WORCESTER POLYTECHNIC INSTITUTE

MAJOR QUALIFYING PROJECT

SecBI: Cyber Threat Intelligence

Author:
Isamu NAKAGAWA

Supervisor:
Dr. Micha HOFRI

A Major Qualifying Project submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the degree of Bachelor of Science

March 23, 2017

Worcester Polytechnic Institute

Abstract

Computer Science

Bachelor of Science

SecBI: Cyber Threat Intelligence

by Isamu NAKAGAWA

The cyber security company SecBI analyzes the network traffic of clients to understand the full scope of cyber threats. Threat intelligence data is knowledge used to help recognize threats. The SecBI infrastructure currently uses threat intelligence from a single source and their solution is not readily expandable to use other sources. This limits the potential to discover and apprehend cyber threats.

I developed a threat intelligence system to collect data from multiple sources that is expandable to use other threat intelligence sources. Additionally, my system allows intelligence discovered by SecBI to be shared with a trusted community.

Acknowledgements

I would like to thank the following for their help and guidance during this project.

- Professor Micha Hofri, Computer Science, WPI, for his assistance and guidance throughout the entire project.
- Professor Isa Bar-On, Mechanical Engineering, WPI, for making this project possible and for working with Worcester Polytechnic Institute to establish an MQP in Israel.

The SecBI executive team for allowing me to work on this MQP project at SecBI.

- Gilad Peleg, SecBI
- Yaron Mashav, SecBI
- Alex Vaystikh, SecBI
- Doron Davidson, SecBI

The other members of SecBI for help and guidance throughout the project.

- Adi Leshem, SecBI
- Ariel Nemtzov, SecBI
- Igor Osetinsky, SecBI
- Liv Aleen Remez, PhD, SecBI
- Chen Dorot, SecBI

Contents

Chapter 1

Introduction

1.1 Cyber Security

The defensive side of security can only deploy a finite number of defense strategies, however the offense has infinite options of attack. Celeste Matarazzo, a data science expert from Lawrence Livermore National Laboratory explained, "It's about more than racial or gender diversity – it's about diversity of thought. Both cybercrime and cybersecurity are only limited by imagination, and we as a nation can't be secure without a diverse set of problem solvers to counter the cyber threat" (Matarazzo, 2014). Cyber security must constantly adapt to counter the latest threats.

1.1.1 Definition

Cyber security consists of, "measures taken to protect a computer or computer system (as on the Internet) against unauthorized access or attack," according to the Merriam Webster dictionary (Merriam-Webster Online, 2009). However, the meaning of this modern phrase is still evolving and it is arguably vague as to what is included in the definition. Other related terms such as *privacy* and *surveillance* are often mistakenly confused with cyber security, although cyber security can help to prevent privacy and surveillance violations (Fischer, 2016).

A cyber attack may be viewed by examining three factors: the identification of the attacker, the vulnerability the attack is exploiting, and the outcome of the attack. Cyber attacks have the potential to cause great harm to national security, the economy, and the safety of citizens (Fischer, 2016).

Cyber security is a subcategory of information security, relating to security specifically in the cyber realm. It is important to understand the goals of information security, which is defined in *federal law (44 U.S.C. §3552(b)(3))*.

> protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide-
>
> (A) integrity, which means guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity;
>
> (B) confidentiality, which means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information; and
>
> (C) availability, which means ensuring timely and reliable access to and use of information (FIPS PUBS, 2004).

1.1.2 History

As modern society becomes increasingly dependent upon computers, cyber security has become a major field in computer science. Computers are ubiquitous in modern times. Cyber security is becoming increasingly integral to almost every facet of business operations (Fischer, 2016). According to the professional services network, PwC's research, the number of security incidents across all industries rose by 38% in 2015. This study has been performed for 12 years and this is the largest annual increase to date (Coopers, 2015).

Both private and government institutions have continued to increase their investment in cyber security. Today, federal agencies spend more than 10% of their annual ICT budgets on cybersecurity (Fischer, 2016).

Cyber security has only recently become a major field of research. In the early 2000s analysts took cyber security less seriously. In 2002, James Lewis of the Center of Strategic and International Studies (CSIS) wrote that cyber warfare would not cause significant harm and declared cyber weapons were, "weapons of mass annoyance." (Shakarian, 2013). Furthermore, Marcus Ranum, a security expert, argued in 2004 that cyber war would not be economically effective. At the time, these theories were more believable because acts of cyber warfare had minimal impact (Shakarian, 2013).

In recent years cyber attacks have become increasingly common and impactful. Some examples of the increase include the 2006 hijacking of Israel IP addresses by the armed group Hezbollah. In 2007 the U.S. Department of Energy demonstrated how a real power generator could be destroyed through cyber attacks alone. In 2008 Russia used cyber attacks against Georgia's media companies to prevent them from communicating internationally. The same year an Iraqi militant group was able to steal drone video feeds for military intelligence purposes. In 2009 researchers found exiled Tibetan leaders' computers had been compromised and monitored (likely by the Chinese government) for over two years. In 2010 a computer worm known as Stuxnet was able to destroy uranium enrichment facilities owned by Iran. In 2010 the cyber group Anonymous used denial of service attacks to disrupt government computers in Middle Eastern countries (Shakarian, 2013).

Today, malicious users are gaining access to more advanced cyber attacks. Organized criminals are now able to rent or buy ready-made cyber attacks from underground markets (Dell SecureWorks, 2014). An example of this was when the threat intelligence firm Crowdstrike found twelve malicious groups in China using the same exploit codes within 24 to 72 hours of each other (CrowdStrike, 2016). These attacks are causing large profit losses, totaling approximately a millions dollars per attack according to research by Kaspersky Labs (Kaspersky Lab, 2015). To adopt to the advanced threats in the cyber world companies such as SecBI are implementing modern solutions such as *Threat Intelligence*, as explained in the following sections (Jasper, 2017).

1.2 SecBI

SecBI is a cyber security company that uses big data algorithms to find patterns in a client's network and seek out anomalies that may indicate a security breach. Specifically, SecBI searches the logs of a client's network traffic and is able to provide a complete picture of the inner-workings of an attack. This is special because sometimes security analysts only see an isolated security incident, but miss the larger context. Detailed forensic information and notifications are displayed to the client's security team using a web interface. On the SebBI website, the company claims to perform, "Advanced Threat Detection", "Accelerate Incident Detection and Response", and "Understand the Full Scope of Evolving Threats".

The logs may be millions of entries or more, so to interpret the data SecBI uses clustering algorithms, using multiple computers working together to find solutions. The system breaks the problem into smaller pieces that can be solved individually to completely solve the larger problem. SecBI has a team of data scientists who seek new ways to optimize the proprietary algorithm and expand its functionality.

It is important to note the distinction between an *intrusion detection system* (IDS) and an *intrusion prevention system* (IPS). An IPS will actively monitor the network and try to prevent security breaches, while an IDS will passively monitor the network and alert the administrators if an intrusion or malware is found. An IDS does not directly handle the flow of network traffic. Both systems together will make a stronger defense. Each situation will determine which of these systems in more beneficial. The SecBI system is an IDS because it passively monitors the network.

SecBI uses cyber security metrics to

1. Confirm that the client's security controls comply with a policy, process, or procedure
2. Pinpoint the client's security strengths and weaknesses and provide analytical feedback
3. Analyze trends in the client's security, inside and outside the organization

Through helping clients monitor security performance, the SecBI system will help to identify changes and recommend ways to improve the organization's security (Voeller, 2014).

While the central system SecBI uses is the distributed big data algorithm to find irregularities, other systems also contribute to the success of the system as a whole. These are sometimes described as *enrichments*, which are fed into the distributed algorithm to improve the results. One of these enrichments is *threat intelligence*.

1.2.1 Threat Intelligence

Data is categorized as intelligence if it is processed in a logical and analytical manner. In the book, *How to Define and Build an Effective Cyber Threat Intelligence Capability*, Dalziel explains that intelligence must meet three requirements, listed below (Dalziel, 2015).

1. Relevant: The information must relate to, or at least potentially relate to, *your* enterprise, industry, networks, and/or objectives.
2. Actionable: It must be specific enough to prompt some response, change, or decision, or to inform and explicit decision *not* to act.
3. Valuable: Even if relevant and actionable, if the data (and the action) do not contribute to any useful business outcome, there is no value.

Threat intelligence systems provide a framework or platform to share the latest known security flaws with a community. Threat intelligence may come in the form of known malicious internet addresses, the location of a sighting of a threat, or the identification of a file that contains malware. Additionally, it may describe the context of the threat. It is a collection of clues to provide insight on a threat. This information helps assess, qualify, prioritize and remediate the threats a system faces.

Today, information is the key to the defense against fast-moving emerging threats (Hacker5, 2012).

There is a strong need for urgent sharing of information about cyber attacks. When actual attacks were observed during 2014 by the cyber security company Risk Analytics, "75 percent of attacks spread from Victim 0 to Victim 1 within one day, while over 40 percent hit the second organization in less than an hour" (The MITRE Corporation, 2015). The faster and more broadly the many types of cyber threat intelligence are shared, the greater the chance other organizations have to stop similar attacks (Jasper, 2017).

Overall, threat intelligence frameworks provide these benefits to an organization:

- Eliminate duplicate work
  - Many organizations find the same analytical work.
- Faster detection
  - Distributed detection across companies.
- Interoperability
  - Share malware information in a standardized format.
- Support automation
  - Import and export features through a computer interface that is performed without human interaction.

(MISP, 2017)

1.3 Project Overview

My project was to both research various threat intelligence frameworks in order to find the options best suited for SecBI and develop a threat intelligence system for SecBI that collects and shares information from threat intelligence feeds. I worked with SecBI to determine their needs from a threat intelligence framework and researched and experimented with over ten threat intelligence options. After agreeing with SecBI on an appropriate option, I implemented a system that would connect the current SecBI infrastructure with the new threat intelligence framework. The system I created allows for easy access of collected data through an application program interface (API). Furthermore, it allows for future expansions of new threat intelligence frameworks and complies with current SecBI architecture.

The new threat intelligence system is developed in a robust way so new features may easily be added. The cyber security software is quickly evolving and it is important to develop with the future in mind. The project was developed with the following design goals:

- Modular and Encapsulated
  - Dynamic settings without code modification
- Pluggable
  - Exchange threat intelligence sources as needed while maintaining the same infrastructure

- Prioritization
  - Analyze and make intelligent decisions about which threat intelligence source to use

Currently, SecBI uses a limited amount of threat intelligence from a single source. SecBI queries a simple threat intelligence aggregation platform database to obtain a blacklist of known threat domains and Internet Protocol addresses (IP). This enrichment feature feeds into the algorithms used by SecBI to monitor network traffic. The new threat intelligence system that I proposed replaces the current intelligence platform and supports more intelligence sources to collect and store information. Licensing is a major factor to SecBI, as the current threat intelligence platform source can not be used in production (without proper licensing license). These factors make an update to the threat intelligence enrichment of SecBI an appealing project.

In addition to collecting data from multiple threat intelligence sources, the new system will also have the ability to share SecBI threat intelligence data with a trusted community. This service initially will be a standalone system to SecBI, but potentially utilize the threat intelligence sharing services should SecBI partner with other companies. The system will ideally run on a central SecBI server to keep a more complete record of threat intelligence data. However another option is to install it locally for various clients of SecBI. The choice is made by the customer clients, as it is their data.

Chapter 2

Background

2.1 Need for Threat Intelligence

There is a strong need for threat intelligence feeds today because of the large growth of cyber attacks and the increase of sensitive information stored digitally (Lingenheld, 2017). The malware population increased exponentially during the last decade and reached a historical highpoint in 2011. Twenty six million newly created unique malware samples were created (Panda Security, 2012) (Gerhards-Padilla, 2012). Furthermore, according to a Techproresearch survey, 84% of IT professionals became more concerned about security and privacy between the years 2014 and 2015 (Tech Pro Research, 2015).

Organizations view cyber threat intelligence as a challenge by itself within the field of cyber security, because no organization has enough information to have an adequate scope of relevant information to access the large expanding threat landscape.

The way to overcome this limitation of knowledge is through the sharing of relevant cyber threat information among trusted partners and communities. By information sharing, each partner has the potential to achieve a more complete understanding of the threat landscape. Furthermore, less duplicate work is performed if companies are able to utilize existing solutions (The MITRE Corporation, 2012).

A threat intelligence feed will allow companies to share their unique point of view of cyber defense with each other to create a more complete defense model - a trusted network of threat intelligence data. Data collected from analysis of threats is organized and indexed to create a threat intelligence datastore. This datastore is shared between multiple organizations so existing solutions can be utilized to solve new problems for a faster and more efficient response. Additionally, the threat intelligence system will allow developers to write programs to automatically collect information from the trusted centralized datastore through the threat intelligence system's API. This means the latest cyber defense knowledge would be collected and utilized immediately by machines, instead of requiring human intervention.

Overall threat intelligence works in a cycle, as shown in figure 2.1. A user will *observe* the behavior of malware, *collect* information about the malware, *analyze* the collected information, *share* the information with a trusted community, and finally *adapt* to the communities' findings. It should be noted that it is important to form a *trusted community* to share information. Otherwise the community is prone to false information by illegitimate sources. Also, shared information should be stripped of identifiable information to maintain anonymity. An organization will be much more likely to share information that does not have the potential to harm themselves by providing information on their internal secrets.

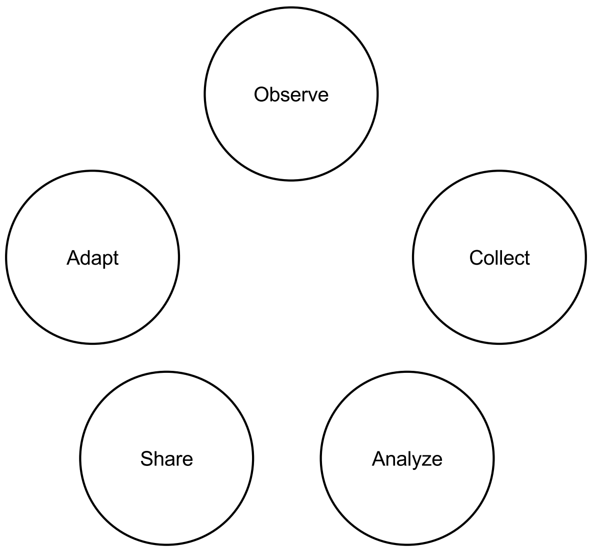

FIGURE 2.1: Threat Intelligence Cycle, Clockwise, (Whalen, 2015)

2.2 SecBI Threat Intelligence

Currently, SecBI uses threat intelligence specifically to obtain black lists for known malicious IP addresses. This is important because an infected computer will likely need to communicate to a malicious controller somewhere in the world so it can receive remote instructions. By blacklisting the malicious IPs on the network, the infected computer will not be able to speak with the malicious controller and be rendered ineffective. However, there are tricks to get around blacklists.

A *domain name generation algorithm* (DGA) is used to calculate the names of new domains on the infected machine. Only the malicious user knows the algorithm so the malicious user will predict the new address and setup a control server listening at the address, and therefore bypass blacklists. This DGA technique strengthens the addressing of a malicious computer and allows a controller to dynamically provide command and control servers (Gerhards-Padilla, 2012).

An example of a blacklist threat intelligence entry is shown below. The first line describes the columns, and the second line is the actual entry. In this example, *xzy* is the malicious domain. Notice other metadata included in the listing.

```
#fields indicator indicator_type meta.source meta.do_notice
xzy.com Intel::DOMAIN from http://mirror1.malwaredomains.com/ via intel.cyber.intelligence.domain F
```

In the effort to stop malicious activity, it is important to stay up to date with the latest blacklists, so newly infected computers can be stopped as soon as possible. For this reason, SecBI collects blacklist information from threat intelligence sources and uses this as enticement data to detect computers trying to connect to known malicious servers.

2.3 Threat Intelligence Standardization

Threat intelligence is a new field, and there is no universal standardization in sharing threat intelligence. Threat-sharing organizations face the challenge of creating the ability to standardize cyber threat information, but not lose the human judgment and control involved in sharing. Organizations often have a need to exchange information in a way that is both human readable as well as machine automated. Some questions to consider are: What and how much content should be shared? What file format should threat intelligence be shared in? What are the accepted categories to organize threat intelligence? Through what protocol should the information be shared? (The MITRE Corporation, 2012).

In recent years the industry has seen some standardizations arise and sometimes the process becomes a cycle of creating more standardizations to standardize the existing standardizations.

The Financial Services-Information Sharing and Analysis Center (FS-ISAC) founded in 1999, is a group of 7,000 organizations, including many banks, who are helping to shape cyber security standardization. Chief security officer Jason Witty at U.S Bank says, "Generally speaking, there's a willingness to share [cyberthreat information among financial institutions]" (Crosman, 2016). In the article, "A Glimmer of Hope for Cyberthreat Data Sharing", Crosman argues that there is hope for cyber security standardization in the future, explaining the recent addition of large corporations to join FS-ISAC. The list includes Goldman Sachs, Bank of New York Mellon, Citigroup, Morgan Stanley, State Street and Wells Fargo (Crosman, 2016).

Threat intelligence software solutions are created as frameworks or platforms that organizations may implement into their infrastructure. The threat intelligence software is integrated into production by communicating through an API. To clarify the definitions, a software framework is defined as:

> A concrete or conceptual platform where common code with generic functionality can be selectively specialized or overridden by developers or users. Frameworks take the form of libraries, where a well-defined application program interface (API) is reusable anywhere within the software under development (Techopedia, 2017)

And a platform is defined as:

> A group of technologies that are used as a base upon which other applications, processes or technologies are developed (Techopedia, 2017).

2.3.1 Communication Standardization

Currently, one of the most widely used standardizations in communication between threat intelligence sources is the *Trusted Automated Exchange of Indicator Information (TAXII), Cyber Observable Expression (CybOX) and Structured Threat Information Expression (STIX)*. In summary, STIX is a language that can use CybOX words, and the communication is possible with TAXII. STIX characterizes what is being told, while TAXII defines how the STIX language is shared (Impe, 2015). The FS-ISAC is implementing STIX and TAXII as standard formats and this will likely cause others to follow their lead (Crosman, 2016).

Most threat intelligence systems have the option to use STIX and TAXII communication standardizations. Since they are currently one of the leading open source standards, SecBI has favored implementing them.

2.3.2 Framework and Platform Standardization

Similar to threat intelligence communication standards, there is no industry standard for threat intelligence frameworks, each has advantages and disadvantages.

Different companies may have different needs and requirements and value different aspects of a threat intelligence framework. There are many threat intelligence options available on the market. Some frameworks are proprietary and sold as a service and require a subscription fee. Others are open source and are community driven. Furthermore, some frameworks are run in the cloud and data is kept in a centralized database. Other frameworks are run locally as a separate instance for each organization.

In the next section 2.4 I explain what I think the properties of a successful threat intelligence framework are, based on my research. They will serve as a metric and ranking system in selecting the best threat intelligence framework for SecBI.

2.4 Threat Intelligence Frameworks and Platforms

The process to select a threat intelligence framework requires researching and experimenting with the many threat intelligence options available. Once I compiled a list of the options, the section could be narrowed down until only the best options remain. The most popular and latest threat intelligence frameworks that I found are shown below.

- FaceBook Threat Exchange
- Microsoft Interflow
- MISP: Malware Information Sharing Platform (MISP, 2017)
- CRITs: Collaborative Research Into Threats (MITRE Corporation, 2017)
- Soltra: Cyber Threat Intelligence Data
- CIF: Collective Intelligence Framework (Collective Intelligence Framework, 2017)
- Blueliv: Targeted cyber threat intelligence and analysis
- AlienVault: Unified Security Management Threat Intelligence
- IBM X-Force Exchange

In order to compare threat intelligence frameworks, it is important to create some metrics and rankings. Not all metrics have the same weight, as some are more important than others (determined by the organizational needs). Below is a list of threat intelligence measures and the potential usefulness. This list was inspired by the table shown in Appendix B.

1. Can the intelligence database be downloaded so the system can run and make API calls offline?
2. Size of the database (number of entries)?
3. Good quality entries in the database?
4. Updated frequency?
5. Frequency of false positives?
6. Popularity (Do potential partners use it?)

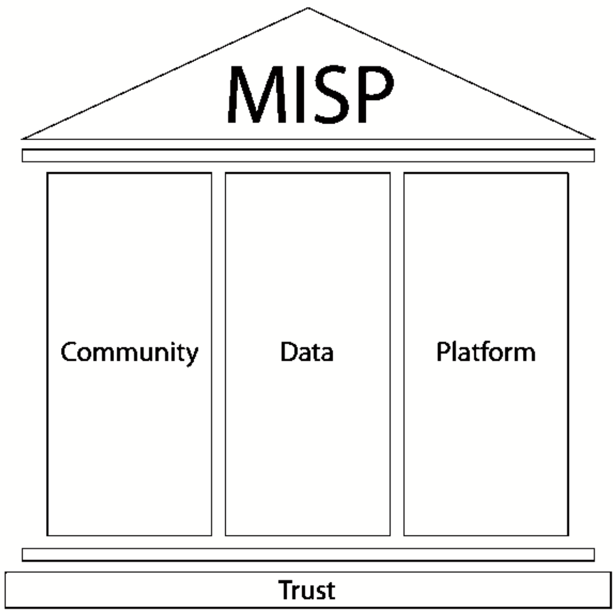

FIGURE 2.2: MISP Framework (NCI Agency, 2016)

7. Exclusivity (Are the database entries unique?)
8. Open source and free?
9. Threat Intelligence format (STIX/TAXI)?
10. Licensing, is modification of software acceptable?

Given this information, the implementation describes what steps were taken to decide upon the best threat intelligence options for SecBI.

2.5 Threat Intelligence Prioritization

There are some considerations to be made before using threat intelligence. One of the greatest difficulties in sharing threat intelligence is selecting the relevant information. Some information may be outdated, or unimportant information may dilute what is actually important. Once obtaining the information, an organization still has to be able to look at the information with enough context and intelligence such that they can take planned actions based upon it (**hacker5**).

2.5.1 Malware Information Sharing Platform

For reasons explained in the results chapter 4, Malware Information Sharing Platform (MISP) was selected as the best candidate and is used in the implementation. The overview of the MISP framework is shown in figure 2.2.

> "Malware Information Sharing Platform is a combination of a community of members, a knowledge base on malware, and a web-based platform." (NCI Agency, 2016)

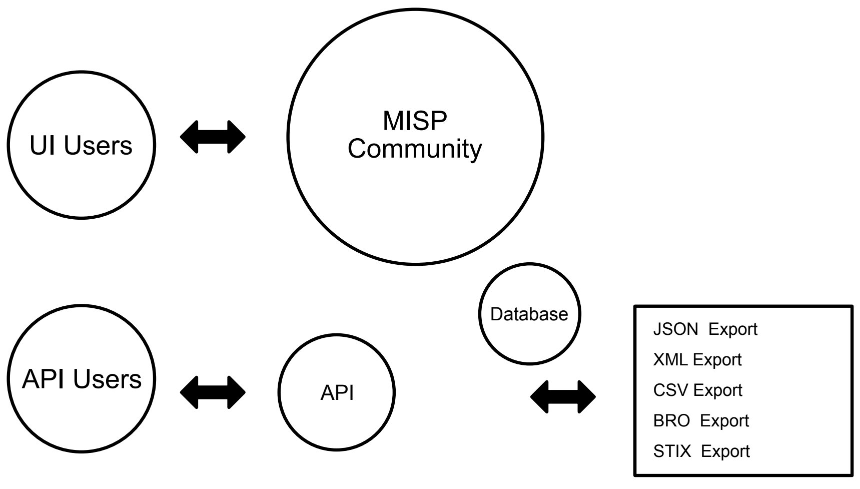

FIGURE 2.3: MISP Diagram (circl, 2017)

MISP is a framework to share malware characteristics with a trusted community. It provides the benefits expected by a threat intelligence framework such as described in the introduction. figure 2.3 shows how MISP allows both automated systems and manual users to access the framework. The label circle labeled MISP represents the trusted community that other users and automated systems may access. Once information is uploaded, it may be exported in many formats such as CSV or JSON as shown in the figure.

MISP has gained a lot of traction over the years since the project's start in 2011. Today over 1,000 organizations use the MISP framework worldwide. Communities form when organizations agree to share threat intelligence with each other and connect their databases. There are several large communities within the MISP framework, up to 500 organizations connected (MISP, 2017).

Some key characteristics of MISP is the ability to share a wide variety of malware characteristics in an organized and standard way, integration with the NATO Incident Response Technical Centre, increased detection speed of incidents, and discovery of malware that is unknown by anti-virus systems (through the sharing within a MISP community). MISP is now led by a team of volunteers and is free and open source.

A central theme to MISP is trusted communities. MISP has the option to work stand-alone without a community as a database of threat intelligence - but the user would miss out on the advantages of the support of a community. Communities in MISP may be public or privately organized. Some communities may require specific rules to join. Communities may team up or work in an isolated island mode. Each organization decides on the type of community they would like to join depending on their needs. One of the key properties of MISP is that it is built on trust as shown in figure 2.2 - so a trusted community is essential. A user in the community has the potential to perform malicious actions toward others in the community by sharing false information. SecBI has currently not joined a MISP community. For SecBI, the choice is heavily influenced by the decision of their client - some organizations do not want to share data.

Chapter 3

Implementation

3.1 Comparing Threat Intelligence

The implementation is done in several sections. Before I developed actual code, I determined the threat intelligence sources that the threat intelligence system would support at first. However, the software is designed so that, as SecBI grows, more threat intelligence sources may be added without modifying the existing threat intelligence framework code. This section describes the process I used to compare threat intelligence sources as the first step. Ultimately, the MISP framework was selected, for reasons described in the results.

In the larger picture, the measures of threat intelligence are:

- *How much will this threat intelligence system reduce the time needed to detect a problem?*
- *How many more problems will this threat intelligence system allow me to detect that I would otherwise miss?*

To break down the problem I worked with SecBI and assessed their current infrastructure to determine what specifically was important to the vision of the company. After gathering this information I created table 3.1 which shows various threat intelligence measures compared with the usefulness of the measure based on the current needs of SecBI. It is important to note that not all of these measures are equal in weight. Furthermore, there is a complexity field, as some measures are difficult to determine. These factors should be considered in to a decision regarding threat intelligence.

In table 3.1 the most important measure was the license of the software. As mentioned in the introduction, this is because SecBI is currently using a simple threat intelligence aggregation platform as their sole threat intelligence enrichment source and they need to expand their search without paying for a license. The second most important measure is the ability to download the database of threat intelligence sources locally without querying a remote server. This is because lookups happen very frequently while the distributed big data algorithm is running and remote calls will slow it down too much. At the same importance level, the framework should be open source and free. SecBI would like the option to modify the software to their needs if necessary and they want to avoid paying a subscription if possible.

On the next level of importance, SecBI wants high quality sources in their threat intelligence selection (meaning less false positives) and they want to comply with the current leading communication standardization format. This is to make the system future proof.

TABLE 3.1: SecBI Threat Intelligence Comparison

Measure	Usefulness (1-10)	Complexity
Can the intelligence database be downloaded so the system can run and make API calls offline?	9 SecBI needs this feature	Easy, do research
Size the database (number of entries)?	5	Unknown, depends on community joined
Good quality entries in the database?	8 Important	Unknown, depends on community joined
Updated frequency?	4 SecBI currently uses its own data sources	
Frequency of false positives?	8 Goes with good quality	
Popularity (Do potential partners use it?)	5 May be important in the future	Ideally the threat intelligence should be pluggable/replaceable
Exclusivity (Are the database entries unique?)	3 Currently not a priority	
Open source and free?	9 No budget for paid service	SecBI is looking for free services
Threat Intelligence format (STIX/TAXII)?	8 Want to use an industry standard data format	SecBI wants to ensure maximum compatibility with partners
License	10 Must comply with SecBI internal production agreements	Must have proper license to be deployed by SecBI

3.2 Software Design

3.2.1 Overview

The general overview of the software is shown in figure 3.1. The SecBI infrastructure connects with the system at the top of the diagram, through the Java API or optionally a REST API. The SecBI infrastructure only communicates at the high level with the *Threat Intelligence Controller*. The controller will interpret the commands and use the various *Threat Intelligence Services* currently plugged into it. In this way, decisions about which services to use and how to use them are abstracted by the controller.

The *Threat Intelligence Controller* handles the logic of the system, including the prioritization of which *Threat Intelligence Service* to use when the SecBI infrastructure requests threat intelligence information. Currently, SecBI exclusively uses threat intelligence to collect information about known malware IP addresses in the form of blacklists. SecBI infrastructure may find new malware IP addresses but the current SecBI system does not share the information. The system I created has the ability to share. Threat intelligence information is requested for a specific domain, and the controller will determine which service to use in order to handle the request based on the priority. Furthermore, the controller can be instructed to get threat intelligence information of a given domain from all available services instead of only one.

The *Threat Intelligence Controller* is initialized based on a universal settings file in the SecBI infrastructure. The SecBI infrastructure can customize how the controller behaves, what services it prioritizes, and where it saves and looks for information.

Services are either local or remote, based on whether they gather information from a *local* file store or if they get information from a *remote* server. The remote services are an extension of the local services meaning they have the same base functionality, but remote services have the additional ability to publish newly found threat intelligence back to the remote server and import a remote database to the local machine. The controller will decide which service to use. Generally, it is best practice to query information from local services but export and share information with remote services. The SecBI infrastructure has full control over this behavior through the configuration file.

The shared and common functions between services are located in the *Threat Intelligence Utilities* class to the right in figure 3.1. By following this pattern, code is not repeated and services can simply use this class to perform tasks such as manipulating the file system, parsing a formatted file, and searching for a specific domain. A common file format that SecBI uses as blacklists are comma-separated values (CSV) and bro files. To parse these a *CSV Parser* class is used as shown in figure 3.1.

Documentation and usage examples are shown in Appendix A.

3.2.2 Strategy Design Pattern

Figure 3.1 depicts how the threat intelligence system implements the strategy software design pattern. The strategy pattern is used to:

- Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from the clients that use it (Sourcemaking, 2017).
- Capture the abstraction in an interface, bury implementation details in derived classes (Sourcemaking, 2017).

In the threat intelligence system I created, the services are interchangeable. They all implement the same *Handler* interface and extend the same *Abstract Handler* class.

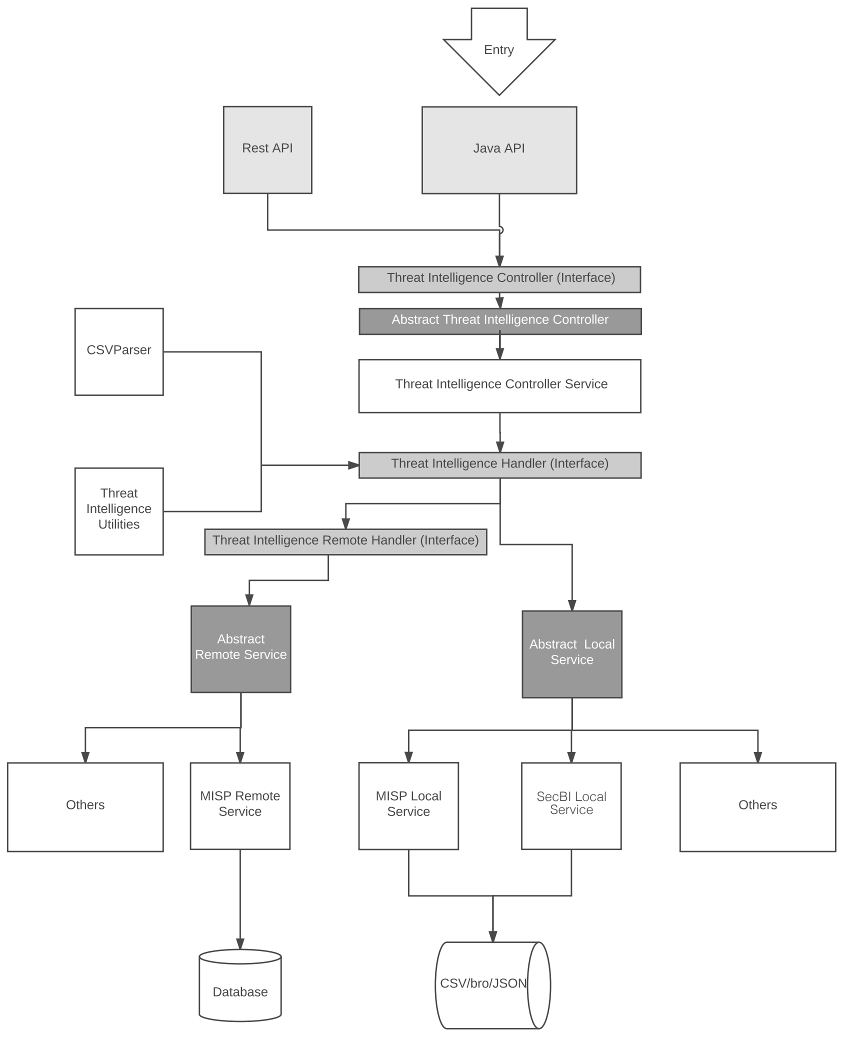
Entry
Rest API
Java API
Threat Intelligence Controller (Interface)
Abstract Threat Intelligence Controller
CSVParser
Threat Intelligence Controller Service
Threat Intelligence Handler (Interface)
Threat Intelligence Utilities
Threat Intelligence Remote Handler (Interface)
Abstract Remote Service
Abstract Local Service
Others
MISP Remote Service
MISP Local Service
SecBI Local Service
Others
Database
CSV/bro/JSON

FIGURE 3.1: Threat Intelligence System Diagram

The *Handler* name is used to describe the abstraction and format that a *Service* will implement. This allows the controller to call the following methods from each service and expect a standard result even though the actual logic used to get the result is different and specific to each threat intelligence service.

```
void init()
String getHostMalwareData(String domain)
```

The init method will initialize the service with the configuration passed into the constructor. The getHostMalwareData method will search the specific data of the service for the given domain name.

The *Remote Handler* interface includes the methods from the *Handler* above, but also includes the following methods.

```
String getEvent(String domain)
void exportEvents()
```

The getEvent method is used to retrieve an event string containing malware data for a given domain name. This is similar to getHostMalwareData, however it is a remote action and the concept of an event is unique to remote services. An event may contain several pieces of information instead of just one like getHostMalwareData.

The strategy design allows for a variable amount of services to be attached to the controller and still maintain uniform functionality across them. The controller understands that each service has these abilities but the details of how they work are abstracted by the services themselves.

3.2.3 Interface Segregation Principle

The Interface Segregation Principle is used in the system and is defined as follows:

> The Interface Segregation Principle states that clients should not be forced to implement interfaces they don't use. Instead of one large interface many small interfaces are preferred based on groups of methods, each one serving one submodule (oodesign, 2017b).

This principle is applied to better organize the software design. Instead of having methods in the interfaces that are not actually used by the class implementing them, there are several interfaces that serve a specific purpose. This means that a class does not have to implement unwanted methods and unsupported operation exceptions can be avoided.

There is both a remote and local handler interface to separate functionality between local and remote communications. Each handler interface serves a specific enough function that there is no overlap. Specific interfaces are selected for each service class. Additionally, some threat intelligence services have special features unique to their functionality. For example, the MISP service uses the concept of identifying events with an identification number, the number is also used to lookup MISP events. It would be poor design to require all services to implement this MISP specific functionality, so by using the Interface Segregation Principle the service is created in a separate interface.

3.2.4 Factory Pattern

The factory pattern is a common way to design object oriented software. It is defined as follows:

- Creates objects without exposing the instantiation logic to the client (oodesign, 2017a).
- Refers to the newly created object through a common interface (oodesign, 2017a).

The advantage of using the factory pattern is that new objects can be added without changing the code in the framework (oodesign, 2017a). In the threat intelligence system, the *Threat Intelligence Controller* needs to create *Threat Intelligence Services* for itself on initialization. The *Threat Intelligence Controller* does not initialize the services itself, but uses a *Threat Intelligence Factory* class instead. This separates to the logic of initialization, making organization clearer and making the service more easily maintainable in the future.

Without the *Threat Intelligence Factory*, the *Threat Intelligence Controller* will have to do all the initializing by itself. This creates a problem when a new *Threat Intelligence Service* is added to the system. The controller would have to be modified to handle the initialization of the new service. It is important to keep the controller as an abstraction service to maintain organization. Instead when a new service is added the *Threat Intelligence Factory* is updated to handle the additional service, but the *Threat Intelligence Controller* maintains the same code.

3.3 Software Features

The software I created is an abstraction between the many threat intelligence sources and the SecBI infrastructure. To implement the system I started by learning how SecBI queries information and what the common high level commands used to communicate with threat intelligence sources are. I discussed with the team how the current system could be improved and what new commands would be useful to the system. After gaining an understanding of the problem and current SecBi infrastructure, I made diagrams similar to figure 3.1 with the proposed solution.

The next stage was to develop the actual program. The threat intelligence framework I developed aimed to have three specific software goals, which are described in detail in the next sections:

- Modular
- Pluggable
- Prioritization

SecBI did not have a robust way of collecting threat intelligence prior to my project. The original threat intelligence solution was to poll a free database from a simple threat intelligence aggregation platform. The system I created has support for queries to the original platform and is also expandable to work with other threat intelligence sources. When a new threat intelligence source is added a new *Threat Intelligence Service* is created to interface with it. This provides standardization as to which features of the threat intelligence source are expected to be available.

3.3.1 Modular System

The system was designed to be modular, with dynamic settings without code modification. The Strategy Design Pattern and the Factory Pattern allow this to happen. Settings are all configured from the *SecBI Configuration* object, loaded from the file system. When the controller is initialized configuration properties are set according to the key-value pair settings of the configuration. Furthermore, the controller passes this configuration object to the factory class which will create services according to the configuration setup. Services also are given a copy of the configuration during initialization.

The *SecBI Configuration* is based on a property file stored in the file system and is shared between the infrastructure of SecBI. Instead of modifying code to change settings, a user will simply have to modify this file and the new settings will be used. I created some support for handling user error, so even if the user inputs incorrect formatting to the configuration file, the system will still accept it to a degree.

3.3.2 Pluggable System

The system was designed so that components may be swapped out - so threat intelligence sources may be exchanged as needed while maintaining the same infrastructure. This design was implemented through the use of the Factory Pattern. When the controller is initialized there is no predetermined amount of threat intelligence sources that will be attached. A dynamic mapping of threat intelligence sources is created and the configuration file determines which services are created and added to the mapping. In this way, the infrastructure of the system is preserved while new threat intelligence sources are exchanged as needed.

3.3.3 Prioritization System

Threat intelligence prioritization is an important component of a successful system. As described in the background, threat intelligence can become useless and potentially harmful if misleading or incorrect information is provided. For this reason having more threat intelligence is not always better - it may only dilute what is actually useful. It is important to sort threat intelligence by priority.

To implement the priority system, I built the configuration system such that the various threat intelligence sources should be listed in order of priority in the configuration file. When the *Threat Intelligence Controller* is initialized it will read the threat intelligence sources in order and maintain a list. When the controller is instructed to perform a task, such as *getHostMalwareData(domain)* it will first try the service with the highest priority. If that service fails to find the data, the controller will try the next best priority. Finally, if all service options are exhausted a null value will be returned.

3.4 Threat Intelligence Features

The threat intelligence system currently has three services, although more can be added without changing the framework as described in the software description. The services are:

- MISP Remote
- MISP Local
- Intelligence-Source Local

While all three services can be used to collect threat intelligence information, only the remote service can be used to publish information. This section will describe the abilities of each service in detail.

3.4.1 Collecting Information

The local services collect information directly from a local file stored in memory, while the remote series use a network connection to query a server.

Before a local service is able to operate, the local data file must be generated. This means that the *exportEvents()* method is usually run prior to local service setup. The controller will use a remote service to download the latest threat intelligence data to a local file. When a local service is initialized it will find the local file based on the path supplied in the configuration file.

The file that was downloaded using *exportEvents()* is read by the program. Before being processed by the system, the modified timestamp of the file is used to determine if it is actually new information. The local handlers may be reinitialized with new data at set time intervals to ensure they are up to date. But this does not necessarily mean the local data has changed. So the timestamp prevents unnecessary updates. Using the *CSV Reader* class it is parsed into strings of data taking advantage of prior knowledge of the file format. All of the local file contents will be added to a hash map in memory for quick lookups in key-value format. Using this method, when *getHostMalwareData(domain)* is called, a fast lookup is made on the table.

Remote services do not have a local file to read from. Most threat intelligence sources are stored online, either in a self hosted server or on a company cloud service. For a remote service to retrieve information a GET request must be made to the server. For achieve this, the AsyncHttpClient is used (AsyncHttpClient, 2017).

Taking advantage of the Java 8 CompletableFuture class, a request is built and the address of the server is added to the request. Furthermore, some security headers are attached, including a secret API key. The key is loaded from a store on the file system and the path to this store is provided in the configuration file. The AsyncHttpClient makes a CompletableFuture *promise* to push information to the server. The *promise* is free to complete after the function ends, preventing slowdowns when larger amounts of data are transferred. When the non-blocking request is complete, the data will be returned to the controller from the remote service.

3.4.2 Publishing Information

To publish information a remote service must be used to post data to the server. Similar to *Collecting Information* a POST or PUT request is built with the secret API key. Essentially, when a new piece of threat intelligence is discovered it is packaged by the service and uploaded to the remote server for future use. Optionally, the intelligence may be shared with others within the organization's trusted community. Currently the only publishing service integrated in the threat intelligence system is MISP (MISP, 2017).

To construct the data to be published, the JSON format is used. The JSONObject from org.json is used to construct a Java object representation of JSON data (JSON-Java, 2017). Using the *put* method from the JSON object, various attributes are attached to the object. An example of JSON construction for a MISP attribute is shown below.

```
JSONObject attribute = new JSONObject();
attribute.put("type", type);
attribute.put("category", category);
attribute.put("to_ids", toIds);
attribute.put("distribution", distribution);
```

```
    attribute.put("comment", comment);
    attribute.put("value", value);
```

3.4.3 MISP Implementation

SecBI is currently experimenting with MISP (not used in production) and an instance is run on a local machine. I set this up using a Docker instance of MISP (Docker, 2017). This is a simple way to run MISP without a dedicated machine. The SecBI threat intelligence system could communicate with the MISP instance through the network. It is important to understand how MISP operates to understand how it is implemented in the SecBI infrastructure. MISP is based on an Apache server with a RESTful interface. The threat intelligence remote services makes HTTP calls to collect and publish data.

MISP uses the concept of events and attributes. According to the MISP Request for Comments (RFC) the defenition of an event is:

> An event is a simple meta structure scheme where attributes and meta-data are embedded to compose a coherent set of indicators. An event can be composed from an incident, a security analysis report or a specific threat actor analysis. The meaning of an event only depends of the information embedded in the event (Iklody, 2016).

An event has a timestamp and a threat level ranking to give context about the event, however most information about an event comes from attributes. Attributes, which make up the information of an event, describe detailed information about a specific event.

> Attributes are used to describe the indicators and contextual data of an event. The main information contained in an attribute is made up of a category-type-value triplet, where the category and type give meaning and context to the value. Through the various category-type combinations a wide range of information can be conveyed (Iklody, 2016)

For SecBI, the attribute category that is used to store malicious addresses is called, *Network Activity*. The type is *Domain*, because it represents a domain address. The value of the attribute is the actual malicious address. Using this format, MISP can take the many attributes of events and compile a list of the results in various formats. This is what happens when *exportEvents()* is invoked by the controller.

Chapter 4

Results and Discussion

4.1 Threat Intelligence Comparison

To compare the threat intelligence candidates, I created a diagram shown in 4.1. After visualizing the threat intelligence options I was able to better explain the current options of threat intelligence to SecBI. It was easier to see the advantages and missing features in each framework. As described in the implementation, using the table 3.1, where I created a system of ranking threat intelligence, the features that SecBI valued in a threat intelligence framework were recognizable in the diagram. The combination of the table, diagram, and background information helped to make more distinctions between frameworks.

4.1.1 Threat Intelligence Candidates

Based on my research and discussions with SecBI I decided that CRITs, MISP, and CIF are the best threat intelligence framework choices for SecBI. They all are free and open source programs and can be run in a private standalone instance on SecBI's or the clients of SecBI's machines. Furthermore, they all either have the *MIT* or *GNU v3* licenses, allowing for complete freedom to market and modify the software. They all have some type of optional social collaboration tool built in. Another important feature valued by SecBI is the ability to export data from the remote instance to a local machine for fast data lookup. The algorithms of SecBI need to get feedback from a framework quickly, and the network delay would make the system inefficient. All of three candidates have this ability, however SecBI currently uses the bro file format, and not all options have bro support. However, file format is not a major factor in choosing a framework because it is reasonable for SecBI to develop support for another format for a worthy threat intelligence framework.

4.1.2 Collective Intelligence Framework: CIF

CIF is a strong candidate because it is fast to setup and simple to use, so it can replace the currently implemented threat intelligence platform faster than other solutions (Collective Intelligence Framework, 2017). It can output using the bro format, which is the same format SecBi is using today. It stores data in Elasticsearch and integrates with Kibana which can be used for visualization. SecBI has experimented with Kibana visualizations previously, so integration with other SecBI services will work well.

Here is an example of the bro output used by CIF, the same format SecBI currently uses.

```
cif --otype ipv4 --feed --confidence 85 --format bro
   --limit 5
```

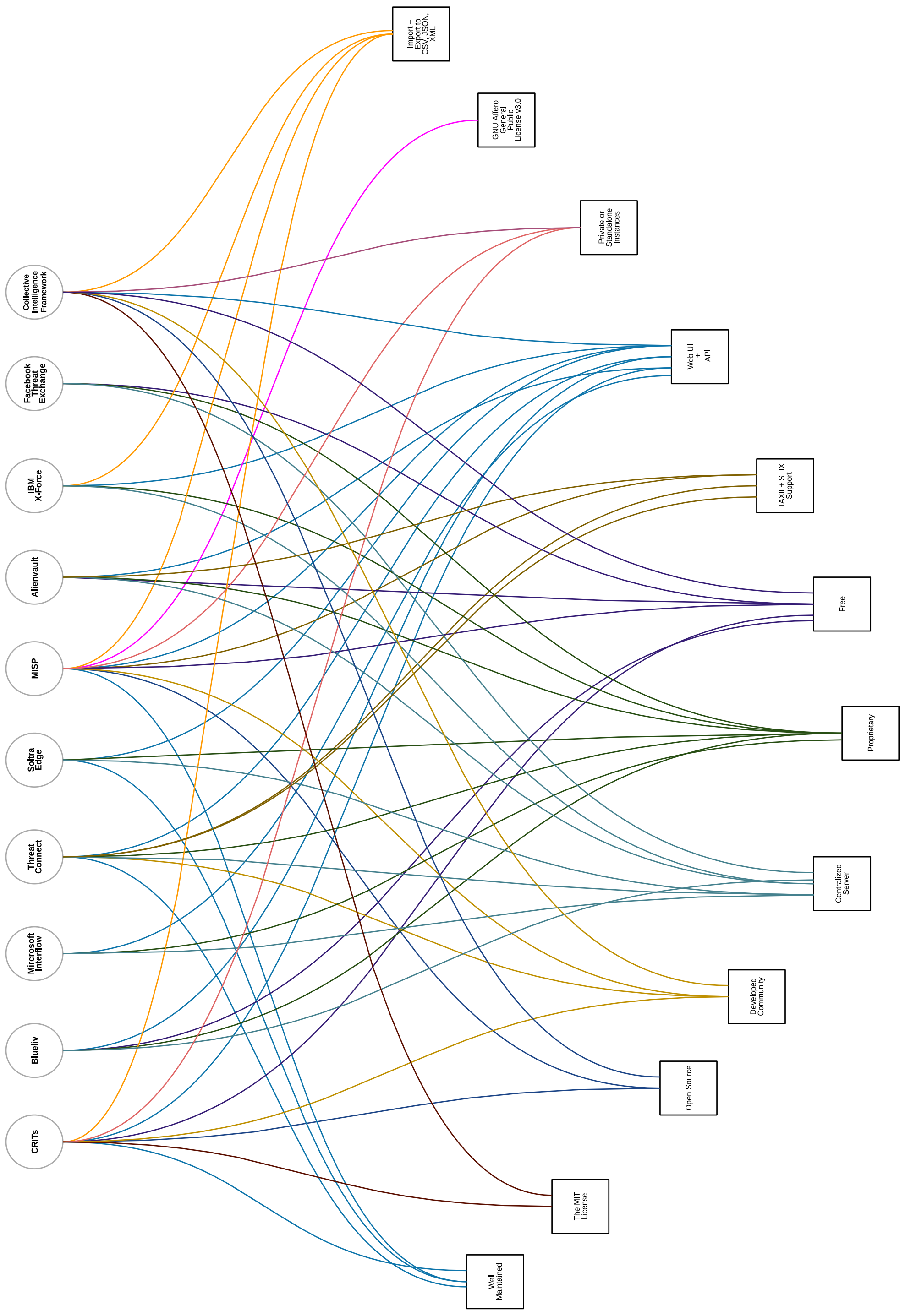

FIGURE 4.1: Threat Intelligence Comparison Chart

```
#fields indicator   indicator_type  meta.desc   meta.
   cif_confidence meta.source
92.50.31.66     Intel::ADDR exploit 95  spamhaus.org
210.4.72.138    Intel::ADDR exploit 95  spamhaus.org
61.150.89.67    Intel::ADDR spam    95  spamhaus.org
68.180.32.194   Intel::ADDR exploit 95  spamhaus.org
221.206.72.203  Intel::ADDR spam    95  spamhaus.org
```

The downsides to CIF is the smaller open source community (compared with the other candidates): 12 contributors and 1000 commits. Additionally, the repository is quiet - updates are relatively infrequent. It seems to have less emphasis on community sharing compared to MISP. There are not as many public groups or communities to join. Finally, CIF recommends a 8 core 16GB server or better in order to run and does not have an official docker build (it likely can run in a VM for testing). SecBI is a small startup and may not want to invest in a machine with this hardware at this time.

4.1.3 Malware Information Sharing Platform and Threat Sharing: MISP

MISP is a strong candidate because it is seemingly well adopted and gaining a lot of traction lately, over 1000 corporations use it today (MISP, 2017). It outputs several files types (CSV, JSON, STIX) but does not have native bro output support. MISP has a large community of 45 contributors and 4000+ commits. A central aspect of MISP is the access to communities and it seems to be simple to connect to a community to share threat information. MISP is very well documented with a 155 page e-book and an additional RFC (Iklody, 2016). MISP has a modern Angular web user interface and RESTful API and uses the LAMP stack. MySQL is used as the datastore. MISP has two official docker builds.

The negatives of MISP is that it lacks native bro support, although there is an old repository project that will actually export MISP to bro. MISP seems to require slightly more overhead to setup and integrate into the SecBI infrastructure, but the large community and support makes this easier.

4.1.4 Collaborative Research Into Threats: CRITs

CRITs has the second largest community out of the three, at 35 contributors and 2000 commits (MITRE Corporation, 2017). It exports data to CSV, JSON, and STIX but not bro. Internally CRITs uses MongoDB and has native support to cluster the database for large scale operations. CRITs has a web UI with data visualizations and API to manage threat entries. Furthermore, MITRE heavily supports CRITs making it stable for long term use. A docker image for CRITs if maintained (but not official). The community is active, but not as large as MISP. The code is well documented.

CRITs is geared more toward collaborative research into malware and threats and is less of a threat intelligence platform replacement. It is a very good solution to store information derived from threat reports. For this reason I think it would be a good addition for data enrichment in the future, but it is not as suitable as a framework to track malicious domains in the way that the currently implemented threat intelligence platform is being used today. CRITs offers a global collaborative network, but compared with MISP it is smaller.

4.1.5 Conclusion

I think MISP is currently the best option because it has the largest community in both software development and threat intelligence sharing collaboration. Also, it has a well developed REST API that can be used for automating tasks which will be essential when dealing with large qualities of threat intelligence information. Since the community is large, many side projects are available to extend the MISP framework such as a python library, bro integration, and *The Hive* project (Hive, 2017). For testing MISP can easily be run in a docker image without the need for expensive machines. Finally, MISP has the most documentation out of the selection and is growing in scale.

4.2 System Performance

After developing a working threat intelligence system I ran integration experiments. The new threat intelligence control I developed performed as expected. To test the system, I first created a new instance of the controller, and initialized it with the global SecBI configuration file. Before looking up malware entries, data must be imported from the SecBI MISP server instance I configured beforehand. The next step was to instruct the controller to use a remote MISP service to download the latest MISP database to the local machine. This stage worked as designed.

Once the data was on the local machine, I was able to instruct the controller to use a local MISP service to lookup threat intelligence data, in the same manner as the previously implemented threat intelligence platform. Finally, although SecBI does not use this feature, I was able to push new threat intelligence to the MISP server using the remote MISP service. I confirmed this worked by using the MISP web user interface to confirm data was successfully uploaded to the MISP server. All of these stages were broken into small individual unit tests, as described in the next section.

4.2.1 Unit Testing Results

In addition to the integration experiments, I developed unit tests for the classes shown in figure 3.1. Unit tests will help ensure the system works as intended and if updates are made, the tests may be run again to maintain a standard of performance. SecBI uses automated testing and this will integrate well with their current system. The following tests were created.

- TIControllerServiceTest
- TIHandlerTest
- MispLocalHandlerTest
- MispRemoteHandlerTest

When a test is run I created a testing platform where a temporary file system is loaded into memory where fake threat intelligence data is stored for testing and later removed automatically by the testing framework. I utilized existing custom SecBI testing tools which allowed for temporary SecBI configuration files to be loaded into the test system.

Overall, I measured a testing coverage of 70% of the system's code. The tests were designed with the goal of trying unusual cases, such as when a null value domain is requested from the services, or if the services try to read from an empty or nonexistent file.

4.3 Threat Intelligence Challenges

Threat intelligence faces various known challenges and after implementing an actual solution myself I gained better insight into the problems. Some challenges of threat intelligence are listed below taken from the research by Scott Jasper (Jasper, 2017).

1. The risk of disclosure of protective or detective capabilities and of sensitive information (to the organization sharing the information)
2. The classification of information which might be difficult to actually use in a threat intelligence system and expensive to request.
3. The trust in the use of the information, possibly time consuming to create and maintain.
4. The ability to consume the information through the infrastructure necessary to access external sources and to incorporate the information into the process for actual decisions.
5. The challenge of establishing interoperability for the secure, automated exchange of data among organizations, repositories, and tools.
6. The challenge of preserving privacy of both individuals and organizations that may participate in sharing communities, but want their contributions to remain anonymous.

The first challenge is the reason why SecBI does not share threat intelligence by default. The customer must make the decision to share because of this challenge. An example situation where this would be problematic is if the organization was compromised, the threat intelligence it shared may indicate a security breach. The organization may not want to disclose this information.

The second challenge does not apply to SecBI because currently the company is only utilizing one type of threat intelligence categorization (malicious addresses). However this may change in the future. Also, the third challenge is not currently something SecBI is considering because it is not part of a threat intelligence community.

SecBI is consuming the threat intelligence information and using it in actual decisions, but this is easier when dealing with just one type of threat intelligence category. Perhaps this will become more problematic as the company grows.

The problem of interoperability for the fifth challenge is partly solved by the framework I created. Having a robust selection of threat intelligence framework options allows for more flexibility in sharing and receiving threat intelligence data. Interoperability is something that the MISP framework aims to solve, along with standard communication formats such as STIX as described in the background.

The final challenge relates to filtering metadata of shared threat intelligence. This seems difficult because it builds more trust if threat intelligence can be linked back to a credible source, when deciding if it should be used. The threat intelligence data that SecBI shares does contain some metadata and it could be removed. By doing this SecBI clients may be more interested in sharing threat intelligence data. However, challenge number one would still exist, as even data without metadata may be linked back to the source using advanced methods.

There is an active effort to remedy some of these challenges, and the National Institute of Standards and Technology is making legal and technical initiatives to assist in this effort (National Institute of Standards and Technology, 2016).

Chapter 5

Conclusion

5.1 Future Goals

SecBI currently does not share threat intelligence with a community, it consumes public threat intelligence information. A major reason for this is that clients of SecBI may or may not want their information shared. This is a well-known problem relating to threat intelligence - how to make the intelligence anonymous. Even if identifiable information is stripped, it may be possible to guess where the source came from using meta data.

Organizations want to remain anonymous for many reasons, for example to maintain a good reputation. When a company falls victim to a cyber attack, their credibility may decrease or they may want to investigate in private, so sometimes they hide it from the public. An example of this was explained in a 2016 Guardian article, revealing Yahoo's choice to hide the security breach where, "data from more than 1bn user accounts was compromised in August 2013, making it the largest such breach in history" (Thielman, 2016). It is a difficult balance to help other organizations stay secure while maintaining a secure reputation. The ideal solution would be to allow for both.

If a partner of SecBI would like to participate in the sharing of threat intelligence, it is already supported in the new threat intelligence system I created. A future goal for SecBI is to incorporate the option to share threat intelligence in the SecBI infrastructure (separate from the threat intelligence system I created). In order for this to happen SecBI will have to perform more research about which threat intelligence community to join and formalize agreements with clients to have their data shared.

Another goal is to share more types of threat intelligence besides only malicious addresses. There is much more threat intelligence information available, but the challenge of how to use the information becomes a larger factor with more broad data. An important goal is to add support for other types of threat intelligence data, perhaps one at a time, into the SecBI infrastructure.

Threat intelligence prioritization is a difficult problem because it is not always clear which source or data will actually be useful. To improve the system a stronger prioritization algorithm could be implemented. Threat prioritization could turn into another project by itself. The Carnegie Mellon University article, Cyber Threat Prioritization, explains some of the process required. One approach discussed is to rank threats for prioritization following the pattern below (Carnegie Mellon University, 2016).

```
Threat = Likelihood + Impact + Risk
Likelihood = Capability + Intent
Impact = Operations + Strategic Interests
Risk = People + Cyber Footprint
```

5.2 Summary

From my research, the future looks promising for threat intelligence. The Ponemon Institute research indicates that 39% of cyber attacks could be prevented by threat intelligence sharing (Ponemon Institute, 2016). A cyber defense system is effective not only because it can detect intrusions, but also because it can perform in a timely manner. Therefore it is in the best interest of an organization to share threat intelligence so information can be quickly passed to others to stop a newly spreading security threat.

The threat intelligence system I created for SecBI is a promising start to a feature that might eventually become a very valuable component to the services SecBI offers. I was able to successfully research threat intelligence frameworks, design a method of ranking, and create a working threat intelligence system that is *modular, pluggable, and prioritized*. Furthermore, the system is highly customizable and designed to be seamlessly upgraded, surpassing the original goals of the project. SecBI now has the basic tool to grow their threat intelligence program and keep up to date with defense of the future.

Appendix A

Documentation

A.1 Threat Intelligence API Documentation

A.1.1 Overview

This threat intelligence module is used to collect and publish threat intelligence information from multiple sources for data enrichment. You may use multiple threat intelligence sources as input. It has a modular design so new threat intelligence sources may be added to the system without changing the framework code.

A.1.2 Setup

1. Setup spark-server-application.properties. This is the configuration the *Threat Intelligence Controller* reads from. Priority: In the .properties file, you configure the priority of threat intelligence, so some sources will be used before others. The priority is determined by the order of the threat intelligence sources (highest priority first). In the example below the *Misp-Local* handler will attempt to search for the domain before the other handlers.

   ```
   ti.priority.list = Misp-Local Misp-Remote Intelligence-
       Source-Other
   ti.misp.endpoint = http://172.17.0.1:443
   ti.export.path = src/main/resources/
   ti.secret.path = src/main/java/com/secbi/ti/KEYS.json
   ti.misp.datafile = misp-data.csv
   ti.intelligence-source-other.datafile = master-public.bro.
       dat
   ```

2. Create files to store threat intelligence data from the sources. For example, MISP requires a .csv file and other threat intelligence sources require a .bro file. The file names should match with the names entered in the configuration file.

3. The files should be moved to the export path specified in the configuration file.

4. Create the secret file KEYS.json, in the secret path specified in the configuration file. Add API keys of sources used in JSON format. An example is shown below.

   ```
   {
     "MISP_SECRET":"abcdefg"
   }
   ```

5. Setup MISP Install a SSL certificate in the MISP apache server or disable SSL in apache configuration files. Otherwise the threat intelligence controller will not accept an invalid SSL certificate.

6. Run MISP in a Docker container or virtual machine

7. Setup other threat intelligence sources (if more have been added) following the specific guidelines for the source.

8. Import and initialize a *Threat Intelligence Controller* object in your Java code. The SecBI configuration must be used to initialize the controller. An example is shown below.

```
SecBIConfiguration configuration;
try {
   String configurationFile = SystemUtils.
       getConfigurationFileLocation();
   configuration = new SecBIFileConfiguration(
       configurationFile);
} catch (Exception e) {
   throw new UnhandledException(e);
}
TIControllerService controllerService = new
    TIControllerService(configuration);
```

9. Use the controller object to call methods to find threat intelligence for domains, or import and export threat intelligence events from MISP server. The methods available are listed in the next section.

A.1.3 Collecting Threat Intelligence

1. To collect threat intelligence data first call *exportEvents()*, using the *Threat Intelligence Controller*.

2. Next, either getHostMalwareData(domain) or getHostMalwareDataAllSources(domain) is called on the *Threat Intelligence Controller*.

3. Example of the getHostMalwareDataAllSources() output. In this case, all three sources were able to find data about a domain from their database, so the list contains three replies (all the same in this case).

```
[from http://mirror1.malwaredomains.com/files/domains.txt,
    attackpage,
from http://mirror1.malwaredomains.com/files/domains.txt,
    attackpage,
from http://mirror1.malwaredomains.com/files/domains.txt,
    attackpage]
```

A.1.4 Publishing Threat Intelligence

1. Currently the MISP framework is the only implemented threat intelligence source to share information. As explained in the implementation 3 section, MISP uses the concept of events and attributes.

2. An event must be created before attributes, as attributes are stored in events. Make an event by calling *setEvent(description)* using the *Threat Intelligence Controller*. Save the event identification returned. Use the description to give context about the event.

3. As new intelligence is discovered relating to this specific event, use *setEventAttribute(eventId, domain, malware)* to attach attributes to the event. The domain is the new address to warn others about, and the malware is the source string.

4. The new data is automatically pushed to the remote MISP server. It can now be exported using *exportEvents()*.

5. To publish the data to a MISP community (after joining a community), use the MISP user interface following the instructions of MISP (MISP, 2017).

6. To understand how MISP data is constructed and organized in communication with the server see section A.1.5 below.

A.1.5 MISP JSON Format

MISP communicates with JSON through the REST API. Below is an example of how the data format is organized. Notice the attributes are nested as a list inside of the event.

```
"Event": {
        "id": "4",
        "orgc_id": "1",
        "org_id": "1",
        "date": "2017-03-01",
        "threat_level_id": "2",
        "info": "master-public.bro.dat",
        "published": false,
        "uuid": "5888aa1a-4570-4745-b179-09deac110002",
        "attribute_count": "9",
        "analysis": "0",
        "timestamp": "1485787469",
        "distribution": "0",
        "proposal_email_lock": false,
        "locked": false,
        "publish_timestamp": "1485351451",
        "sharing_group_id": "0",
        "disable_correlation": false,
        "Org": {
            "id": "1",
            "name": "ORGNAME",
            "uuid": "58873b8a-cc4c-4e4d-905d-02ffac110002"
        },
        "Orgc": {
            "id": "1",
            "name": "ORGNAME",
            "uuid": "58873b8a-cc4c-4e4d-905d-02ffac110002"
        },
        "Attribute": [
            {
                "id": "20",
                "type": "domain",
                "category": "Network activity",
                "to_ids": true,
                "uuid": "5888aa1a-d6fc-4802-9d06-099fac110002",
                "event_id": "4",
                "distribution": "5",
                "timestamp": "1485351450",
                "comment": "from http:\/\/mirror1.malwaredomains.
                   com\/files\/domains.txt,attackpage via intel.
                   cyber.intelligence.domain",
                "sharing_group_id": "0",
                "deleted": false,
                "disable_correlation": false,
                "value": "squirtvidz.com",
                "SharingGroup": [

                ],
                "ShadowAttribute": [

                ]
            }
        ],
        "ShadowAttribute": [

        ],
        "RelatedEvent": [

        ],
        "Galaxy": [

        ]
    }
```

A.1.6 Threat Intelligence Interface

The *Threat Intelligence Controller* is the object the SecBI infrastructure communicates with. In the background, the controller will use the tools of the threat intelligence system to accomplish the requested task. Below are the methods exposed for use to interface with the program.

- void exportEvents()

```
Export events from endpoints to local files. Call
   before attempting to use getHostMalwareData().
```

- void exportEvents(ThreatIntelligenceType property)

```
Export events from endpoints to to local files from a
   specific threat intelligence source.
```

- java.lang.String getEvent(int eventId)

```
Get an event from remote server, given event
   identification number.
```

- java.lang.String getEvent(java.lang.String domain)

```
Get an event from remote server, given attribute domain
    name.
```

- java.lang.String getHostMalwareData(java.lang.String domain)

  ```
  Get malware data from handler, either remote or local
     based on priority.
  ```

- List<java.lang.String> getHostMalwareDataAllSources(java.lang.String domain)

  ```
  Get malware data from all handler available. Return a
     list of results.
  ```

- void init()

  ```
  Initialize the controller object.
  ```

- int setEvent(java.lang.String description)

  ```
  Share event with remote server. Returns the event
     identification number.
  ```

- void setEventAttribute(int eventId, java.lang.String domain, java.lang.String malware)

  ```
  Add an attribute to an event on remote server.
  ```

Appendix B

Diagrams

B.1 On Comparing Threat Intelligence Feeds

TABLE B.1: On Comparing Threat Intelligence Feeds (Chuvakin, 2017)

Proposed measure	Ease of getting it	Usefulness for the TI user
Number of entries	Easy – just count'em	Debatable – is more better? Or noisier?
Certainty of entry badness	Moderate– providers may not know due to algorithms, blind aggregation, lack of context, etc	Important, but not sufficient; even proven badness does not convey relevance to your environment
Type of entry badness	Varied – some feeds only cover certain types (like C&C or exfiltration)	Useful
Additional context data, extended schema fields, etc	Easy – look at the data to see what context is provided	Useful, but as an auxiliary
Update frequency	Easy – ask the vendor or check the data	Useful as long as it can be utilized at the same speed
Frequency of matches with your IT environment	Hard – requires operational usage of TI data	Yes, this is what makes the feed relevant and "actionable"
Frequency of matches in your environment NOT connected to an ongoing investigation	Hard – requires operational usage and an active IR team	The most useful
Frequency of "false positives"	Hard – requires operational usage	Useful in combination with the above
Popularity	Medium – need to ask the provider or peers about who else uses the data	Somewhat useful, but requires detailed interviews on usage with peers for maximum value
Durability – continued use for detection over time	Hard – requires operational usage for a long time	Yes, this is what makes the feed not just actionable, but reliably actionable
Preprocessing by the provider	Medium – need to ask the provider and trust the answer	Sort of – presumably feed provider processing makes it "better", but how?
Exclusivity	Hard – no trustworthy way of getting it	Yes, but needs to be combined with some relevance metrics – if the provider TI feed has unique threats that don't matter to you

Bibliography

AsyncHttpClient (2017). *Async Http Client*. URL: https://github.com/AsyncHttpClient/async-http-client.

Carnegie Mellon University (2016). "Implementation Framework – Cyber Threat Prioritization". In: URL: http://www.sei.cmu.edu/about/organization/etc/citp-cyber-threat-prioritization.cfm.

Chuvakin, Anton (2017). *Thoughts on On Comparing Threat Intelligence Feeds*. URL: http://blogs.gartner.com/anton-chuvakin/2014/01/07/on-comparing-threat-intelligence-feeds/.

circl (2017). *Malware Information Sharing Platform MISP - A Threat Sharing Platform*. URL: https://www.circl.lu/services/misp-malware-information-sharing-platform/.

Collective Intelligence Framework (2017). *Malware Information Sharing Platform*. Open Source CIF Project. URL: https://github.com/csirtgadgets/massive-octo-spice/.

Coopers, Pricewaterhouse (2015). *The Global State of Information Security® Survey 2017*. Survey. http://www.pwc.com/gx/en/issues/cyber-security/information-security-survey/assets/gsiss-report-cybersecurity-privacy-possibilities.pdf. Pricewaterhouse Coopers.

Crosman, Penny (2016). "A Glimmer of Hope for Cyberthreat Data Sharing". In: *American Banker*. URL: http://bi.galegroup.com/essentials/article/GALE%7CA460877707?u=mlin_c_worpoly&sid=summon&password=AnaxAndron&ugroup=outside.

CrowdStrike (2016). "2015 Global Threat Report". In: pp. 12–13.

Dalziel, Henry (2015). "How to Define and Build an Effective Cyber Threat Intelligence Capability". In: *Syngress Publishing*. URL: http://www.sei.cmu.edu/about/organization/etc/citp-cyber-threat-prioritization.cfm.

Dell SecureWorks (2014). "Underground Hacker Markets: White Paper". In: pp. 1–14.

Docker (2017). *Docker*. URL: https://www.docker.com/.

FIPS PUBS (2004). "Standards for Security Categorization of Federal Information and Information Systems". In: *Information Technology Laboratory* R43831. FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION. URL: http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf.

Fischer, Eric (2016). "Cybersecurity Issues and Challenges: In Brief". In: *Congressional Research Service* R43831. URL: https://fas.org/sgp/crs/misc/R43831.pdf.

Gerhards-Padilla, Thomas Barabosch1 , Andre Wichmann1 , Felix Leder , Elmar (2012). "Automatic Extraction of Domain Name Generation Algorithms from Current Malware". In: *Information Assurance and Cyber Defense*. URL: http://publica.fraunhofer.de/documents/N-237723.html.

Hacker5 (2012). "Threat Intelligence." In: *General OneFile*. URL: libraries.state.ma.us%2Flogin%3Fgwurl%3Dhttp%3A%2F%2Fgo.galegroup.com%2Fps%2Fi.do%3Fp%3DITOF%26sw%3Dw%26u%3Dmlin_c_worpoly%26v%3D2.1%26id%3DGALE%257CA295995900%26it%3Dr%26asid%3D96d3eb128790cf742f4d99ca0cbafbac.

Hive (2017). *The Hive: Scalable, Open Source and Free Security Incident Response Solutions*. URL: https://thehive-project.org/.

Iklody, A. (2016). "misp-rfc". In:

Impe, Koen Van (2015). *How STIX, TAXII and CybOX Can Help With Standardizing Threat Information*. URL: https://securityintelligence.com/how-stix-taxii-and-cybox-can-help-with-standardizing-threat-information/.

Jasper, Scott E. (2017). "U.S. Cyber Threat Intelligence Sharing Frameworks". In: *International Journal of Intelligence and CounterIntelligence* 30.1, pp. 53–65. DOI: 10.1080/08850607.2016.1230701. eprint: http://dx.doi.org/10.1080/08850607.2016.1230701. URL: http://dx.doi.org/10.1080/08850607.2016.1230701.

JSON-Java (2017). *JSON-java*. URL: https://github.com/stleary/JSON-java.

Kaspersky Lab (2015). "Cyber Attacks Mean Bill Bills for Business , Worldwide Survey of 5,500 Companies". In: p. 1.

Lingenheld, Michael (2017). *Malware Information Sharing Platform and Threat Sharing Framework*. URL: http://www.forbes.com/sites/michaellingenheld/2015/04/27/the-unfortunate-growth-sector-cybersecurity/.

Matarazzo Celeste M , Hansen, Rose (2014). *A dynamic introduction to cybersecurity*. Online. Lawrence Livermore Nation Labratory. URL: https://www.llnl.gov/news/dynamic-introduction-cybersecurity.

Merriam-Webster Online (2009). *Merriam-Webster Online Dictionary*. URL: http://www.merriam-webster.com.

MISP (2017). *Malware Information Sharing Platform*. Open Source MISP Project. URL: https://github.com/MISP/MISP.

MITRE Corporation (2017). *Collaborative Research Into Threats*. Open Source CRITS Project. URL: https://crits.github.io/.

National Institute of Standards and Technology (2016). "Guide to Cyber Threat Information Sharing (Draft)". In: pp. 8–9.

NCI Agency (2016). *Malware Information Sharing Platform*. URL: https://www.ncia.nato.int/Documents/Agency%20publications/Malware%20Information%20Sharing%20Platform%20(MISP).pdf.

oodesign (2017a). *Factory Pattern*. URL: http://www.oodesign.com/factory-pattern.html.

— (2017b). *Interface Segregation Principle (ISP)*. URL: http://www.oodesign.com/interface-segregation-principle.html.

Panda Security (2012). *Pandalabs annual Report- 2011 summary*.

Ponemon Institute (2016). "Flipping the Economics of Attacks". In:

Shakarian Paulo , Shakarian, Jana , Ruef Andrew (2013). "Introduction to Cyber-Warfare : A Multidisciplinary Approach". In: *ProQuest ebrary. Web.*

Sourcemaking (2017). *Strategy*. URL: https://sourcemaking.com/design_patterns/strategy.

Tech Pro Research (2015). *IT Security and Privacy: Concerns, initiatives and predictions*. URL: http://www.techproresearch.com/downloads/it-security-and-privacy-concerns-initiatives-and-predictions/.

Techopedia (2017). *Techopedia*. URL: https://www.techopedia.com/definition/14384/software-framework.

The MITRE Corporation (2012). “Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX^{TM})”. In:

— (2015). “2015 Data Breach Investigations Report”. In:

Thielman, Sam (2016). *Yahoo hack: 1bn accounts compromised by biggest data breach in history*. The Guardian. URL: https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached.

Voeller, John G (2014). “Cyber Security (1)”. In: *John Wiley Sons, Incorporated*.

Whalen, Sean (2015). *Open Secrets of the Defense Industry: Building Your Own Intelligence Program From the Ground Up*.