 1.4 Incidence Response

To prepare an organization from internet security threats it is important to have a Incidence response plan. This plan should include these topics [40]:

- Planning and preparation
- Detection and analyses
- Containment, eradication and recovery
- Post-incident activity's

These topics should be a part of an continuously learning and improvement model, where the organization learn from events and observations. Figure 1.1 illustrates the cyclic NIST (National Institute of Standards and Technology) incident response, which is further described below the figure 1.1.

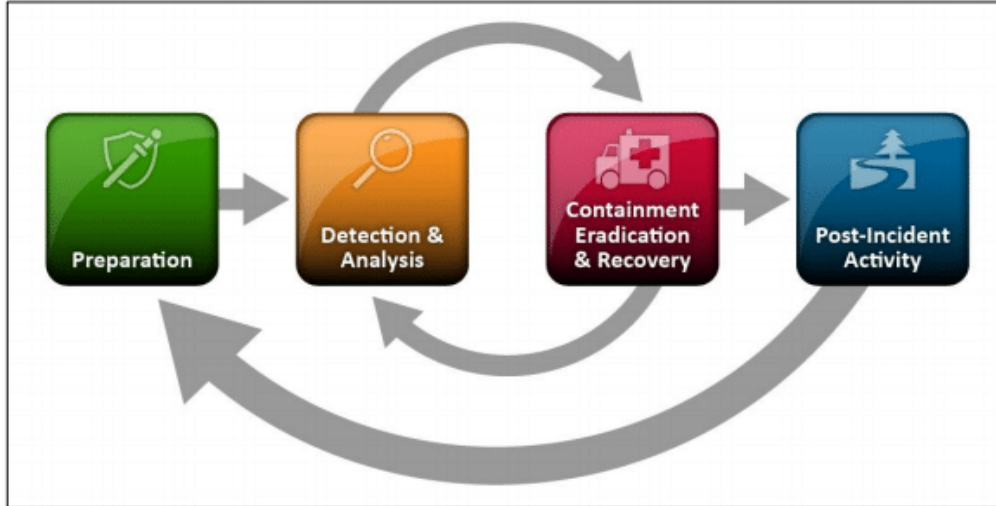


Figure 1.1: The NIST Incident Response Life Cycle [40]

Preparations done can be gathering knowledge of networks, servers and endpoints. Characterising them according to importance and content on sensitive data. It is important to gather a baseline of normal activity, to be able to get a indication if there is an unreasonable high traffic on the network. What type of events that should be investigated and response steps should be described.

Detection can be identifying signs that an attack might happen or observing data showing an ongoing attack. While analyzing is used to see how events affect systems by measuring affected systems against baseline measurements.

Containment is the strategy of how to stop an attack and avoid organization resources from getting overwhelmed or cause any other damage. When developing the containment strategy, it is important to evaluate how much damage an incident can cause, and the amount of availability needed related to redundancy for both costumers and employees. An organization should be able to identify attackers IP addresses, to be able to block them, and thereby avoid the attack. By identifying the IP address, it will also be possible to identify the threat actor, and understand their motivation and TTP.

After an incident have been contained, it is time for eradication. All elements of an incident must be identified and removed. Malware have to be removed. and user accounts affected should be removed, or get their password reset. Also, other weaknesses exploited should be improved before recovering normal operation.

When the system has been recovered to normal operation there should be a post-incident activity where the organization investigates the incident. What happened and when. Can something be done with already existing preparations, routines, detection or analyzing tools? Were established processes followed. Did any actions taken cause damage or delay recovery? Should new tools or resources be made available to easier capture future cyber-attacks?

## 1.5 Training

One effective way to reduce the amount of successful cyber-attacks on computers and networks is training of the users. So that they can tell the difference from normal information and malicious information. Increased knowledge about cyber threats will reduce the amount of people fooled by attacks like phishing, crypto lock and downloading of malware. But even then, there are often someone that are careless, and get tricked by this kind of cyber threats.

## 1.6 Network Security

It is possible to hire a security analyst or pay an external company to set up firewall and network control systems. There are three goals for network security: prevent, detect and respond. Each of these should be in focus when working with network security. Prevention in general is preventing an attack from happening, by for example using a firewall. It is recommended that information related to possible threats and attacks is gathered.

## 1.7 Detecting attack

There are several ways to detect an attack. It is recommended to include advanced cyber security tools, that have been set up based on updated information from most possible cyber threat and attack cases. It is also recommended to perform a baseline test on your network, to verify what the "normal" network performance and load. By doing regular tests on network performance and load, it is possible to determine if there are any suspicious activity by comparing it to the base case (normal condition). There are several Intrusion Detection Systems that can be used, these systems have default settings for alerts related to possible cyber-attacks, and also have the possibility for tailor made rules and actions. It is also possible to perform white-listing and black-listing on access points. Making sure to drop unwanted packages.

## 1.8 Safety Threat Intelligence

*Structured Threat Information eXpression* (STIX) is defined by the *Cyber Threat Alliance* (CTA) [44] and is a structured language used to exchange cyber threat intelligence (CTI). STIX in combination with *Trusted Automated eXchange of Indicator Information* (TAXII) is one of the main current systems for CTI sharing in a machine-readable format. STIX and TAXII is standardized by *Organization for the Advancement of Structured Information Standards* (OASIS) [29] and enables CTI communities to represent their intelligence in a common language, facilitating information sharing. However, there are several issues with the current system. One of the main issues is the readability of the STIX and TAXII formats. This is because the STIX and TAXII formats are not human readable, as they are based on XML format. The other issue is the lack of a standard parser for STIX and TAXII formats. The current version of STIX is 2.1, which includes a set of vocabularies that define the entities and relationships used in CTI.

TAXII defines the message formats and protocols used to exchange CTI. The current version of TAXII is 2.1. The TAXII protocol is a transport protocol for STIX that uses HTTP as a transport mechanism. TAXII also defines a set of services that are used to exchange CTI.



CTI has for many years been gathered by safety engineers to increase the knowledge about attacks, and to help themselves, and others to be able to protect against future cyber threats. There have been several projects the last years building machine learning models that gather information from courses like twitter, facebook, blogs, articles and the dark net. These models search through information and determine if the information has cyber threat intelligence (CTI) content.

The number of files like these are many, and time consuming for a person to read through.

## 1.9 Natural language processing

In deep learning, there are several pre-trained open-source libraries for *Natural Language Processing* (NLP). These libraries can be used for text classification, information extraction, and text generation. The natural language processing toolkit (NLTK) and SpaCy are Python-based open-source library for NLP [36][38]. These libraries also provide a set of tools for text pre-processing. In the thesis, the Spacy library is used for Entity extraction.

### 1.10 Project Task

As mentioned above, current parsers for STIX and TAXII are not accurate. This paper proposes a system that converts the STIX and TAXII formats into natural language by using SpaCy. The system then translates the natural language into *intrusion detection system* (IDS) rules. The system was evaluated using a real data set from the MITRE organization. The results show that the system can convert STIX and TAXII formats into natural language with an accuracy of over 95%. The system can then generate accurate IDS rules from the converted natural language.

The thesis is a proof-of-concept type of assignment. To see if it is possible to create a program that can scan through STIX files, and automatically create rules in an intrusion prevention system. These are the main sub-tasks:

- Collection of STIX files from free STIX servers.
- Extract relevant information like IP addresses, port and transportation protocol from STIX files using NLP (Natural Language Processing).
- Verifying that the IP addresses found are active.
- Generate alerts for a Intrusion Detection System with relevant information from STIX files.
- Set up snort
- Verify that Snort rules work, by simulate malicious traffic.

The program code is available at <https://github.com/hvatsvag/Application>. Shodan private key 'FoUpFRTGpPF78nsu3EUVB7GEsYcXPBZH' must be put in the shodan program file. It is set up to collect 200 files from each source each sequence. This has been done to make each sequence run fast. But be aware that for some reason the from\_date function in first files does not work properly for alienvault and vxvault, causing the same files to be downloaded several times. By increasing this amount to 20 000, the program runs better, but each sequence takes more time. During the 'real run' for the project. the amount was set to 400 000 for each run.

## 1.11 Related work

This section will give a short overview of related work. Strategies of how to protect against cyber threats has historically been of a responsive type. First an organization make their preparations, and set up tools to detect and block cyber threats [40]. If any cyber-attack causes an incident, do necessary adjustments to improve the cyber defense.

Ying He et al [51] proposed an upgrade to the NIST model, using CTI to be able to make the model proactive and adapt to possible incoming attacks. The proposed solution use CTI recived from CTI sharing platforms in all steps in the existing NIST model. Ying He et al propose an approach where human analysts have an important role for implementing the changes. The reason for this is that an intrusion system may have thousands and even millions of alerts each day, due to false positive indications.

A lot of work has been conducted to make Cyber Threat Information (CTI) standardized and available to get a better understanding of cyber landscape and Threat Actors [29] [31]. Daniel Schlette gives an overview regarding CTI evolution and points out that now the availability of information and possible actions to prevent cyber threats are increasing, but there is still needed to develop dedicated incident response formats, and it will also be necessary to secure a high quality of the CTI.

Katy Nguyen et. al. does also point out a relevant issue regarding the way CTI is shared and distributed [23]. Even when the technology to share information is available. It seems to be an issue with organizations contribution. Some of the reasons for organizations not wanting to share Intel is because of fair of exposure, possible reputation damage and lack of incentives. Katy Nguyen et. al. proposed a model that will increase both privacy and a higher focus on quality assurance.

A study performed by Soham et al used existing named entity recognition (NER) algorithms and strategies to collect CTI [34]. The test dataset includes data from different sources, with different complexity and length. Algorithms made for general named entity recognition does not always work well for the cyber threat information domain. They found that a combination of domain-specific embeddings gave better results than domain-independent embeddings, and that a BERT fine-tuned with a small cybersecurity corpus can yield better results than a model using Word2vec embeddings trained on a huge cybersecurity corpus.

# Chapter 2

## Theory

In this project the following libraries has been used. Application graphical user interface has been created with PySimpleGUI. Database has been set up by using sqlite3 and aiosqlite (to get asynchronous connection to the database). Cabby and taxii2client.v20 has been used to connect to STIX databases. Stix2elevator has been used to elevate STIX xml files to JSON. SpaCy has been used for NLP. Scapy has been used to create and simulate network traffic. The json package has been imported to be able to handle JSON objects. Datetime has also been used for sequential actions and limit activity in some parts of the program.

This chapter has the following sub chapters Cyber Threats, STIX files and TAXII, Natural Language Processing with SpaCy, Intrusion Prevention and ScaPy. Cyber Threats contain a overview of some normal types of attacks and tools that can be used by hackers. STIX files and TAXII include a dive into the STIX structure and TAXII. Natural Language Processing with SpaCy gives a short introduction to the SpaCy library. Intrusion Prevention includes information related to Intrusion Prevention Systems (IPS) and Intrusion Detection Systems (IDS). Scapy gives a short introduction to the Scapy library.

### 2.1 Cyber Threats

#### 2.1.1 Information gathering

The first step of an attack is usually information gathering. Where the attacker tries to get a good overview of the target, like what tools that are used, what ports that are open, if possible any details about the network, and information about the owners of the target (company, school, hospital, etc.).

One approach to get information is by forced browsing. By forced browsing the attacker tries to get access to hidden or forgotten resources as files, directories, backup or configuration files [6]. These resources can contain sensitive documentation important for the approaching attack. Several applications are easily available, and can be used for this purpose, for example dirb by Kali Linux [4].

Port scanners can be used to check for open ports. Open ports can be possible access points for the intruder. The most common ports are in the range of 0 - 1023. It is possible to use tools like

dmitry from Kali Linux to get information on open ports and other information like subdomains, email addresses and up-time information [5].

General information gathering can be obtained by using OSINT (Open Source intelligence) tools. These tools gather general information that can be used for example during a phishing campaign. An example of a OSINT tool that can be used is maltego from Kali Linux [10].

Web application vulnerability scanners are programs used to scan web applications for vulnerabilities. These programs look for weaknesses in applications, like possibilities to perform SQL-injection, XSS (Cross Site Scripting). There are several free available tools that can be used. For example Nikito, that can scan an application for files and programs that can be potentially dangerous, outdated servers and servers with specific problems [11]. The tool also crawls for multiple index files and attempt to gather server information. The information gathered are not guaranteed to be safety issues, but usually security flaws.

### 2.1.2 Phishing

With information gathered it is possible to set up phishing campaigns, trying to fool people in the target organization by bait and switch, click-jacking or downloading a key logger, viruses or Trojans.

Bait and switch is a tactic where the attacker buys add space at a web page through shady companies, putting up an add that would seem like a normal add [49]. The ads will later be changed to spread malware, browser locking or to compromise target systems. Clickjacking is in general making a target click a invisible or a page disguised as another element. This kind of an attack is typically used to mirror login pages and thereby gaining usernames and passwords. But this kind of attack can also be used to spread malware or other wanted activities.

Keyloggers which is short for keystroke logger, is a type of spyware - malware that is used to capture every key the victim is typing, and sending all the information to the hacker. In addition to logging keystrokes, this kind of malware can also take screenshots when certain keystrokes are typed, collect information about how long apps have been in use, browsing and search history, record both sides of conversation in emails and messaging apps. It can also perform actions like taking remotely control of devices, logging in and out of devices and send email with logs back to hacker [25].

There are different types of viruses. Some virus types are file infector, macrovirus and polymorphic viruses [28]. File infector type of virus attach to executable files to spread through a network. The main goal of this type of virus is usual to overwrite operating system or try to reformat hard drives. Macro viruses are usually spread through spam mails with infected files, for example word or excel [17]. This type of virus targets the programs itself and can infect both Mac and PC. Typical effects of macro virus infection can be new files created, data that gets corrupted, text that get moved, files that get sent, hard drives get formatted, pictures gets inserted, or they may even be used to deliver other malware. Polymorphic viruses are a type of virus that replicates itself, but changes just enough to avoid many computer defense systems [24]. Many types of viruses can have a polymorphic characteristic. The virus can be combined with other malicious routines as for example botnet, keylogger or ransomware. Polymorphic viruses are getting more popular and are dominating as the main type of virus [33][24].

### 2.1.3 DOS (Denial of Service)

Denial of Service is an attack where a target network, user or server is flooded with requests making it unavailable for the attended user [46]. This means that the attacker tries to send more requests than the target can handle. The two most normal types of floods are ICMP flood and SYN flood. SYN flood is used when attacking one specific target, for example a server. A SYN request is an initiation of the 3-way handshake that is used to connect to a server. When a user sends a SYN (synchronization) request to a server, the server replies with a SYN-ACK (synchronization acknowledgement). After that, the server waits for a response (an acknowledgement) from the user. Depending on capacity, a server has a maximum number of users it can await to get connected to. ICMP/UDP flood is used when an attacker is trying to overwhelm a device ability to process and respond requests. In addition to the device to be unavailable for "normal" traffic, the firewall can also be exhausted, causing a denial of service. This attack can be used when one or more devices in the network is incorrectly configured, allowing an attacker to send spoofed packets that trigger a massive activity in the network. a simple method to avoid UDP flood, is to drop all UDP traffic that is not related to DNS.

HTTP flood and QUIC flood are also used. HTTP flood is in general the attacker(s) flooding the target with GET or POST requests, making a server unavailable for normal users [7]. QUICK flood sends a lot of data sent over quick. It is hard to filter out non-legitimate requests, because QUIC use UDP that provide little information of the data recipient [15].

Other types of DOS attacks can be targeting specific flaws in the application, that cause applications to crash or high consumption of resources.

The most popular DOS programs 2022 are [8]:

- HULK - HTTP Unbearable Load King
- Tor's Hammer
- Slowloris
- LOIC - Low Orbit Ion Canon
- Xoic
- DDOSIM
- RUDY
- PyLoris

Of these, LOIC is one tool that has been used a lot by non-ethical hackers, as it is a simple tool, that has a simple GUI that makes it easy to use a bot net to attack a target [9]. Also HOIC (High Orbit Ion Canon) is getting more popular, as it is more powerful, because it generates more packages [2].

### 2.1.4 Password hacking

Password hacking is one way to get access to an application. Usernames can be guessed, and some times be easy to find if application login page has not been built in a good way. The most common password hacking techniques are brute force, dictionary attack and rainbow table. The brute force

strategy is in general a costly approach for the hacker, because there are a lot of possibilities for a password. If it is eight characters long, it can have above  $3 * 10^{16}$  possibilities. Using a simple brute force strategy can be time consuming. If you assume a computer can try 1 000 000 passwords per second, it can take up to 95 years to crack it. This is where dictionary attacks come in handy. Most people use similar passwords, and there are lists available online [12]. Brute force with rainbow table is even faster than dictionary attacks, but require more storage space. There are also rainbow tables available online [13]. Some well known tools are [1]:

- CrackStation
- Password Cracker
- Brutus Password Cracker
- Aircrack
- Medusa
- THC Hydra
- RainbowCrack
- John the Ripper
- ophCrack
- Wfuzz

### 2.1.5 Hacker goals

The main goals for unethical hackers are mainly to get access to information or to disturb operation [35]. Information can have a lot of value for example for competitive organizations, and in some cases, it can hurt a business reputation. So by gaining access to information it is easy to blackmail a company, or selling information to others. Another approach can be to encrypt or in other ways steal data, and charging a ransom for giving it back. It is also possible to exploit the targets resources. For example as a part of a bot net, also called zombie network. When a machine is controlled in a network like this, it is possible to use the machine to perform attack against other targets, mine for cryptocurrency or other needed actions.

### 2.1.6 Why protect a business from hackers

To understand potential cost with a data breach, here are some details from Cost of Data Breach Report 2021 from IBM Security [32]. The increase of average cost of a data breach has increased from the year 2020 to the year 2021 by 10 % to an average cost of 4.24 million dollars per data breach. Typically, the cost of a breach where the target uses remote office is about 1.07 million dollars more expensive than the ones that do not use remote office. Health care has the most expensive breaches with an average at 9.23 million dollars per breach.

The average number of days to identify and contain a data breach was 287 days. In average, the breaches that were detected and contained within 200 days did have a lower cost with an average at 3.61 million dollars. While breaches that used more than 200 days to identify and contain the

breach had a higher cost, with an average at 4.87 million dollars.

Mega data breaches containing from 1 million and up to 65 million records were much more costly than smaller breaches where 1 000 to 100 000 records were affected. Where more than 50 million records were affected, cost related to the breaches were 100 times higher than average.

Organizations using security AI fully or partly had a much lower cost related to the breaches that were not using security AI. The organizations using full deployed security AI and automation did have a much lower average cost of a data breach with an average at 2.9 million dollars, compared to the average cost of a breach for organizations that did not implement any security AI and automation which was 6.71 million dollars.

Organizations that store data in hybrid cloud environments has had an average of 3.61 million cost per breach, which is 1.19 million dollars lower than the average compared to public, private and on-premise cloud models. Organizations with complex systems and high level of compliance failures also tend to have a higher cost than organizations that have low level of complexity.

In average 38 % of total breach cost is loss of business.

## 2.2 STIX files

### 2.2.1 Origin

Before STIX files was introduced, safety threat information was often exchanged by human-to-human interaction. For example, by email and blog posts. As cyber-attacks and threats were increasing, the need for a standardized format for safety threat information increased [16]. The goal of the work that MITRE Corporation conducted was to create a language and file structure that was "expressive, extensible, automatable and readable". STIX gives the possibility to analyze cyber threats, specifying patterns, manage cyber threat activities and sharing cyber threat information. An overview of the STIX (version one) file structure is shown in figure 2.1. From the start of the work published in 2013, a lot of work has been put into the STIX language and structure. Making it possible to extract information as JSON info, which is easier to forward to applications and also making it easier for humans to read. But still the

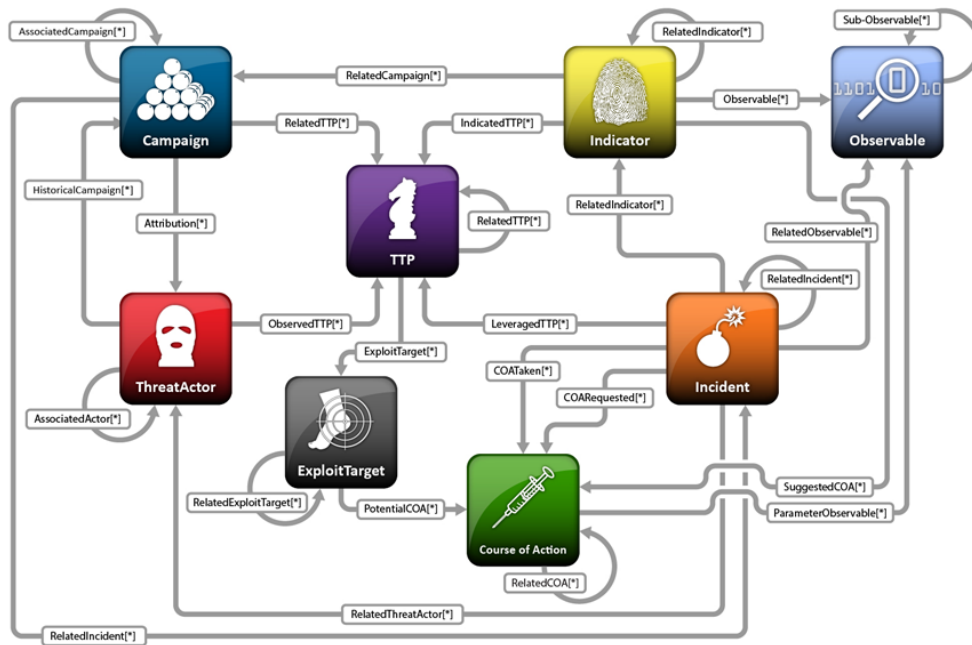


Figure 2.1: STIX Structure [16]

### 2.2.2 Content of STIX files

**Observed data** are information gathered in something called STIX Cyberobservable objects (SCOs) [29]. SCOs contain data from one instance observed at a network or a host. To get a better overview of the threat, it is possible to connect SCOs to STIX Domain Objects (SDO). From STIX 2.1 documentation these objects are defined as “Attack Pattern, Campaign, Course of Action, Grouping, Identity, Indicator, Infrastructure, Intrusion Set, Location, Malware, Malware Analysis, Note, Observed Data, Opinion, Report, Threat Actor, Tool, and Vulnerability”. By connecting SCOs and SDOs or SDOs with other SDOs it is possible to create **relationships**. These relationships can for example be used to map patterns and methods used on types of organizations. Thereby it will be easier for organizations to know what to look for in their network. It might be files, processes or maybe traffic to a IP address. Relationships can be stored as STIX Relationship Object (SRO). There is also a type of relationship called Sighting SRO. Sighting SRO stores information from an observation of for example an indicator. These SROs has the possibility to use count. In STIX version 2.1 data is stored as JSON data, and the number of objects has increased to 18 STIX Domain Objects (SDOs).

**Tactics, Technique or Procedures (TTP)** are observed behavior, resources used and victims targeted by a threat actor. This information can be used to check if you or your organization is a typical target of an attack, what to expect when being targeted, and what they typically would try to acquire (information, cash, denial of service, etc.)

**Threat Actor** are malicious actors representing a cyber-attack threat. It can vary from one or several individuals to large groups or organizations. Threat Actors can be characterized by



motivations, goals and past activities. These characteristics can often be used to evaluate if an organization is an assumed possible victim. Other characteristics are capabilities, sophistication level and resources the threat actor has access to, which gives an indicator on the power of the threat actor.

**Tools** can be software developed for white hat hacking, or tools developed purely for malicious purposes. Threat actors can use a wide range of tools to perform an attack. The STIX Domain Object for a tool can have information about the tool function, how it can be used, and examples of kill chain phases it has been used for.

**Attack Pattern** stores information about tactic, technique or procedure that has been used to infiltrate a target. By knowing these patterns, it is possible to get information on how an attack can be performed, and activity to expect as a part of a kill chain. Examples of observed patterns can be unexpected scanning activity of ports/applications, "spear phishing" and other activities.

**Campaigns** are observations of Threat Actors through incidents and/or TTP that has occurred over a period of time. Campaigns can include information regarding suspected intended effect, characterization and assertion of aggregated intent of the campaign, TTP used in the campaign incidents that are believed to be part of the campaign and Threat Actors that are believed to be responsible. Campaigns that might be related to current campaign can also be linked to these objects. Additional information regarding activity's taken in response to the campaign, source of information, handling guidance and more can also be added to these campaign objects.

**Intrusion set** can be used for a very long period of time and is a gathering of the entire attack package. One intrusion set can be used for multiple Campaigns that has been used for multiple purposes, that is believed to be performed by the same Threat Actor.

**Course of action** gives information about basic use cases, but do not include automated actions. This can contain information regarding how to prevent an attack or how to react to an ongoing attack. Information included can be necessary patches, configurations of firewall or even training of employees in the organization.

**Identity** is used to identify entities from several perspectives. Source of information that made a stix file, threat actors identities, information source, targets of attack and more.

**Grouping** is an object that can be used to characterize an ongoing analysis process. First it starts out to be a set of data, which over time, with sufficient analysis can be developed to be a STIX Report Object.

**Indicator** is a type of STIX object that contain a pattern. Indicators can be used to represent a set of malicious domains by using **STIX Patterning**. STIX Patterning enables the possibility to match observations against observations gathered from historic attacks. These patterns usually contain IP addresses that has been observed in correlation with other IP addresses or activity, and the time duration the observation is valid for. Indicator SDO can be used to detect suspicious cyber activity (malware, tool and attack pattern), what phase in a kill chain the behavior is involved in and the time window the indicator is relevant for.

**Infrastructure** SDO is a type of TTP used to describe resources like servers used by a treat actor, servers or devices used as a defense, or servers that has been targeted by an attack.

**Location** is a geological location, must be presented as ether by region, civic address or latitude and longitude. Location SDO can for example be used with Identity or Intrusion Set to identify the location where a threat actor operation is located. Or it can be used with malware or attack pattern to view locations that threat actors are targeting.

**Malware** is a type of TTP. As the name implies, it describes malicious code. The object can include information regarding goal of the code, with regards to impacts it can cause if it is inserted

into a host or system. It might be used to compromise integrity, availability or confidentiality within the target. Causing disturbance in the targets data, applications or operation system.

**Malware Analysis** is used to store results and metadata from statistics or analyses that has been performed on malware or malware families.

**Note** can be used to add information to an already existing STIX object. For example, adding information on a Campaign that another organization or user. It can contain information like observations made in hacker forums.

**Opinion** is meant as an assessment of an existing STIX Object. It is a possibility to give feedback and to rate existing objects. This is mainly done by human analysts.

**Report** object is used to store collections for one or more topics as a more comprehensive cyber threat story with references to relevant STIX Objects. Topics can be for example Threat Actor, malware or attack technique.

**Vulnerability** are flaws found in software or hardware that can easily be exploited by a hacker to have negative impact on a host or system confidentiality, integrity or availability. Common Vulnerabilities and Exploits (CVE) are stored in a list, and can be linked to by Malware objects. When Vulnerabilities are being targeted during malicious cyber activity these vulnerabilities will be referenced in other relevant SDOs.

**Incidents** are related to discrete instances of indicators, and consist of data as time-related information, parties involved, assets affected, impact assessment, related Indicators, related Observables, leveraged TTP, attributed Threat Actors, intended effects, nature of compromise, response Course of Action requested, response Course of Action taken, confidence in characterization, handling guidance, source of the Incident information and log of actions taken.

### 2.2.3 Trusted Automated eXchange of Indicator Information (TAXII)

TAXII is an application protocol that defines a RESTful API for exchanging CTI over HTTPS [39]. TAXII servers contain CTI in a logical repository. Producers of CTI can host CTI data in the server, that can be requested by clients. Information stored from one source that can be collected by clients are defined as a collection. See figure 2.2 for an illustration. Channels provides an interaction where producers of CTI can deliver to many consumers, and consumers can receive CTI from many producers.

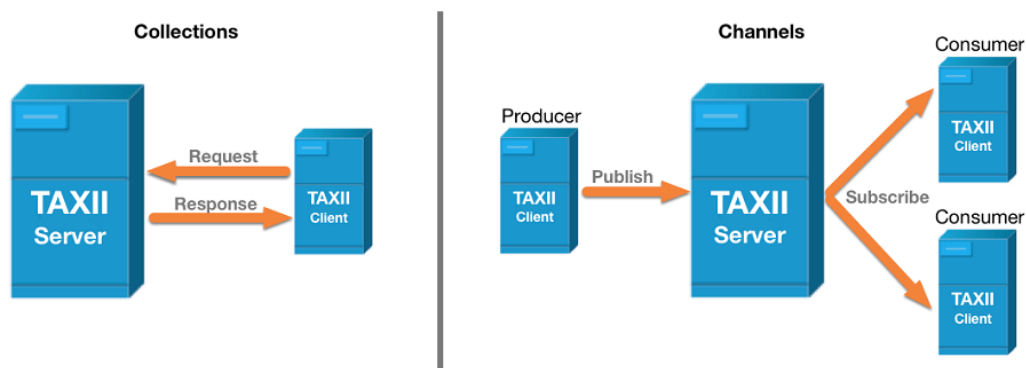


Figure 2.2: TAXII architecture [39]

## 2.3 Natural Language Processing with SpaCy

Natural Language Processing algorithms can be used to add structure to unstructured text [26][38]. Features built in to SpaCy are tokenization, Part-of-speech tagging, dependency parsing, lemmatization, sentence boundary detection, named entity recognition, entity linking, similarity, text classification rule-based matching training and serialization. Simple trained pipelines available in spaCy typically provide binary weights, lexical entries, data files, word vectors and configuration options.

First spaCy separates the text into words, signs sentences and paragraphs. The splitting into words and signs is called tokenization. Thereafter each word is assigned with part-of-speech (POS), lemma, TAG, DEP, shape, alpha and stop. This gives each word a set of values needed for further analysis. POS tells the type of word, for example proper noun, verb, adjective or number. Lemma is the base form of a word. TAG a tag that is related to the POS. DEP is the dependency between tokens. Shape gives a description of the shape of the word, as for example capitalization, punctuation and digits. Alpha is a true/false Boolean, and tells if the token is an alpha character. STOP is a true/false Boolean that indicates if the token is in the category most common words in the language. In English STOP words are for example is, and, the and is.

The dependency between the tokens in a text is stored, and can be illustrated using a visualizer. As an example, see the correlation in the sentence "School is cool, and the teacher is awesome!" in figure 2.3.

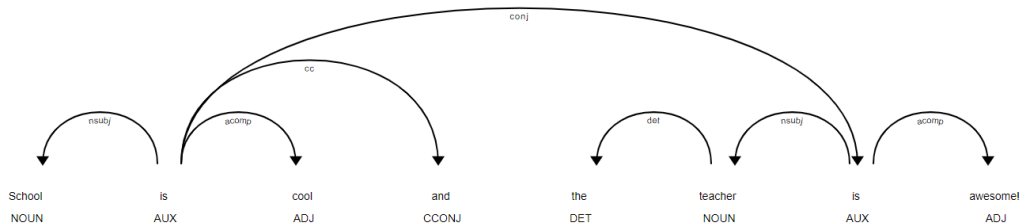


Figure 2.3: SpaCy dependency illustration

Named entities are real named objects. Examples are countries, dates, movies, names, companies or currencies. As an example of output, here is the named entities from a STIX file description field. In figure 2.4 the small English model is used, with no extra patterns or rulers added.

This **IP ORG** address **185.7.151.29 DATE** has been identified as malicious by feodotracker.abuse.ch.  
 For more detailed information about this indicator go to [ CAUTION!!Read-URL-Before-Click ] [ <https://feodotracker.abuse.ch/host/185.7.151.29> ].

Figure 2.4: Named entities

It is possible to add matcher and/or entity ruler to the spaCy model. The matcher can be used together with rules to find information in a doc object. Rules are added to a dictionary of rules in the matcher. Information found is stored as a list of tuples, where each match has a match\_id, start and end value. The match\_id value represent a value in nlp.vocab.strings dictionary while start and stop is the matched span (slice) where the information has been mapped to in the doc

object. An entity ruler can be added to the pipeline. When it has been added, it is possible to add new patterns that will be found by the NER (Named Entity Recognition). It is important that the entity ruler is put in front of NER, for the new rules to be added in the NER process.

By using medium sized model, word vectors are also included. Word vectors gives the possibility to comparing words and sentences. These vectors can be generated by using an algorithm like for example word2vec.

When SpaCy is being used, the text is processed through pipelines. Each pipeline ads context to the text. First the tokenizer processes the text, and generates a doc object. Depending on complexity of the pipeline, the doc object is processed further, and for each component a doc object is the output. Each component is a machine learning or a data learning model, and the output depend on how the model has been trained. In SpaCy, there are ready models that are pre-trained for several languages. It is also possible add available models or self-made models to the pipeline.

segmentation, and a representation of words, names, signs and other information into tokens. Each token has a lot of information connected to it that can be used when analyzing the output of the algorithm. The algorithm also splits the document/file into sentences, with the possibilities to predict the dependence of the content of the sentence. Named entity recognition (NER) is also available. More complex algorithms provide word vectors, which gives the possibilities to compare words with each other to find similarity between words and sentences.

## 2.4 Intrusion Prevention

It is possible to use both Intrusion Prevention systems (IPS) and Intrusion Detection systems (IDS) to observe and inspect traffic for a network [48]. Both IPS and IDS alert system administrators if it detects suspicious activity. The main difference with the two is that IPS also has the possibility to react to events that are suspicious and malicious. An IPS can thereby prevent an attack by for example blocking IP addresses or closing ports that are targeted. Threats that an IPS can detect and protect against can be DoS type of attacks, worms, viruses and also other types of exploits. When inspecting packets these systems look for malicious code. If malicious code is found a IPS could be configured to terminate TCP sessions, block IP addresses, reconfigure firewalls and also remove and replacing malicious content. Methods used for detection of malicious activity can be for example signature-based, anomaly-based and policy-based.

Signature-based detection is detection of pre defined signatures and patterns that are known to be malicious. A signature can be a specified sequence of bytes, anomalies by use of network privileges, or it might be unauthorized software execution, network access or directory access [45]. The detection system needs to be kept up to date with regards to static signatures, to be able to detect as many threats as possible. Signature-based detection is a traditional and fast approach which has been used in antivirus software for many years. This type of detection typically finds the well-known, but also potential threats.

Anomaly-based detection is detection of unexpected behavior in the network. when a IPS detect an anomaly, it will block access to the host or hosts involved. Anomaly-based detection need to be trained with a baseline of how the network usually behaves to be able to tell anomaly behavior from normal behavior [37]. The advantage of anomaly-based detection is that it can protect against unknown threats. If an event is out of ordinary behavior an alert is sent, and actions can be made to set the network back to normal operation.

Policy-based detection is detection of activities that are triggered by security policies that has been set up by system administrator. When violations are triggered, administrators are alerted, and activities can be triggered.

Endpoint Detection and Response (EDR) is a type of intrusion system that use rules, analysis and automated response to detect and react to threats on hosts and endpoints. The high number of devices in a network can be difficult to control [50]. EDR help system administrators to monitor and collect data from endpoints, analyze data and have automated response to known types of security breaches.

Fire Walls are the first line of defense in networks and internet applications [47]. A rule set defines what traffic to let through and what should be stopped. The fire wall can be hardware and/or software. There is a variety of different types of firewalls. The general purpose of firewalls is to avoid direct connection to the internet and other networks for racecourses in a network. Packages are filtered based on state, port, protocol and IP address. Filtering can be performed based on input from administrator, predefined settings and updates from intrusion prevention and anti virus systems.

## 2.5 Scapy

Scapy is a tool that has several usage areas. It can be used to perform package scans, it can modify packages and can be used to create new tools. It can also be used to forge new packages [14]. Packages can be created with a wide range of protocols, and the packages forged in Scapy can be sent to simulate traffic in a network. Which is useful when for example testing reactions from firewalls and other intrusion prevention systems. Scapy can be used to for hacking, for example for port scanning and ARP spoofing [22]. Port scanning are tools used to find out what ports are open at an endpoint, as for example a network interface. While ARP spoofing is done when an attacker is using its own MAC address together with for example the default gateway IP, it can trick a machine to believe the attacker is the default gate way, and thereby sending packages to the attacker. This would be a typical man in the middle attack, where the attacker receives all traffic between a target computer and router.



# Chapter 3

## Methods

### 3.1 Problem and its Investigation Methods

As mentioned in chapter 2.2.6, detection time and response time of a cyber-attack is crucial with regards to cost related to a data breach. This is why automating detection and response is of high importance. In this work the goal has been to automate detection of potential threats. STIX files contain CTI that can be used by safety engineers to increase the cyber security of a network. But resources used for safety engineering is often limited, and the amount of time to review these files can be overwhelming. Because the amount of STIX files available is huge.

In this project, the goal was to collect STIX files (data) from cyber threat intelligence servers. Then collect relevant information represented with natural language from the STIX files using SpaCy. Verify if IP addresses found in the files were active. For IP addresses that were active, information gathered from the STIX files were used to generate alert rules for Snort. Figure 3.1 gives an overview of the process of how the workflow in the application is. These steps are described in more detail in the next chapters.

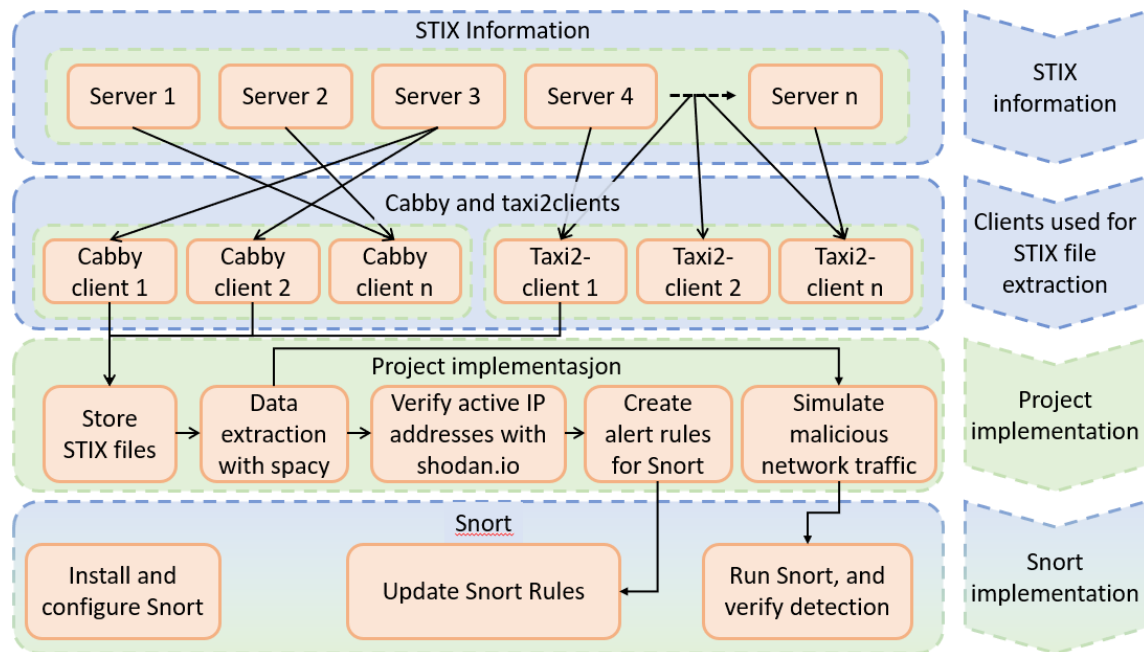


Figure 3.1: Project workflow

## 3.2 Collecting CTI from servers

The tool `cabby` in python were used to collect STIX files from CTI servers. `Cabby` can be used to connect to different `cabby` services. From the user guide, standard implementation is by using `taxiistand.com`. But this setup did not give a stable connection to collect STIX files. It was possible to collect some files from `vxvault` (server), but the rest of the servers, that were `hailataxii` servers, were near impossible to collect files from due to bad connectivity and timeout.

During the project the `alien` client was also tested. This client is live (during time of writing), and provided the project with relevant data on a daily basis.

With regards to quantity, the best free source found for CTI was `hailataxii.com`. But to be able to poll (collect) STIX files it was necessary to get authenticated by using the username "guest" and password "guest". This source gives access to a high amount of STIX files ( 15 million files). But one thing to be aware of is that the time stamp on the files collected is the time that the files were collected. The issue with that is that it is not possible to use the parameters `begin_date` and `end_date` in the poll request. For the other clients, it is possible to choose data from specified start and end days. If someone tries to poll files several times from the sources at `hailataxii.com`, they will get several copies of the same files.

In this project the program was hard coded to sequentially try to collect new files from the relevant sources. The relevant sources are collected in a list, and the list are run through a for loop. This function is initiated when there are no more files to process, if there are no IPv4 addresses to be searched for. The reason for this is to avoid using resources collecting more files, when resources are needed elsewhere. The pseudo code for creating clients are shown in algorithm 1.



---

**Algorithm 1** Async get\_client pseudocode

---

```

async def get_clients():
    clients = [] # List for clients
    create client for otx.alienvault.com, open.taxiistand.com and \
        hailataxii.com
    add each client to clients
    return clients

```

---

A function was made for collecting STIX files produced later than the last file collected from the server. This approach will let the user poll from any amount of sources. The approach for collecting from hailataxii.com is a little different. Because begin\_date statement does not work for hailataxii.com. If begin\_date is used it result in timeout and no data collected. As an alternative a skip loop has been set up, where you skip the amount of files that is already collected. One negative aspect of this skip loop is that it is time and recourse consuming. To make this worth the consumption of resources the amount of files are adjusted quite high. For this skip loop to work, the files has to be downloaded. for this loop to work, the files has to be downloaded. Of cause it would be possible to just collect all files at the same time, as it seems that no more files are added to these servers. But the strategy has been that it should be a solution that could be used on servers that are supplied with new files regularly. The reason why there is a try/except statement used when collecting files, is because there is a possibility that the pipeline (content\_block) has a slow response, and initiate a HTTPError 502. This was more of a problem in the start of the project, because the program was set up to collect from all servers at the same time. This caused a timeout HTTPError 502. After changing this to a for loop as illustrated in algorithm 2.

---

**Algorithm 2** Async auto\_poll pseudocode

---

```

Collect clients by using get_clients()
for each client:
    for collection in client.collections:
        collect relevant information from db
        if client name is "hailataxii.com":
            add a count to a variable #used to skip files already downloaded
        connect to the server:
            if files are to be skipped:
                skip files
            collect the wanted amount of files
        save to db

```

---

### 3.3 Storing STIX files

In the start of the project STIX files were stored in a local folder. But because of the possibilities to add information to each entry and easily be able to find the STIX file if necessary, information was stored in an sqlite3 database. Values stored for each entry were content\_id, name, content\_text,

and source. `Content_id` is the primary key, which is an int with auto increment. `Name` is the time stamp for the STIX file, and is used to be able to set the `begin_date` when collecting files. `Content_text` is the value of the STIX file itself. `Source` is the name of the server that the STIX file has been collected from.

### 3.4 Running SpaCy

During the start of the project, small, medium and large English model in spacy were tested to see if any of them managed to collect relevant CTI from the STIX files and elevated STIX files. But the Intel did not get marked and collected in a usable manner. IP addresses were labeled as GPE (Geopolitical Entety), CARDINAL (numbers that does not fall under another type) and DATE. Some relevant information was also tagged as WORK\_OF\_ART as for example "ET CNC Shadowserver Reported CnC Server Port 6667 Group 5". This support some of the staitments from Soham et.al. [34] that NER used for general named entity recognition may not always work well when collecting CTI.

Since it did not give any contribution in collecting CTI by using larger models, the small model was used in most of this project. STIX files were analyzed, to find out what kind of information that would be beneficial to collect. The relevant information that could be of use would be protocols used, IP addresses and port or ports that are relevant. These types of information are found in two different approaches. Relevant protocols for snort are TCP, ICMP, UDP and IP. IP is not relevant because it will be found in many of the cases where "IP:" followed by the IP address is in the free text.

To be able to collect information regarding transport protocol `entity_ruler` was added before NER (Named Entity Recognition) in the SpaCy pipeline. Patterns added to the ruler all have label "transport", and the patterns are "tcp", "TCP", "icmp", "ICMP", "UDP" and "udp". The `entity_ruler` is simply added to the nlp model as shown in algorithm 3.

---

#### Algorithm 3 Async pattern\_ruler partial code

---

```

async def find_relevant_spacy_stix(list_of_stuff):
    nlp = spacy.load("en_core_web_sm")
    ruler = nlp.add_pipe("entity_ruler", before="ner")
    transport_protocol = ["tcp", "TCP", "icmp", "ICMP", "udp", "UDP"]
    patterns_ruler = [{"label": "transport", "pattern": protocol} \
                      for protocol in transport_protocol]
    ruler.add_patterns(patterns_ruler)

```

---

Matches has been used to find ports, IPv4 addresses, MD5-values and SHA-256 values. The code used to find these values are shown below and is a continuation of the function above. Match is set up in the nlp model, and can only be used on a doc object produced by the nlp model. See algorithm 4 for details regarding the matcher implementation.

During the project it was tested to collect the information gathered by the matcher by using the same rules in `entity_ruler`. But it seems like regex is not easily used in `entity_ruler` compared to the matcher.

**Algorithm 4** Matcher partial code

---

```

octet_rx = r'(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)'
patterns_matcher = [{ "TEXT": \
{"REGEX": r"^\{0\}(?:\.\{0\})\{3\}$".format(octet_rx)}]}]
patterns_matcher2 = [{ "TEXT": "MD5"}, {"TEXT": ":"},
{"IS_ASCII": True}]
patterns_matcher4 = [ {"TEXT": "SHA256"}, {"TEXT": "-"}, {"OP": "?"},
{"TEXT": "None"}, {"OP": "!"}, {"IS_ASCII": True}]
patterns_matcher5 = [{ "TEXT": \
{"REGEX": r"^\{0\}(?:\.\{0\})\{3\}$".format(octet_rx)}},
{"TEXT": ":"}, {"OP": "?"},
{"TEXT": {"REGEX": r"[0-9]{2}?"}, {"OP": "+"}]
patterns_matcher6 = [{ "TEXT": "Port"}, {"TEXT": \
{"REGEX": r"[0-9]{2}?"}, {"OP": "+"}]
patterns_matcher7 = [{ "TEXT": "Ports"}, {"IS_SPACE": True, "OP": "?"},
{"TEXT": ":"}, {"IS_SPACE": True, "OP": "?"},
{"TEXT": "{"}, {"IS_ASCII": True, "OP": "+"}, {"TEXT": "}"}]
patterns_matcher8 = [{"TEXT": "port"},
{"TEXT": {"REGEX": r"[0-9]{2}?"}, {"OP": "+"}]

```

---

To make sure to collect all relevant data when using SpaCy it is important to format the input data to make it representative. During the project the string formatting shown below has been found sufficient to extract relevant information. Mattingly [26] has been a great inspiration in general for the SpaCy implementation. Not all STIX files can be elevated to JSON format, and for some files, even if they got elevated, all content was not included in the JSON output. Some of the STIX files contained JSON objects with information marked as "raw data", this data was not elevated. Which was weary unfortunate, since a lot of relevant information like IPv4 addresses, ports and more were stored in these objects. Because of this, raw STIX files have been run through the SpaCy model. The SpaCy function used in the project receives a list of lists with text content of the STIX file and the content\_id. Content\_id is used to be able to know which STIX file that has been processed. Content from each file was standardized to make sure that the entity\_ruler and matcher could find the wanted information. See algorithm 5 for details.

**Algorithm 5** String preparation partial code

---

```

information = information.replace("\'", "")
information = information.replace("'", " ")
information = information.replace("[", "[ ")
information = information.replace("]", " ]")
information = information.replace("(", "( ")
information = information.replace(")", " )")
information = information.replace(",", " , ")
information = information.replace("<", " < ")
information = information.replace(">", " > ")
information = information.replace(":", " : ")
information = information.replace("/", " / ")

```

---

The SpaCy implementation returns a list of lists that contain the doc object, content\_id and the matches that belong to each file. Some of the files were too big for spacy to process, as there is a limit for maximum entities collected. When this was noticed, each file was split up into several files if they were larger than 600 000 bytes. This value was chosen, because files of this size were able to be processed (by testing). The collected info is inserted to a table (called spacy) in the database. Each entry has the values info\_id which is a int with auto-increment, info which is the content that was collected, info\_label which tells what kind of info (IPv4, port, MD5, no info, SHA-256 or transport protocol) is stored and the content\_id of the file it has been collected from. URLs found in the STIX files were not collected, because the "LIKE\_URL" found so many irrelevant matches, that the match for "URL" got flooded. In the end phase of the project it was noticed that using an empty model with pattern\_ruler and matcher gave the same results. Therefore, the empty model were used for the rest of the project. Files were processed again by the SpaCy implementation with the same results.

## 3.5 Shodan

To be able to use Shodan on a large scale, it is necessary to create a user at shodan.io. It is possible to get a user registered as a student user to get access to more resources than a normal free user. The basis for the search was found at shodan.readthedocs.io [3]. Some modifications were needed. First of, if more than one request is sent per second an error will occur. Sleep was introduced in the code to avoid this error. In general, the only function for the use of Shodan in this project is to check if the IP address is active. In addition, information from the IP address search was stored in the database. A lot of testing was done to get the most efficient implementation. Some IP addresses had to be searched for several times. Because of this in addition to the slow search (one search per second), it was chosen to get one IP address at the time, and search for it until a result was gathered without error. If no information was found in shodan, the result in the database was added as "No information in shodan". Information stored for each entry was shodan\_id which is an int with auto-increment, info which is the result "data" from the JSON file received from shodan, additional\_info which is the whole JSON file received from shodan, info\_label which is the label of the info that was searched for, info\_spacy which is the info that was searched for in shodan, spacy\_id was stored, to be able to find the source of the information. Every entry in the database represented one match in the json file returned from shodan, e.a. every IP address could have more than one hit in the database table. The reason for this was that early in the project snort rules were based on information gathered in additional\_info field. But as the project developed, only information from STIX-files was used to create these rules.

At the end stage of the project the search was changed to a IP address specific search (host search) [43]. It was also observed that some of the results from shodan.io did not have the 'data' field that is in the standard implementation. This caused a lot of the searches to not be stored in the early phase. The final implementation did not use this data content in the info field in the database. Instead it was tagged with "found data" if data was found, and if data was not collected, the error received was added to the info instead.

## 3.6 Snort

Snort was installed and configured according to instructions from Just Ed at youtube [19]. The command used to initiate snort was "snort -i 5 -c c:\Snort\etc\snort.conf -A console" (this depends on the setup on the computer used). The command was initiated from c:\Snort\bin. Final rules made with the program, and configuration are available at <https://github.com/hvatsvag/Snort>. If configuration are to be used by anyone else, be aware that this setup is for use by windows, and that the "ipvar HOME\_NET" value in the config file has to be changed to match the network where it is used. In this project the file local.rules in Snort has been used to append new rules. These rules consist of action (in this case alert), source protocol, source IP address, destination protocol, destination IP address and an alert message. an example of structure is shown below.

```
"alert tcp 209.133.10.45 [7000] -> any any (msg:"Alert, this ip 209.133.10.45 has been found malicious, see STIX file 25280";sid:1001060;)"
```

For the rule to be valid it needs a set value for action, protocol and message. Port and IP address can be "any", depending on what is interesting to get actions for. To be able to build these rules, relevant information produced by SpaCy (from STIX files) was used. Towards the end of the project the reference to the content\_id (STIX-file) was not used in the msg part of the snort rule because there were issues with creating msg larger than 2048, which is the longest msg allowed.

When making new snort rules, the function collects all IP addresses that did not get "No information found" from the shodan search and is not already represented in the snort rules and adds them to a dictionary. Dictionary was chosen to make sure not to get any duplicates. For each IP address, it search for relevant information in the spacy table. Ports, transport protocols and content\_id are collected for each IP address. Then it makes a snort rule for every IP address in the dictionary.

There is also a function to delete the snort rule table, and create a new blank local.rules file. The reason for this is to be able to include information from STIX files that has been processed after the snort rules has been made. It was first tried to implement this with a timer, but in some cases the database table did not get deleted, which is a lot worse. Because the only rules in the snort table would then be new rules made after the reset. Because the table keeps track of which IP addresses there has been created snort rules for in the rule file.

One problem with snort was observed. When the amount of rules got high, snort has issues starting up. There is some kind of limit for the amount on content that snort include from the local.rules file. It might be related to the count of bits used or total length of string. Because, it can stop collecting in the middle of a rule, which will cause "Fatal Error" for snort, as the rule are not correct. At the end of the project an implementation to the code were introduced to compress the snort rules, to see if it could help with this problem. The update in the code adds IP addresses that have the same ports and transport associated to it on the same line. This made a big improvement.

But a new Error were observed. Even when the IP addresses had been verified by checking them in shodan, this might not be enough. The IP address 01.01.01.01 was in the rules. But this IP address could not be processed by Snort. This could be because this IP belong to "APNIC and Cloudflare DNS Resolver project" and is used for DNS resolver service [27]. This IP address has been hard coded to not be included in snort rules.

## 3.7 Database structure

**Content** table contain all the STIX files. The primary key (`content_id`) is an integer with auto increment. `Info` is the timestamp from the STIX file and has been set up with `varchar(255)`, `content_text` is the content of the STIX file and has been set up as `text`, and the `source_db` is the source where the STIX file has been collected from and has been set up with `varchar(255)`. To increase query efficiency there is an index on the content table for the values `content_id` and `source`, as these are regularly queried.

**Spacy** table contain information collected from STIX files. Each STIX file can have many entries in the spacy table. The primary key is an integer with auto increment. `Info` and `info_label` is the values and the labeled type gathered by the ruler or matcher in SpaCy and are set up with `varchar(255)`. The `content_id` is stored to have a reference to the original STIX file, and has been used to be able to verify information that has been gathered. To increase efficiency there is an index on the spacy table for the values `info_label` and `info`, as these are regularly queried.

In the start of the project a unique constraint was used for the `info` value. The reason for this constraint was that SpaCy was also used to find matches that was "LIKE\_URL". This type of match gave a lot of matches that was standard, and the same for many for the STIX files. Typically 50 - 60 matches for each file, that was not relevant. By using unique constraint the values found was mostly relevant. But as the project developed, the URL matches was not used. URL matches was therefore not gathered by spacy and the unique constraint removed. This was an advantage when it comes to retrieving information for IP addresses, protocols and ports. As the same IP address can be referred to in several STIX files, and other information like ports and protocol can be different in the different STIX files.

**Shodan** table contains information from all the IP address searches that has been performed towards shodan.io. The primary key is an integer with auto increment. `Info` either contain an error message, or "found data" and is set up with `varchar(255)`. Earlier information from the "data" field of the JSON element received from shodan was stored in "info". Because of the issues with al JSON elements not having this value, this was not used in the final implementation of the program.

Additional information is also set up with `varchar(255)`, is a dictionary that contain a lot of information like ports used, transport protocols used, location of the server, may contain known Vulnerabilities. During the startup of the project information regarding protocol and port used for snort was collected from these data. But as the project developed, it was decided that all data used in the snort rules were to be collected from STIX files. `Info_label` is the same as in the spacy table. The reason why this is in the table is because it is possible to search for other information, as for example URL. But it has not been used, since URLs has not been collected. `Info_spacy` is the information used for the search. This information is included, to avoid the same IP address to be searched for several times. It also has `info_id` as foreign key, to have a reference back to the source of information. To increase efficiency there is an index on the shodan table for the values `info_spacy` and `info`, as these are regularly queried.

**Snort** contains information for the snort rules that has been made. When a row is going to be inserted into the table. The program collect IPv4 addresses that are not in snort, and has a positive hit in shodan. Then spacy table is queered for information for the IPv4 addresses. Information from all STIX files that have the IPv4 address are combined into one rule. The primary key

is an integer with auto increment. Protocol is the protocol info that has been gathered from STIX files. If no information regarding protocol has been found in a STIX file, tree rules are generated. One for each of the following protocols TCP, UDP and IP. ICMP was not included, because it is less likely to be used, and would also be triggered by IP rules. Ipv4\_src is the IPv4 address that has been found in a STIX file. Msg is the message used in the snort rules, which refer to the STIX file it has been found in. both these values is set up with varchar(255). To increase efficiency there is an index on the snort table for the value ipv4\_src, as these are regularly queried. This will most probably be one of the smallest table in the database, but an index was included even then, to optimize the queries.

The database structure is illustrated in the figure 3.2.

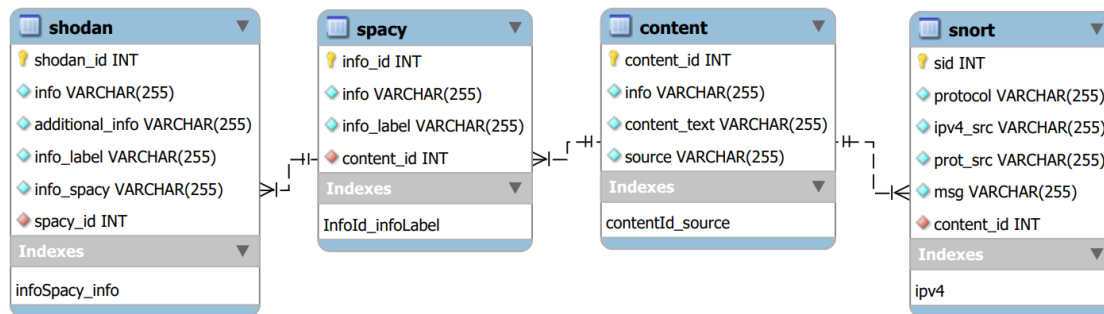


Figure 3.2: Database Structure

## 3.8 Verifying with ScaPy

ScaPy was used to simulate data traffic in a home network, to test if Snort could detect packages coming from malicious IP addresses. The ScaPy implementation is set up to count the number of snort rules. Then, pick random rules, and create packages based on the content of the rule. If no ports has been specified in the rule, port 80 and 443 is used.

When testing this, it is important to change the IP address to make it match the IP address of someone in the network you are monitoring or for example the default gateway of the network you are connected to.

## 3.9 Experiments

To collect data for the experiments a taxii client for hailataxii.com, otx.alienvault.com and open.taxiistand.com was used. For all services the program was set to poll up to 400 000 files from each of them. The program was stopped after running two sequences (collecting up to maximum 800 000 files). Sources having less than 800 000 did not have more files than the ones that were downloaded. The distribution of files are shown in figure 3.3.

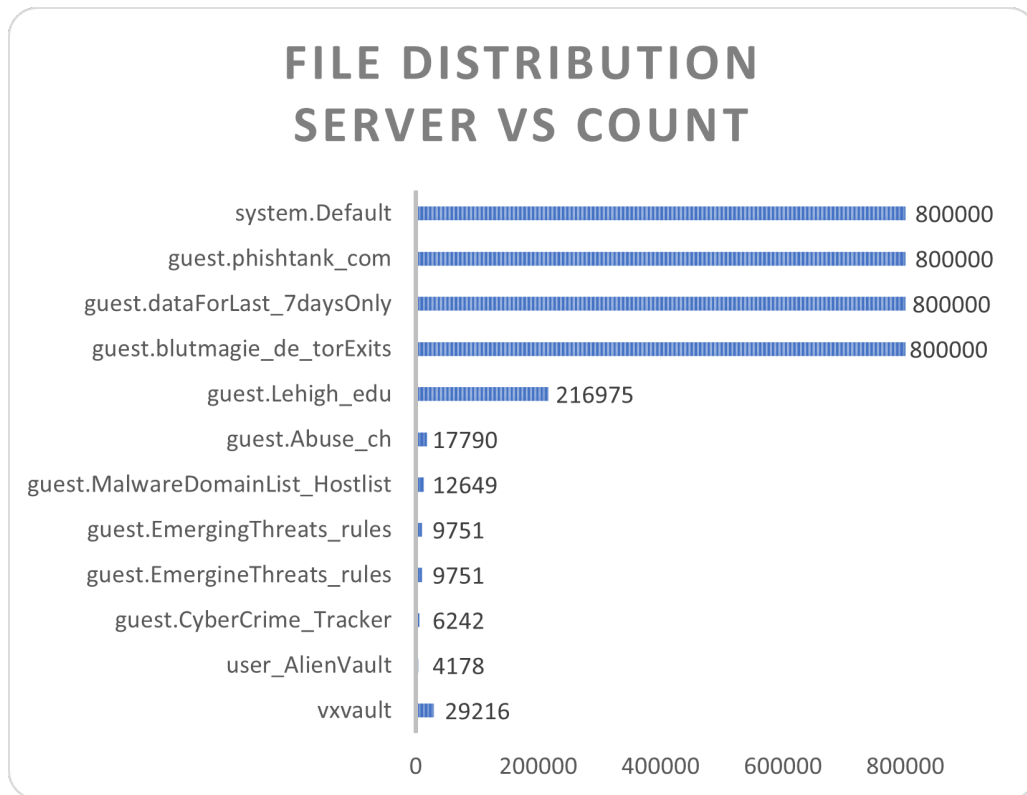


Figure 3.3: File distribution

When the program was processing the information, the spacy implementation is using approximately 2 minutes and 30 seconds on 1 000 files. In average 1 000 files gave 100 distinct IPv4 addresses.

### 3.10 Overall program run

as this program is a proof-of-concept there are several things that are hard coded that would not be in a normal program. As for example, clients created to collect STIX files and shodan user key. These would of cause be input values in a normal program. But this has not been a focus for the development, as the goal has been to check if the task itself were possible. When the program is starting up the main frame of the program is started in addition to three coroutines. The coroutines are one for snort, one for shodan and one for SpaCy. This has been done to be able to accelerate the process, making it possible to run as much processes as possible at once. The first time the program is started. The three coroutines will start. Snort will try to run, but will not have anything to do, and will sleep waiting for shodan to verify IPv4 addresses. Shodan will also try to run, but will also sleep, waiting for IPv4 addresses to search for in the spacy table (database). The spacy implementation check if there is any STIX files that are ready to be processed. When there is not, the spacy coroutine will check if there is any IPv4 addresses left for the shodan implementation of



the program left to process. If it is not, it will initiate downloading of STIX files. First three clients for otx.alienvault.com, open.taxiistand.com and hailataxii.com are created. From taxiistand, the only collection used is vxvault, because the other collections are collected from by using hailataxii. After this the program try to download 400 000 new files from each collection (server). When files are collected, SpaCy will start processing the files, shodan will search for new IPv4 addresses and snort will also sequentially add rules to the database, and refresh the snort rule table.



# Chapter 4

## Results

An overview of all the results for gathering information by SpaCy, and verification of the IPv4 addresses found are shown in figure 4.1. This figure gives the results by percent related to amount of STIX files in the server. This has been done, because there is a huge difference in the number of files for each server. Raw data used to create the figures are summarized in figure 4.2. Radar diagrams has been made for each of the servers, to illustrate the findings. Dates presented under each sub chapter below are the newest and oldest date found by spot checks for all the servers starting with guest. As these are not sorted by date.

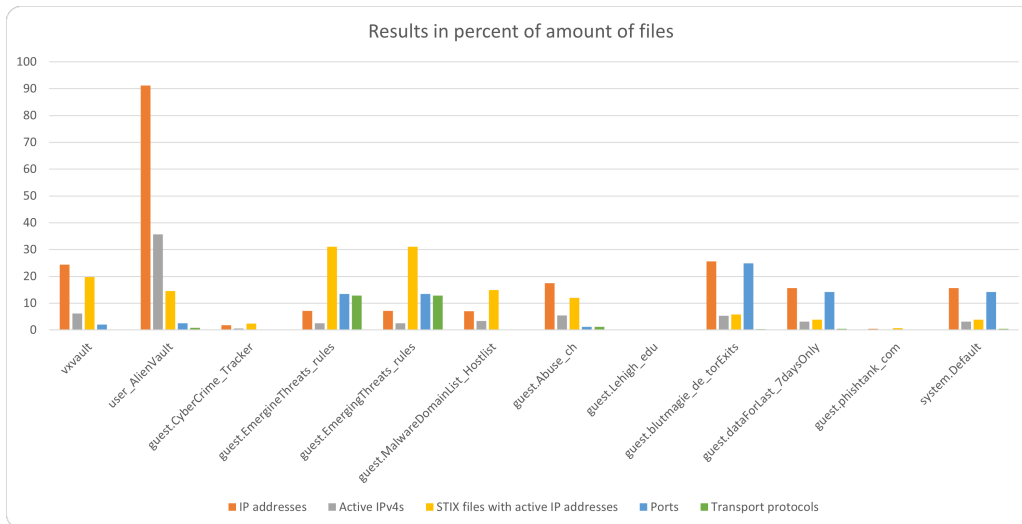


Figure 4.1: Results in relation to found files

Server	Files	STIX files with active IP addresses	IP addresses	Ports	Transport protocols	Active IPv4s
vxvault	29216	5776	7106	602	1	1817
user_AlienVault	4178	607	3812	106	32	1491
guest.CyberCrime_Tracker	6242	149	110	15	0	33
guest.EmergingThreats_rules	9751	3033	695	1306	1248	250
guest.EmergingThreats_rules	9751	3033	695	1306	1248	250
guest.MalwareDomainList_Hostlist	12649	1892	886	0	0	432
guest.Abuse_ch	17790	2132	3100	207	203	971
guest.Lehigh_edu	216975	10	1	0	2	1
guest.blutmagie_de_torExits	800000	46548	204641	198875	3061	42049
guest.dataForLast_7daysOnly	800000	31226	125558	112976	3761	24853
guest.phishtank_com	800000	5147	3306	366	14	1328
system.Default	800000	31226	125558	112976	3761	24853

Figure 4.2: Raw data from results

## 4.1 vxvault

Results from vxvault is illustrated in figure 4.3. one unique IPv4 address were found in approximately each 4th STIX file. Approximately 25 percent of the IP addresses found were active. Each IP address is mentioned in multiple STIX files. Files from this server gives some contribution to port and very little contribution to transport protocol information. Information stored in this database were gathered in the time span 26.08.2015 - 05.11.2015. Most of the files are related to Trojans downloaded when clicking on images, movies or other types of files.

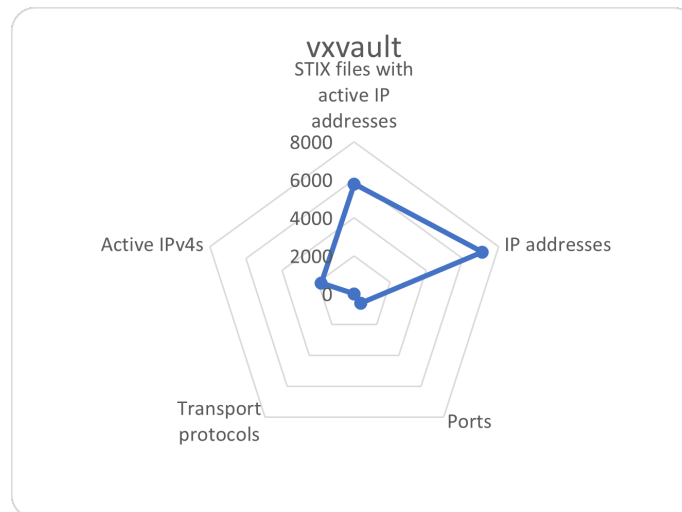


Figure 4.3: Results vxvault server

## 4.2 user\_AlienVault

Results from user\_AlienVault is illustrated in figure 4.4. Approximately 40 percent of all IP addresses found are still active. In each file that contain active IPv4 addresses, in average approximately 2.5 addresses were collected. Files from this server gave some contribution to port and transport protocol information. Information stored in this database were gathered in in the time span 11.19.2014 - 05.05.2022, this being the only server used feeding live data.

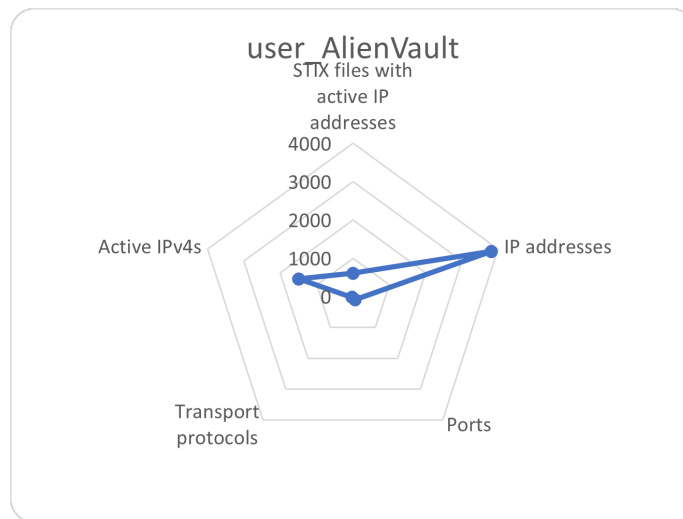


Figure 4.4: Results user\_AlienVault server

## 4.3 guest.CyberCrime\_Tracker

Results from guest.CyberCrime\_Tracker is illustrated in figure 4.5. There were not found many IPv4 addresses in the files from this server. Approximately 30 percent of the IPv4 addresses found were active. Each active IPv4 address were found in several files. Files from this server gives some contribution to ports but no contribution to transport protocol information. Information stored in this database were gathered in the time span 04.11.2014 - 23.02.2015.

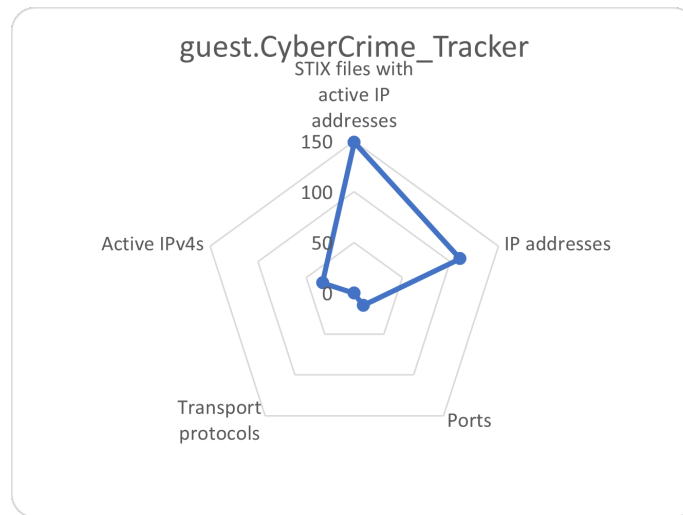


Figure 4.5: Results guest\_CyberCrime\_Tracker

#### 4.4 guest.EmergingThreats\_rules and guest.EmergingThreats\_rules

Results from guest.EmergingThreats\_rules is illustrated in figure 4.6. There were quite some IPv4 addresses in the files from this server. Approximately 35 % of IPv4 addresses were active. Each active IPv4 address were found in several files. Files from this server gives relatively high contribution to ports and transport protocol information. Information stored in this database were gathered in the time span 23.10.2014 - 07.11.2014. Both servers guest.EmergingThreats\_rules and guest.EmergingThreats\_rules gave exactly the same results, and are believed to contain the same data.

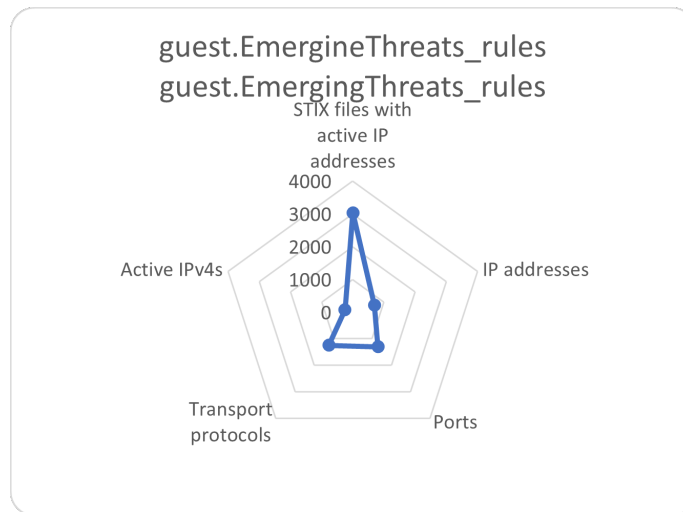


Figure 4.6: Results guest.EmergingThreats\_rules

## 4.5 guest.MalwareDomainList\_Hostlist

Results from guest.MalwareDomainList\_Hostlist is illustrated in figure 4.7. There is a unique IPv4 address in approximately each 14th STIX file. Approximately 50 % of IPv4 addresses found were active. Each active IPv4 address were found in several files. Files from this server gives no contribution to ports and transport protocol information. Information stored in this database were gathered in the time span 05.09.2014 - 04.06.2015.

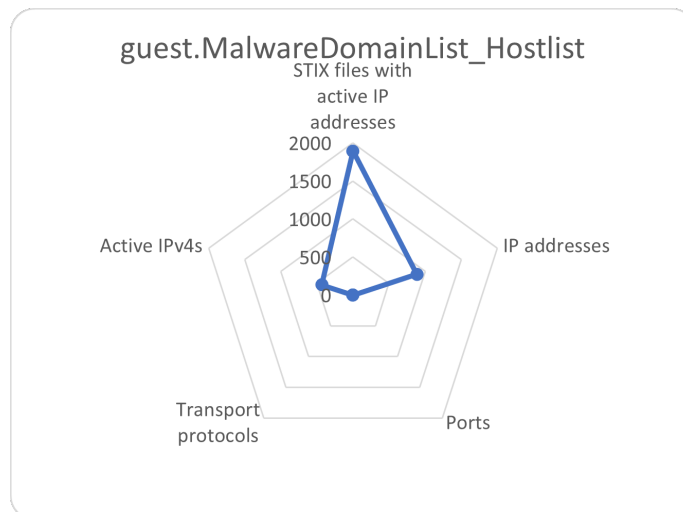


Figure 4.7: Results guest.MalwareDomainList\_Hostlist

## 4.6 guest.Abuse\_ch

Results from guest.Abuse\_ch is illustrated in figure 4.8. There is a unique IPv4 address in approximately each 6th STIX file. Approximately 30 % of IPv4 addresses found were active. Each active IPv4 address were in average found in a couple of files. Files from this server gives some contribution to ports and transport protocol information. Information stored in this database were gathered in the time span 20.10.2014 - 11.10.2018.

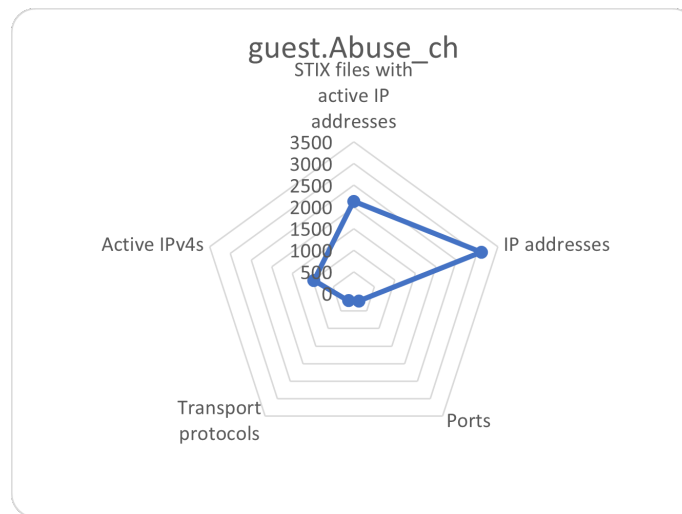


Figure 4.8: Results guest.Abuse\_ch

## 4.7 guest.Lehigh\_edu

Results from guest.Lehigh\_edu is illustrated in figure 4.8. From the over 200 000 files checked, there were one IPv4 addresses found. Files from this server contained little information regarding transport protocol in two files, Information stored in this database were gathered in the time span 17.02.2014 - 11.10.2018.



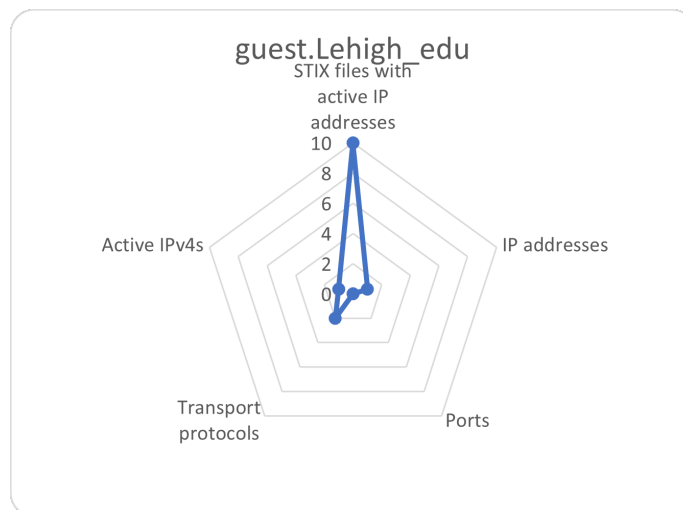


Figure 4.9: Results guest.Lehigh\_edu

## 4.8 guest.blutmagie\_de\_torExits

Results from guest.blutmagie\_de\_torExits is illustrated in figure 4.10. From the 800 000 files checked, One unique IPv4 address were found in approximately each 4th STIX file. Approximately 20 % of IPv4 addresses found were active. 90 % of all active IPv4 addresses found, were also found in files from this database. This server had high contribution to transport protocol information and very high contribution to port information. Information stored in this database were gathered in the time span 27.01.2015 - 21.01.2016.

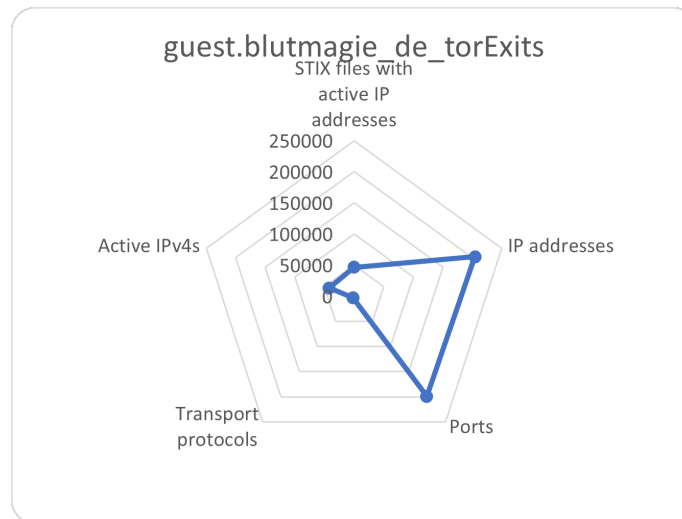


Figure 4.10: Results guest.blutmagie\_de\_torExits

## 4.9 guest.dataForLast\_7daysOnly and system.Default

Results from guest.dataForLast\_7daysOnly is illustrated in figure 4.11. From the 800 000 files checked, one unique IPv4 address were found in approximately each 6th STIX file. Approximately 20 % of IPv4 addresses found were active. This server had high contribution to both transport protocol and port information. Information stored in this database were gathered in the time span 26.03.2015 - 13.07.2016. The results for guest.dataForLast\_7daysOnly and system.Default gave exactly the same results, and are believed to contain the same data.

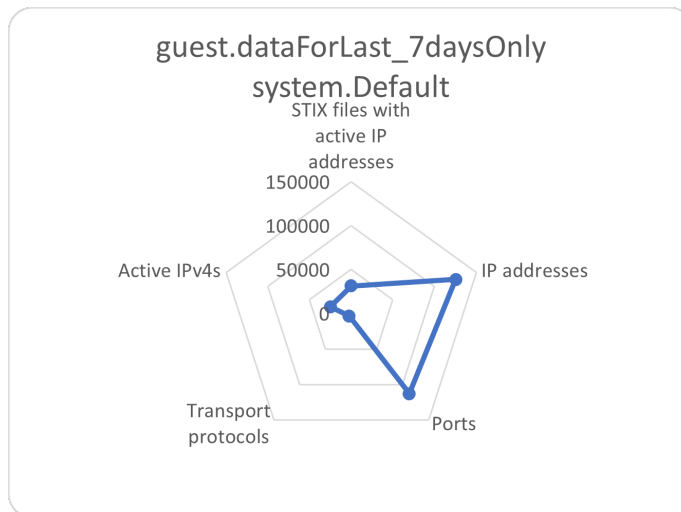


Figure 4.11: Results guest.dataForLast\_7daysOnly

#### 4.10 guest.phishtank\_com

Results from guest.phishtank\_com is illustrated in figure 4.11. From the 800 000 files checked, one unique IPv4 address were found in approximately each 240th STIX file. Approximately 40 % of IPv4 addresses found were active. This server had little contribution to transport protocol and some contribution to port information. Information stored in this database were gathered in the time span 19.03.2015 - 14.11.2017.

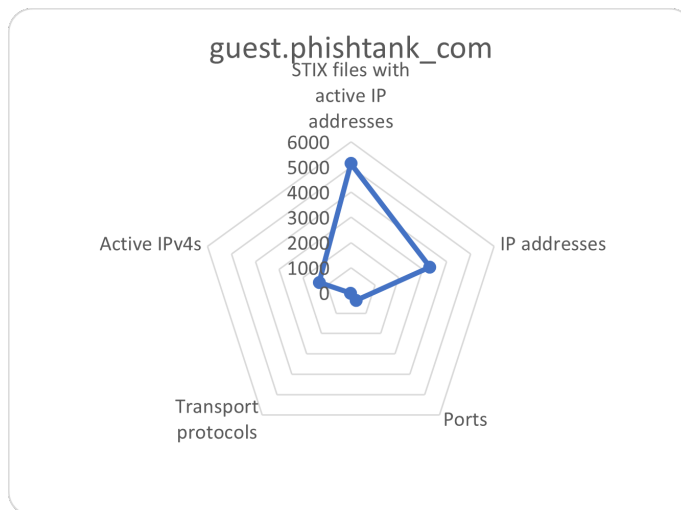


Figure 4.12: Results guest.phishtank\_com

## 4.11 Snort and ScaPy

After the files had been processed by SpaCy and IPv4 addresses verified by shodan.io, there were 48 904 unique IPv4 addresses that were found to be active, and were inserted into the snort table in the database. 308 of these IPv4 addresses were associated with a transport protocol. This resulting in 308 IPv4 addresses having specified with the protocol type. The rest of the IPv4 addresses were set up with an alert for IP, that trigger on all transport protocols over IP. 48 597 rules were specified with IP as protocol. 302 of the IPv4 addresses that had specific transport protocol associated with it, also had information regarding ports that were used in the snort rules. 4 514 of the other 48 599 rules had information regarding ports. These entries were combined into compact rules for the local.rules file in the snort program. Each rule containing up to 500 IPv4 addresses. The IPv4 addresses combined into the same rule share the same ports and the same transport protocol.

The program is set up with a button for testing Snort. When clicking this button, the ScaPy implementation of the program collect information from three random snort rules, and creates packages that Snort should react to. An example of a test is illustrated in figure 4.13. When running Snort the last line in the command prompt will be "Commencing packet processing (pid="A\_UNIQUE\_NUMBER)". Everything underneath this statement is output from Snort alerts. In this example, the alerts chosen were set up with IP as transport protocol. For these cases, ScaPy is set up to send both four UDP and TCP packets. Both types of package initiates alerts. Which again verify that using IP as protocol when the protocol is unknown is sufficient.

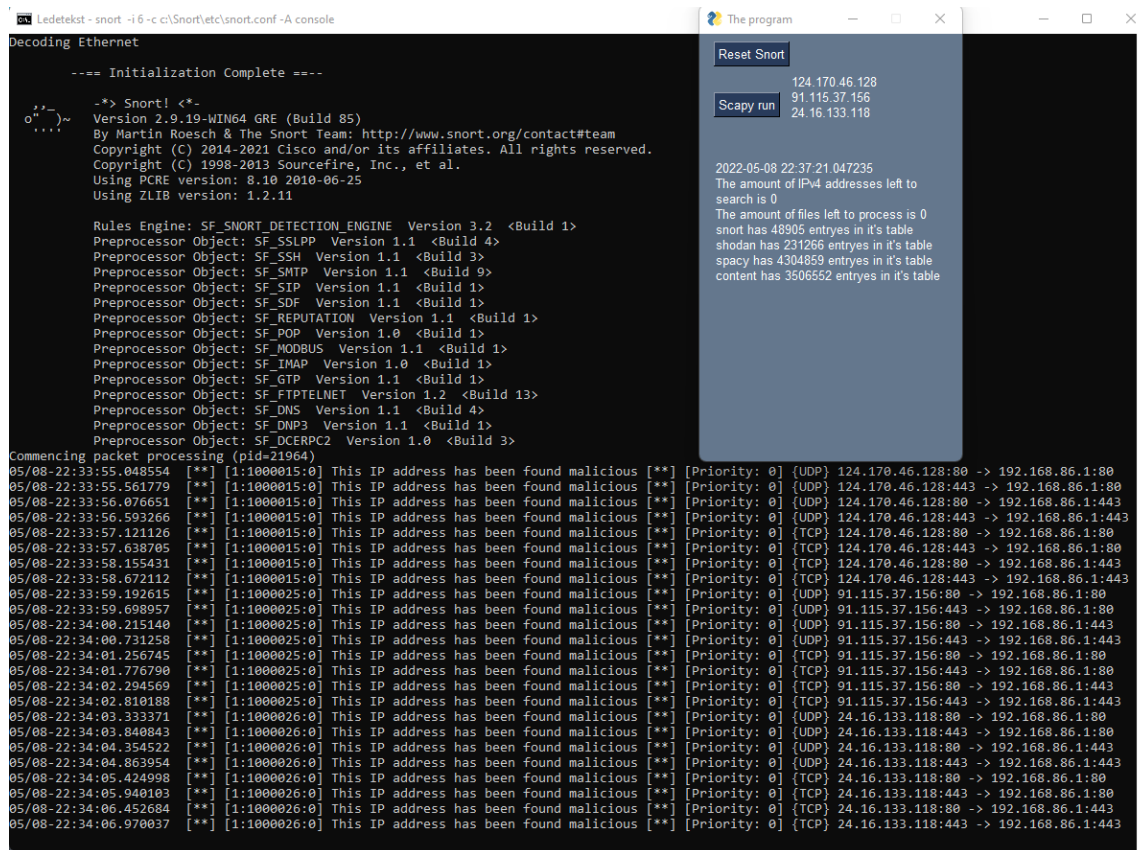


Figure 4.13: Testing Snort implementation



# Chapter 5

## Recourse's

### 5.1 Financial overview

This project is performed as a proof-of-concept project. The aim has been to verify that it is possible to set up an automatic collection of STIX files, collect relevant data from the files, create alert rules based on the information, and at the end, test if rules made can alert packages coming from the IP addresses and the specified additional info collected from STIX files. As this is a proof-of-concept, it was not seen as necessary to use tools, data and software that was not free. At least since there are usable options that are free, and can be used for real case testing. A normal private laptop with a intel core i7 8th gen and 16 GB memory was used.

### 5.2 Environmental accounts

When the program is running with a high number of STIX files available, the power consumption is high. CPU was running at 10 to 17 %, approximately 240 - 290 MB memory and disk capacity at approximately 1 MB/s. But when adding information to the spacy table in the database, the disk capacity peak to approximately 5.2 MB/s.





# Chapter 6

## Discussion

### 6.1 Data

Data used for this study were data from free and open source. A lot of the data from these sources were older data, that may not be relevant. As the STIX-files may also include weaknesses in web pages and endpoints that are not necessarily a IP address that is owned by people or organizations with malicious purposes. These weaknesses may not still be relevant, even if the IP addresses are active. Issues might have been patched, and hardware equipment might have been upgraded.

### 6.2 Spacy

The main purpose of the project was to collect information from the description field in the STIX-files. But as the files were old, and does not exist as JSON files, it has been difficult to collect only this field. During the startup of the project elevate from the tool stix2elevator was used. All files could not be elevated, as they needed to be STIX version 2 for the tool to be applicable. Approximately 30 to 50 % of the files that were processed could not be elevated. Also, there were observations where the elevated files did not elevate the description field when it was present.

To make sure to not miss out on information, the whole STIX files were run through spacy. Using the whole file should not be of to much trouble, as one STIX file usually is related to one incident, observation, campaign e.a. Also usually the rest of the information, except from description are usually references (PUID - Public Unique Identifier) for objects or other files.

The spacy implementation has been created to find data based on the files available. If this application would be used on another data set, the data can have another structure. This can result in information not being collected. Also, the STIX files are stored as a binary sequence of data. If other data should be stored, the program may need to be updated. It would have been beneficial to store the data as a normal string, and not binary for future use.

It may also be the case that all relevant data has not been gathered. A little bit more than 3.5 million files were processed in this project. Because of the quantity, only spot checks were performed. About 10 000 files has been verified manually, and the code has been updated to collect all relevant data found in these files. As only about 0.3 % of the files has been manually verified, there may be relevant data that has not been collected by the program. During end phase of the

project some extra formatting (removing /) were performed on the input to the spacy program. This resulted in a massive increase in gathering ports.

In this project data collection has been performed as simple as possible. During the project, the English large, medium and small model was tested. For the purpose of this project these models did not supply any value themselves. To gather information the "entety\_ruler" and "matcher" modules in SpaCy were used. After testing the different models it was tested to remove some parts of the small English model, but there were some contingencies that was not that easy to understand. Therefor, during large parts of the project the small English model was used.

Towards the end of the project a empty model was used, and "entety\_ruler" and "matcher" was added to it. Time needed to process documents decreased to about 5 % of the processing time. Even when this is a great improvement, it does not help the total program that much, because the big time thief in this program is the shodan implementation.

In this project the reliability of the STIX files has been assumed to be 100 percent reliable. This is unlikely, because the internet environment changes over time. Old bugs may be fixed, and new threats may appear. Also, the reliability of the source of the information has not been verified.

### 6.3 Shodan

When reviewing the data received from shodan.io about 20 percent of the found IPv4 addresses got result from the search. First, Snort rules was made for these IPv4 addresses. But some of the IPv4 addresses returned information from shodan containing "HTTP/1.1 301 Moved Permanently" and "HTTP/1.1 302 Moved". This type of response has been found for approximately one percent of all responses from shodan.io. In the last implementation of the program snort rules were not made for these IPv4 addresses.

A test was performed on the shodan application. All searches done on IP addresses that did not get "found info" was deleted, and all these IP addresses were searched again. This gave an additional 0.5 % increase in results for the IP addresses. Which means that this part of the program will miss some of the relevant IP addresses. The reason for this might be errors like "unable to parse JSON" occur some times, and may be it will be able to parse the JSON object if it tries again. Speed of the searches were an probably will be an issue, as it has not been found an asynchronous API towards shodan.io in python. In addition, each search can take from approximately 0.3 seconds up to above 18 seconds. Usually each search did take less than one second when using the api.host in shodan. When using api.search many of the searches did take two to four seconds. There is an asynchronous API in Ruby [30], so hopefully a similar solution will be made for python as well.

Even with this when the program was running the last time it did make an average of 0.77 search per second while also processing approximately one million files for one day. The intervals for the searches were set to 0.6 seconds, which is the lowest found that does not give error for too many searches executed. A picture of the amount of searches performed the last period running the program is available in VI.1

### 6.4 Snort

Snort has been used as Intrusion Detection System in this project. Rules has been made for snort based on information gathered from the SpaCy implementation of the program.

All information used are gathered from STIX files. All relevant information gathered in SITX files that the IP address has been mentioned in, is used to generate rules for that IP address. If no info regarding protocol has been found, the rule will use IP as transport protocol. This will cause all protocols carried by IP to cause an alert. This approach has been chosen to make sure that contact with these IPv4 addresses are alerted. If no ports have been found "any" has been used in the alerts, causing an alert independent of what port has been used.

There are some drawbacks for this method. If port or transport protocol is referenced in one STIX file and not in other files referenced, it could be negative. Imagine that another port or transport protocol were used in the case that the other STIX files refer to, just that they were not observed, or for other reasons were not added to the STIX file. This could lead to specifying a snort file to not include relevant activity, because the information was not specified. The result of this would cause other port or transport protocol being used by the IPv4 address to be undetected.

As mentioned earlier, one of the biggest issues for safety engineers are to filter out relevant alerts [51]. This is why the alerts are made as specifics as possible.

There is one issue that has not been verified by any source. Having too many rules seems to be an issue. As when the amount of rules are higher than approximately 138 200 rules it seems to stop collecting rules. But it might be related to the count of bits used or total length of string. Because, it can stop collecting in the middle of a rule, which will cause "Fatal Error" for snort, as it would cause one rule to be incorrect. At the end of the project an implementation to the code were introduced to compress the snort rules, to see if it could help with this problem. The update in the code adds IP addresses that have the same ports and transport associated to it on the same rule. This reduced the time for starting up snort, as well as having the possibility to have rules for many more IPv4 addresses. 500 IPv4 addresses was set as a max limit for IPv4 addresses. But this will most probably not be the highest usable amount for each rule. It was also tested with 5 000 IPv4 addresses. But each rule has a limit at 32 768 characters, which was exceeded with that many IPv4 addresses.

## 6.5 ScaPy

In this project ScaPy was simply used to create data traffic that Snort should be able to alert. The implementation of the tool selects three random entries from the database snort table. Then it generates traffic based on the available information. In general, this only test if the rules are written correctly, and results in alerts.



## Chapter 7

# Conclusion

The main goals for this project were to collect STIX files from STIX servers and store them. Then use NLP to extract relevant information from the collected STIX files. Verify that IPv4 addresses found were active. Create alerts rules in an Intrusion Detection System. Test the rules with simulated network traffic.

With regard to the STIX files, these files were gathered and stored in a suitable amount for the purpose. Most of these files were gathered from the MITRE organization and represent real data (not dummy or test data).

A high amount of data was collected by SpaCy from the STIX files. Spot checks has been performed. But the amount of STIX files were high and it is not unreasonable to think that some information has not been collected.

Shodan was used to verify if IP addresses were active. Some of these did have information like "HTTP/1.1 301 Moved Permanently" and "HTTP/1.1 302 Moved". Resulting in some false positives when creating the Snort rules.

Snort was set up successfully, and were able to detect the suspicious network traffic.

This project proves that it is possible to automatically collect STIX files, collect relevant information and use that information to automatically create snort rules.



## Chapter 8

# Discussion and future work

This program has been developed to create rules in a Intrusion Detection System. Which is a small step in the direction of making a software that can really help out safety engineers in their every day work. The program helps by creating new rules based on STIX files that are published, and will try to collect new files frequently.

By implementing this program "as is" will possibly lead to flooding the user with alerts. If it does, the program will not be of any help. Because it can be hard to find alerts that correspond to an actual cyber threat.

For any future work taxii2client or similar should be used to extract JSON STIX files, to handle the data easier. In general, there are two main types of further work. One is collecting differently, and verifying the quality of the data. The other one will be generating rules that not only generate alerts, but also initiate actions to protect against possible threats.

In this project there were not found any free client for taxii2client with enough free STIX files to collect. That is why cabby was used, and approximately 3 500 000 files were used for this project.





# Chapter VI

## VI. Appendix A

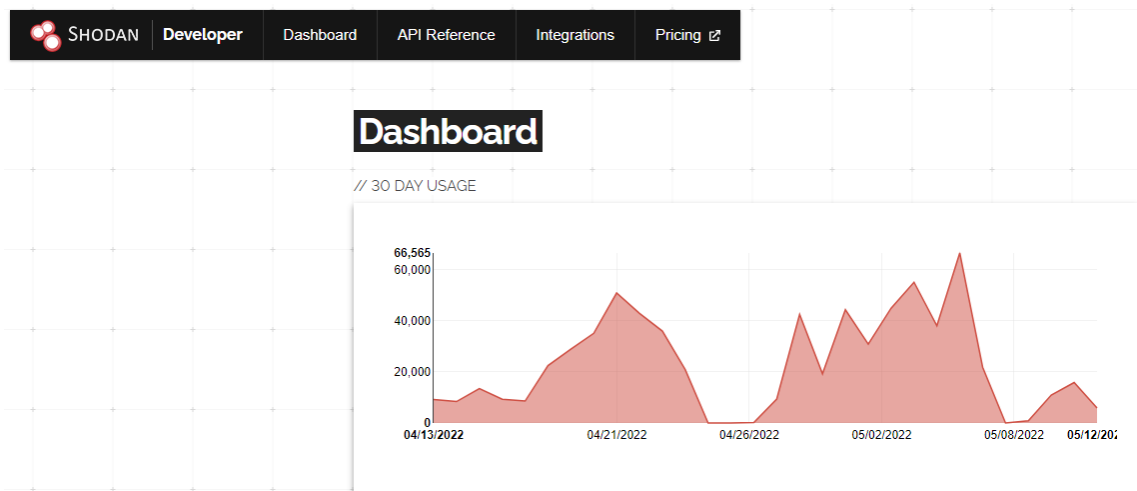


Figure VI.1: Shodan.io 30 days usage

## VI. Acronyms

- STIX** Structured Threat Information Expression
- TAXII** Trusted Automated eXchange of Indicator Information)
- CTA** Cyber Threat Alliance
- CTI** cyber threat intelligence
- OSINT** Open Source intelligence
- XSS** Cross Site Scripting
- NLP** Natural Language Processing
- DOS** Denial of Service
- DDOS** Distributed Denial of Service
- IOT** Internet of Things

**DOS** Denial of Service  
**SYN** synchronization  
**SYN-ACK** Synchronization acknowledgement  
**SDO** STIX Domain Objects  
**SCO** STIX Cyberobservable objects  
**SRO** STIX Relationship Objectn  
**TTP** Tactics, Technique or Procedures  
**CVE** Vulnerabilities and Exploits  
**API** Application programming Interface

# Chapter VII

## Bibliography

- [1] 11 password cracker tools (password hacking software 2022). available at <https://www.softwaretestinghelp.com/password-cracker-tools/>, Accessed: 24.01.2022.
- [2] 16 best ddos attack tools in 2022. available at <https://lab.wallarm.com/16-best-ddos-attack-tools-in-2022/>, Accessed: 23.01.2022.
- [3] Basic shodan search. Available at <https://shodan.readthedocs.io/en/latest/examples/basic-search.html>, Accessed: 21.03.2022.
- [4] Dirb, forced browsing. available at <https://www.kali.org/tools/dirb/>, Accessed: 22.03.2022.
- [5] Dmitry, forced browsing. available at <https://www.kali.org/tools/dmitry/>, Accessed: 22.03.2022.
- [6] Forced browsing. available at [https://owasp.org/www-community/attacks/Forced\\_browsing](https://owasp.org/www-community/attacks/Forced_browsing), Accessed: 21.03.2022.
- [7] Http flood attack. available at <https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/>, Accessed: 22.03.2022.
- [8] Http flood attack. available at <https://www.softwaretestinghelp.com/ddos-attack-tools/>, Accessed: 22.03.2022.
- [9] Low orbit ion cannon. available at <https://www.imperva.com/learn/ddos/low-orbit-ion-cannon/>, Accessed: 22.03.2022.
- [10] Maltego, forced browsing. available at <https://www.kali.org/tools/maltego/>, Accessed: 22.03.2022.
- [11] Nihto2. available at <https://cirt.net/nikto2>, Accessed: 22.03.2022.
- [12] Password lists example. available at <https://github.com/duyet/bruteforce-database>, Accessed: 24.01.2022.

- [13] Rainbow lists example. available at <http://project-rainbowcrack.com/table.htm>, Accessed: 24.01.2022.
- [14] Scapy. Available at <https://scapy.net>, Accessed: 21.03.2022.
- [15] What is a quic flood. available at <https://www.cloudflare.com/learning/ddos/what-is-a-quic-flood/>, Accessed: 22.03.2022.
- [16] Sean Barnum. *International archives of the photogrammetry, remote sensing and spatial information sciences*. Available at <http://stixproject.github.io/getting-started/whitepaper/>.
- [17] Danielle Bodnar. Macro virus: What is it and how to remove it. Available at <https://www.avast.com/c-macro-virus>, Accessed: 03.04.2022.
- [18] Brooks Chuck. Available at <https://www.forbes.com/sites/chuckbrooks/2022/01/21/cybersecurity-in-2022--a-fresh-look-at-some-very-alarming-stats/?sh=5ee2b9896b61>.
- [19] Just Ed. How to install and run snort on windows. Available at <https://www.youtube.com/watch?v=naLbhKW62nY&t=237s>, Accessed: 12.02.2022.
- [20] Lauren Feiner. Cyberattack hits ukrainian banks and government websites. 2022. Available at <https://www.cnbc.com/2022/02/23/cyberattack-hits-ukrainian-banks-and-government-websites.html>.
- [21] Herjavec Group. Available at <https://www.herjavecgroup.com/wp-content/uploads/2021/10/2021-Healthcare-Cybersecurity-Report.pdf>.
- [22] Javier Gómez. Introduction to scapy ? Available at <https://santandercto.com/en/guide-using-scapy-with-python/>, Accessed: 07.04.2022.
- [23] Zahra Jadidi Ali Dorri Raja Jurdak Kathy Nguyen, Shantanu Pal. A blockchain-enabled incentivised framework for cyber threat intelligence sharing in ics. 2021. Available at [https://www.researchgate.net/publication/356711532\\_A\\_Blockchain-Enabled\\_Incentivised\\_Framework\\_for\\_Cyber\\_Threat\\_Intelligence\\_Sharing\\_in\\_ICS](https://www.researchgate.net/publication/356711532_A_Blockchain-Enabled_Incentivised_Framework_for_Cyber_Threat_Intelligence_Sharing_in_ICS).
- [24] Kaspersky Lab. What is the polymorphic virus? Available at <https://www.kaspersky.com/resource-center/definitions/what-is-a-polymorphic-virus>, Accessed: 03.04.2022.
- [25] Nica Latto. What are keyloggers and how do they work? Available at <https://www.avast.com/c-keylogger>, Accessed: 03.04.2022.
- [26] William Mattingly. Introduction to spacy 3, a free course for beginners. 2021. Available at <http://spacy.pythonhumanities.com/intro.html#>.
- [27] Kevin Meynell. Cloudflare launches 1.1.1.1 dns service with privacy, tls and more. Available at [https://www.internetsociety.org/blog/2018/04/cloudflare-launches-enhanced-dns-service/?gclid=Cj0KCQjwsdiTBhD5ARIsAIPw8CJUs3FqUFyxM3i91rbc3paBCF2TYdpOMWFsEqY7s9YXtPkFjfHnU04aAlcREALw\\_wcB](https://www.internetsociety.org/blog/2018/04/cloudflare-launches-enhanced-dns-service/?gclid=Cj0KCQjwsdiTBhD5ARIsAIPw8CJUs3FqUFyxM3i91rbc3paBCF2TYdpOMWFsEqY7s9YXtPkFjfHnU04aAlcREALw_wcB), Accessed: 07.05.2022.

- [28] None. What is the difference between malware and a virus? Available at <https://www.mcafee.com/enterprise/en-us/security-awareness/ransomware/malware-vs-viruses.html>, Accessed: 03.04.2022.
- [29] Rich Piazza OASIS, Edited by Bret Jordan and Trey Darley. Stix™ version 2.1. Available at [https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_muftrcpnf89v](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_muftrcpnf89v), Accessed: 29.03.2022.
- [30] picatz. picatz/shodanz. Available at <https://github.com/picatz/shodanz>, Accessed: 17.04.2022.
- [31] Daniel Schlette. Cyber threat intelligence. 2021. Available at [https://www.researchgate.net/publication/349427184\\_Cyber\\_Threat\\_Intelligence](https://www.researchgate.net/publication/349427184_Cyber_Threat_Intelligence).
- [32] IBM Security. Cost of a data breach report 2021. 2021. Available at <https://www.endpointprotector.com/blog/the-cost-of-a-data-breach-in-2021/#>.
- [33] Renatta Siewert. Polymorphic viruses — best practices to prevent them. Available at <https://www.mimecast.com/blog/polymorphic-viruses--best-practices-to-prevent-them/>, Accessed: 03.04.2022.
- [34] Anantaa Kotal Anupam Joshi Soham Dasgupta, Aritran Piplai. A comparative study of deep learning based named entity recognition algorithms for cybersecurity. 2020. Available at <https://ieeexplore.ieee.org/abstract/document/9378482>.
- [35] Christopher Tozzi. What do hackers want, anyway? a look at different cyberattack goals. Available at <https://www.channelfutures.com/strategy/what-do-hackers-want-anyway-a-look-at-different-cyberattack-goals>, Accessed: 24.01.2022.
- [36] Unknown. Documentation, natural language toolkit. Available at <https://www.nltk.org/>, Accessed: 01.05.2022.
- [37] Unknown. Election security spotlight – signature-based vs anomaly-based detection. Available at <https://www.cisecurity.org/insights/spotlight/cybersecurity-spotlight-signature-based-vs-anomaly-based-detection>, Accessed: 09.04.2022.
- [38] Unknown. Industrial-strength natural language processing. Available at <https://spacy.io/>, Accessed: 03.04.2022.
- [39] Unknown. Introduction to taxii. Available at <https://oasis-open.github.io/cti-documentation/taxii/intro>, Accessed: 22.04.2022.
- [40] Unknown. Nist incident response plan: Building your own ir process based on nist guidelines. Available at <https://www.cynet.com/incident-response/nist-incident-response/>, Accessed: 01.04.2022.
- [41] Unknown. Outsteel, saintbot delivered by spear phishing attacks targeting ukraine. Available at <https://otx.alienvault.com/pulse/621ccab95c4b796eeea2ee78>, Accessed: 12.04.2022.

- [42] Unknown. Outsteel, saintbot delivered by spear phishing attacks targeting ukraine. Available at <https://otx.alienvault.com/pulse/621ccab95c4b796eaea2ee78>, Accessed: 12.04.2022.
- [43] Unknown. Search shodan, getting started. Available at <https://shodan.readthedocs.io/en/latest/tutorial.html#searching-shodan>, Accessed: 22.04.2022.
- [44] Unknown. We are the cyber threat alliance. Available at <https://cyberthreatalliance.org/>, Accessed: 30.04.2022.
- [45] Unknown. What are signatures and how does signature-based detection work? Available at <https://home.sophos.com/en-us/security-news/2020/what-is-a-signature>, Accessed: 09.04.2022.
- [46] Unknown. What is a denial of service attack (dos)? Available at <https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos>, Accessed: 20.04.2022.
- [47] Unknown. What is a firewall? Available at <https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html>, Accessed: 09.04.2022.
- [48] Unknown. What is an intrusion prevention system (ips)? Available at <https://www.forcepoint.com/cyber-edu/intrusion-prevention-system-ips>, Accessed: 08.04.2022.
- [49] Unknown. What is bait switch attack and how is it different from clickjacking? Available at <https://cyware.com/news/what-is-bait-switch-attack-and-how-is-it-different-from-clickjacking-d33b450a>, Accessed: 02.04.2022.
- [50] Unknown. What is endpoint detection and response (edr)? Available at <https://www.mcafee.com/enterprise/en-us/security-awareness/endpoint/what-is-endpoint-detection-and-response.html>, Accessed: 09.04.2022.
- [51] Aliyu Aliyu Ying He, Leandros Maglaras and Cunjin Luo. Healthcare security incident response strategy - a proactive incident response (ir) procedure. *International archives of the photogrammetry, remote sensing and spatial information sciences.*, page 10, 2022.