Heimdallur

Vulnerability detection through Internet Scanning

Thesis of 12 ECTS credits submitted to the School of Computer Science at Reykjavík University in partial fulfillment of the requirements for the degree of **Bachelor of Science in Computer Science**

Heiðar Karl Ragnarsson

Hlynur Óskar Guðmundsson

Tinna Þuríður Sigurðardóttir

December 15, 2017

Supervisor
Marcel Kyas, PhD

Examiner
Gylfi Þór Guðmundsson

Advisors
Theódór R. Gíslason
Ýmir Vigfússon, PhD

Acknowledgements

We would like to give special thanks to these individuals and organizations. Syndis was very generous to accommodate us by providing us with an office space, free lunches and caffeine. Opin Kerfi was nice enough to allow us to perform our scans when no other server provider was willing to host us. Ýmir Vigfússon for taking time out of his busy schedule to provide us with guidance on a weekly basis. Theódór Ragnar Gíslason for encouragement and being there when we needed to consult his expertise. Tryggvi Farestveit for server support and ideas. Bjarki Ágúst Guðmundsson for going through our technical report and giving us thorough feedback. James Elías Sigurðarson for allowing us to bounce ideas off of him regarding our threat score. Björn Símonarson for giving us a demonstration of Nessus, the vulnerability scanner.

[This page is intentionally left blank]

Contents

Abstract

As the number of Internet-connected devices grows, so does the likelihood of hackers successfully breaching these devices. In this paper, we present the threat scanner Heimdallur that scans networks to gain an understanding of the security landscape and be a port keeper, watching out for possible threats. In addition to enumerating networks, Heimdallur also gives the IP addresses a threat score, indicating their estimated vulnerability level. This threat score is a novel innovation and our primary contribution. The score serves as an attempt to predict the probability of the network being compromised. It is a cumulative score of weighted factors, such as the number of ports exposed to the Internet and the severity of known vulnerabilities connected to the services running behind those ports. Heimdallur was tested on all Icelandic networks that are publicly accessible but could be used to scan the entire Internet.

1 Introduction

There is an ever-growing rate of devices being connected to the Internet every year. Gartner, Inc. predicted that 11.2 billion connected things would be in use worldwide in 2018, up 75% from 2016, and will reach 20.4 billion by 2020 [1]. Each of these devices will expose a plethora of remotely accessible services. These services can be anything from a web server, file sharing server or even a live stream from a web camera used for monitoring babies [2]. And as the number of Internet-connected devices grows, so does the likelihood of hackers successfully breaching these devices as well as the networks that they are connected to.

In the current state of the cyber security landscape there is a general lack of insight into vulnerabilities, how widespread they are and how devastating they could be. It has taken far too long to identify and notify affected organizations when new cyber threats have surfaced in the past. An example of this is when WannaCry effectively took out multiple NHS hospitals in the UK [3]. In many cases the underlying flaw has been known for months, or even years, but when hackers combine a vulnerability with a powerful exploit it is possible to cause devastating impact. To counter this threat, it is important to have the information on new exploits available at a moment's notice. That way people can be notified when the vulnerability has been connected to an exploit, so that they can make the appropriate arrangements to protect their systems before it is too late.

In order to gain information about servers and what they are running we use two scanning methods. First, network scanning is the process of going through a network to check for alive hosts. And second, port scanning is the process of checking for open ports on these hosts. Different types of scanners can be used for different purposes. For example, Internet scanners cover the whole Internet, but do not give in-depth analysis for individual IP addresses. In contrast, vulnerability scanners cover smaller networks and give detailed information about specific devices. There is a lack of scanners with the coverage and speed of an Internet scanner and the depth of a vulnerability scanner—in fact such scanners do not exist.

The problem we want to solve is to gain an overview of the status of cyber security on a specific network, big or small. We want to assess the security of each IP address on the network and give each of them a score indicating the risk of them being compromised. Our hope is that this increased insight into the state of Internet security will raise awareness of governments, companies and individuals and encourage them to implement precautions. Additionally, we hope that this will lead to a shorter response time when a new threat surfaces and thus lead to increased cyber security in general.

This problem has been partially solved by projects like Shodan [4] and Censys [5] which scan the Internet and provide a search engine interface for its results. What these tools lack is the known vulnerability correlation and in the case of Shodan this has only been done for a few vulnerabilities. Other tools like Nessus [6], are vulnerability scanners that focus on giving detailed reports on security issues. Nessus uses methods that can be quite invasive and if default settings are used, it even can crash the hosts due to excessive traffic or exploits being tested. Thus, in order to be effective and give detailed results, the process needs to take place over a period of time, possibly days for one company. This is quite slow compared to Internet scanners, but the results are also much more detailed.

What we propose is the creation of a scanning infrastructure that is able to assess the security state of all IP addresses on a given network. We tested our infrastructure on the IPv4 addresses registered to Icelandic organizations [7]. This reduces the number of networks to scan which enables us to iterate faster on our scanning process. We call this infrastructure Heimdallur, after the Norse god who stood guard at Bifrost. Heimdallur is described in Gylfaginning:

> *He dwells in the place called Himinbjörg, hard by Bifröst: he is the warder of the gods, and sits there by heaven's end to guard the bridge from the Hill-Giants. He needs less sleep than a bird; he sees equally well night and day a hundred leagues from him, and hears how grass grows on the earth or wool on sheep, and everything that has a louder sound. He has that trumpet which is called Gjallar-Horn, and its blast is heard throughout all worlds [8].*

Much like the Norse god, our system tirelessly watches out for possible threats. Heimdallur scans networks for all accessible services and gathers information to enumerate each individual service on whichever of the ports it may be listening on. By conducting such a scan, we can enumerate all the services that are accessible on the Internet, often with specific details about each service. Using this information, we implement novel approaches whereby we correlate the scan information for each accessible service or server with known vulnerability sources such as CVE. CVE stands for Common Vulnerabilities and Exposures, often referred to as CVE Identifiers for publicly known security vulnerabilities [9]. Combining the scanning results with the CVE database we can assess the overall security of the entire network. This then allows for a much faster response time when a new threat is discovered. In addition to finding CVEs we assign a threat score to IP addresses.

The main novelty of our research is the threat score that we assign to the servers, as well as the integration of the scanning and vulnerability detection. This has not been done before in any of the research we have been able to find. There are scanning tools and there are tools that can search for vulnerabilities based on scanning results. However, none of these tools assign a threat score to the IP addresses and none of them combine all these different parts of the process we have described. New scanning tools can easily be integrated into Heimdallur's scanning process, since it is an iterative process.

In this paper, we will address the matter of how a correspondence between CVE identifiers and services can be created. We will introduce a threat score that uses this correspondence and explain the extent of coverage it provides. Our discussion begins with the background and motivation for this research. Some ethical concerns that affect our work will also be addressed. Then we will describe the design of Heimdallur, its implementation and how we evaluated the system. Finally, we will discuss some related work and possible future work.

2 Background

2.1 Motivation

Research on Internet security in Iceland is lacking. It is difficult to find information about the security of servers and whether they have been breached. This increases the difficulty of predicting whether a server is in immediate danger of being compromised or not. The main purpose of this research is to gain a better understanding of the reality of Internet security in Iceland, to see what the nation's main security concerns should be.

In the digital age, more and more software is being developed and exposed to the Internet, and as the number of lines of code increases, so does the number of flaws.

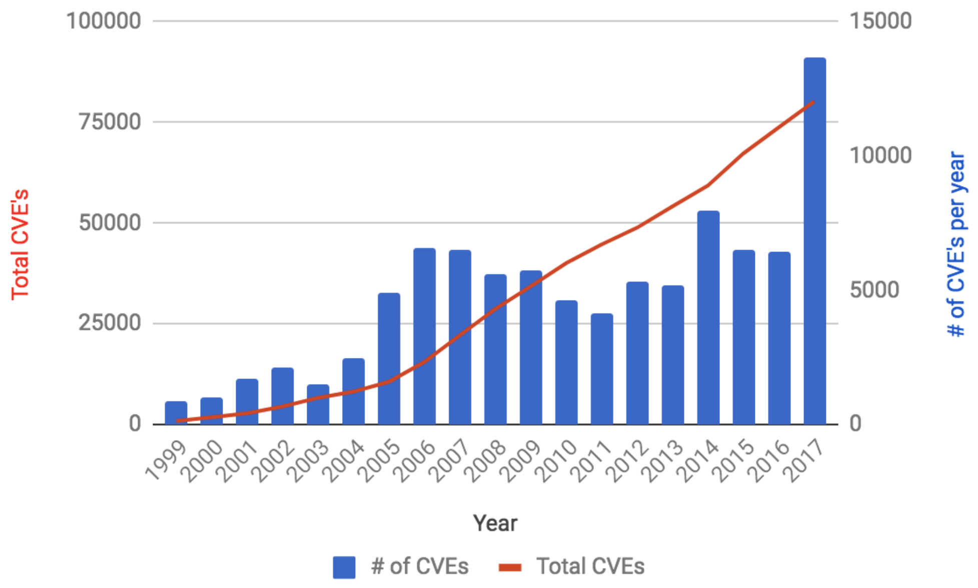


Figure 1: CVEs added to the CVE list each year and the total number of CVEs in the list

Figure 1 shows us that the number of vulnerabilities published to the National Vulnerability Database each year is growing. The number of new vulnerabilities found in 2017 was roughly double the number found in 2016. One could speculate the reasons behind this development, but regardless of the reasons these numbers are troubling and underline the fact that we need to be vigilant when it comes to computer security. Even though these numbers are known and publicly available, the situation has not improved. Our goal is to identify IP addresses that are affected by these vulnerabilities and to gather data to see how the number of vulnerable IP addresses changes over time. By making this information available to the proper authorities, we hope to advance the state of Internet security in Iceland.

2.2 Standards and Definitions

Key concepts needed to understand the research that Heimdallur is built upon will now be introduced.

2.2.1 Common Vulnerabilities and Exposures (CVE)

This project heavily depends on the standards defined for Common Vulnerabilities and Exposures (CVE) and Common Platform Enumeration (CPE). The CVE list was created with the purpose of standardizing communication about vulnerabilities. Before the CVE list, different groups of people would use different names for the same vulnerabilities, which made it difficult to discuss or analyze the extent of certain vulnerabilities. The CVE list is designed to be a dictionary, where every vulnerability has one name and one standardized description. The CVE list is publicly and freely accessible via the Internet [9].

2.2.2 Common Platform Enumeration (CPE)

In Common Platform Enumeration: Naming Specification [10], CPE is described in the following way:

> *Common Platform Enumeration (CPE) is a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets. CPE can be used as a source of information for enforcing and verifying IT management policies relating to these assets, such as vulnerability, configuration, and remediation policies. IT management tools can collect information about installed products, identify products using their CPE names, and use this standardized information to help make fully or partially automated decisions regarding the assets.*

A well-formed CPE name, known as WFN, is a way to represent the name and version of a software product. A WFN can include a vendor, product, version and update. This name can then be converted into machine-readable form by using methods of binding. The CPE specification defines two methods of binding: Uniform Resource Identifier (URI) and formatted string. URI was used in CPE version 2.2, and has been kept a part of version 2.3, which is the current version, for backwards compatibility. Formatted string binding supports additional product attributes and the syntax of these two methods is somewhat different [10].

2.2.3 Common Vulnerability Scoring System (CVSS)

The Common Vulnerability Scoring System (CVSS) is a way to assign a score to vulnerabilities. A CVSS consists of metrics that are divided into three groups: base, temporal and environmental. The metrics in the base metric group represent the characteristics of vulnerabilities that are constant over time and across environments. The temporal metric group contains metrics that represent characteristics that are constant across environments but not over time. The environmental group contains metrics that represent characteristics that are dependent on the environment.

Here we will focus mainly on the base metric group. This group can be further divided into exploitability metrics and impact metrics. The exploitability metrics are: attack vector, where a higher metric value indicates the possibility for a remote attack; attack complexity, where a higher value indicates a more complex attack; privileges required, that describes the level of privileges required before being able to use the exploit; and user interaction, that indicates that a third party has to be involved. The impact metrics are: confidentiality-, integrity- and availability impact. Confidentiality impact measures the impact that a successful exploit would have on the confidentiality of the information that should be protected. Integrity impact measures if the information can be altered or corrupted. Finally, the availability impact measures how accessible and available a component is.

The attack vector is of much interest to us, since it indicates the context in which the exploit is possible. A high attack vector indicates that the exploit does not require physical proximity, but allows the attacker to use it remotely [11]. Since Heimdallur only checks for remote access to hosts, vulnerabilities that have a higher attack vector are of more relevance to this research than those with lower values.

2.2.4 National Vulnerability Database (NVD)

The NVD is a database that was created by NIST Computer Security Division, Information Technology Laboratory and is maintained by the U.S. government. The NVD was originally created in 2000, under a different name, but has gone through several changes and improvements since then. The NVD is sponsored by the Department of Homeland Security's National Cyber Security Division. It contains information on CVEs from the CVE Dictionary along with the corresponding CVSS and CPE. The NVD can be queried with a CPE, and will then provide information about the CVEs connected to the platform, along with their CVSS association. The database is regularly updated with new CVEs and corrections to older entries [12].

2.2.5 Vulners

Vulners is a vulnerability database with an open API. It contains a number of exploits that have been gathered from different security researchers. The data Vulners aggregates comes from sources that include Metasploit and ExploitDB, which are the two largest curators of community sourced exploits. CPEs can be used to query Vulners for known exploits and information about them [13].

2.2.6 Scanners

There are mainly two types of scanners used to scan things on the Internet. Fast scanners where the main goal is to scan as many IP addresses and ports as quickly as possible, often the entire IPv4 range in just a few minutes, and slow scanners where the focus is on being as thorough and accurate as possible. The fast scanners are sometimes called connectionless or SYN scanners. They only check if the port is open and responsive, but do not complete the TCP handshake. These slow scanners are sometimes called connection-oriented scanners. They complete the TCP handshake and try to get additional information about the service running behind the port. Here we will discuss the scanners that we used for this project.

Nmap (Network Mapper) is a free, "open source tool for network exploration and security auditing". It is a connection-oriented scanner that produces a list of scanned targets along with information such as port number, protocol, service name and state. The state of the port tells the user whether the port is open or not. Nmap can also provide additional information about the target servers, such as reverse DNS names, operating system, device types and MAC addresses. Nmap has a scripting engine that allows users to write simple scripts to automate a wide variety of network tasks [14].

Masscan is a free, open source port scanner. It can be used on a variety of operating systems, from Windows to Mac OS X and FreeBSD, but its main platform is Linux. Masscan is a connectionless scanner and claims to be the fastest Internet port scanner, scanning the entire Internet in under 5 minutes [15]. Masscan is a command line tool and was made to have an interface that is similar to Nmap, so that it would feel familiar to Nmap users. However, its internal operation is different from Nmap and it uses asynchronous transmission. What this means is that transmit and receive threads are mostly independent from each other. In addition to detecting whether ports are open, Masscan "can complete the TCP connection and interaction with the application at that port in order to grab simple 'banner' information". Its default transmission rate is 100 packets/second, but the user can adjust that along with other parameters [15].

Masscan sends a SYN packet to the target service when performing a connectionless scan. It then waits for a SYNACK response from the target, this means that the service is active. In order to make sure that the scan does not disrupt the target with an abundance of open connections just waiting to time-out, Masscan sends a reset packet which terminates the connection.

2.2.7 Banner Grabbing

Banner grabbing or NULL probing is a method used by scanning tools like Nmap. This method involves sending a request to a server on a specific port and waiting for a reply. The reply often contains a banner. Banners can include information about the service running on a given port. When a banner is received, we know that the port is open and available. Not all servers will reply with a banner. In addition, banners can be modified and could therefore contain incorrect or misleading information. Even though it is possible that some banners may give us incorrect data, the banners will in general be the best guess possible without using a more invasive method. Heimdallur uses the banners to extract information about the services and find the corresponding CPE.

2.2.8 IP Address

When we scan an IP address, we cannot be sure whether there is a single host behind it or if there are multiple devices. For this reason, we will mostly refer to IP addresses in our discussion, but this should be thought of as interchangeable with hosts or networks.

3 Design

The design goals for Heimdallur include being fast, modular, non-invasive, ethical and assigning an accurate and predictive threat score to each device covered. There is a trade-off between speed and non-invasiveness. The scanning process should be fast enough to cover all of Iceland in a week, but sending out too many packets at the same time might crash servers or trigger warnings about an attack. Being non-invasive is a part of having an ethical scanning system. The security is ensured by using a secure database system and storing the data on our server that is located in Iceland. Thus, the system is protected both by law, since the Icelandic law does not permit the government or others to access our server without permission, and by access control.

3.1 Ethical Concerns

Before starting on a project such as this one, it is important to be aware of ethical concerns and decide on some sort of ethical code of conduct. We discussed our concerns with a lawyer, as well as our instructor and advisers. Following that discussion, we decided on guidelines that we would follow throughout the project. We will now discuss these guidelines.

First of all we decided to scan only Icelandic IP addresses, to avoid any legal issues. In addition, we do not run any exploits nor do we send malformed packets. The addresses and port numbers are scanned in random order and the traffic is kept around or below 10 Mbit/s. This should ensure non-invasiveness and minimize disruption. The purpose of the scan is explained on our website `https://scan.syndis.is` and the user-agent string of Heimdallur includes an email address to allow people to opt-out. We maintain a blacklist, containing the IP addresses that should not, and will not be scanned. The reasons behind this ethical code are further explained in the Discussion and Future Work section.

3.2 Metrics

In order to measure the goals for Heimdallur, we have a few metrics. The first metric we will consider is the throughput. Throughput of the scan will be measured in the number of IP addresses scanned by Masscan and Nmap in one hour. The total time for scanning all Icelandic IP addresses will also be measured.

The coverage of the system is defined by two main factors: the range of CVEs covered and the range of live IP addresses that are covered. Internet scanners do not get a reply to all the requests that they send. There are a couple of reasons for this, the first being that not all IP addresses correspond to an active device. The second reason is that not all active devices will respond with useful information such as service banners. For these reasons, we cannot expect a high response rate, but we will focus on extracting information from the answers we receive. To evaluate the coverage of the vulnerabilities that are exposed on the Internet, the proportion of possible CVEs that the system covers will be measured. Together, these two measures should provide an estimate of the overall coverage. In addition, we will find the percentage of live IP addresses that provide a CPE, since CPEs are used to look up CVEs, so the coverage of the results will depend heavily on CPEs.

The accuracy of the system will be an indirect result of how much coverage it provides, since the number of CVEs that Heimdallur finds for a randomly selected machine, will depend on the extent to which CPEs are covered. To measure the accuracy of Heimdallur, we will set up a number of containers with software that is either fully patched or affected by a CVE. In this way, we can conduct a controlled experiment and measure the accuracy of the results of Heimdallur, that is the proportion of CVEs found by Heimdallur compared to the number of CVEs planted in the containers.

The threat score should be a score that predicts the vulnerability level of each IP address. This means that a network with a high threat score should have a higher probability of being compromised then a network with a low threat score. No IP address should have the absolute minimum score, since 100% security is not possible and no IP address should have the absolute maximum score, since that would mean that it could not possibly become more vulnerable. Since the threat score is the first of its kind and highly subjective, providing a good and objective measurement for it is tricky. We will use the random sample from the accuracy test, score each machine and then use the score to rank them. These results will then be evaluated by an expert in the field.

3.3 Architecture

There are many ways to structure the system we desire and here we will describe the general architecture and layout of our approach. There are two parts to the operation of the system, the data gathering and the data processing. Figure 2 shows a diagram of the architecture.

For the data gathering, we start with a list of IP addresses to scan. A random port is selected and a connectionless scan is subsequently performed on the entire list of IP addresses. The order of the IP addresses is randomized to minimize disruption. Ports that respond are then fed into a connection-oriented scanner after they have been identified. The connectionless scan and the connection-oriented scan are performed in parallel.

A connection-oriented scan is then performed on the given port for the list of IP addresses which attempts to identify the service behind each port. The scanner extracts the banner of the service and tries to generate a CPE based on that. If there was insufficient information in the banner to generate the CPE, then further interrogation of the service is required. The scanner sends simple requests to the service. This may include a GET request to an HTTP server or a HELP command to an FTP server, for example. Based on the responses of the service, the scanner tries to generate a CPE once again. The CPE is used to make it easier to connect a service to vulnerabilities later in the process. The output of the connection-oriented scan is then parsed and stored in a database. This concludes the data gathering and begins the data processing.

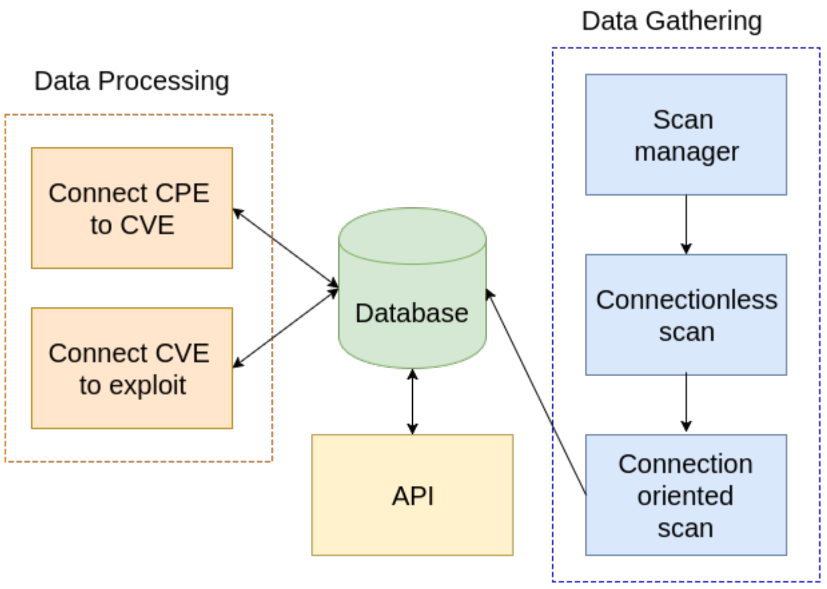


Figure 2: The architecture of Heimdallur's two processes, the gathering of data via scanning and processing the vulnerability score

The main purpose of processing the data is to acquire information about the exploitability of each host. When the information about vulnerabilities connected to each open port on a network has been gathered, these can be aggregated to create a combined threat score for the IP address. Other aspects of processing involve creating an API and visualization of the data.

In order to detect which platforms are affected by which vulnerability, we depend on a database which stores a mapping between CVEs and affected CPEs. We find the CPE for the service and look it up in a CVE database to determine the vulnerabilities of specific service. This will result in a list of software weaknesses that the given service is susceptible to.

Finding which vulnerabilities have known exploits easily accessible on-line is invaluable to determining if a service is under immediate threat. To do this the system should depend on an exploit database which maps vulnerabilities to known exploits. This database should be able to receive a query with a CVE and respond with a list of exploits. Interactions with the system should go through an application programming interface (API). The API acts as a middleman between the database and the client. It is then possible to create a separate web user interface that communicates with the API. A custom web user interface is not a part of our project.

4 Implementation

As described in the Design section, the functionality of Heimdallur is twofold and each part is composed of modules. To fulfil our design goal of being decoupled we split each section of the system into individual modules. The manager module is where we control when and how to scan a given IP address. The manager calls the scan module where connectionless and connection-oriented scans are performed. To parse the data from the scan, we use the convert module which the processor uses to prepare the data before it goes into the database. Interactions to the database go through the models module.

Here we will describe the current implementation of Heimdallur and all of its parts. First, we will discuss the previously existing scanners that the system makes use of. We will then go on to discuss the implementation of the two parts of Heimdallur: the data gathering and the data processing.

4.1 Tools

As discussed before, Heimdallur uses two types of scanners, one connectionless and one connection-oriented. When we were choosing a connection-oriented scanner we were looking for something that could extract banners, identify common services, generate CPEs and be easily extensible to send custom requests and store the response. We looked at several scanners and then decided that Nmap would be the best fit for our implementation. Nmap is able to parse banners to identify common services and furthermore, sends additional requests to generate a CPE. As well as being able to determine services, Nmap has a highly extensible scripting engine (NSE) which includes a rich repository of user submitted scripts.

When looking for a connectionless scanner we wanted something that was able to perform a fast SYN scan and closes the TCP connection after a response. Among other things, we looked at Nmap as a possible candidate, but because Masscan wrote their own custom network stack it manages to be way faster than Nmap. Masscan was therefore our ideal candidate to detect open ports in our implementation and conveniently, Masscan's interface is very similar to Nmap.

4.2 Data Gathering

Our data gathering process involves collecting as much information as we can about Icelandic IP addresses. Limitations were put on the scope of our scan due to legal reasons as we discussed in the ethical section. This limits our scan to 910,720 IPv4 addresses. Along with limiting our network range we decided that we would only scan the 1,000 most frequently open ports in the interest of time. According to Nmap's research, "[t]he top 1,000 (out of 65,536 possible) finds roughly 93% of the open TCP ports" [16]. A list of these ports is provided by Nmap's "–top-ports 1000" option.

We wanted to be able to do complete scans of Iceland and still manage to iterate on the infrastructure to improve it within the given time frame of the project. We decided that scanning UDP would not be worth the effort and focused on TCP instead. Identifying services on UDP is much harder than on TCP and it would almost double our current scan time according to our estimate. Our system is implemented in Python which was decided due to the general-purpose nature of the language, which makes it ideal to quickly setup prototype systems. Here we will go in depth of how each module is implemented and its interactions with subsequent modules in the system.

4.2.1 Manager Module

The manager module is responsible for initiating threads that perform connectionless and connection-oriented scans. We modeled the scanning process after the producer-consumer problem. The producer-consumer problem describes two processes, the producer and the consumer, who share a queue. The producer generates data and adds it to the queue while the consumer runs in parallel consuming the queue one piece at a time. In our context, this means that we have a producer thread that adds alive services to a queue while the consumer threads gather more information about the services.

The producer is passed a Masscan configuration file, a list of ports and a shared queue as input parameters. In the configuration file is a list of IP addresses or IP ranges to scan. The order of the list of ports is randomized in order to not raise suspicion when scanning consecutive ports within the same IP range, for example. The port number along with the Masscan configuration file are passed to the scan module to perform port detection. Figure 3 describes how the producer functions.

```
Data: A shared queue (queue), A Masscan configuration file (config) and a list of
      ports (ports)
Result: Masscan output files for each scanned port
Randomize the order of ports;
foreach port in ports do
    Scan.port_detection(config, port);
    Add the result of the scan to queue;
    Store scan result in a JSON file labeled with port;
end
```

Figure 3: Producer

The consumer runs in parallel to the producer. A shared queue is passed to the consumer where it continually dequeues items as they get added by the producer. These items are a port paired with a list of IP addresses. The consumer performs service detection on the list of IP addresses it removed from the queue and stores the result as described in Figure 4.

```
Data: A shared queue (queue)
Result: Nmap output files for each scanned port
Take a port, IP list pair ips off the queue;
Scan.service_detection(ips, port);
Store scan result in an XML file labeled with port;
```

Figure 4: Consumer

4.2.2 Scan Module

The scan module is split into two parts, finding open ports and detecting the service. To discover open ports, we receive a list of IP addresses along with a port number. We initialize a new instance of Masscan with the following parameters:

```
masscan -c <config> --port <port>
    --excludefile <exclude file> --output-filename <output file>
    --http-user-agent <excuse>
```

The configuration file and port is provided by the manager module. A list of IP addresses to scan is contained in the configuration file. The exclude file is a list of IP addresses that are on our blacklist which we do not scan. The output file is a JSON file which is labeled with the scanned port. We put an explanation of our scanning project into the HTTP user-agent string of any web server that we scan, so that administrators can see what we're doing in their server logs. The log will contain the following message:

> This is a part of a university research project. See more info at https://scan.syndis.is. Opt-out by sending an email to scan@syndis.is

When Masscan has finished executing, we parse the generated output file to find which IP addresses responded on the given port. This list is paired with the port that was given as an input parameter and returned to the manager module. In order to perform service detection, we run an instance of Nmap with the following parameters:

```
nmap -Pn -sV -sC --script http-get -T4 -oA <results folder> -p <port> <IPs>
```

Because we already know that we only get hosts that are alive from the port detection phase, we can disable host discovery using "-Pn". The option "-sV" is used to determine the version of the service running. We also run all the scripts in the Nmap Script Engine (NSE) labeled as being "safe", this is done with the "-sC" option. The scripts marked by Nmap as safe, are non-invasive scripts that should not send malformed packets or cause disruption.

We wrote a custom HTTP GET script in Lua through the NSE which stores the response from a GET request to the root of any web server that we find. The "–script http-get" is used to active our custom script. This allows us to query our scan results further, even though we do not use it as of now. Finally, we save all the outputs Nmap offers to disk using "-oA".

4.2.3 Models Module

SQLAlchemy, which is a Python library, is used to set up and interface with our PostgreSQL database. SQLAlchemy is an SQL tool kit and an object relational mapper that allows us to define Python classes for the tables in our database. The database consists of three main tables, along with a few smaller connection tables. One table contains the scanning results, another has information about vulnerabilities from NVD [12] and the third contains known exploits from Vulners [13]. The connection tables connect CPEs from the scanning table to CVEs in the NVD table and CVEs to exploits. Figure 5 shows a diagram of all the tables in the database and how they are connected. A more detailed diagram can be found in Appendix I. Below is a list of the tables, which describes each table and its purpose.

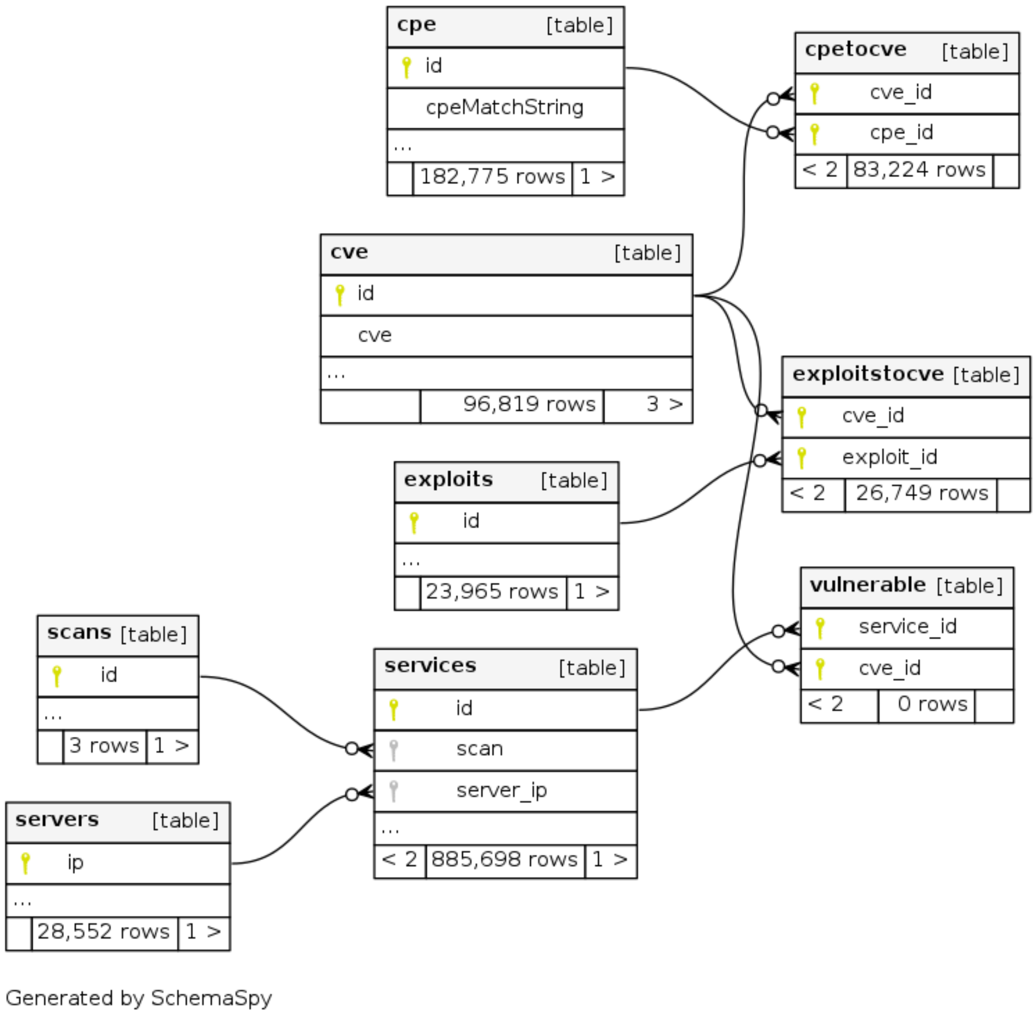


Figure 5: Relationship diagram with omitted attributes

- The **servers** table contains data about a specific network where the IP address of the network is the primary key. We call this table **servers** even though there could be many devices on this specific IP address.
- The **services** table stores information about a given port on a IP address. Services are connected to the **server** table through the *server_ip* foreign key. The ID of the scan references the **scans** table so we are able to identify which complete scan this service snapshot belongs to. The service banner and a JSON list of matching CPEs are all stored in the database along with the output of all the safe Nmap scripts in JSON format.
- In order to store individual scans we have a table called **scans**. This is only stores complete scans of Iceland, when they started and when they finished.
- The **cve** table stores a partial CVE database gathered from the NVD. It contains the CVE identifier, the CVSS score and a breakdown of the variables that are used to calculate it for both version 2 and 3 of CVSS.
- The **vulnerable** table is a connection table between the **service** and **cve** tables. It stores a service ID and a CVE ID which indicates that the service is susceptible to a given CVE.
- The **cpe** table stores CPE match strings for common platforms. We also store the CPE version 2.3 match string which we do not utilize as of now.
- The **cpetocve** table is a connection table between the **cpe** and **cve** tables. It stores a CPE ID and a CVE ID which indicates that the given platform is susceptible to a CVE.
- The **exploits** table stores exploits gathered from Vulners. We store the title of the exploit, type (either Metasploit or ExploitDB), a link to the exploit and when it was released.
- The **exploitstocve** table is a connection table which connects a CVE to an exploit. This table indicates that an exploit exists for a give vulnerability.

4.3 Data Processing

A couple of public database, known as the National Vulnerability Database (NVD) and Vulners, are used to process the scan results. These databases help us connect software weaknesses to platforms and exploits. This data is stored in the database such that it can be served by our API.

4.3.1 Converter Module

The converter module is responsible for parsing the XML output files generated by Nmap into a Python object. The parser depends on the python-libnmap library. Libnmap is a Python library which makes it easier to parse and manipulate Nmap data. Python classes are provided with libnmap to store and work with Nmap results.

4.3.2 Processor Module

When processing the scan results of Nmap, we start off by creating a new scan entry. The entry in the scan table indicates when the scan started and when it finished. This also allows us to group services based on when they were scanned, which makes us able to research historical data. Services are processed by iterating through output files which contain information about individual ports. Each file is passed to the converter module in order to get a Python object that we can work with. We first check if we have already added the current host to the database, if not, we create a new entry in the servers table. Subsequently, we create a new entry for the service that we just scanned which contains a list of CPEs, banner, output of Nmap scripts among other things. The Nmap results directory is then compressed and archived just in case we would need it at a later date.

In order to link software weaknesses to services that we scanned, we depend on the NVD. This database maps CPEs to CVEs which indicates that a particular platform is susceptible to some vulnerabilities. We downloaded their database which contains CVE mappings dating back to 2002. We decided to ignore all CVEs that were marked as either "** DISPUTED **" or "** REJECTED **". This occurs when the vendor of a given product believes the entry not to be valid. For each valid CVE, we find affected CPEs and create a new entry for both the CVE and the CPE, making sure that they don't already exist. The two are then connected through the **cvetocpe** connection table.

These software weaknesses are then searched in the API provided by Vulners. Vulners stores a mapping between CVEs and known exploits. We decided to limit our search to exploits only found on Metasploit and ExploitDB. When an exploit is found, a new entry is created in the database, with the source of the exploit, a publishing date and more. A connection is then made between a CVE and an exploit through the **exploitstocve** table.

Heimdallur iterates through each service and looks at its list of CPEs to determine which services have known vulnerabilities. It ignores CPEs that it is unable to determine the version of, along with CPEs of operating systems. We found that operating systems CPEs were way too general, there was too much uncertainty for us to able to determine if a specific operating system was vulnerable or not.

Heimdallur determines if a service is vulnerable by querying the database with a CPE and checks if a there is a CVE associated with it. If such a mapping exists, a new entry is added to the **vulnerable** table which contains the CPE and CVE IDs. Our database now has enough data for us to start speculating how secure these services are.

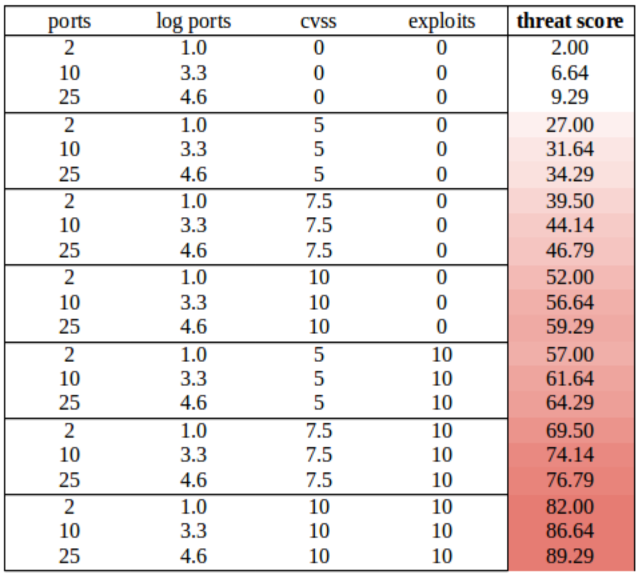
4.3.3 Threat Score

We now start looking at the implementation of our proposed threat score. The threat score will depend on CVSSs for CVEs connected to the services running on the device and whether an exploit exists for those CVEs. In addition, the number of ports open will increase the score, since more open ports indicate a larger attack surface. The proposed formula is as follows:

$$\textbf{threat\_score} = 10 \cdot (\beta_1 \cdot \max(CVSS) + \beta_2 \cdot \mathit{exploit\_exists} + \beta_3 \cdot \log_2(\mathit{num\_ports}))$$

Here $\max(CVSS)$ is the highest CVSS of the CVEs associated with any service on the IP address in question. $\mathit{exploit\_exists}$ is either 0 or 10, depending on whether an exploit exists for any of the CVEs, and $\log_2(\mathit{num\_ports})$ is the base 2 logarithm of the number of open ports on the IP address, rounded up to one decimal place. There can be at most 1,000 ports open out of the 1,000 ports scanned, and $\log_2(1000)$ rounded to one decimal place is ten. Furthermore, if there is only one port open, the port score defaults to 0.5 instead of 0. CVSS scores also range from zero to ten and the Boolean $\mathit{exploit\_exists}$ is multiplied by ten in order to have ten as the maximum value. The beta constants are weights that can be adjusted, but their total sum should be one. That way the score ranges from 0 to 10. Heimdallur's threat score should be a number between 0 and 100. Thus, we multiply our formula by 10.

After consulting with an expert in the field, we chose values for the constants so that $\beta_1 = 0.5$, $\beta_2 = 0.3$ and $\beta_3 = 0.2$. Figure 6 shows the table used to determine the weights for each part of the formula. The color of the score indicates the severity of it, white indicates relatively little risk, while red indicates danger.



ports	log ports	cvss	exploits	**threat score**
2	1.0	0	0	2.00
10	3.3	0	0	6.64
25	4.6	0	0	9.29
2	1.0	5	0	27.00
10	3.3	5	0	31.64
25	4.6	5	0	34.29
2	1.0	7.5	0	39.50
10	3.3	7.5	0	44.14
25	4.6	7.5	0	46.79
2	1.0	10	0	52.00
10	3.3	10	0	56.64
25	4.6	10	0	59.29
2	1.0	5	10	57.00
10	3.3	5	10	61.64
25	4.6	5	10	64.29
2	1.0	7.5	10	69.50
10	3.3	7.5	10	74.14
25	4.6	7.5	10	76.79
2	1.0	10	10	82.00
10	3.3	10	10	86.64
25	4.6	10	10	89.29

Figure 6: Threat scores for different combinations of open ports, CVSS and exploits

4.3.4 API

We implemented an API using GraphQL [17], which is a query language for APIs. The API was written in Python using the Graphene-Python [18] and the Graphene-

SQLAlchemy [19] libraries. GraphQL has a graphical interactive in-browser integrated development environment (IDE), called GraphiQL, which we enabled on top of GraphQL to make it easier to explore the data gathered.

A GraphQL schema was created for all tables in the database, giving our end users full access to all the data. To secure the API endpoint, the web server was configured to only allow a specific IP address through, the IP address of Syndis' office. For client authentication, a SSL certificate is needed to establish access to both the GraphQL API endpoint and the GraphiQL web user interface.

5 Evaluation

The questions we set out to answer concerned the throughput, coverage and accuracy of Heimdallur, as well as the accuracy of the threat score. We wanted to know how fast Heimdallur could scan Iceland, how many IP addresses would respond and how well we could identify the services running behind the ports we scanned. We also wanted to know how well our threat score could identify IP addresses that are vulnerable, according to an expert's opinion. Our discussion now moves on to how we sought to answer each of these questions and evaluate our results.

5.1 Throughput

We conducted several complete scans of the 910,720 Icelandic IP addresses, which progressively got better and faster over the course of the development process. Three of these scans are listed in table 1. As we can see, the total scanning time went from just over a week to just under 2 days. Since our goal was to do a complete scan in less than a week, the final results are well within the time frame.

Scan started	Scan completed	Total time
27.10.2017 at 15:56	03.11.2017 at 18:00	8 days, 2 hours, 4 minutes
29.11.2017 at 16:23	01.12.2017 at 15:15	1 day, 22 hours, 52 minutes
11.12.2017 at 16:42	13.12.2017 at 17:18	2 days, 0 hours, 36 minutes

Table 1: Complete scans of Icelandic IP addresses

5.2 Coverage

All of our scans covered the same 910,720 IP addresses. The first scan, that was completed on October 19th, covered the first 1000 ports (1-1000). 20,556 IP addresses were found to have at least one port open. We made the decision to scan the top 1,000 most open ports in hopes of increasing the coverage without increasing the time it takes to complete a full scan of all 910,720 IP addresses, as described in the Implementation section.

We noticed that roughly 86% (517,887 of the 602,296) of the ports Nmap detected were filtered. When Nmap was scanning the service, it completed a full TCP handshake, but the remote host closed the connection without receiving any data. This could indicate that some sort of network security device like a firewall or intrusion prevention system is protecting the port. Many of these network security devices are configured to respond to TCP port scans. This behavior can slow down a port scan and cloud the results with false positives. Filtered ports have been removed from the results below.

Scan completed	Servers	Scanned ports	Open ports	CPEs matched	CPE Coverage
27.10.2017	24,103	1,000	84,409	45,439	53.83%
29.11.2017	23,985	1,000	82,871	44,587	53.80%
13.12.2017	24,065	1,011	83,717	44,142	52.73%

Table 2: Scan results

On October 27th we completed our second scan and found 24,103 servers with 84,409 ports open. Our system managed to find CPEs associated with 45,439 services on those ports. Thus, we were able to identify the service running behind 53,83% of the accessible ports that were found.

A second scan was conducted, which concluded on October 29th. The scan found 23,985 servers with 82,871 ports open. The system managed to identify services running behind 53,80% of the accessible ports. In effort to compare our system with what is available on the web today, we compared this scan with Shodan [4]. We queried Shodan for information on the same ports and IP addresses to compare to our results. Shodan managed to find 36,748 services on these ports which is less than half of the services Heimdallur discovered. When looking at the complete scan result of Shodan which contained roughly 230 ports, they found 52,354 services. This includes a number of ports we did not explore.

A final scan, for this part of the project, was completed on the December 13th. This time around Heimdallur found24,065 servers and 83,717 ports by scanning the top 1,000 ports along with 11 additional ports. We were informed by our resident expert that these ports are associated with commonly misconfigured services that should not be exposed on the Internet.

5.3 Accuracy

We setup a test bed with 34 different services, ranging from FTP servers to content management system (CMS). Docker images hosted on DockerHub were used to test each service. All services were tested with default settings and each service was tested with the latest available version and, if available, an outdated version. The outdated version was picked randomly from the available images on DockerHub. In total, we ran tests on 64 services.

We chose to use Docker and images hosted on DockerHub for a couple of reasons, images are tagged with the version they are running for that service which makes it easy to specify which version we wanted to run a test on. Docker images are pre-built therefore we did not have to go through the installation process for each service.

It is worth mentioning that there are a few limitations to our methodology. One of those limitations is that DockerHub only has a handful of services available, and for each of those services they usually only go back a year or two. This limits the available services we can run our tests on.

Tests were done using Docker version 17.09.0-ce-mac35 on a MacBook Pro 2016 running MacOS 10.13.2 and Nmap version 7.60 with the parameters:

```
nmap -Pn -sV -sC -T5 -oA results/nmap/$SERVICE -p $PORT(S) $IP
```

Here SERVICE is the name and version of the service being scanned, PORT(S) is the ports that the services has open and we are scanning, and finally, IP is the IP of the host we scanned.

5.3.1 Results

In order to evaluate the accuracy, we looked at the services that Nmap was able to find and checked if it found the correct version. The CPE Nmap returned indicates whether that was the accurate CPE for the version of the service that was found.

	Services found	Correct services	CPEs found	Accurate CPEs
Latest version	31/34 (91%)	18/34 (53%)	27/34 (79%)	11/34 (32%)
Outdated version	23/30 (77%)	14/30 (47%)	22/30 (73%)	8/30 (27%)
Total	54/64 (84%)	32/64 (50%)	49/64 (76%)	19/64 (30%)

Table 3: Service and CPE detection accuracy

As the results in table 3 show, CPEs were found for 76% of the services we scanned but only 30% of those CPEs were correct. Thus, we can only find CVEs for 30% of the services. The results also indicate that we only found CPEs for 59% of the services that were identified correctly. This indicates that the data needed to find the CVEs for those services is available, but since the CPE is not accurate we are unable to match them.

One of the main reason for the accurately identified CPEs only being 30% is that Nmap only does basic service detection and tries to enumerate the underlying services and not the services that sit on top of them. For example, it is unable to find a CPE for Jenkins but does find a CPE for the Jetty HTTP server which Jenkins uses to host their service on.

5.4 Threat Score

Figure 7 shows the distribution of threat scores for the responsive IP addresses from the scan conducted on November 29th. As we can see, the distribution is skewed to the left, and most of the scores are below 60. The median score is 2 and the average score is 7, with a standard deviation of 14.6. A score of 2 is equivalent to there being two ports open on that IP address.

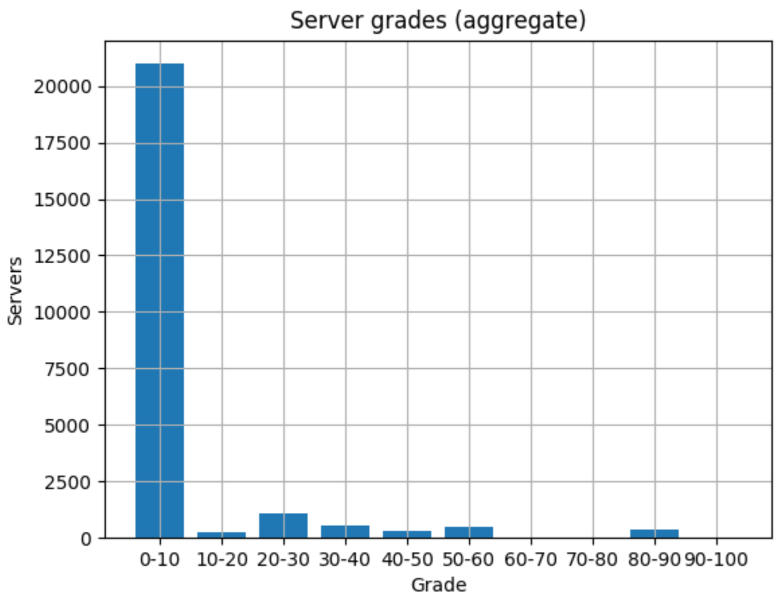


Figure 7: The distribution of threat scores

Out of the 24,119 responsive IP addresses, 87% received a score of less than or equal to 10 out of 100. These IP addresses can be considered relatively low risk. The addresses that have a score of 20 or above are the ones we need to take a closer look at. In order to be able to view those results better, we removed the addresses that scored below 10 and plotted the score for the rest. The results are shown in figure 8. Only a few IP addresses receive a score between 60 and 80 and then there is a spike between 80 and 90. None of the IP addresses score above 90.

As we mentioned before, the CVSS contains an attack vector, that indicates whether an attack requires physical proximity. If the attack vector is "Network", that means that a remote attack could be possible. For this reason, we removed all CVEs that did not have an attack vector of "Network", scored the IP addresses again and compared the results. Figure 9 shows this comparison.

As we can see, the number of scores between 0 and 10 increased, while the number of higher scores decreased. This is reflected by the decreased average, which is now 6.2, with a standard deviation of 13.9. The median score is not effected and remains as 2. The highest scores are not influenced much by this change, which is to be expected, since the highest CVSS scores are for network vulnerabilities. Less false positives should now be found in the results for network vulnerabilities. That means that the probability of an IP address being marked as vulnerable, even though it isn't, should be lower when we limit the vulnerabilities to network vulnerabilities.

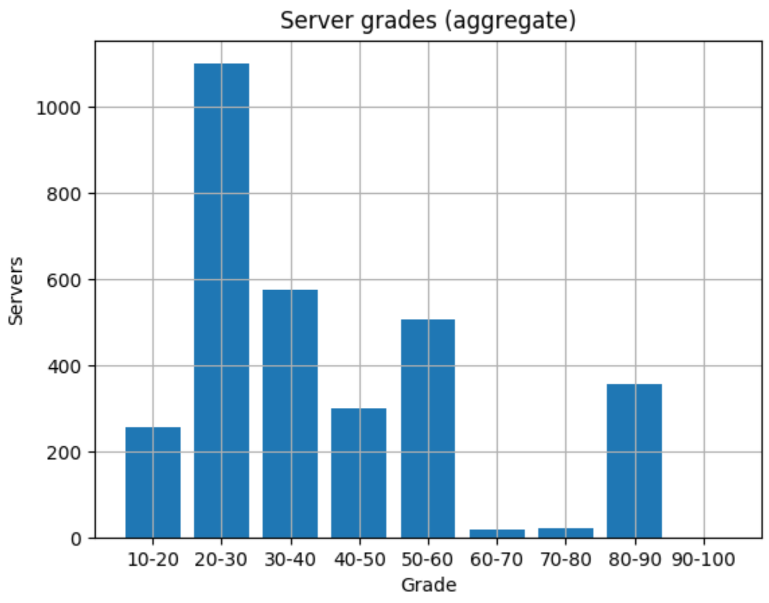


Figure 8: The distribution of threat scores of more than 10

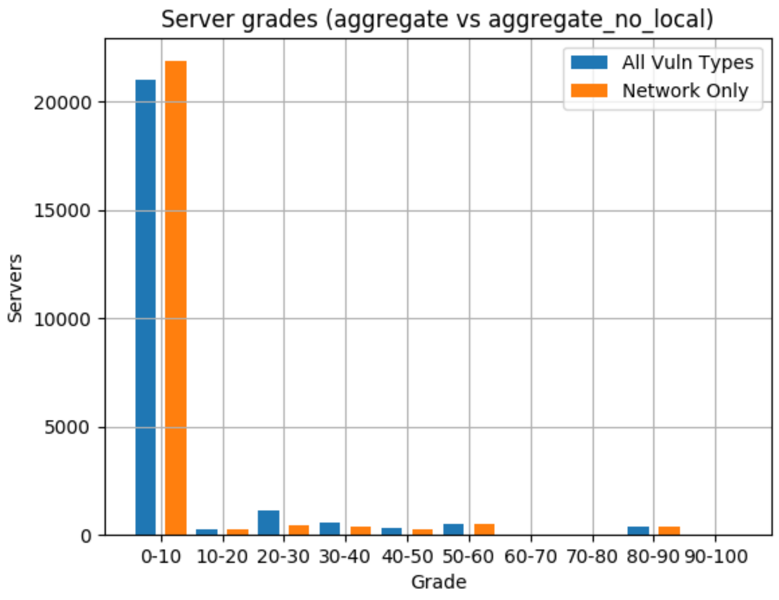


Figure 9: The distribution of threat scores for all CVEs compared to the distribution of threat scores for remotely exploitable CVEs

6 Related Work

Previous work on Internet scanning and vulnerability detection has mostly been focused on either scanning the Internet or detecting vulnerabilities, but rarely both. Here we will discuss some of the research that has been done on mapping the Internet's security landscape and how it relates to our current work.

6.1 Internet Scanners

Durumeric, Bailey and Halderman conducted research where they analyzed Internet scanning, that is, who were running the scans and in what purpose. Their focus was on horizontal scanning, that is focusing on a single port at a time. They found that researchers could usually be identified, since they provided information on the purpose of the scan. Malicious scans were often from bullet-proof hosting providers with no identifying information. Durumeric et al. found that after more than two years of regular scanning, companies were still discovering it for the first time. Furthermore, over half of the companies and corporations that opted out discovered the scan after more than a year had passed. This indicates that companies are not actively detecting whether they are being scanned, but discovered it by accident or as a part of maintenance [20].

The first search engine for Internet-connected devices was Shodan. Shodan is regarded as one of the most popular search engines for Internet services. Its creators conduct regular port scans to gather fresh information about the Internet. People can use this search engine to make queries about online surveillance cameras and default passwords, to name a couple of things. They have done vulnerability assessments for a few vulnerabilities, such as HeartBleed, but vulnerability scanning is not their focus. In order to use Shodan, people must sign up as users on shodan.io and are charged for substantial scan results [4].

In 2015 Durumeric et al. introduced a new search engine for software vulnerabilities, called Censys. They use ZMap and ZGrab (which were created by some of the same authors) to do Internet-wide scanning. Censys' approach is less aggressive than Shodan, and they focus on "good Internet citizenship", publicly providing information on who they are and why they are scanning [5]. This approach is similar to the one we are taking, although our method is somewhat different.

6.2 Vulnerability Detection

Software vulnerability is an extensive and pervasive problem. Security experts and governments have tried to collect and publish known vulnerabilities, so that people can discover if their computers might be compromised. There have also been attempts to scan multiple devices, or even all devices connected to the Internet, and assign possible vulnerabilities to them. Some of these attempts were made by Shodan and ShoVAT. However, detecting services and assigning vulnerabilities to them is no easy undertaking and there have been mixed results. Nevertheless, this is an important task and one that can help reduce the problem of software vulnerability.

In the field of vulnerability detection there is a variety of methods used. These methods can be classified as either passive or aggressive detection. When doing aggressive scanning, an exploit for a security flaw is tested against a target server. Running exploits on servers owned by companies is frowned upon and in most cases, illegal. However, this has been done in the past by projects such as Shodan which aggressively scanned for Heartbleed when it was first discovered [4].

Passive vulnerability detection is more common (and legal), especially for researchers that are trying to assess the security of the Internet as a whole. However, they have several shortcomings due to their passivity. Passive methods usually analyze banners returned by servers, generate a CPE from the banner and link the CPE to a CVE. We only looked at other projects that were doing passive vulnerability detection even though aggressive scanning yields more accurate results.

There is an inconsistency between CVE and CPE, such that not all CVEs have CPEs listed and not all CPEs are assigned a CVE. This can lead to false negatives, that is, not identifying vulnerabilities that are never the less present. An automated system could try to guess the correct CPE for the CVEs that do not have any listed, but this could lead to false positives. Misspelling in CPEs has also created a problem, since the CVE list might have different spelling than the actual CPE. According to Sanguino and Uetz the CPE dictionary contained 2,614 deprecated entries in February 14th 2017, due to name correction [21]. These entries may not have been corrected in the CVE list, which might lead to even more false negatives. Sanguino and Uetz point out that on February 14th 2017, the CVE feeds contained a shocking number of 105,591 CPE entries that are not in the CPE dictionary. They also propose a solution for this, and have created open source software that should reduce false negatives and false positives, and thereby improve the matching. They use the Levenshtein algorithm to compare CPEs, so that if two CPEs have the Levenshtein distance of less than 3, they are called similar. This method provided better results for some CPEs, but did not fully solve the problem. The software they created also does not fully automate the process, but requires manually assigning CPEs to products. Therefore, although Sanguino and Uetz's research will be of value for us, we will probably not be able to make use of their software directly. [21].

ShoVAT is a vulnerability assessment tool that relies on Shodan to crawls the Internet and index discovered services. The output of Shodan queries are taken by ShoVAT and it parses the banner information of each service and automatically generates a CPE for each service. ShoVAT proposes a way to generate CPEs through an in-memory data structure that matches strings in banners. They use a heuristic function to help them prioritize strings in the banner to create more accurate CPEs. Their claim is that they can produce more accurate CPEs than other tools such as Nmap [22]. When the CPE has been generated, ShoVAT looks up the CPE in the National Vulnerability Database (NVD) to identify vulnerabilities that affect the CPE. Since Shodan's services are no longer free, ShoVAT's approach is no longer a feasible option.

7 Discussion

The data we have gathered is full of potential and we have only been able to scratch the surface of it. As is to be expected, we ran into some obstacles and had to address various issues during the process of this project. The nature of these issues also varied; some of them were ethical, while others were technical. Here we will discuss some issues that came up and how we dealt with them. We will also go into further discussion about the results and their implications.

7.1 Ethics

As we mentioned in the Ethical concerns section, it is important to decide on an ethical code for a project such as this one. We began by asking for permission to scan a small IP range of 4,096 addresses from a local Internet Service Provider (ISP), which we used to develop a prototype of our scanner. The throughput of the first scan was too high and triggered a distributed denial of service (DDoS) alert with our server host. This lead to a temporary loss of that server. The next scan was from an Amazon server, but as we found out that scanning is against their terms of service. A special permission can be requested to perform scans on a specific network. This would have required us to specifically request permission to scan each network in Iceland, which is not feasible. Thus, we ended up acquiring a server at a local data center.

We made a deal with Opin Kerfi to be our server provider. Opin Kerfi previously hosted Shodan, so they have dealt with large scale scanning before and know what that entails. Our deal with Opin Kerfi states that they will forward any abuse complaints to us as they are usually handled by the provider. We handle abuse complaints by responding with an email that explains our project and add the relevant IP address to our blacklist.

After setting up our server at Opin Kerfi, we realized that data that we managed to extract from that small IP range was very limited so we knew we had to do a larger test. We started scanning all of Iceland with the prototype scanner we had developed and one night into the scan we got our first abuse complaint. The complaint made us aware of an issue we should have addressed, but did not: we had not provided publicly accessible information of what we were doing and for what purpose. Therefore, we added a description of the project along with an opt-out email address in the user-agent string of our scanner. After adding the opt-out feature, we have only received one request to be added to our blacklist.

Our intention was to randomize the scan, so that both IP addresses and port numbers would be randomized. However, the first abuse complaint revealed to us was that the scan was not nearly as random as we thought. In order to cause as little disruption as possible, we aimed to scan hosts and ports randomly. It turned out that ports were not being randomized at all. We selected a random host and then we proceeded to scan 65,536 ports on that network, obviously triggering any scanner detector. Our scanner was promptly changed to also select ports at random and we began scanning anew.

We discussed with the head of operations at Opin Kerfi and we soon realized scanning too fast might cause disruption to the servers being scanned. Thus, it would not be feasible to aim purely for speed and try to match Masscans speed claims. The guidelines we got from our service provider was to keep the traffic around or below 10 Mbit/s. They also specified that occasional spikes of higher bit rates were acceptable if they would not persist. This put an upper bound on the scanning process. Nevertheless, we did not want the scanning to be too slow, since we wanted to scan continuously and conduct several complete scans during the project. Thus, our goal was to have the scanning process span anywhere between a day to a week.

7.2 Limitations

One of the problems we ran into is that CPE list is not up to date with NVD. As referenced before, Sanguino and Uetz have tried to find a way around this problem, but a good solution has not been found. This has a negative effect on the coverage, for example the CPEs for Redis, a popular database system, are outdated. This causes our system to miss out on important data. As mentioned in the Related work section, typos also pose a problem, since different spelling of CPEs in different databases will not match.

A limitation of this method of scanning is firewalls and network level access control, that keep us from acquiring information about the services running behind them. Such filters resulted in a reduction from 632,937 open ports, to 83,717. This is limitation that we need to be aware of, and will reduce our coverage, but we cannot do anything about it since that would require methods beyond our ethical boundaries.

All our scans came from a single IP address, so it is possible that our IP has been blocked by some of the servers being scanned, without our knowledge. Distributing the scanning could prevent this. Future work includes scaling the system up, exchanging threads for nodes and distributing the scanning process. Different servers would then scan different IP ranges. This should be relatively easy, since Heimdallur was designed to be modular and scalable and the long-term goal is to distribute the system and scan the entire Internet.

7.3 Throughput

The throughput of our scan was improved greatly over the duration of the project. The scan started off taking just over 8 days. This is because we experimented with splitting our IP ranges into individual buckets. A bucket had a port assigned to it and $\frac{1}{10}$th of the IP ranges because we had 10 buckets. Each bucket spawned a new Masscan and Nmap thread. This means that instead of scanning all the IPs in one thread, we did it in 10, increasing our overhead drastically. We found this out by trying different bucket sizes until we found out that having a single bucket for all the IP addresses was the fastest. This decreased our scan time to just under 2 days.

7.4 Threat Score

When assessing the threat score, it is important to keep in mind that it is the first of its kind and that it is subjective. This makes it tricky to evaluate it objectively. New factors could be added to our formula to account for other components, such as services that should not be exposed. The goal is that the score would have a predictive value, correctly identifying servers that are in danger of being tampered with. If more data were available on breaches, that could be used to make predictions of what hosts are in danger of being breached. However, such data is sparse and that complicates predictions.

This score should thus indicate how easy it is to breach a given IP address or whether it has already been compromised. This score is not a complete measure of vulnerability and we are not able to score the IP addresses that do not respond. Thus, the score should not be regarded as a comprehensive evaluation of the IP addresses security. It is merely an indicator that should identify IP addresses that can be associated with known vulnerabilities and exploits.

Over 300 IP addresses received a score between 80 and 90, while almost all the others scored below 60. Since the existence of an exploit accounts for 30% of the score, all the IP addresses that score above 70% can be associated with an exploit. If an IP address has a vulnerability with a CVSS of 10 and there exists an exploit for a CVE associated with it, the score will be at least 80. When an exploit is found, it drastically changes the score and that explains the lack of scores between 60 and 80.

8 Future Work

We have gathered a large amount of data and although we have began to analyze it, the data could be analyzed more to focus on a specific area of interest. A certain port or CVE could, for example be analyzed in more detail. The data could be visualized, for example by creating an interactive map of vulnerable servers in Iceland and their threat score. Ideally this would be a part of a graphical user interface. If that interface were to be publicly available, identifiers like IP addresses and exact geographical locations would need to be filtered out. This could be solved by creating a map that contains data and statistics for each country, without specifying anything about individual servers. A separate interface could also be created for privileged and trusted users like security professionals.

Some editions could be made to the threat score. For example, if a device is running a service that has a history of being compromised, its threat score could be raised. These services include a popular database and the Remote Desktop Protocol, which are easily misconfiguration, thereby providing a way in for malicious hackers. CVEs do not cover this type of vulnerability, so adding it to the threat score should increase its accuracy.

Adding other tools to get detailed information about a subset of services already found is one way to address the CPE to CVE problem we encountered. There are tools available that can be easily integrated into Heimdallur. One such tool is wappalyzer [23], which identifies technology used on websites. With the data from wappalyzer it is possible to map the newly detected services to CPEs, which are services we are not able to identify currently with the scanner being used.

We discussed with the head of operations at Opin Kerfi and he proposed an idea where we could try to associate IPv4 addresses to IPv6 addresses. This would allow us to answer interesting research questions such as if firewalls are identical for both protocol versions. Firewalls for IPv6 are often misconfigured to allow interaction with services that should not be accessible.

We would like compare the networks that we found to networks found on abuse lists. These abuse lists contain IP addresses of network that have been reported to partake malicious activities, such as email spam and DDoS. This would allow us to cross reference these data sets to see if there is a correlation between having a high threat score and being on one of these lists. Further comparison could be done, for example by scanning a single network and comparing the resulting threat scores to the national threat score. Scanning more countries and comparing the average and median scores as well as the distribution of scores would also be interesting.

9 Conclusion

In conclusion, we have successfully scanned all the 910,720 addresses several times, and gathered data on more than twenty thousand Icelandic IP addresses that were responsive and unfiltered. The scanning process has improved during the process of this project and a complete scan of Iceland now only takes about two days. Our comparisons with Shodan shows us that we gathered information about more services than they did. This could be partially explained by the fact that we scanned more ports than them. This shows that Shodan could be missing out on important data that could be the pinnacle of detecting and containing the next big malware epidemic. We have began to process this data, which opens up a variety possibilities for future work. The preliminary evaluation of the threat score looks promising, and the score could be used to identify IP addresses that are in danger of being breached or might have been breached already. This type of score could be very useful to help security professionals identify immediate threats.

10 Appendix

10.1 Database

In interest of space the relation diagram of the database was truncated. Figure 10 is the complete diagram of Heimdallur's database.

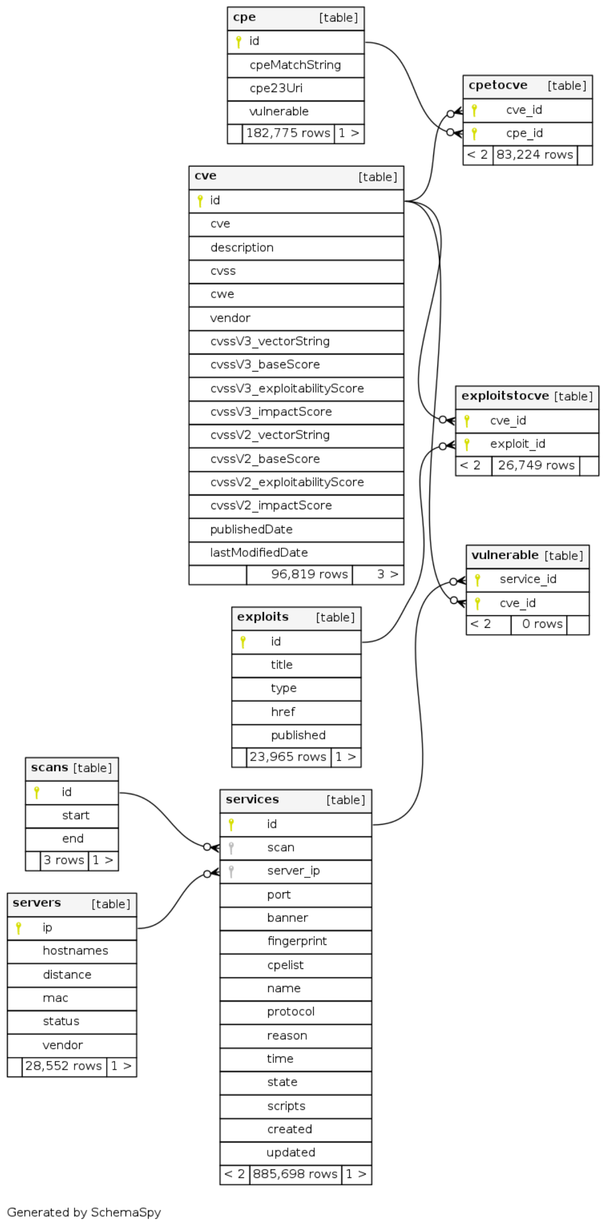


Figure 10: Full relationship diagram

References

[1] “Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016.” [Online]. Available: `https://www.gartner.com/newsroom/id/3598917`, [Accessed: Dec. 11, 2017].

[2] “Vísir: Leitarvélar finna barnapíutæki.” [Online]. Available: `http://www.visir.is/g/2017171009052/leitarvelar-finna-barnapiutaeki`, [Accessed: Dec. 11, 2017].

[3] “Forbes: Medical devices hit by ransomware for the first time in us hospitals.” [Online]. Available: `https://www.forbes.com/sites/thomasbrewster/2017/05/17/wannacry-ransomware-hit-real-medical-devices`, [Accessed: Dec. 11, 2017.

[4] “Shodan - the world’s first search engine for internet-connected devices.” [Online]. Available: `https://shodan.io`, [Accessed: Dec. 11, 2017].

[5] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, “A search engine backed by internet-wide scanning,” *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 542–553, 10 2015.

[6] “Nessus vulnerability scanner.” [Online]. Available: `https://www.tenable.com/products/nessus-vulnerability-scanner`, [Accessed: Dec. 11, 2017].

[7] “Reykjavík internet exchange: Icelandic ipv4 ranges.” [Online]. Available: `https://www.rix.is/english/is-net.txt`, [Accessed: Dec. 11, 2017].

[8] A. G. Brodeur, *The Prose Edda.* American-Scandinavian Foundation, 1916.

[9] R. A. Martin, “Managing vulnerabilities in networked systems,” *Computer*, vol. 34, no. 11, pp. 32–38, 2001.

[10] K. S. Brant A. Cheikes, David Waltermire, “Common platform enumeration: Naming specification, version 2.3,” *NIST Interagency Report 7695*, 8 2011.

[11] P. Mell, K. Scarfone, and S. Romanosky, “Common vulnerability scoring system,” *IEEE Security & Privacy*, vol. 4, no. 6, 2006.

[12] “The national vulnerability database.” [Online]. Available: `https://nvd.nist.gov/`, [Accessed: Dec. 11, 2017].

[13] “Vulners - vulnerability data base.” [Online]. Available: `https://vulners.com/`, [Accessed: Dec. 11, 2017].

[14] G. F. Lyon, “Nmap network scanning.” [Online]. Available: `https://nmap.org/`, [Accessed: Dec. 7, 2017].

[15] “Masscan - tcp port scanner, spews syn packets asynchronously, scanning entire internet in under 5 minutes.” [Online]. Available: `https://github.com/robertdavidgraham/masscan`, [Accessed: Dec. 11, 2017].

[16] G. F. Lyon, “Nmap network scanning: Host discovery and port scanning performance and features.” [Online]. Available: `https://nmap.org/5`, [Accessed: Dec. 11, 2017].

[17] “Graphql | a query language for your api.” [Online]. Available: `http://graphql.org/`, [Accessed: Dec. 13, 2017].

[18] “Graphene-python.” [Online]. Available: `http://graphene-python.org/`, [Accessed: Dec. 13, 2017].

[19] “Github - graphql-python/graphene-sqlalchemy: Graphene sqlalchemy integration.” [Online]. Available: `https://github.com/graphql-python/graphene-sqlalchemy`, [Accessed: Dec. 13, 2017].

[20] Z. Durumeric, M. Bailey, and A. Halderman, “An internet-wide view of internet-wide scanning,” *Proceedings of the 23rd USENIX Security Symposium*, pp. 65–73, 2014.

[21] L. A. B. Sanguino and R. Uetz, “Software vulnerability analysis using cpe and cve,” *arXiv preprint arXiv:1705.05347*, 2017.

[22] B. Genge and C. Enăchescu, “Shovat: Shodan-based vulnerability assessment tool for internet-facing services,” *Security and communication networks*, vol. 9, no. 15, pp. 2696–2714, 2016.

[23] “Wappalyzer - identify technology on websites.” [Online]. Available: `https://www.wappalyzer.com/`, [Accessed: Dec. 11, 2017].