

# Nationwide service and vulnerability detection

*Identifying vulnerable services using  
non-intrusive techniques*

Harald Aarseth



Thesis submitted for the degree of  
Master in Informatics: Information Security  
60 credits

Department of Informatics  
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2023



# **Nationwide service and vulnerability detection**

*Identifying vulnerable services using  
non-intrusive techniques*

Harald Aarseth

© 2023 Harald Aarseth

Nationwide service and vulnerability detection

<http://www.duo.uio.no/>

Printed: Representralen, University of Oslo

# Abstract

This Master's thesis explores the field of internet-accessible devices in Norway, their vulnerability status, and the challenges faced in developing a scanning platform to identify these vulnerabilities. Initial findings indicate that the developed platform is capable of conducting a nationwide network scan, providing a platform that identified known vulnerabilities in outdated software from open ports, showcasing its potential for detailed vulnerability assessments. Furthermore, the thesis highlights the critical need for continuous monitoring and vulnerability assessment of internet-accessible devices in order to ensure their protection against potential cyber threats.

# Acknowledgments

This project is the final project of my Information Security master's degree. I am thankful to supervisor Vasileios Mavroeidis for assigning me this interesting thesis. I am also grateful to the NREC cloud services at UiO for allowing me to conduct extensive internet scanning activities from their networks. Their provision of resources and infrastructure greatly contributed to the data collection and analysis, enhancing the quality of my research.

Lastly, I want to acknowledge the support from my family and friends, throughout this academic journey. Their encouragement and belief in my abilities played a significant role in overcoming challenges and reaching the completion of this thesis.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	2
1.2	Objective . . . . .	3
1.2.1	Research questions . . . . .	3
1.3	Thesis structure . . . . .	3
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Related work . . . . .	5
2.2	Literature review . . . . .	5
2.2.1	Literature review conclusion . . . . .	7
<b>3</b>	<b>Analysis of Vulnerability Scanners</b>	<b>8</b>
3.1	Open source solutions . . . . .	8
3.2	Closed Source Alternatives . . . . .	9
3.3	Exploring Available Search Engine Solutions . . . . .	9
3.3.1	Shodan . . . . .	10
3.3.2	ZoomEye . . . . .	11
3.3.3	Censys . . . . .	12
<b>4</b>	<b>Fundamentals of Internet Scanning</b>	<b>13</b>
4.1	IP Addresses . . . . .	13
4.2	TCP Protocol . . . . .	14
4.3	UDP . . . . .	14
4.4	Network ports . . . . .	15
4.4.1	Network Ports in the Context of Firewall and NAT . . . . .	16
4.4.2	Port ranges . . . . .	16
4.5	In-depth Port Scanning: Which Ports to Prioritize . . . . .	17
4.6	Detecting Software using Banner Grabbing . . . . .	20
4.7	Connection-Oriented, and Connectionless Scanning Techniques . . . . .	21
4.8	Nmap - Network Mapper . . . . .	22
4.8.1	The Nmap-services File: Mapping Port Numbers to Known Services and Protocols . . . . .	25
4.8.2	Optimizing Nmap . . . . .	26
4.9	Masscan: A High-Performance Port Scanning Tool . . . . .	27
4.10	Rate limiting . . . . .	28

<b>5</b>	<b>Identifying Services and Vulnerabilities</b>	<b>29</b>
5.1	Uncovering Services: Alternative Techniques . . . . .	29
5.1.1	Utilizing Ping for Host Discovery . . . . .	29
5.1.2	TLS Fingerprinting . . . . .	30
5.1.3	Favicon Hash . . . . .	31
5.1.4	Website Screenshots for Service Discovery . . . . .	32
5.2	An Overview of Common and High-Risk Vulnerability Types	32
5.3	Vulnerability Detection with Nmap . . . . .	34
5.4	Classifying vulnerabilities: CVSS System . . . . .	34
<b>6</b>	<b>Evaluation</b>	<b>36</b>
6.1	Performance and Detection Analysis: Nmap vs. Masscan . .	37
6.2	Assessing Public and Individually Collected Port Information	38
6.3	Assessing Throughput for Large-Scale Port Detection . . . .	39
6.4	Identifying the Most Frequent Network Ports . . . . .	40
6.5	Nmap Vulnerability Discovery Rate . . . . .	42
6.6	Assessing Bandwidth and Network Usage in Large-Scale Port Scanning . . . . .	43
6.7	Comparison of Nmap and Masscan for Banner Grabbing . .	45
<b>7</b>	<b>Implementation and development</b>	<b>46</b>
7.1	Choosing the Right Tools: Combining Nmap and Masscan .	46
7.2	Optimizing Network Discovery: Masscan and Nmap Integ- ration . . . . .	47
7.3	Storing and Visualizing Obtained Results: Elasticsearch . . .	48
7.3.1	Ingesting results into Elasticsearch Database . . . . .	50
7.3.2	Visualizing using the Kibana Dashboard . . . . .	53
7.4	Web Interface to launching Service Discovery . . . . .	54
7.5	Working with Docker . . . . .	55
7.6	Exploring Hosting Provider Policies on Port Scanning . . . .	58
7.6.1	Norwegian Research and Education Cloud . . . . .	58
<b>8</b>	<b>Results</b>	<b>60</b>
8.1	Comparing Identified ports against the Nmap-Service record	60
8.2	Nation-Wide Port Scanning: An In-Depth Investigation . . .	62
8.3	Results for Service and Vulnerability Scanning . . . . .	64
8.3.1	Amount of Identified Vulnerabilities . . . . .	65
8.3.2	Exploitable against Non-Exploitable vulnerabilities .	66
8.3.3	CVSS scoring . . . . .	67
<b>9</b>	<b>Discussion</b>	<b>68</b>
9.1	Port Detection Compared to Shodan . . . . .	68
9.2	Service Detection Compared to Shodan . . . . .	69
9.3	Vulnerabilities Detected Compared to Shodan . . . . .	71
9.4	Analyzing the Detected Results . . . . .	72
9.4.1	Unveiling False Positives: SYN Cookies . . . . .	74
9.5	Complains and Detection for the Conducted Scanning Activity	74



9.6	Choosing Appropriate Scan Intervals for Network Assessments . . . . .	75
9.7	Limitations . . . . .	76
9.8	Further Research and Work . . . . .	76
9.9	Legal and Ethical Implications . . . . .	78
<b>10</b>	<b>Conclusion</b>	<b>79</b>
<b>11</b>	<b>Appendix</b>	<b>82</b>
11.1	Code for the Main Scanning Function . . . . .	83
11.2	Script to detect Favicon Hash . . . . .	86
11.3	Masscan Service Discovery Test . . . . .	86
11.4	Nmap banner XML results . . . . .	86
11.5	Logstash-codec-nmap . . . . .	87
11.6	Webpage . . . . .	88
11.6.1	Scantypes . . . . .	88
11.6.2	Custom port scan . . . . .	89
11.6.3	Custom Nmap or Masscan command scan . . . . .	89

# List of Figures

4.1	The example banner displays the status for an Apache web server running on Linux Red Hat [31]. . . . .	20
4.2	TLS 3-way handshake and retrieval of banner information [32]	24
6.1	Results of exhaustive TCP port scanning, sorted by the number of ports detected . . . . .	41
6.2	Comparing Nmap and Masscan banner grabbing . . . . .	45
7.1	Stored XML information for port discovery . . . . .	47
7.2	Example of a document with banner information stored in ElasticSearch Discovery page for a discovered service . . . . .	52
7.3	Screenshot of Kibana Dashboard created for the UiO SecurityLab robot . . . . .	53
7.4	Homepage for the UiO SecurityLab robots web interface . . . . .	54
7.5	Dockerfile for running a provided Python script . . . . .	56
7.6	Overview of implemented Docker containers . . . . .	57
8.1	Comparing Nmaps stored port popularity against the UiO scanner . . . . .	61
8.2	Chart for the top most responding ports . . . . .	63
8.3	Most frequent service detected from banner grabbing . . . . .	64
8.4	Most frequent CVE detected from banner information . . . . .	65
9.1	List of the 20 most frequent open ports as of March 24, 2023.	69
9.2	Comparing Shodan and UiO robot service detection results	70
9.3	Comparison of Shodan vs. UiO Robot CVE detection . . . . .	71
11.1	Web interface for starting a new scan . . . . .	88
11.2	Web interface for default scan type . . . . .	88
11.3	Web interface for custom port scan type . . . . .	89
11.4	Web interface for custom command scan type . . . . .	89

# List of Tables

4.1	TCP and UDP ports required for different effectiveness rates [28]. . . . .	19
5.1	Favicon Hash Table [39] . . . . .	31
6.1	Network Scanning Tool Performance . . . . .	37
8.1	Exploitable vs Non-Exploitable data classification . . . . .	66
8.2	Distribution of vulnerabilities by CVSS Scores . . . . .	67

# Chapter 1

## Introduction

According to estimates, the number of devices connected to the Internet in 2023 surpassed 51 billion [1]. Alongside the rise of disclosed vulnerabilities [2], the probability of owning a vulnerable device connected to the Internet escalates. This escalation has provided cybercriminals with opportunities to exploit these vulnerabilities, eliminating the need for physical intrusion to pose a substantial threat to personal assets. To better understand the current attack surface, it is essential to have a way of monitoring the number of vulnerable devices online.

While existing search engines are available to access vulnerable device data, there are still questions about the accuracy and completeness of the information provided, particularly regarding closed-source data. There is concern that certain information may be censored, altered, or ignored, which can compromise the reliability of the data. Given these limitations, exploring other possible solutions and conducting experiments is essential to ensure that the information provided is trustworthy.

This thesis will examine strategies for overcoming obstacles related to implementing an efficient vulnerability scanning service with the ability to conduct on a large scale. The outcome will be a network port scanning service designed for UiO SecurityLab, which aims at identifying vulnerable services and devices within the Norwegian IP range for research purposes.

## 1.1 Motivation

Within the current digital landscape, the usage of the Internet has become an important part of our lives. It assists us in providing access to a wide range of information and services. This convenience however comes at a cost, as the Internet is filled with potential threats and vulnerabilities, where bad actors seek to gain unauthorized access to sensitive data. These malicious actors constantly search for ways to exploit vulnerabilities to gain unauthorized computer access. In today's information age, data has become increasingly important and is now considered one of our most crucial assets.

Attackers use various methods to gain unauthorized access to computer systems. These methods pose a significant threat to individuals and organizations. Before launching an attack, reconnaissance tools are often utilized to gain valuable information about the target systems. Port scanning is a popular reconnaissance technique attackers use to help identify open network ports, and running services on the target systems. This information can allow the attackers to identify potential vulnerabilities, which can be exploited to gain access to the target.

From a defensive perspective, it is crucial to thoroughly understand the available connections on a computer system to prevent such attacks. This can be achieved by deploying an authorized port scanner, which can aid in identifying potential entry points that an infiltrator might utilize. Medium to large organizations often conduct network scans to identify and address potential internal security risks before attackers can exploit them.

Utilizing a vulnerability scanner for research purposes can provide a valuable understanding of the security status of a computer system or network. Possessing such information allows a security researcher to locate potentially vulnerable machines quickly. By identifying and assessing vulnerabilities, researchers can better understand potential attack vectors and develop effective countermeasures to prevent the attacks.

Developing a dedicated vulnerability scanner named the *UiO SecurityLab Robot* grants researchers comprehensive control over the scanning process. By constructing the scanner internally, we maintain absolute control over every aspect of the scanning process, including the types of scans performed, frequency, and detail level. This approach will allow for control over the collected data, guaranteeing its reliability and authenticity. In addition, by focusing on particular sections of the Internet, we can increase the intensity of our scans and allow for a more comprehensive awareness of the security flaws in those regions.

## 1.2 Objective

The objective of this master thesis is to research and analyze the existing security robot solutions available on the Internet, investigate their technical gathering methods and functionalities, and use this knowledge to develop a demo version of the UiO SecurityLab Robot, capable of scanning specific parts of the Internet. Achieving the objective of this thesis will contribute to the network security scanning field by providing insights into the obstacles faced when developing a large-scale scanning service. The purpose of this study is not to perform exploits on public machines but to allow researchers to gather data to determine the number of devices that are vulnerable to different types of vulnerabilities.

### 1.2.1 Research questions

The following research questions have been specified for this study.

- **Q.1** How many devices in Norway are accessible via the Internet?
- **Q.2** How can we determine the vulnerability status of internet-facing devices in Norway, considering their exposure to different types of vulnerabilities, and how does this compare to available solutions?
- **Q.3** What are the obstacles in creating a platform that can perform vulnerability scans on specific infrastructure targets and store the obtained data in a format that can be retrieved easily?

## 1.3 Thesis structure

Provided is a short overview of the thesis. The thesis totals ten chapters.

### 2 Background

This chapter covers previous research and experiments related to the topic of vulnerability identification.

### 3 Assessing the Landscape of Vulnerability Scanning Software

Introduces available solutions for service and vulnerability scanning and their capabilities.

### 4 Fundamentals of Internet Scanning

This chapter covers relevant concepts and terminologies necessary to understand when tasked with creating a large-scale vulnerability scanner. The chapter also introduces techniques used by some popular tools to detect services.

## **5 Identifying Vulnerabilities in computer systems**

Describing different types of vulnerabilities and the ways of classifying them. The chapter also describes a method of identifying vulnerabilities using a non-intrusive technique.

## **6 Evaluating and Benchmarking of port scanning tools**

Here, different tools and parameters are compared and tested, including a small-scale scan, to assess the most commonly open TCP port.

## **7 Implementation and Development**

Describing the tools and configuration implemented to make the UiO SecurityLab Robot.

## **8 Results**

Presenting the results gathered using the previously implemented tool.

## **9 Discussion**

Comparing the results gathered against already available solutions. Further, this chapter discusses the limitations of the project and recommends further work for the project. Finally, a section about the legal implications of the techniques utilized in the thesis is also discussed.

## **10 Conclusion**

Assessing the research questions against the provided results

# Chapter 2

## Background

In this chapter, previous research and experiments related to the topic of vulnerability identification are discussed. Several papers are listed, including a summary.

### 2.1 Related work

Two master theses at the University of Oslo investigate vulnerability detection of online devices. The first thesis, "Development of a large-scale web scanner for detecting vulnerabilities" by Torjus Dahle in 2020, involves the creation of a tool to scan the most commonly used websites for known OWASP top 10 vulnerabilities. [3] The second thesis, "An extendable internet security scanner and Analyzer" by Kristian Helgesen Torkveen, examines website security by assessing their use of secure protocols and techniques. [4]

Both theses concentrate on globally popular websites to evaluate their security, reflecting the importance of assessing the security of these websites, given their broad reach and usage. As a result, these findings provide valuable insights into the vulnerabilities on commonly used websites and highlight the importance of implementing robust security measures to mitigate user risks.

### 2.2 Literature review

In the preliminary research on internet-wide scanning, a search was conducted for port and vulnerability scanning topics and reviewed relevant literature on different widely known tools used for network scanning. The techniques presented in the research can be broadly categorized into discovery techniques, results, and the legality and ethical considerations of scanning the Internet. The literature review provides valuable insights into the different approaches and challenges associated with internet-wide scanning, including identifying vulnerable devices and the need for ethical and legal considerations.



In the paper "Large scale port scanning through Tor using parallel Nmap scans to scan large portions of the IPv4 range", researchers from the University of Arizona discuss a method for performing anonymous port scans by using the TOR network to target specific areas of interest in the IPv4 range and then scanning those areas anonymously with parallelized scanners. The results demonstrate that this approach is feasible and effective for collecting internet scan data anonymously. [5]

"The Design of Large Scale IP Address and Port Scanning Tool" is an article from the Institute of Modern Physics, Chinese Academy of Sciences in China. The report provides a detailed comparison of port scanning results using different parameters of Nmap. It also compares their created scanning tool with Nmap for speed and against the Shodan database for open port detection. The result is a tool capable of performing a higher scanning rate than the Nmap tool. [6]

The article "Who is scanning the Internet" by Roman Trapickin from the Department of Informatics at the Technical University of Munich introduces the publicly available tools for internet-wide scans and answers the question of who is actively scanning the Internet for open ports. The article also discusses the legality and ethical aspects of scanning the Internet. [7]

"A review of network vulnerabilities scanning tools: types, capabilities and functioning" is an article by Andrea Tundis, Wojciech Mazurczyk, and Max Mühlhäuser. The study investigates some available Internet-wide scanning tools, highlighting their main objectives, application context, and distinguishing features. [8]

The article "An Intelligent Improvement of Internet-Wide Scan Engine for Fast Discovery of Vulnerable IoT Devices" by Hwankuk Kim, Taeun Kim, and Daeil Jang from the Korea Internet & Security Agency aims to perform Internet-wide scanning to detect vulnerable IoT devices. This paper proposes an intelligent internet-wide scan model that employs advanced IP randomization, reactive protocol (port) scanning, and OS fingerprinting scanning, applying the  $k^*$  algorithm to locate vulnerable IoT devices. The proposed model demonstrates improved performance compared to existing internet-wide scanning solutions, such as ZMap and Shodan. [9]

The article "Identifying Devices across the IPv4 address space" by Ryan Jicha, Mark W Patton, and Hsinchun Chen at the University of Arizona centers around the comparative analysis of connection-oriented and connectionless scanners. The findings of this study indicate that a connectionless scanner, such as Masscan, can perform scans at a rate exceeding 200 times that of a connection-oriented scanner like Nmap. [10]

The web blog article "Finding the Balance Between Speed & Accuracy During an Internet-wide Port Scanning" posted by the user "CaptMeelo" has conducted various tests involving simultaneous usage of multiple in-

stances of Masscan and Nmap, which can potentially accelerate scanning speed, but may also decrease detection rate. Based on these tests, the blog suggests using the Masscan tool to identify open ports before conducting a comprehensive scan with Nmap is a useful technique. [11]

The article "Improving Accuracy of Applications Fingerprinting on Local Networks Using NMAP-AMAP-ETTERCAP as a Hybrid Framework" was authored by Waheed Ali H. M. Ghanem and Bahari Belaton, affiliated with Universiti Sains Malaysia, explores the subject of service detection on remote hosts using different fingerprinting techniques. The study assesses the efficiency of various methods to detect running services and applications. The authors also propose a hybrid solution that combines active and passive fingerprinting techniques. [12]

The Carna botnet was a large-scale network of compromised devices used to conduct a comprehensive internet survey. A researcher created the botnet to map out the entire Internet and gather information on the number of services and devices online. The researcher developed a tool to brute-force attacks on weak passwords towards SSH servers. It would then distribute the network mapping tool Nmap, and a list of addresses the infected machine will be configured to probe against. After scanning the entire IPv4 part of the Internet, the Carna botnet found 1.41 billion addresses in use. The researchers performed this scanning activity within half a year by infecting over 420,000 computers, each performing scanning activities. The data was transferred back to a central server. Although its origins are highly controversial, the Carna botnet has proved to be a valuable source of information for cybersecurity researchers, as it has provided insights into the structure and security of the Internet. The results gathered from the botnet are still relevant today, over ten years later. [13]

### **2.2.1 Literature review conclusion**

Looking at the papers mentioned above, we gain insights into the different techniques and challenges associated with internet-wide scanning. Several studies discussed using widely-known tools for network scanning, such as Nmap and Masscan, in which they compared the results obtained using different parameters. Additionally, a few articles focused on the ethical and legal implications of internet-wide scanning, highlighting the legality of different behavior and the potential risks associated with unauthorized scanning and information disclosure.

## Chapter 3

# Analysis of Vulnerability Scanners

In today's cybersecurity landscape, the detection of vulnerabilities in a network has become an essential component of maintaining a secure infrastructure. There exist numerous software solutions capable of identifying vulnerabilities in computer systems. These programs come in open-source and commercial offerings, providing organizations with options for selecting the best software for their needs. The topic of the proposed master thesis is to collect and analyze the existing examples of security robots, meaning tools designed to perform automated network scanning for port, service, and vulnerability detection to enhance system security. Their strengths and weaknesses will be discussed.

### 3.1 Open source solutions

The most popular open-source solutions, Nmap and OpenVAS, provide basic network scanning capabilities, which allow for the detection of open network ports on systems, vulnerabilities, and other information about the systems, like operative system detection.

Nmap is a popular open-source software tool widely used by network administrators, security professionals, and researchers to explore and audit network security. Nmap can perform port scanning, host discovery, and vulnerability detection towards hosts on a network. Nmap is highly modular and allows for customization using various scripts to perform more specialized scans. A more detailed look at Nmap is in Section 4.8 provides a more detailed look at Nmap.

OpenVAS (Open Vulnerability Assessment System) is also an open-source vulnerability scanner. The tool provides a user interface for scanning and detecting security issues in servers, software, and other network devices. OpenVAS utilizes a database of known vulnerabilities to identify and analyze possible security threats. Even with multiple advanced features, the OpenVAS tool is simple to use by utilizing the provided interface.

[14]

IVRE (*Instrument de veille sur les réseaux extérieurs*, or Dynamic Recon of Unknown Networks in English) is an open-source network reconnaissance framework designed for collecting, processing, and managing network scan data. The framework relies on information from open-source tools such as Nmap, Masscan, and others and provides a way of storing the gathered data in a database. IVRE provides a web-based interface for searching and visualizing the collected data, making it simple for service and vulnerability detection. [15]

### 3.2 Closed Source Alternatives

Commercial software solutions, such as Qualys, Rapid7, and Tenable.io Nessus, offer more advanced features, including compliance auditing and web application scanning. These proprietary tools are provided at an enterprise level, often requiring expensive license keys.

Nessus, developed by Tenable, is a commercial vulnerability scanner focused on in-depth local network scanning. Nessus is a popular commercial software tool that provides advanced vulnerability detection. Developed by Tenable, the vulnerability scanner focused on in-depth local network scanning. It performs network-based vulnerability scanning activities and a more thorough system configuration analysis. Nessus uses various techniques, such as exploit-based scanning and web authentication scanning and exploitation. The tool produces detailed reports on the identified vulnerabilities, including remediation suggestions. Nessus is licensed software and can be costly, making it less accessible than open-source tools such as Nmap. It is also not designed for scanning more extensive network ranges, e.g., a country. [16]

The Norwegian security agency, known as the National Security Authority (NSM), offers an automated vulnerability scanning service called Allvis NOR to public agencies and owners of critical infrastructure. Allvis NOR functions by performing port scanning to detect exposed services, followed by automated tests to identify the protocol and software in use. Subsequently, a test is launched against the specific identified software to detect vulnerabilities. Upon identifying any vulnerabilities, the signed-up agency is notified accordingly. [17]

### 3.3 Exploring Available Search Engine Solutions

In this section, we will examine search engines that offer access to previously conducted scans, enabling users to view the results of internet-wide information gathering without having to perform the scanning themselves. These search engines simplify the process of retrieving the desired network-gathered data.

### 3.3.1 Shodan

Shodan is a website that allows users to search for information about computers with open ports on the Internet. The website was launched in 2009, allowing users to do an advanced search for open source information (OSINT) about computers connected to the Internet. The service is intuitive to use and allows for information about open services without conducting a port scan on your own. The service indexes information from open internet ports from all over the Internet and lets users search through the scanner's findings. These findings range from web servers with default credentials, cameras exposed to the Internet, and industrial control systems, to possible vulnerable software. The search results often include detailed information on the location of the device. [18]

The Shodan search engine receives its information from multiple sources, such as crawling the web, receiving data feeds from third-party providers, and getting inputs from the Shodan community. Shodan is arguably the most known platform for accessing worldwide information about internet-exposed machines, providing insights into otherwise not easily discoverable devices. This tool's capabilities extend beyond vulnerability assessment and can serve various purposes. Notably, researchers have utilized Shodan to monitor malware's spread across the Internet and identify devices possibly being misused. [19]

The feature *Trends* display the growth and decrease of open ports and technology exposed to the Internet. With Trends, it is possible to see unique insights into the evolution of services over time. Further, it allows users to track and monitor specific property values, allowing for continuous updates for a given search query. Shodan's facet analysis and trend tracking are both features that the search engine provides. The facet analysis allows tracking specific property values, giving a bigger picture of the most commonly open port or active vulnerability, for example, the current amount of operational Cobalt Strike command and control servers.

Shodan *Vuln* is a feature for academic and professional users. The features allow users to search for known device vulnerabilities, not just software information. The search engine will append a Common Vulnerabilities and Exposures (CVE) identifier to the detected vulnerable machine. This identifier calculates the number of vulnerable devices exposed on the internet. The vulnerabilities that have been identified, although not verified by Shodan, can not be confirmed as an accurate indicator of a device's vulnerability status.

These features mentioned above make Shodan a feature-rich and overall solution for analyzing the behavior of internet-connected devices. While Shodan can be a valuable tool for vulnerability assessment and other research purposes, the tools could also be utilized by adversaries seeking potential targets for cyber-attacks. This is why it is important that users

follow ethical guidelines to prevent any misuse of the tool.

In conclusion, Shodan is a powerful search engine that provides insights into details about internet-connected devices. Given the always-expanding internet, the significance of Shodan's role in monitoring internet-connected devices is becoming even more important.

### **3.3.2 ZoomEye**

ZoomEye, created by Knownsec, is a search engine designed to assist security professionals, researchers, and enthusiasts obtain information about internet-connected devices and web applications [20]. This simplifies the identification of potential vulnerabilities and security risks. Knownsec was founded in 2007 and is based in Beijing, China. Like the Shodan search engine, Zoomeye was developed with the Chinese market in mind.

The search engine continuously scans the internet and collects data about hosts, devices, and other web-facing applications. The collected data includes open ports, banner information, and software configuration, among additional information. ZoomEye indexes and makes the information from their scanners easily searchable and accessible for users.

ZoomEye offers a web-based interface and an Application Programming Interface (API) for users to access the gathered data. Users can search using parameters such as IP address, domain, hostname, software, and services. This enables the search engine to provide insight into the distribution of specific technologies, vulnerable systems, and potential attack surfaces. In addition to having search capabilities, ZoomEye provides aggregation analysis, which helps users identify patterns and correlations in the cybersecurity landscape. By analyzing this aggregated data, the users can better understand the overall security aspect of various technologies and systems. This information can be used to anticipate potential threats, prioritize security efforts, and develop effective mitigation strategies.

By default, ZoomEye does not retest the results it gathers after a specific time, resulting in a potentially high number of devices displayed by the search engine service. Some of these devices may have been identified a long time ago and may no longer be online, affecting the results' accuracy. This limitation in retesting gathered results may impact the service's usefulness for certain applications and should be considered when interpreting the data retrieved from ZoomEye.

The search engine functionality is primarily designed to explore internet-connected devices and web services. Therefore, their mapping towards vulnerabilities that apply to internet-connected devices is not as accurate and up-to-date as the Shodan search engine.

### 3.3.3 Censys

Censys Search is a search engine, intended to provide data about the various devices connected to the Internet [21]. This project emerged from the academic research efforts at the University of Michigan in 2013, wherein the Zmap tool was developed as a part of the initiative. The Censys Search engine utilizes the open-source software tools Zmap and Zgrab to gather its data. These tools function similarly to other central tools in the field, such as Nmap and Masscan.

The function of Censys search is to display gathered data about devices, specifically focusing on service configuration data. This is stored in a large search engine, making it easily accessible for research purposes. The regular scanning of the Internet forms a significant part of this search engine's operations, ensuring that the data remains updated and relevant. Censys provides services similar to those offered by the Shodan search engine, however with fewer features for advanced searches.

Censys Search also allows interested individuals to request access to its comprehensive dataset [22]. This dataset is built up by scanning the IPv4 address space, known IPv6 addresses towards more than 3,500 ports, and about 100 protocols.

## Chapter 4

# Fundamentals of Internet Scanning

Detecting vulnerable devices in computer networks is critical for ensuring their security. However, before diving into the details of these techniques, it is essential to understand some fundamental concepts in networking and network security. This chapter will discuss topics such as the TCP protocol, banner grabbing, firewalls, and other essential aspects necessary to understand when creating a large-scale vulnerability scanner.

### 4.1 IP Addresses

An IP (Internet Protocol) address is an identifying number assigned to every device connected to a computer network. The addresses serve as unique identifiers for the devices, allowing them to communicate with each other by transmitting data packets. IP addresses can be either set as either public or private. Public IP addresses are assigned by Internet Service Providers (ISP) and allow for connection between devices throughout the entire Internet. A private IP address is only accessible within a local network. Currently, two types of IP addresses exist IPv4 and IPv6. IPv4 is assigned 32-bit numbers represented in a dotted decimal notation; for IPv6, the number of bits assigned is 128-bit. This makes the total amount of available IPv4 addresses  $2^{32} = 4,294,967,296$ , which may seem large; however, it quickly became apparent that this was insufficient to meet the growing demand for IP-connected devices. This led to the creation of IPv6 addresses, which allows for a virtually limitless number of unique addresses.

In Norway, about 16 million IPv4 addresses are allocated according to ipinfo.io.'s list of IP ranges. [23] The website provides a list of IP addresses in the CIDR format, which includes all the subnets and Autonomous System Numbers (ASNs) allocated to Norway. ASN (Autonomous System Numbers) is an extensive network of IP addresses controlled by a company or organization. An instance of this is Telenor, the leading network provided in Norway, which controls the autonomous system number AS2119, incorporating nearly 7 million IP addresses.



## 4.2 TCP Protocol

The Internet Protocol (IP) is an essential component of network communication. Its primary function is transporting packets from a sender to a receiver device. Once a package arrives at its destination, a different protocol manages the transmission of data byte streams between the sender and receiver computers. The Transmission Control Protocol (TCP) takes charge of this task. TCP is a dependable and connection-oriented transport layer protocol that ensures the accurate and efficient data transmission across networks. To establish a data transfer, a three-way handshake process must occur between the sender and receiver computers. This process involves three steps.

1. The client sends an SYN (synchronize) packet to the server, signaling its intention to establish a connection.
2. The server will then respond with an SYN-ACK (synchronize-acknowledge) message, responding to the initial SYN packet. This indicates that the server is open to establish a connection with the client.
3. After the client receives the SYN-ACK packet, it sends an ACK (acknowledge) message in response.

Once the connection is established, data can be reliably sent between the two parts. For an illustrated look at the three-way handshake, look at Figure 4.2. TCP uses different flags to manage data transmission between the two devices. The SYN (synchronize) flag is used to request an initiate connection. The ACK (acknowledge) flag is sent to confirm the arrival of a packet. Lastly, the RST (reset) flag is used to terminate a data connection. These messages are important to understand when we later discover different port scanning techniques. They are explained in more detail, and practically in section 4.8.

## 4.3 UDP

Similar to TCP, UDP (User Datagram Protocol) is also a communication protocol used for the transmission of data over a network. However, unlike TCP, it does not provide the same reliability, making it fast but more prone to data loss. The UDP protocol is more suitable for speed over reliability, where packet loss is not as critical.

Although UDP is an important protocol on the Internet, it is not the focus of this master thesis. The primary reason for this is that detecting UDP services can be challenging. In addition, UDP does not have the same connection-oriented characteristics as TCP, making it harder to detect open ports and services. Nonetheless, it is important to acknowledge the existence of UDP and its relevance to certain types of applications and services.

## 4.4 Network ports

Computer ports function as communication endpoints, enabling interaction between devices. With ports, a computer can establish multiple connections simultaneously between different devices. An IP address is allocated 65,535 ports, utilized for both TCP and UDP connections. This number of available ports is constrained by a 16-bit unsigned integer. Linked to a port number is the source IP address and also the type of transport layer protocol used.

Port binding is utilized by computers to enable communication between different processes and services across a network. This involves associating a specific port number with a particular process or service, and listening for incoming network connection requests. Port binding ensures that only that process or service can receive incoming network traffic on the specific port.

To be able to listen to incoming traffic, ports utilize sockets to establish network connections and exchange data. The operating system directs incoming network packets to the relevant process by matching the packet's IP address and port number with a corresponding socket. When port binding, only a particular process can receive incoming network traffic on the specific port. When multiple programs attempt to utilize the same port number, IP address, and protocol, common application failures or port conflicts can occur. These conflicts arise due to the overlap in resource allocation, causing issues in the proper functioning of the applications involved.

The activity and connectivity expected from ports result in three distinct states. The first state is the "open" state, indicating that a particular port is actively listening for incoming connections, and is available to receive data packets. This indicates that a service is most likely bound to the port. The second state is "closed", indicating that the particular port is not actively listening for incoming connections. This port state is typically associated with systems which not require any network traffic to be routed toward the device.

The third and final state a port can have is the "undetermined" state, also commonly known as the filtered state. This state indicates that a firewall or network intrusion prevention system mechanism is blocking access to the port, which prevents incoming traffic from being delivered to the process or service. This measure is in place to protect systems and services from unauthorized access, or incoming attacks. In some cases, ports also may be intentionally filtered to avoid detection by port scanners. Devices within the same network can still communicate with filtered ports as the filtering is directed towards external sources only.

#### 4.4.1 Network Ports in the Context of Firewall and NAT

While filtering ports may improve security, it could also hinder legitimate communication between devices within the same network. Network Address Translation (NAT) is a technology used for mapping private IP addresses to a single public IP address, enabling multiple devices on a local network able to communicate with the internet from one IP address. Without opening a port on the firewall, all incoming traffic from the internet will be blocked. If a user wants to make a web server accessible from the Internet, they need to open the corresponding port bound to the web server service on their firewall and map the port opening to the internal IP that is listening for incoming requests. Common vulnerability scanners such as Nmap, Nessus, and Shodan are only able to look for services that are accessible from the internet, and not behind a firewall. Opening a port on the firewall poses a potential security risk by allowing attackers and scanners to access internal network devices. This risk is important to take into account.

An example of the consequences is the data breach incident at Norkart in May 2022. [24] Norkart is a company that specializes in providing digital mapping and geographic information system (GIS) solutions for both public and private sector clients. They informed about a security breach 10th of May, where unidentified attackers gained entry to an unsecured Application Programming Interface (API), where a single port was exposed to the internet. [25] The attackers found this single open port using a port scanning tool, which allowed them to retrieve a copy of Norway's official property register. This register provides an overview of who owns what in Norway, including names, addresses, and dates of birth of over 3,3 million Norwegian citizens. With the installation of a firewall, it is likely that the occurrence of such incidents would have been averted as the firewall could have prohibited access to the port from IP addresses outside the local network.

#### 4.4.2 Port ranges

As previously known, the port range exists of 65,535 possible ports. This range is divided into three categories: well-known ports, registered ports, and dynamic or private ports. The ports ranging from 0 to 1023 is classified by the Internet Assigned Numbers Authority (IANA) as well-known ports. This range contains some of the TCP and UDP ports that are most commonly used by server applications. For instance, web servers commonly use ports such as 80 and 443, and Secure Shell (SSH) is commonly used on port 22.

Ports from the range 1024 to 49151 are registered ports. Vendors can apply to IANA for the assignment of specific services registered to a port. The ports within this range are often assigned to specific services by the operating system or other software.

Ports assigned the number above 49151 are known as dynamic or private ports and can be used by any application. This range is also used for ephemeral ports. Ephemeral ports are temporary ports that are used for only a short period of time, within the time frame of the data session. They are managed by the operating system and allow for multiple connected clients to establish connections with a server simultaneously. Once a connection is established, the ephemeral port is used to send and receive data between the client and the server. With this port, a client is then able to send and receive data using the specified port. [26]

## 4.5 In-depth Port Scanning: Which Ports to Prioritize

Port scanning is a widely used reconnaissance technique used in computer networks to identify open ports on target systems. The process involves systematically iterating through a range of port numbers toward one or multiple hosts to determine which ports are open, closed, or filtered. The technique can be used for various purposes, such as network security assessments, vulnerability scanning, and penetration testing. However, it can also be used by a malicious adversary when scoping out a potential target. This is why port scanning should be conducted with caution and with proper authorization and in compliance with legal and ethical guidelines to avoid any legal consequences.

We often differentiate network scans into two types: Service sweep and regular port scanning. Even though both techniques aim to discover information about a system or network, there are important differences between the two. The service sweep technique is used to identify a particular service running on a system, or network without having to iterate through all possible ports. One illustration of this concept is when there is a need to identify all servers within a network that have SSH (Secure Shell) enabled. In such a scenario, a straightforward approach is to execute a service sweep, which involves sending requests to each machine within the network using the established port 22, which is commonly utilized by the SSH protocol. This differs from a port scan, which is a broader scan that attempts to identify all open ports on a target system or network, not targeting a specific software as in service sweeping.

When performing a port scan, it is important to be aware of the different categories of port ranges to determine which ports should be given priority. Conducting port scanning activities over a wide range of ports and hosts is a time-consuming process that also produces excessive network traffic. Consequently, it is preferable to constrain the number of scanned ports, focusing primarily on those that are more likely to yield a greater number of discovered services. As a result, it is uncommon for port scanning software to examine all 65,535 available ports, as this would generate a large amount of network traffic. The aspects of performance and network traffic will be

discussed in more detail in Chapter 6.

We can examine the known port ranges to determine which most likely contain responsive ports. The "Well-known" port range is utilized for widely recognized protocols and services that are commonly employed on both Linux and Windows systems. This range is an obvious option to scan, which is why the popular port scanning tool Nmap used to look towards these ports by default when a user did not specify a port range in the tools parameters. Running the tool on the port range 0-1023 would most likely detect a large amount of services, and only need to send around one thousand requests per host.

The registered ports range is the largest out of the three categories. About one-third of the ports are registered at IANA. [27]. Among the ports that are registered, it is evident that many of them correspond to outdated and less frequently used services and applications. This characteristic reduces the efficiency of scanning the entire registered port range.

When it comes to the ephemeral ports, they are not that important to scan because it often does not host any services, do not listen for incoming requests, and are temporarily used by clients. Applications however can be hosted within this port range as well, so a middle ground when port scanning needs to be selected.

In terms of the ephemeral ports, their importance in scanning is relatively low since they usually do not host any services, do not listen for incoming requests, and are only temporarily utilized by clients. Applications, however, can be configured to operate within this port range, necessitating a balanced approach when conducting port scanning.

The Nmap Project has conducted independent calculations regarding the required number of ports to be checked in order to assess the scanner's effectiveness in detecting open ports.

Effectiveness	TCP ports required	UDP ports required
10%	1	5
20%	2	12
30%	4	27
40%	6	135
50%	10	1,075
60%	18	2,618
70%	44	5,157
80%	122	7,981
85%	236	9,623
90%	576	11,307
95%	1,558	13,035
99%	3,328	15,094
100%	65,536	65,536

Table 4.1: TCP and UDP ports required for different effectiveness rates [28].

This table provides an analysis of the relationship between the different levels of effectiveness and the number of TCP and UDP ports when performing network scanning. The data presented in the table indicates that as the desired level of effectiveness rises, there is a corresponding increase in the number of ports that need to be scanned. The effectiveness levels range from 10 % to 100%.

The table reveals interesting findings demonstrating how scanning just a specific set of 10 TCP ports can enable the discovery of 50 percent of all running services. Also when scanning a selected 576 ports, which is only 0.878% of the total available 65535 ports, one can expect to discover 90 percent of services and applications. The data demonstrate a significant enhancement in efficiency by sending requests to ports that frequently respond. In subsequent sections of the thesis, the specific common ports will be identified and discussed.

It is important to mention that the data has remained unchanged for an extended period. Originally published in Gordon "Fyodor" Lyon's 2009 book, "Nmap Network Scanning" [29], the information continues to be hosted on the Nmap.org website without any updates. The popularity of various services has undoubtedly evolved since 2009, requiring an updated table to utilize the provided data in a more scientific approach.

Looking at the specified ports some of the available search engine solutions prioritize, it is hard to find a list or a definite answer, since their solutions are proprietary, and closed source. Compared to the search engine Shodan, their service computers reportedly crawl toward a list of 1225 ports, as of May 2020 [30]. This number is only 1.87 percent of all the possible ports, however looking at the effectiveness table when focusing on TCP, the Shodan search engine would then be able to identify anywhere from 90 to 95 percent of all the services.

## 4.6 Detecting Software using Banner Grabbing

Now that we have a better understanding of network ports, we can go into more detail on how we can use port information to detect software and services on computer networks.

A banner is a piece of information that a service transmits to the connecting client upon establishing a connection between two ports. The banner includes information such as hostname, protocols, software, and software version. Additionally, the banner may indicate the status code of the request, such as 404 for "not found" or 200 for "successful." Essentially, the banner serves as metadata that provides insight into the service associated with a TCP or UDP port.

```
[root@prober]# nc www.targethost.com 80
HEAD / HTTP/1.1

HTTP/1.1 200 OK
Date: Mon, 11 May 2009 22:10:40 EST
Server: Apache/2.0.46 (Unix) (Red Hat/Linux)
Last-Modified: Thu, 16 Apr 2009 11:20:14 PST
ETag: "1986-69b-123a4bc6"
Accept-Ranges: bytes
Content-Length: 1110
Connection: close
Content-Type: text/html
```

Figure 4.1: The example banner displays the status for an Apache web server running on Linux Red Hat [31].

The banner grabbing technique can be utilized for fingerprinting, which is the process of identifying the operating system, software, and other characteristics of a system or network. Fingerprinting is an important step in reconnaissance, as it helps identify software, and possible vulnerabilities linked to the software. The banner often reveals information about what version of the software assigned to the port is running, which allows us to look for outdated software that is affected by known vulnerabilities. The tool Nmap allows users to perform banner grabbing on open ports. The tool will then connect to an open port and store all traffic sent by the listing service within a specified amount of seconds. The Nmap tool is described in more detail in Section 4.8, where we also look at the features of detecting vulnerable devices.

Netcat is a command line utility that is used for performing networking tasks to interact with network ports. It is commonly used to obtain a banner by establishing a connection with the target server on a desired port and afterward requesting the port's banner information. Netcat is a command line tool frequently employed for executing networking operations

and interacting with network ports. One of its features allows for requesting a banner by establishing a connection with the intended server on a specified port and subsequently requesting the banner information associated with that port. Banners are typically in the form of a byte stream, and the request can vary depending on the specific service or protocol being utilized. To obtain the banner, Netcat must wait for a response from the target to ensure that the entire banner is received. The amount of time it takes to receive a response can vary depending on a number of factors, such as network latency, distance, the load on the target host, and the specific service or protocol being used. In cases where no response is received, Netcat will terminate the connection by terminating its ephemeral port. This is a practical feature as it eliminates indefinite waiting periods for non-responsive hosts. Waiting insufficient duration pose the risks of only partial capture of the banner or not receiving it at all. On the other hand, waiting too long can overcrowd the scanning process, causing delays in performance when performing banner grabbing across multiple ports and devices.

Although Netcat is capable of performing banner grabbing, Nmap's dedicated tool, Ncat, provides numerous advantages compared to its predecessor. The Ncat tool is created to extend the Netcat tool, providing additional features that make it more flexible and efficient in performing banner grabbing. Ncat extends the waiting time required for banner grabbing, by adjusting the waiting time itself during a banner-grabbing scan. The tool will learn from the response time of the previously scanned ports, and adjust the waiting time accordingly. It provides the ability to automate banner grabbing tasks, enabling the user to perform banner grabbing on multiple servers or ports simultaneously.

## **4.7 Connection-Oriented, and Connectionless Scanning Techniques**

When talking about port scanners, there are two main different types that perform in slightly different ways. A connection-oriented scanner is a type of network port scanner used to identify ports on a target. This scanning technique establishes a full TCP connection with the target and sends packets similar to a real connection's behavior. This way of scanning allows the scanner to identify any open ports and their associated services accurately. Additionally, a connection-oriented scanner will bypass certain firewalls designed to block network packets that do not behave like genuine TCP traffic. This type of activity is how the popular port scanning tool Nmap operates by default.

On the other hand, a connectionless scanner is a port scanner that identifies open ports on targets without establishing a full TCP connection. Instead of establishing a TCP connection as the Connection-Oriented scanner, it sends packets designed to obtain a response from the target system, allowing it to identify open ports. Connectionless scanners have the ability



to operate faster than connection-oriented scanners since the entire TCP connection does not need to be established. However, they can also be less accurate, as the reliability of distinguishing between an open and closed port with the same level of certainty as a full TCP connection. Connectionless scanners operate in an asynchronous manner, sending packets independently and without the need for establishing and maintaining TCP connections or waiting for responses before proceeding to the next packet or target.

However, connectionless scanning comes with a trade-off in accuracy as it relies on the target system to respond within a short time frame to the sent packet. The send packages are also more commonly blocked by receiving firewalls, which results in the scanning type being less accurate and reliable in detecting open ports. The scenarios suitable for a connectionless scanner would be more large-scale scans of large networks or the Internet, where speed is more important than accuracy. A popular tool that is implemented using this technique is called Masscan, which is described in greater detail in section 4.9

## 4.8 Nmap - Network Mapper

Throughout this thesis, the tool Nmap has been referenced multiple times in various contexts, highlighting its significance in network scanning and reconnaissance abilities. Section 3.1 provides a brief overview of the tool; however, a more comprehensive look at Nmap capabilities and fundamentals is required to recognize its abilities in network reconnaissance. Therefore, this section aims to provide a more detailed exploration of Nmap, including its features and capabilities.

Nmap is a commonly employed tool utilized by security professionals to examine network systems for accessible ports. The tool is short for "*Network Mapper*", and is free and open-source and was first released in 1997 by Gordon Lyon. Nmap can be used to perform network-wide scans of hosts and services and offers various advanced features, such as OS detection, version detection, and stealth scanning. The features can be extended by writing supported Nmap Scripting Engine (NSE) scripts in the Lua programming language. These scripts automate the Nmap tool to search for particular patterns or data. Nmap has the capability to perform both connection-oriented scanning and connectionless scanning methods.

Nmap supports a wide variety of scanning techniques, including the following:

- **TCP scan:** The default option for Nmap. The scan works by sending an SYN packet to the target system and waiting for an SYN-ACK response. If the target responds with SYN-ACK, then to complete the three-way handshake, Nmap will send an ACK packet. Once the TCP connection has been established, Nmap sends an RST packet to terminate the connection and registers the port as open. Allowing for a full connection with each port adds additional time, making it slower than most other scanning techniques. However, it is the most reliable way of accurately determining a port's status.
- **ACK scan:** Sending a packet with only the ACK flag set is a distinct scanning technique that differs from the methods mentioned earlier. This technique is used to map out if the target is protected by a firewall. When scanning a system not protected by a firewall, the targeted host will answer with an RST packet if the port is open or closed. If the device does not respond, we can assume the host is behind a firewall.
- **SYN / Stealth scan :** Also known as the "half-open" scan. This scanning technique aims to hide its activity from being detected by firewalls. The client sends an SYN packet, and if open, the server will answer with an SYN-ACK packet. This scanning technique differs from the standard TCP SYN scan in that, immediately after receiving the SYN packet from the host, the client promptly responds with an RST (reset) packet directed back to the host. By doing so, the TCP handshake is effectively halted, preventing the establishment of a complete connection. This approach limits the number of detected connections that a host may initiate. SYN scanning typically involves creating and using raw sockets, which on Linux require sudo privileges.
- **UDP scan:** The abovementioned techniques are used for detecting TCP ports. To detect UDP ports, Nmap, unlike TCP, does not establish a connection or return an ACK flag. Instead, Nmap uses the characteristic of UDP to determine if a port is open or closed by sending a UDP packet to the target port and awaiting a response. If an ICMP "Destination Unreachable" message is received, it signifies that the port is closed. On the other hand, if no response is received, the port is considered open.
- **ICMP:** The ICMP, also called ping scan, is a technique used to check if a target is alive and reachable. Nmap will then send an ICMP Echo Request packet, commonly called a ping, to the target. It will then wait for an Echo Reply. If the device does not answer the ping request, Nmap assumes that the host is offline and will not respond on any port scan. ICMP does not use TCP or UDP and is not tied to ports. This technique is explained in greater detail in Section 5.1.1.

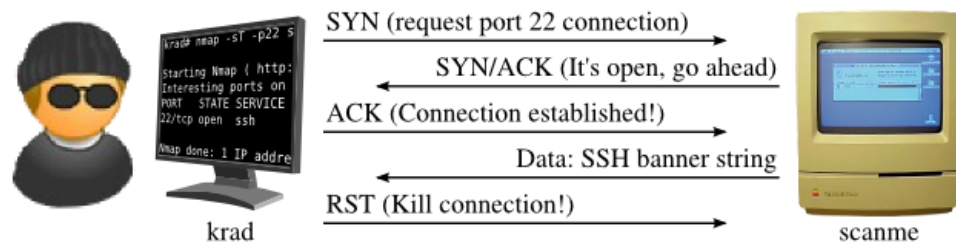


Figure 4.2: TLS 3-way handshake and retrieval of banner information [32]

Figure 4.2 displays a client named "*krad*" performing banner grabbing activities towards the server "*scanme*". The figure shows which TCP flags are used to establish and kill the connection.

When selecting an appropriate approach for network scanning, it is important to be familiar with the various scanning techniques available. Different techniques are suited for different reconnaissance purposes, with some prioritizing being undetected and others prioritizing speed. Given all the possible scanning techniques available, selecting a suitable technique for the UiO SecurityLab robot is important. The discussion of what approach the SecurityLab will adopt is explored in detail in Section 7.2 .

As mentioned, Nmap supports different scripts using the Nmap Scripting Engine (NSE). It allows users to write or reuse scripts written in Lua programming language to automate various tasks. The NSE scripts are organized into categories that separate the scripts by how they interfere with the targeted device. Some scripts are marked as "intrusive", a categorization of scripts that can use significant resources or are likely to crash the system or service. Scripts within the category "exploit" are scripts that actively attempt to exploit some vulnerability in a system [33]. Utilizing these types of scripts may not be considered legally acceptable towards devices not owned by oneself.

The NSE scripts can greatly enhance the functionality of Nmap by integrating with various external tools and accessing additional data sources. Additionally, the NSE provides a secure and flexible platform for writing custom scripts and sharing them with the community. As a result, the NSE has become an integral part of the Nmap toolkit and has significantly expanded its capabilities beyond simple network mapping. For example, NSE was used in the Carna botnet to brute force default telnet login usernames and passwords toward thousands of hosts.

```

local socket = require("socket")

function action(host, port)
    local s = socket.tcp()
    s:settimeout(1)
    local success = s:connect(host, port)
    if success then
        s:send("GET / HTTP/1.0\r\n\r\n")
        local banner = s:receive()
        print(host .. ":" .. port .. " banner: " .. banner)
    end
    s:close()
end

function run(host, port)
    for p = 1, 65535 do action(host, port) end
end

```

Listing 1: Example of a Nmap NSE script for banner grabbing

The code example in Listing 1 represents a Nmap script implemented in the Lua programming language. The script aims to establish a connection towards all possible ports using sockets. First, a new TCP socket is created with the timeout set to 1 second. The script then attempts to connect to the designated host and port. If the connection attempt is successful within the specified time limit, an HTTP request is sent to the server to retrieve the server's banner as a response. This request will repeat for every specified port, which is all possible 65,535. The results of this script are subsequently printed out as an output. The provided script in Listing 1 offers a basic example of what is possible with Nmap scripting. More advanced scripts can automate and look for specific features conveniently. Furthermore, it allows integration with third-party libraries and tools, further extending its capabilities.

#### 4.8.1 The Nmap-services File: Mapping Port Numbers to Known Services and Protocols

The Nmap-services file is a registry connecting port numbers with their corresponding known services and protocols. The file acts as a lookup table, with mapped descriptions and popularity for most available ports. Initially, the file was based on the IANA assigned ports list (0-1023), discussed in Section 4.4.2. Over time, the registry file has undergone multiple expansions to encompass a broader range of services for identification purposes. Additionally, these expansions have enabled the identification and detection of port numbers associated with potentially malicious activities, such as trojans, worms, and other attacks indicating a compromised device.

The file includes a frequency number, which indicates in percentage

how often the port is discovered as open. This frequency is utilized when users only want to iterate over the most commonly open ports.

After some research and investigation of the frequency percentages, it is apparent that a significant number of the top most listed ports and associated services are in less use today. Ports related to non-secure HTTP traffic, printer ports, Telnet remote access, and Microsoft Netbios are listed among the top 10, all services that have declined in popularity since 2023. Given the mentioned findings, an investigation was conducted into how the frequency data was collected and the specific time frame for the data gathering research. It appears that the data was collected through a large-scale internet scan conducted during the summer of 2008 by Nmap's founder "Fyodor" [34].

The utilization of the service file to iterate through the most common ports has become less effective compared to its effectiveness at the time of data collection in 2008. Over the span of 15 years, the adoption of new protocols has gained population, which is now listed with lower frequency in Nmap's services file. Consequently, the frequency percentages reflected in this file are regarded as outdated and may lead to inaccurate results. Therefore, relying entirely on this outdated file for port analysis would result in an improper assessment of current network configurations and service usage. A comparison is conducted in Section 8.1, where newly gathered results are compared to the ones listed in the Nmap-services file.

## 4.8.2 Optimizing Nmap

We previously discussed in Section 4.6 why having as low as possible run time is essential for not causing delays and overcrowding the scanning process. Therefore, Nmap includes several options that allow for optimizing the scanning time, enabling the adjustment of various parameters. During the testing phase of this master thesis, a considerable amount of time experimenting with various command parameters to enable and disable different techniques in Nmap. Some of these tests are explained in detail in Chapter 6.

A theoretical possibility would be to run multiple instances of jobs of Nmap simultaneously, dividing the target list evenly to each job; in the article "Finding the Balance Between Speed & Accuracy During an Internet-wide Port Scanning," Capt. Meelo researched optimizing large-scale port scanning by experimenting with executing both Nmap and Masscan in parallel jobs and with different host timeout parameters. [11]

The article's conclusion revealed that a high number of multiple parallel jobs containing Nmap tasks had a negative effect on the amount of discovered ports. Nmap has built-in parallelization capabilities that already enable the execution of numerous concurrent scanning processes of multiple targets. Therefore, executing Nmap in multiple parallel jobs will lead to resource conflict and performance issues, resulting in the

discovery of fewer ports. The topic of parallelism is also written about in the Nmap's webpage "Execute Concurrent Nmap Instances", [35] which concludes that running multiple parallel Nmap scans against a single target is inefficient and slow while dividing the scan into large groups and utilizing different hosts within the network improves speed and efficiency.

## 4.9 Masscan: A High-Performance Port Scanning Tool

Masscan is an open-source network scanning tool created by Robert Graham in 2013 [36]. The tool has the capability to identify open ports and services on a large number of target hosts at high performance. In this section, we will explore the essential features and capabilities of Masscan, highlighting its unique abilities.

The most notable feature of Masscan is its ability to perform port scanning at a high rate using the connectionless traffic technique. The Masscan tool was developed to run faster than Nmap, leveraging an asynchronous TCP scanning methodology. This approach involves sending TCP SYN packets and later analyzing the response received from the incoming packets. The tool does not promptly wait for an answer after sending out the SYN packet, which allows the tool to continue to send out SYN packages to the following hosts on the list. This technique is also known as Asynchronous I/O. It allows the program to start multiple SYN connections simultaneously without waiting for each previous connection to finish. If the target port is open, it will send back a TCP SYN-ACK packet, which indicates that the machine is willing to open a TCP connection. Upon receiving an SYN-ACK packet, Masscan assumes that the port is open without completing the three-way TCP handshake typically required to establish if the port is reachable. This eliminates waiting for the slow three-way TCP handshake. The program will then send a TCP RST packet toward the target immediately to close the connection.

Masscan implements a custom TCP/IP stack that facilitates the utilization of asynchronous I/O within the program. This custom stack also minimizes the overhead sent in the Ethernet frame header, which typically includes packet metadata. Using raw sockets, the operating system bypasses certain internal checks and enables the capture of all incoming traffic, effectively functioning as a packet sniffer. Consequently, Masscan can accurately associate the received SYN-ACK packets with the corresponding scanned host machines, thereby predicting their source.

Using these techniques, Masscan can send millions of packets per second, allowing for the rapid detection of services on a large scale. The tool has gained attention after claiming to be able to scan the whole internet within 5 minutes, using a 10GB internet connection, combined with a network card which allows for a protocol called PF\_RING [36].

Masscan supports various scanning options allowing for a high level of customization regarding rate, ports, and output. The tool also supports banner grabbing features that enable users to obtain information about detected running services. Section 6.7 compares the banner information obtained through Masscan and Nmap's corresponding data, examining the results' differences and similarities.

## **4.10 Rate limiting**

Rate limiting refers to the practice of restricting the number of actions a device or application can perform within a specific time period. Websites often limit the amount of GET and POST requests one IP address can do within a given timeframe. Network firewalls and applications may also limit the number of requests one machine can make to a given host. When scanning, a firewall can block the traffic from a port scanner after X requests within Y seconds. This is to prevent the scanner from being able to do reconnaissance on the host and to limit the amount of network traffic the server must handle.

Conducting rapid and extensive scanning activities is prone to triggering firewalls to impose restrictions on the source IP of the scanning device. For example, it is common for organizations and online services to enforce limits on the number of requests per second originating from a single source IP address. Consequently, this may lead to the denial of further requests and subsequently cause ports that could have been potentially accessible to appear as closed. Further, the source IP address of the scanner might be blocked entirely, thereby potentially influencing the results of a reconnaissance scan.

## Chapter 5

# Identifying Services and Vulnerabilities

In this chapter, we will discuss further the possible ways of detecting services and vulnerabilities on services connected to the Internet, including methods like TLS fingerprinting, favicon hashing, and screenshot analysis for service identification. Host discovery using ICMP ping is discussed, along with the limitations imposed by devices blocking ICMP requests. This chapter also covers software and network vulnerabilities, including buffer overflow, XSS, and SQL injection, and a way of identifying and scoring the identified vulnerabilities.

### 5.1 Uncovering Services: Alternative Techniques

The use of Nmap and other port scanning tools provides a straightforward and efficient approach to identifying services on target networks. It is although important to consider alternative techniques for detecting services. In this section, we will discuss various other methods which allow for service identification, going through TLS fingerprinting, favicon hashing, and screenshot analysis.

#### 5.1.1 Utilizing Ping for Host Discovery

The Internet Control Message Protocol (ICMP) is a network protocol utilized for error reporting and network diagnostics. This protocol can also be utilized for host and service discovery on network-connected devices. To perform this reconnaissance, a message called the ICMP Echo Request message, commonly known as a 'ping' is sent towards the target. If the target device responds with an ICMP Echo Reply message, then it can be assumed that the target is reachable. The ICMP messages are sent within the IP layer, meaning features from the transport layer, such as port information is not present.

Ping is often utilized by network administrators to identify and troubleshoot network-related issues. Additionally, the technique can also



be utilized at the beginning of a network reconnaissance, to quickly discover active and reachable network devices, without having to scan a wide range of ports against it. When identified, a port scan towards the devices that answered with an ICMP Echo Reply message can be launched. The port scanning tool Nmap implements this feature by default.

There are however downsides to implementing ping scanning, which makes the technique not always recommended. Many devices, such as Windows are set up by default to block incoming ICMP, so utilizing the above-mentioned method would classify the host as down, even though the host has active running services. Firewalls are also often configured to block incoming ICMP requests, further falsely identifying services as unreachable.

### 5.1.2 TLS Fingerprinting

TLS fingerprinting is a method used to identify the features of a TLS handshake. The technique can reveal information about the server's running software version and operating system. The TLS protocol is used for establishing a secure network communication session between a client and a server. In contrast, the TLS handshake is the process of how the securely created connection is established. TLS fingerprinting relies on the fact that there can be variations in the TLS handshake process due to the diverse implementations of the TLS protocol.

To achieve TLS fingerprinting, one must analyze the specific characteristics of the TLS handshake for the different operating systems and software versions. For instance, various versions of the same web server software may implement TLS differently, resulting in variations in the TLS handshake process. By analyzing these variations, one can determine the specific TLS implementation used.

Ncat, the Netcat-modified tool by Nmap provides the ability to detect operative system versions by sending a series of TCP/IP packets to the target system and analyzing the responding target's TLS fingerprints to determine the characteristics of the operating systems. It compares the response towards a database of known patterns and signatures associated with known various operative systems [37]. This analysis is known as OS fingerprinting.

An example TLS fingerprint is provided, written in the ja3 format. The TLS traffic is generated from a host infected with the Neutrino malware. [38]

```

{
"desc": "Malware Test FP: neutrino-traffic",
"ja3_hash": "fd6bbdf835788b3c7d33372127470a06",
"ja3_str": "769,57-56-53-22-19-10-51-50-47-7-5-4-21-18-9-20-17-8-6-3-255,,"
}

```

Listing 2: TLS fingerprint example for the Neutrino malware

Looking at the above-mentioned code listing displays *ja3\_hash* and *ja3\_str* string values. JA3 is a TLS fingerprinting technique that extracts a distinctive hash value during the establishment of a connection between a client and a server. This method helps in identifying and categorizing TLS client applications. Utilizing the *ja3\_hash* technique can assist in the tracking of Command and Control (C2) server, to detect the widespread of both infected devices and servers used for spreading various malware. This enables security analysts to identify and track the usage of specific TLS client applications and servers associated with C2 activities.

### 5.1.3 Favicon Hash

A favicon is a small icon present in the browser's page tab. The word favicon is short for "favorite icon", and is used to quickly help users identify web pages in the computer's browser. Favicon hashing is a technique that is used to identify web services on a target system, by analyzing the favicon sent to the client, when requesting the website. The technique requires computing a hash value of the favicon's small 16x16 pixel image and comparing the hash to a database of known favicon hashes. Table 4.2 presents an illustrative example showcasing various web applications along with their respective favicon hashes.

http.favicon.hash	Product/Application
81586312	Jenkins
743365239	Atlassian
2128230701	Chainpoint
-1277814690	LaCie
855273746	JIRA

Table 5.1: Favicon Hash Table [39]

To identify the hash, a simple Python script can be developed to retrieve the favicon image and generate a hash for the retrieved file. This script can be expanded by integrating it with a database containing known hashes, which allows for service detection. For a practical implementation of favicon hashing for websites, refer to Appendix 11.2 , which includes an example code demonstrating the process.

One limitation of using the favicon hashing technique for detecting services is its inability to provide information regarding the version of the detected service. While the method is efficient at detecting services, it may not always be reliable, in terms of not all websites possessing a favicon. Due to these limitations, the techniques were deemed beyond the scope of this thesis.

#### **5.1.4 Website Screenshots for Service Discovery**

Implementing a feature that allows for taking screenshots of webpages, can help to identify services hosted on machines. By visually displaying the target machines software, users could view a large gallery of screenshots, and conduct a more in-depth analysis of the services they find interesting. This feature is provided by the Shodan search engine, allowing users to view images from services such as websites, web cameras, and open dashboards for 3D printers [40].

The implementation of website screenshots can be achieved through the utilization of a Python script or by employing pre-existing tools such as EyeWitness. EyeWitness facilitates the capturing of screenshots for services that are detected within the output file of a Nmap scan [41].

This functionality may prove useful in situations where the Nmap tool fails to identify a service, even though it is evident from browsing the location that the server is a website containing a web camera feed, for instance.

## **5.2 An Overview of Common and High-Risk Vulnerability Types**

In this section of the master thesis, we will discuss the types of vulnerabilities to look for when designing a vulnerability scanner. There are several different types of vulnerabilities that exist in computer systems. One of the more common types is software vulnerabilities, which occur when programming errors or flaws in software design leave the system open to exploitation, which is when the program gives out information that is not intended. Another type is network vulnerabilities, which arise when weaknesses in network protocols, configuration, or devices allow unauthorized access or data interception. Having knowledge of the most dangerous vulnerabilities enables us to concentrate on addressing them as a primary priority.

Within software vulnerabilities exist multiple different types of specific vulnerabilities that pose a significant threat to computer systems. One such vulnerability is a buffer overflow, which occurs when an application or system attempts to store more data in a buffer than it was designed to hold, causing the excess data to spill over into the neighboring memory locations. This can result in system crashes, data corruption, and even unauthorized code execution. The vulnerability is ranked as one of the most common

software vulnerabilities categorized by the CWE. Common Weakness Enumeration (CWE) is a standardized method of identifying and classifying security vulnerabilities that may exist in software systems. The standard is developed and maintained by the MITRE Corporation [42]. It is created as a comprehensive reference for both developers and security professionals to better understand potential weaknesses and take appropriate countermeasures. Furthermore, it helps in the development of secure coding practices, vulnerability detection tools, and effective mitigation strategies, ultimately contributing to the overall improvement of software security.

Buffer overflows are identified as CWE-787 (Out-of-bounds Write), and according to the CWE list for 2022, buffer overflow vulnerabilities have a Common Weakness Scoring System (CWSS) score of 64.20 out of 100, making them the highest-ranked vulnerability on the list of the "2022 CWE Top 25 Most Dangerous Software Weaknesses" [43].

Another highly ranked CWE is CWE-79, referred to as "*Improper Neutralization of Input During Web Page Generation*", commonly known as 'Cross-site Scripting' (XSS). XSS vulnerabilities occur when an application includes data provided by users on a web page, without properly validating the input. This can allow attackers to execute malicious scripts within the context of the users' browsers. The vulnerability can lead to theft of sensitive data, session hijacking, and defacement of web content. This is why the score for the CWE-79, is set to 45.97 out of 100.

According to Miter, the third most dangerous software weakness is the Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'). SQL Injection vulnerabilities occur when an application allows for user-supplied data into SQL queries without proper sanitation or validation of the input. Attackers can exploit this weakness to manipulate the SQL queries, allowing for modification, deletion, and access to sensitive data stored in the underlying database.

Verifying attacks like SQL injection poses challenges as they require active system exploitation. However, when we know about the vulnerability of a particular version of a database service, it becomes possible to search for active instances of that specific version. This enables us to easily identify if the server is vulnerable to SQL injection.

These three mentioned critical vulnerabilities all share a common theme: insufficient input handling and validation. They all can have severe consequences, including unauthorized access to sensitive data, execution of malicious code, and system crashes. This is why vulnerabilities like these are important to detect, and patch to mitigate the exploitation of devices that occurs daily.

### 5.3 Vulnerability Detection with Nmap

As discussed in the previous chapter, Nmap is a versatile port-scanning tool, that allows additional features using the Nmap Scripting Engine (NSE). A resourceful NSE script named "*vulners*" allows for vulnerability detection through an analysis of the banner data distributed by the target service [44]. The script performs software and host version checks and compares the version with data stored in the Vulners database. This database collects and archives information about information security vulnerabilities in software and hardware computer systems [45]. The database is integrated with the "*vulners*" script using a simple API. If a correlation is found between the scanning results and the database, it suggests that the target system could potentially possess vulnerabilities related to the identified threat. This vulnerability lookup activity is automatically done by Nmap and a Vulners database API. Nmap will output the corresponding vulnerabilities in a list assigned to the host.

The identified vulnerabilities will be allocated a designated Common Vulnerabilities and Exposures (CVE) identifier. A CVE is a publicly disclosed computer-related vulnerability in software products. Each CVE is assigned to a uniquely identifiable number. An identified CVE contains a detailed description of the disclosed vulnerability, including the severity level, and the possible impact. It allows for simple tracing and sharing of vulnerabilities, enabling organizations to implement necessary adjustments to mitigate potential risks. The CVE system is maintained by the Mitre Corporation and is widely used in the security industry.

The *Vulners* NSE script is considered non-intrusive as it does not require exploding the target to detect any possible vulnerability. It stands out as one of the few present non-intrusive techniques that enable vulnerability assessment of devices at a large scale.

Further in this thesis, a benchmark test of the Vulners script is present in Section 6.5. In Section 8.3, the results from utilizing the script towards a large portion of the nation's devices are presented.

### 5.4 Classifying vulnerabilities: CVSS System

Having a way of categorizing different identified vulnerabilities allows for prioritizing the ones classified as the most severe. The CVSS system, short for Common Vulnerability Scoring System, is a standard used to evaluate the severity of publicly disclosed vulnerabilities. The system was developed by the US *National Infrastructure Advisory Council* (NIAC) and first published in 2004 and is currently maintained by the *Forum of Incident Response and Security Teams* (FIRST) [46]. The scoring system takes multiple factors into consideration, including ease of exploitation, level of user interaction requirements, and the impact of the vulnerability. The assigned

score ranges from 0 to 10, with 10 being the most severe. Vulnerabilities with a CVSS score of 7 or higher are classified as high and are more likely to cause significant harm to computer systems. As such, these vulnerabilities should be prioritized in order to mitigate the potential risk of exploitation. The system can be used to assess the severity of any security vulnerability, not only limited to CVEs, however, CVE identifiers are commonly used in combination with the CVSS scoring system. Utilizing the CVSS scoring system, organizations can effectively identify the most critical vulnerabilities and address them as a priority.

Section 8.3.3, presents a CVSS scoring table with provided data from the outcomes of a comprehensive vulnerability scan conducted on a significant number of devices nationwide.

## Chapter 6

# Evaluation

As explored in Chapter 3, a variety of port scanning tools have been identified, each with its own unique features and limitations. Consequently, the assessment and comparison of these tools are important in order to determine their effectiveness and suitability for different scenarios of port scanning. In this chapter, we will perform a comprehensive analysis of the strengths and weaknesses of a selected set of tools, which will aid in selecting the most appropriate tool and configurations for a given scenario. The findings presented in this chapter will contribute to the decisions in implementing the UiO SecurityLab Robot.

In the initial phase of the thesis, the possibility of implementing a port scanning tool from scratch was considered and tested. However, this approach presented several challenges that hindered the detection of vulnerabilities on a large number of devices. The tool encountered issues in regard to features such as banner grabbing for retrieval of service information, especially in estimating the time interval in which the tool would listen for replies from the server, before closing the connection. This timing implication was discussed in greater detail in Section 4.6. Additionally, the tool demonstrated poor performance, with only having the capability to reliably scan two ports per second in the initial implementation. This slow pace was due to the reason of lack of parallelization. These challenges, which included difficulties in determining an appropriate time interval for banner and port grabbing, and ensuring effective overall performance of the tool led to the exploration of alternative and reliable tools.

## 6.1 Performance and Detection Analysis: Nmap vs. Masscan

This study aims to compare and evaluate the performance and detection capabilities of two widely used open-source port scanning tools, namely Nmap and Masscan. The assessment focuses on detecting services and measuring the duration of the scanning process. The target network selected for this evaluation is Autonomous System number AS29695, encompassing a subnet mask of /17 and comprising approximately 32,768 available IP addresses. The scanning is conducted specifically to identify web servers operating on port 80. The evaluation is carried out using a Virtual Private Server (VPS) hosted in the cloud, equipped with a 1 Gbps fiber internet connection. Various parameter configurations are applied to the tools to assess their impact on the scan duration. Masscan is executed four times with different packet transmission rates, while Nmap is tested with two distinct timing templates: the default -T4 option, which balances speed and reliability, and the more aggressive -T5 option, intended for faster scanning but potentially leading to incomplete or inaccurate results due to possible packet drop. The results of this study provide insights into the detection capabilities and time efficiency of the scanning tools, aiding in the selection of the most suitable tool and configurations.

Tool Used	Detected Services	Time (s)
Nmap T4	106	6259
Nmap T5	107	1836
Masscan 1000	112	35.3
Masscan 10,000	112	13.8
Masscan 100,000	112	6.5
Masscan 1,000,000	113	3.5

Table 6.1: Network Scanning Tool Performance

The findings of the study indicate that the Masscan tool is significantly faster to detect open ports, and also, in this case, reports more ports as being open. The increased speed achieved by the tool can be attributed to its higher rate of request transmission per second in comparison to Nmap. It is worth noting that with the specific virtual machine and configured network card used to perform this test, the traffic rate within Masscan reaches a maximum value of 200,000 packets per second (PPS). This is why we only see a two-time speed increase from one hundred thousand requests per second to one million requests per second. The findings also suggest that although Masscan demonstrates a considerably faster speed than Nmap, the rate of operations remains within reasonable limits. Further, the accuracy of service detection achieved by Masscan is comparable to that of Nmap.



Upon comparing the two different Nmap timing profiles, it becomes evident that the T5 profile, in this instance, resulted in a scan time that was three times faster, while detecting nearly the same number of services. However, it is worth noting that the application of these two profiles possibly becomes more different when considering banner grabbing, as the T5 profile could potentially lead to a reduced collection of complete banners, due to the aggressive rate.

Furthermore, it is important to mention that a comparison between the most aggressive scan of Masscan and the most aggressive scan of Nmap reveals a substantial speed difference. Specifically, Masscan is approximately 524 times faster than Nmap in this particular scenario.

## **6.2 Assessing Public and Individually Collected Port Information**

The purpose of this next test was to compare publicly available port information obtained from the search engine Shodan, in contrast to utilizing self-collected data through the usage of open-source port scanning tools.

As of February 9th, 2023, Shodan reports a total of 104,102 open ports associated with port 80 HTTP within the Norwegian internet infrastructure [47]. It is worth noting that the number of open port values experiences monthly variations, with fluctuations of approximately 5-10 percent in either direction, as indicated by the Shodan trends feature.

To perform a comparative analysis, the Masscan port scanning tool was employed. The scan targeted the entire registry of Autonomous System Numbers (ASNs) assigned to Norway, containing a total of 16,099,072 distinct IP addresses. The Masscan tool, operating at its maximum rate, required 245 seconds to complete a port sweep across the entire IP address range. The scan successfully identified 97,646 open ports on port 80, seen in Appendix 11.3. Comparing these results with those obtained from Shodan, a 6 percent variation was observed, suggesting that the obtained data is not significantly different. Furthermore, the execution of 16 million requests within 245 seconds calculates to an approximate rate of 65,710 requests per second. Notably, this rate is lower than that achieved in the previous performance test, indicating that the performance of Masscan varies depending on the characteristics of the targeted network infrastructure.

### 6.3 Assessing Throughput for Large-Scale Port Detection

An investigation was conducted to assess the theoretical maximum speed of service discovery by conducting a test scan on a subset of allocated IP addresses in Norway. This evaluation aimed to determine the approximate time required to perform discovery scans across different port ranges. The Masscan tool was configured with a maximum scanning rate of 1,000,000 requests per second. The scan focused on approximately 2.7 million IP addresses, which included approximately one-fifth of the total allocated IP addresses in Norway. This subset was chosen as a representative sample to evaluate the performance of Masscan on a large scale, eliminating the necessity to gather data for all 16 million addresses in the country.

The choice of port 443 was based on its association with HTTPS web server protocol, which is a frequent activity on the internet and, would most likely result in a large number of detected services. The rate of transmitted packets was calculated by dividing the total number of destination IP addresses by the overall scan duration of 52 seconds.

$$\frac{2744320}{52 \text{ sec}} = 52775 \text{ packets/sec (pps)}$$

Masscan achieved an average transmission rate of 52,775 packets per second. Despite configuring the tool with a rate parameter of 1 million, it could only execute approximately 50,000 packets per second. This limitation is attributed to operating the tool within a virtual machine, which restricts the output to a maximum of 300 kpps (kilo packets per second) due to buffer overhead. Further, the disk and network configuration of the virtual machine may also contribute to the low packet rate. Possible ways to tackle this limitation are discussed in detail in Section 9.8

To estimate the time required to scan the entire IP range allocated to Norway against a range of 1000 ports, similar to the search engine Shodan's scanning approach, the following calculation can be performed by dividing the total number of packets by the packets sent per second.

$$16 \text{ million IPs} \times 1000 \text{ ports} = 16 \text{ billion packets}$$

$$\frac{16 \text{ billion packets}}{50000 \text{ pps}} = 320000 \text{ seconds} = 88.8 \text{ hours}$$

For scanning all available TCP ports (65,535), the calculation is as follows:

$$16 \text{ million IPs} \times 65535 \text{ ports} \approx 1 \text{ trillion packets}$$
$$\frac{1 \text{ trillion packets}}{50000 \text{ pps}} = 5820.5 \text{ hours} = 242 \text{ days}$$

Utilizing the maximum rate of the server hosted for this project, more discussed in Section 7.6.1, it would take 242 days to execute TCP port scanning for all available ports. These calculations will be useful for further analysis in determining the possibility of conducting continuous scanning for trend analysis.

## 6.4 Identifying the Most Frequent Network Ports

Following the identification of the outdated Nmap-services file in Section 4.2.5, a subsequent test was conducted to determine a more current and accurate list of the most commonly open ports in Norway. In order to gather this data, an analysis was conducted targeting the largest assigned Autonomous System (AS) in Norway, AS41164, which is affiliated with the Internet Service Provider (ISP) Telia. This AS contains a /13 subnet comprising of 524,286 assigned IP addresses. The IP range is primarily allocated to residential networks, which may provide insight into the nature of devices targeted in the scan, indicated by few servers and a high number of personal devices.

The port scan was configured to evaluate all possible ports, totaling 65,535 distinct ports. While this approach will require a significantly longer duration compared to scanning a more conventional range of 0-1024 ports, it offers more comprehensive insights into potential open ports within less frequently scanned ranges. The following command was employed to execute the port scan:

```
masscan 84.208.0.0-84.215.255.255 -p 1-65535 --max-rate=1000000
```

The command specifies a port range from 1 to 65,535, assessing all possible ports. The rate was set to a maximum of million packets per second; however, the actual scanning rate achieved by the machine was discovered in the previous section to achieve about 52,000 packets per second.

The scan lasted for a duration of 2 days, 1 hour, 40 minutes, and 57 seconds. Throughout this period, the Masscan tool detected a total of 235,305 open ports, resulting in an estimated rate of approximately 1.3 ports detected per second. Upon closer inspection of the results, it becomes evident that certain IP addresses reoccur numerous times within the data. Four IP addresses, in particular, respond to tens of thousands of requests, consequently saturating the scan results file. Potential explanations for this

occurrence include firewalls configured to report every port as responsive in order to obfuscate genuine open ports, or honeypot devices designed to intercept oncoming attacks and identify scanner IP addresses. Such activities is discussed further in Section 9.4.1.

After removing the records for IP addresses that potentially caused false positive responses, the number of open ports was reduced to 42,545, which accounts for only 18 percent of the previous count. With this revised dataset, we can proceed to determine the frequency of occurrence for the most commonly encountered open port.

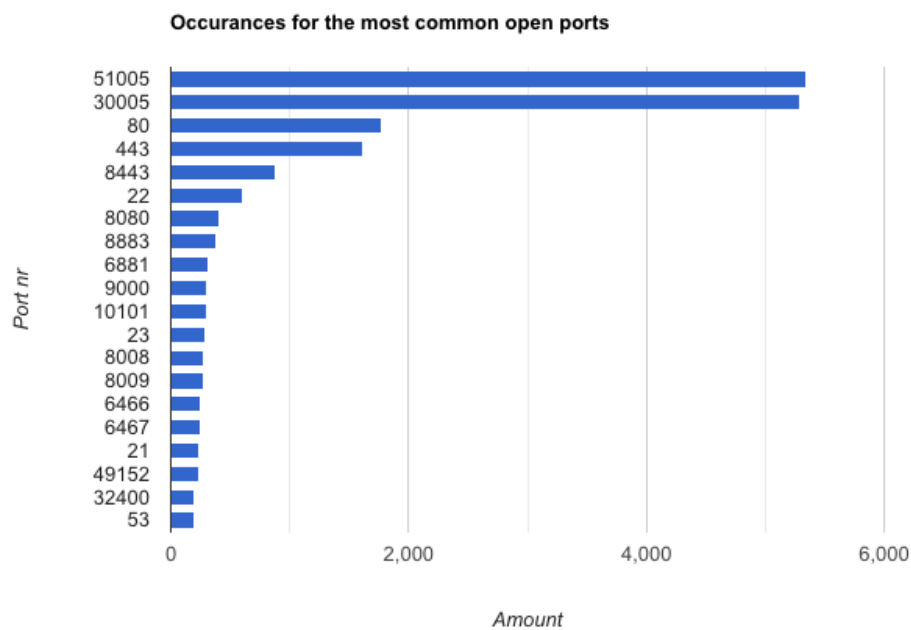


Figure 6.1: Results of exhaustive TCP port scanning, sorted by the number of ports detected

The findings reveal a substantial number of open ports on 51005 and 30005, which could potentially be associated with the TR-069 protocol. TR-069, also known as the Customer Premise Equipment WAN Management Protocol (CWMP), is a widely utilized standard allowing service providers and equipment manufacturers to remotely manage, configure and update their customer’s network-connected devices. This allows the Internet Service Provider (ISP) to remotely diagnose and resolve issues as well as apply security updates to routers and modems. Further examination is necessary to validate this hypothesis and evaluate the security ramifications associated with these exposed ports.

Upon further analysis of the identified open ports, it becomes apparent that numerous frequently encountered ports are present, with a significant

number of them being associated with web services. Notably, ports 80, 443, 8443, and 8080 are well known for their connection to web servers, handling HTTP and HTTPS traffic. These findings indicate a considerable presence of web servers within the scanned IP range, suggesting that many individuals may be hosting web services from their residential networks. The results from this experiment will be used to generate an own list of top ports in the results Section 8.2.

## 6.5 Nmap Vulnerability Discovery Rate

In order to determine the time required for conducting Nmap's vulnerability detection on known open ports, a benchmark test was conducted. The process involved utilizing Masscan to initially identify the open ports on the target hosts, followed by employing Nmap banner grabbing, along with the NSE script "vulners" for vulnerability lookup. This test was performed on a limited set of hosts to calculate an approximate average duration per host. By multiplying the seconds per host by the total number of network-connected devices, a rough estimation can be obtained for the overall time required to conduct vulnerability scanning across all internet-connected devices in the country.

In this experiment, the data from the conducted experiment in Section 6.2 were used. The previous experiment focused on conducting a service sweep on port 80, identifying 97646 services responding. Subsequently, a new experiment was conducted to identify vulnerabilities for the responding services utilizing the Nmap tool, with the NSE script *vulners*, previously mentioned in Section 5.3. The goal for this new experiment was to identify as many possible vulnerabilities on the target systems. The duration required for this experiment will provide the thesis with an estimated timeframe for conducting large-scale vulnerability identification.

The process of conducting a service sweep combined with banner grabbing took a total of 402,693 seconds, equivalent to over four days. Within this timeframe, the tool successfully identified 91,335 out of the previously detected 97,646 services. The output file generated by the Nmap tool consists of an XML file with over 119 MegaBytes in size. This file contains detailed information such as service version, potential vulnerabilities, and additional data related to each service. To calculate the rate at which the Nmap tool performs banner grabbing towards the detected hosts, we can divide the number of services by the number of seconds.

$$\frac{91335 \text{ services}}{402693 \text{ seconds}} = 4.4 \text{ seconds per service}$$

Based on the calculated rate of 4.4 seconds per service, the Nmap banner grabbing feature, when combined with the *vulners* NSE scrip, demonstrates a relatively slow performance. When compared to the discovery in Section 6.3, which revealed that the Masscan tool is capable of transmitting over 50,000 packets per second to different services, it becomes evident that the Nmap banner grabbing process is considerably time-consuming.

According to Shodan, there are approximately 800,000 services in Norway [48]. Utilizing the Masscan tool on Shodan's indicated port range of 1200, it is possible to identify these ports within approximately 88 hours using previously calculated rates. To estimate the total time required to perform vulnerability detection towards the approximately 800,000 ports, we can multiply the average duration of 4.4 seconds per service by the number of services, resulting in approximately 40.7 days. This highlights the significant time investment necessary for scanning all reported open ports in Norway. Consequently, conducting large-scale scans on a weekly or monthly basis is often impractical due to the extensive duration of the scanning process.

## 6.6 Assessing Bandwidth and Network Usage in Large-Scale Port Scanning

This section is dedicated to calculating the necessary bandwidth for conducting port scanning reconnaissance on a large number of hosts. The results of this calculation can assist in determining the appropriate configuration for the host machine that will be utilized for network operations. During service discovery using the Masscan tool, a total of three packets are sent to each target port, each containing a specific number of bytes:

- 54 bytes for the initial SYN packet request.
- 60 bytes for responding RST and ACK if the port is closed, or SYN and ACK if open.
- 60 bytes for closing the connection with an RST packet.

The number of bytes is retrieved by listening on the network traffic using the software tool Wireshark. Within this tool, we can see the byte length for each packet. SYN, ACK, and RST are different TCP flags that can be present in an IP packet. The TCP flags are explained in greater detail in Section 4.2.

The number of bytes remains consistent for both open and unresponsive ports. In the case of the Masscan tool, the total traffic sent to a TCP port is determined to be 172 bytes. By utilizing the request rate per second, as calculated in Section 6.3, the outgoing throughput for the initial SYN packet requests sent by the Masscan tool can be calculated as follows.

$$54 \text{ bytes} \times 52,775 \text{ PPS} = 2,848,770 \text{ bytes/s} = 22,79016 \text{ Mbps}$$

The total traffic generated by scanning all ports of a single IP address can be determined by multiplying 174 bytes by the number of ports (65,535), resulting in 10.87 megabytes of data transfer. While this data volume may appear manageable for an individual IP address, the data amount become significant when scaled up. For instance, if we were to scan the entire Norwegian IP range consisting of 16 million IP addresses, targeting all 65,535 possible network ports, the total number of packets would approach nearly one trillion, as derived from Section 6.3. To calculate the total data traffic for such a network operation across the entire nation, we can multiply the amount of traffic for one IP address by the total number of IP addresses assigned to Norway.

$$10.87 \text{ MB} \times 16,000,000 \approx 173.92 \text{ TB}$$

Performing a comprehensive network assessment that involves a simple SYN scan targeting every assigned service, totaling over 1 trillion, would result in network traffic exceeding 173 Terabytes. This extensive scanning process emphasizes the significant volume of data required for conducting such comprehensive network assessments. It is important to reduce the number of requests, and network bandwidth sent when performing large-scale network operations. This reduces the load on network switches and routers around the nation, which is often not designed to handle such an amount of traffic.

## 6.7 Comparison of Nmap and Masscan for Banner Grabbing

The Masscan port scanning tool also includes a banner-grabbing feature, which allows for service detection. After some initial testing, it is evident that the features are limited by the fact that the tool does not maintain an open connection to the target, which makes banner grabbing more difficult than using more common tools like Nmap or Netcat. As mentioned before in Section 4.9, when Masscan sends a SYN packet as part of the handshake, it does not maintain the connection open to await the SYN-ACK response. Instead, upon receiving the SYN-ACK response from the target, the tool promptly responds with an RST packet. This immediate termination of the connection prevents the tool from capturing detailed banner information before the connection is closed.

When experimenting with banner grabbing search across various IP addresses, comparing the Nmap and the Masscan tool, it is evident that the Nmap tool is more reliable at retrieving banner information reliability.

```
harald@prodesk:~$ nmap -sV -p1-1000 192.168.1.2
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-16 12:20 UTC
Nmap scan report for 192.168.1.2
Host is up (0.00025s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.52 seconds
harald@prodesk:~$ sudo masscan -p1-1000 --banner 192.168.1.2
[sudo] password for harald:
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2023-04-16 12:21:03 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [1000 ports/host]
Discovered open port 22/tcp on 192.168.1.2
Discovered open port 111/tcp on 192.168.1.2
```

Figure 6.2: Comparing Nmap and Masscan banner grabbing

The comparison of the two tools is shown in the figure. The target device used in the experiment is a Proxmox Debian host. Observing the banner results, it is clear that both tools successfully identify the two open ports. However, only Nmap is capable of detecting the corresponding services and their versions. The banner grabbing feature in Masscan is deemed ineffective when employed for service detection. Consequently, this limitation restricts the tool's effectiveness in performing vulnerability detection.



## Chapter 7

# Implementation and development

This chapter focuses on the integration and development of a comprehensive tool for automated network scanning and result storage. Building upon the previous testing and experimentation with various tools, the goal is to create a complete solution that integrates the various steps in gathering and presenting network data. The chapter outlines the steps taken to integrate different components and technologies, enabling the automated launch of scans and efficient storage of the obtained results. The focus is placed on the practical implementation and development aspects, allowing for a more efficient and effective scanning workflow. By leveraging the knowledge gained from previous experiments, this chapter presents a comprehensive approach to scanning, resulting in a robust and automated tool for network reconnaissance.

### 7.1 Choosing the Right Tools: Combining Nmap and Masscan

After conducting performance testing, it is evident that Masscan outperforms Nmap as a port scanning tool in terms of speed. However, the effectiveness of both tools in determining the software version running on a port differs. The Masscan banner grabbing test indicates that it is not well-suited for this purpose, while Nmap proves to be more suitable.

The choice came to implement both Nmap and Masscan, which allow for the speed of Masscan, and the accuracy of Nmap. To implement both these tools, Masscan will first be used to identify services, and then Nmap will be used to perform detailed banner and vulnerability lookup from the detected services. Further, the usage of other Nmap NSE (Nmap Scripting Engine) scripts will be possible, for example using the recent NSE scripts created by the UK's National Cyber Security Center (NCSC), which aims at helping system owners to detect specific vulnerabilities [49].

The web article "Finding the Balance Between Speed & Accuracy During an Internet-wide Port Scanning" [11] explores the approach of combining the Nmap and Masscan tools, using various Linux commands to filter and pass on the output. Taking inspiration from this web article, we can integrate the results from both tools, explained in more detail in the next section.

The decision was made to exclude the implementation of Favicon hashing and website screenshots due to their insufficient ability to provide a reliable indication of services and vulnerabilities. Furthermore, these methods have the potential to expose personal or sensitive information, which also justifies not implementing these features.

## 7.2 Optimizing Network Discovery: Masscan and Nmap Integration

This section will go through the steps implemented for combining the Nmap and Masscan tools. The combination will result in a detailed and efficient scanning service. A Python script was created to handle the tool integration. The script first performs a port discovery scan using Masscan. Here, the user can provide parameters to specify the port range and rate. The results from this initial service discovery will be stored in an Extensible Markup Language (XML) file.

```
<host endtime="1679331505">
  <address addr="89.8.156.51" addrtype="ipv4"/>
  <ports>
    <port protocol="tcp" portid="6112">
      <state state="open" reason="syn-ack" reason_ttl="247"/>
    </port>
  </ports>
</host>
```

Figure 7.1: Stored XML information for port discovery

The result stores records for every discovered port, as displayed in the figure. The record includes most importantly timestamp, IP address, port number, and port state. With this data, the before mentioned Python script will create a file for every unique port, seen in Appendix 11.1.

Every IP host responding for each unique port will be added to the file, resulting in the creation of multiple files, containing a large list of IP addresses. Additionally, a separate file will be generated to store a list of all the distinct ports identified, sorted in descending order based on their frequency of occurrence. Further, Nmap will be executed for every port in the unique ports list, focusing on the corresponding file containing IP

addresses associated with that specific port. The Nmap process will be executed for every unique identified port, with a transition to the subsequent port as soon as the previous one completes its execution.

The Nmap process will scan the IP addresses once more, with version detection enabled, and the NSE script named "*vulners*". Nmap will capture the services banner information and store the results in XML files for each unique port. Here, each discovered port would have a record where banner information and output from the "*vulners*" NSE (Nmap Scripting Engine) script are stored. An example of one of these records is provided in Appendix 11.4.

This example displays a service on port 80, where the banner reveals the Apache service, with its according version. Further, the NSE script provides related CVE vulnerabilities for the Apache services. In this particular example, a service running on port 80 is illustrated, wherein the banner information discloses the presence of the Apache service along with its corresponding version. Additionally, the utilization of the NSE (Nmap Scripting Engine) script facilitates the identification of relevant CVE (Common Vulnerabilities and Exposures) vulnerabilities associated with the Apache services.

This approach builds upon the methodology outlined in the web article titled "Finding the Balance Between Speed & Accuracy During an Internet-wide Port Scanning" [11], which employed Nmap to conduct supplementary scans for each identified port. However, in the present implementation, a more efficient strategy is adopted. Instead of conducting scans for all identified ports across all hosts, a more intelligent approach is employed, targeting only the identified open ports for each individual host.

### **7.3 Storing and Visualizing Obtained Results: ElasticSearch**

The generated scan result files from Nmap lack readability, and their presentation appears chaotic, as illustrated in Appendix 11.4. To facilitate a more favorable presentation of the results, a decision was made to adopt a database as a solution for storing the data. This choice enables the management of a more robust and simple search for the data.

When deciding on a storage method for the scanning results produced by Nmap, careful consideration should be given to the data format. Nmap offers various options for storing results, with the XML format being one of them. Storing this format in conventional relational databases like MySQL can pose challenges due to the structured nature of such databases. Relational databases rely on well-defined schemas, which can make it difficult to store unstructured data like XML files, which is the chosen approach utilized by Nmap.

NoSQL databases, which are specifically designed to handle unstructured data, may initially appear to be a suitable option for storing Nmap output results. However, it is worth noting that NoSQL databases, such as MongoDB and Cassandra, are not ideally suited for efficiently searching and analyzing extensive datasets. These databases lack the advanced searching capabilities that we need for analyzing and making sense of the obtained data.

ElasticSearch is a widely used search and analytics engine, built on the Apache Lucene library, written in Java. It employs a distributed architecture, where data is stored across multiple nodes. The data is organized into indices, which are collections of documents that have similar characteristics. In ElasticSearch, data is stored in a JSON format, and all fields within each document are indexed to make searching efficient and fast. The search engine provides a query language (EQL), a powerful and intuitive search language, which enables users to retrieve and analyze data from ElasticSearch indices.

ElasticSearch is a component of the ELK stack, which consists of three different components; ElasticSearch, Logstash, and Kibana, which all work together to provide a feature-rich database. All of the components are open-source and are designed to be integrated with each other. The ELK stack provides an efficient solution for storing data, transforming, and visualizing the data. The following provides a brief description of each component:

**Elasticsearch** is a search and analytics engine that stores and retrieves data. It is designed in mind to be able to handle large amounts of data in real time. The search engine provides a feature-rich set of APIs that allows for integration and interaction with the engine.

**Logstash** is a data processing pipeline that is designed to manage and analyze large amounts of log data. The tool is capable of ingesting log data from a variety of sources and transforming the data into a common format. The tool can parse, manipulate and enhance data before forwarding it to other tools, such as Elasticsearch for indexing and searching or Kibana for visualization.

**Kibana** is a visualization and dashboard tool that provides a web interface for searching, and visualization data stored in Elasticsearch. Kibana can generate interactive visualizations based on data stored in Elasticsearch, such as pie charts, bar charts, heat maps, and more. This makes Kibana a great tool for creating visualization dashboards.

By combining these three components, the ELK stack is formed. In essence, Elasticsearch provides the data storage, Logstash ingests and processes the data, and Kibana visualizes the data, and provides a web dashboard. The ELK stack makes a suitable solution for storing Nmap

results because it is the ability to process and parse data using Logstash. This is why the storage solution was chosen for the project.

### 7.3.1 Ingesting results into ElasticSearch Database

In order to store the Nmap results in ElasticSearch, the data must go through a transformation process. This is enabled by the Logstash codec plugin *logstash-codec-nmap*, which enables the conversion of Nmap's stored data into a format that can be utilized by ElasticSearch [50]. A codec plugin is used to decode and parse data from a specific format, in this case, an XML file, to a readable format by ElasticSearch.

The *logstash-codec-nmap* contains a mapping of how the data should be parsed and mapped to specific fields. Logstash uses codec mapping to understand the structure of receiving data, and how to parse it into individual fields. In the case of the Nmap codec, the mapping contains instructions for how different entries in the Nmap XML file shall be transformed, such as IP address, port, service, and timestamp. The mapping is used to define structure and data types of fields within ElasticSearch indexes for storing the result data from previous network scans using the Nmap tool. Manually mapping the data can be a time-consuming process. Therefore, ElasticSearch, in combination with the Logstash Nmap codec, proves to be a suitable option since it creates a pipeline to automate the tedious process.

During the exploration of methods to index Nmap results, a project appeared, which aims at solving the task of ingesting Nmap result data in ElasticSearch. The GitHub project "*elk-nmap*" by Bennett Warner combines the projects of 5 different public GitHub repositories, to an easy-to-deploy ELK stack, with the capabilities of ingesting Nmap's XML scan result files into an ElasticSearch database [51]. The subsequent bullet points provide a brief overview of each of the merged projects.

- The project implements *docker-elk*, a popular repository that allows for running the ELK stack with Docker [52]. The repository includes a configuration file used by Docker Compose to configure the different services for the ELK stack, which is run within its own containers.
- *VulntoES* is a GitHub project created by Chris Rimondi, created to ingest and import results from a range of vulnerability scanner outputs into Elasticsearch [53]. A provided Python script parses Nmap XML results and uploads the results to extracted data to Elasticsearch using a REST API. It extends on the *logstash-codec-nmap* Logstash codec plugin, by allowing for mapping of output by the Nmap NSE script *Vulners*.
- "Docker\_offensive\_elk" updates and improves the code from *VulntoES*, which is over 6 years old as of February 2023 [54]. The repository also implements code from "docker-elk", to deploy the ELK stack using Docker.
- Lastly, the combines code from the repositories *scan2elk* [55] and *vulnscan-parser* [56] by the developer happyc0ding, which allows for parsing scan results from the output of the tools Nessus, testssl, Nmap, SSLYZE, and Burp Suite. This allows for ingesting data from other tools than just Nmap, which is something that can be good for future-proofing the project.

All of these repositories combined make up the *elk-nmap* repository. The deployment of the ELK stack, alongside the provided data ingesting container is straightforward using the provided docker-compose file. Using the "elk-nmap" repository for ingesting the Nmap results offers a time-saving advantage. Prior to discovering this consolidated GitHub repository, considerable effort was invested in attempting to ingest XML files using the default Nmap codes, followed by the manual inclusion of mappings for data records associated with the vulners vulnerability lookup Nmap NSE script.

Once successfully indexed, the data is visible in Elasticsearch's web interface. Each index is stored as a JSON object, and the discover search functionality within Elasticsearch offers users an interface to explore extensive datasets while enabling filtration through the use of a straightforward query language. The stored data is organized into separate indices, and the JSON objects can be displayed as a table, where every mapped data using Logstash codec is present.







 ip	79.160.4.86
<b>Multi fields</b>	
ip.raw: 79.160.4.86	
 method	probed
<b>Multi fields</b>	
method.keyword: probed	
 name	http
<b>Multi fields</b>	
name.raw: http	
 port	443
 product	D-Link DCS-930L_17 webcam ht tp interface
<b>Multi fields</b>	
product.raw: D-Link DCS-930L_17 webcam ht tp interface	
 protocol	TCP

Figure 7.2: Example of a document with banner information stored in Elasticsearch Discovery page for a discovered service

### 7.3.2 Visualizing using the Kibana Dashboard

The Kibana component allows for presenting data in a more visually-oriented format. The dashboard feature lets users create customizable dashboards to visualize data. These customizable dashboards allow for displaying data using various visualization types, such as bar charts, pie charts, tables, and histograms. A visualization in Kibana uses Elasticsearch queries to retrieve data. A user can apply Elasticsearch queries and filters to retrieve only the desired data. This feature is used as a front-end dashboard for visualizing the results from the large-scale vulnerability scan conducted in the results Chapter 8. Utilizing this functionality, it becomes feasible to generate a dashboard that emulates the Facets feature of the Shodan search engine, as mentioned in Section 3.3.1.

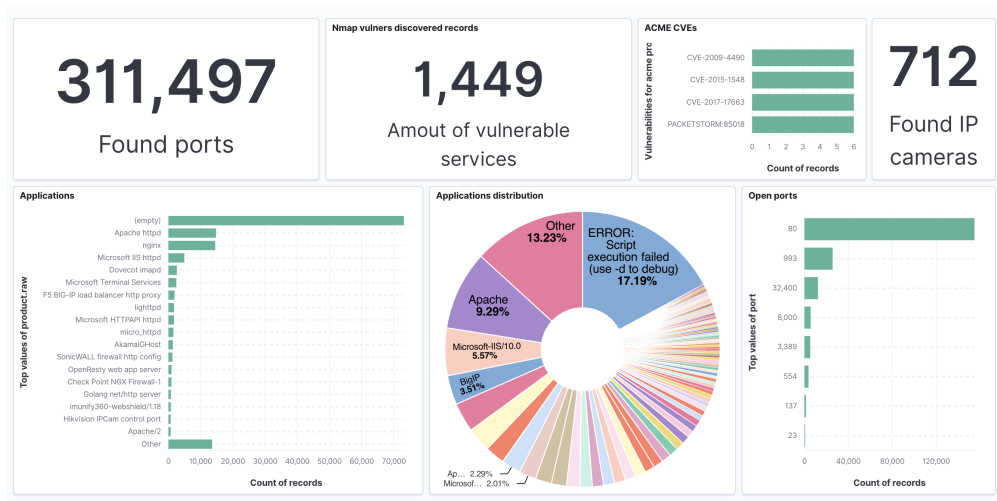


Figure 7.3: Screenshot of Kibana Dashboard created for the UiO SecurityLab robot

The visualized data presented in the figure was collected before conducting a full nationwide scan of all ports and IP addresses. As a result, the data displayed in the figure does only represent a selected portion of the nation’s infrastructure.



## 7.4 Web Interface to launching Service Discovery

To facilitate vulnerability scanning without requiring users to utilize the command line shell, a website was developed. The website provides a simple interface for launching different types of network reconnaissance activities. The creation of the website involved using Django, a widely-used open-source web framework designed for building web applications. Django, written in Python, effectively manages incoming HTTP requests by leveraging Python's native web server, commonly known as the "WSGI" server.

The utilization of Django enables simple handling of incoming POST and GET requests from clients, and allows the execution of Python scripts on the server for specified types of actions. The implemented website utilizes this functionality to execute Python scripts to initiate the running of Docker images configured to launch scans aimed at targets specified by the user.

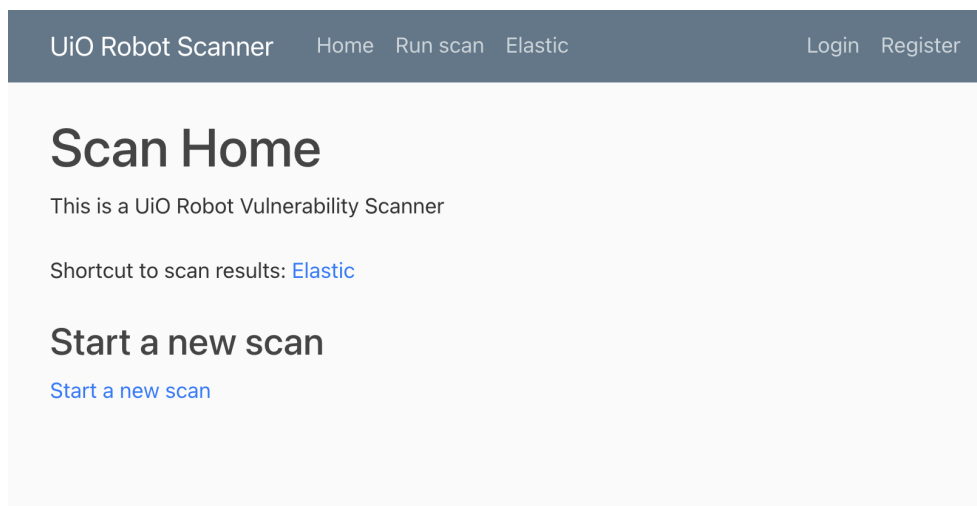


Figure 7.4: Homepage for the UiO SecurityLab robots web interface

CSS was utilized to implement simple styling for the website. The design was kept simple, and visible. The website contains three main pages, starting with the home page, where the user is greeted with a page giving a brief explanation of the purpose of the website. The home page links to the page for starting a new scan, the most important feature of the website. This allows for the execution of Masscan and or Nmap scans by the user. The "Start a new page/Run scan" page initiates access to three different scan methodology pages, enabling the user to conduct a "Default" scan, a "Custom Port" scan, or a "Custom Command" scan.

The default scan page is designed to conduct a scan command against the top 1000 most frequently open ports. The user can initiate the scan by simply entering an IP range or list and pressing the "Start Scan" button. A screenshot of this page is provided in Figure 11.2 in the Appendix.

In contrast, the custom port scan feature allows a user to specify both an IP address list and a port list. This feature is great for conducting a service sweep against a specific port. Figure 11.3 in the Appendix displays this page.

Lastly, the custom command scan page allows the user to conduct a scan using a user-generated command for either Nmap or Masscan. This page opens up the possibility for the user to conduct a scan utilizing different parameters and Nmap NSE scripts. User input is required, which will be appended after either the Nmap or Masscan command execution. This feature is displayed in Figure 11.4 in the Appendix.

The results from all of these scan techniques will be ingested into the Elasticsearch database. A URL redirecting to the Elasticsearch web interface is also provided on the web server. There, the user can view visualized graphs and the database entry for the recently conducted reconnaissance scan.

## 7.5 Working with Docker

In Section 7.3.1 of the thesis, the reference was made to the utilization of Docker for integrating database features. Docker is an ideal choice for hosting the running services of a port scanner due to its features which allow for robust and scalable deployment of services. Docker runs each service as an individual container, which provides an isolated environment for the software. Within this container, all necessary dependencies and libraries are packaged within the container image, ensuring consistent deployment across different deployment environments. Utilizing Docker enables simple deployment processes for services and allows for great control of configurations to provide a suitable environment. The conversion of Docker container environments into images enables simple sharing of the service or application.

These capabilities assist during the deployment of the previously mentioned port scanner service, ELK stack database, and web server. Docker will be utilized to host all of these services on one host device. Every service has a configuration specified in a "Docker-compose" file. This file specifies configurations such as storage paths, source ports, and networks, enabling the containers to launch with predetermined configurations. The usage of "Dockerfiles" is also utilized, which enables the building of a custom container, allowing for the possibility of deploying preconfigured Linux images, in this case, used for conducting scanning

activities. Provided is the code used to implement the scanning container in a Dockerfile.

```
FROM ubuntu:20.04
ENV DEBIAN_FRONTEND noninteractive

RUN apt update && apt-get install -y
python3 python3-pip nmap net-tools
masscan libpcap-dev

COPY scan.py .
COPY requirements.txt .
COPY hosts.txt .
RUN mkdir /finalOutput
RUN python3 -m pip install -r requirements.txt
ENV IP=${ip}
ENV PORT=${port}
ENTRYPOINT ["python3", "scan.py"]
```

Figure 7.5: Dockerfile for running a provided Python script

The provided Dockerfile code in the figure builds a Ubuntu container that performs a port scan toward a list of host and port values provided as environment variables. The Ubuntu container is configured to update to the latest version of Ubuntu prior to installing the necessary Linux tools. Once installed, a Python program script provided will be executed to initiate the scan.

A Dockerfile is also used to deploy the Django web server, which is configured to install and run the necessary commands for the instance. Figure 7.6 displays a diagram of the container flow between the different Docker containers.

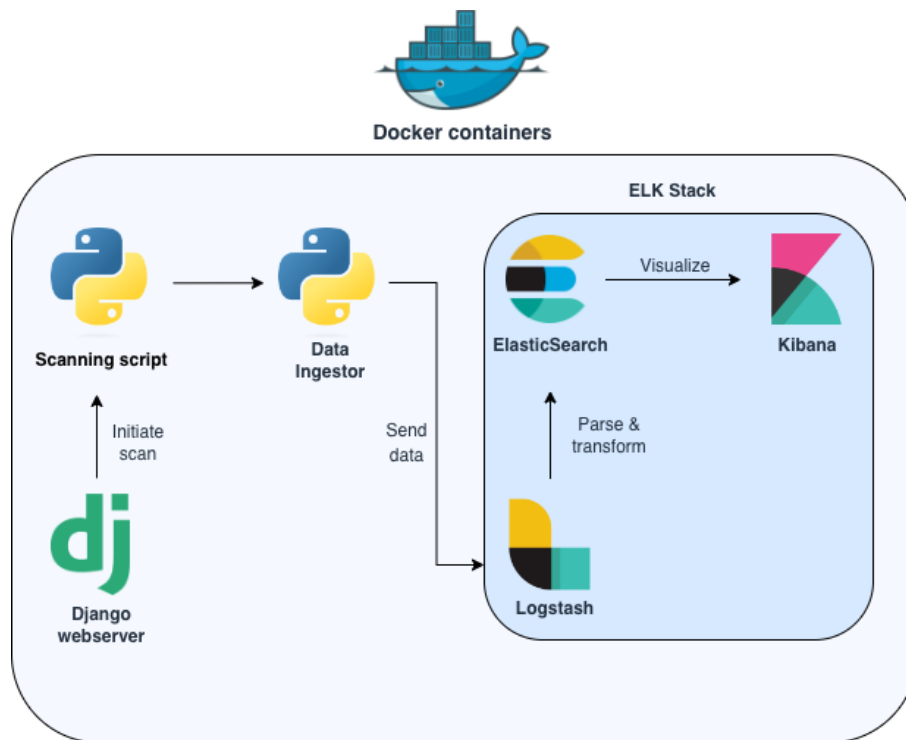


Figure 7.6: Overview of implemented Docker containers

The containers communicate with one another, and their interactions are dependent on the data provided by the prior service. The data flow process can be described as follows:

1. To initiate the process, a user launches the scanning activity on the created Django website by selecting the "start scan" button.
2. Thereafter the Django webserver is prompted to execute a script that launches a docker container with specified entry point variables provided by the web server.
3. Upon the completion of the scanning activity, the data ingestion container is launched. This container will parse Nmap XML output files to extract findings, certificates, ciphers, hosts, and services, which will be stored in ElasticSearch.
4. During this phase, Logstash is utilized to parse and transform the scan results data from Elasticsearch using the designated pipeline file for Nmap.
5. Once the data is transformed, Logstash transmits it to the Elastic-Search database.
6. Finally, Kibana is employed to execute queries towards ElasticSearch, retrieving the data required for visualization purposes.

Docker containers can by default not talk to each other. Therefore the Docker mounting feature is used, for allowing the containers to have a "shared folder" used to transfer data between each other.

## 7.6 Exploring Hosting Provider Policies on Port Scanning

While the web server for launching scans and the database storing the results do not pose any issues hosting, the port scanner is expected to generate a considerable amount of SYN packet traffic towards potentially millions of hosts. This kind of traffic may be perceived as potentially malicious by network and hosting providers, which often take appropriate measures in order to mitigate potential legal consequences that may occur from the generated scan traffic.

Virtual Private Server (VPS) and cloud computing hosting providers offer a platform for users to host Software as a service (SaaS) applications and hardware as a service (HaaS) virtual machines. This type of service is commonly used to host projects. The providers may, however, not allow their services to generate large amounts of SYN traffic or perform port scanning activities. Both the VPS hosting providers and Internet Service Providers (ISPs) may have to comply with legal or regulatory compliments in order to protect their infrastructure against potential malicious activities. Consequently, they may adopt policies that prohibit users from conducting activities such as port scanning.

To comply with these legal and regulatory compliments, the hosting and internet providers require their users to comply with terms of service that prohibit any form of possible malicious activities. They also may implement measures to detect possible malicious traffic, utilizing network monitoring and intrusion detection systems (IDS). The legality of port scanning activities is discussed in greater detail in Section 9.9.

### 7.6.1 Norwegian Research and Education Cloud

In order to deploy a large-scale scanning activity and avoid violating the above-mentioned terms of services, the request was made to Norwegian Research and Education Cloud, NREC, to use one of their machines for the network scanning activity. NREC is a cloud service for research and education purposes, provided in collaboration between the University of Oslo UiO and the University of Bergen UiB. The service is available for students and researchers within numerous major universities in Norway and provides Infrastructure-as-a-Service(IaaS) for users to create instances of different Linux and Windows distributions. The platform allows for selecting between different resource tiers, limited by a quota, that is tied to projects.

The request towards NREC asked for permission to use an instance within their cloud computers to conduct large-scale, non-intrusive service and vulnerability discovering using methods previously explained in this chapter. After some internal discussion upon receiving my request, they allowed for the activity and would notify the national cyber security center (NCSC) at the National Security Authority (NSM) in Norway about the planned activity.

# Chapter 8

## Results

This chapter will present the results from a series of conducted network reconnaissance assessments. By analyzing the identified ports and comparing them to established port popularity data, we gain insights into the changing landscape of service usage and potential vulnerabilities. The purpose of this activity was also to enable a comparison between existing data and the data we gathered ourselves. Furthermore, we present an overview of the vulnerabilities detected during the scan, including their exploitability and severity score. This chapter emphasizes the importance of up-to-date port popularity data and targeted scanning approaches for effective network security assessment. Within this chapter, the various data-gathering activities will complement one another, as the results obtained from one scan will be utilized in the subsequent analysis.

### 8.1 Comparing Identified ports against the Nmap-Service record

This section presents a comparison between the popular ports list used by Nmap for TCP ports and the results obtained from our own test conducted in Section 6.4. The previous experiment involved conducting reconnaissance activity by sending SYN packets to all available 65535 TCP ports within a specific AS IP address range. The objective was to determine the port that occurred with the highest frequency.

The port popularity data stored by the Nmap services file was obtained from research conducted back in 2008, as documented in Section 4.8.1. It is important to acknowledge that the collected data differ in terms of timestamp and scale. The conducted scan covered 524,286 IP addresses, while the data from Nmap is reported to be gathered from tens of millions of Internet hosts.

The gathered data is converted from a count format to a popularity percentage format. This conversion is done to ensure consistency in data representation and align with the data format used by Nmap for storing port popularity values.

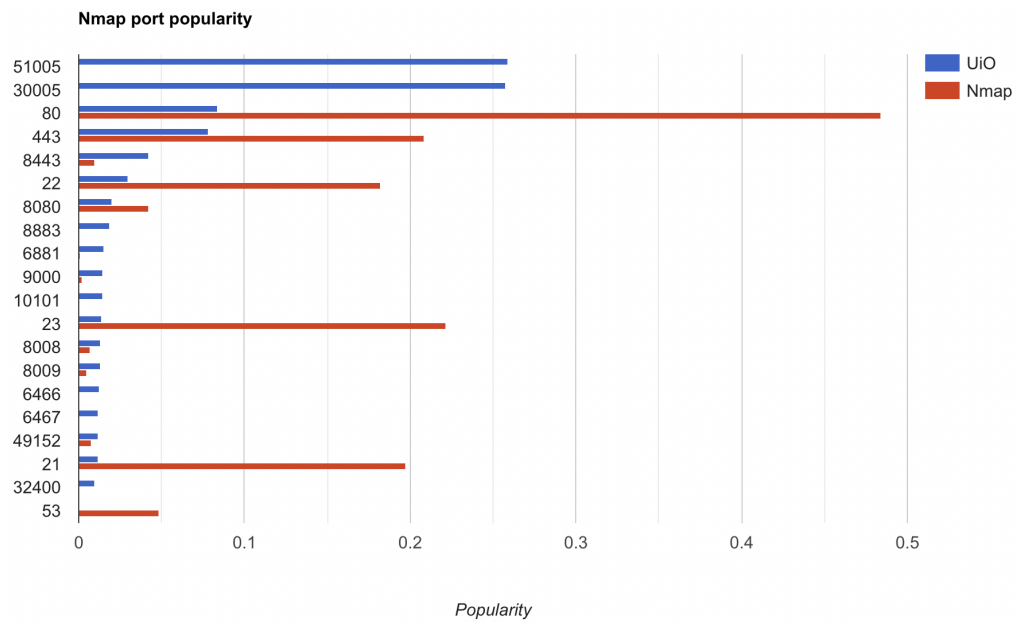


Figure 8.1: Comparing Nmaps stored port popularity against the UiO scanner

The y-axis of the figure presents the most frequently discovered port from the server hosted at UiO in descending order. On the x-axis, the popularity frequency number for each port is displayed, indicating the likelihood of the port being open. Analyzing the figure reveals significant differences in reported port popularity.

Observing the data, it becomes evident that the usage of certain ports has undergone significant changes over the past 15 years. For instance, ports like 23, commonly used by Telnet, have experienced a considerable decline in popularity over time. These substantial variations suggest that the stored popularity data in Nmap is likely outdated and requires updating to reflect the current landscape of service usage. This issue is further discussed in Section 9.4.



## 8.2 Nation-Wide Port Scanning: An In-Depth Investigation

Recognizing that the commonly used Nmap ports list is no longer up to date, we took the initiative to update the list and initiate an extensive investigation encompassing the entire nation's IP ranges. This assessment aimed to offer a comprehensive analysis of the distribution of open ports throughout the nation, which would serve as an indicator for determining the types of services that will be prioritized when conducting a vulnerability assessment later.

The recent scan conducted in the previous section has provided us with the results that can be filtered and sorted to identify the most frequently open ports. The results from this sorting process offer an accurate representation of the ports and services typically exposed to the Norwegian Internet. The scan confined all possible TCP ports, allowing for the identification of the ports that were most frequently responsive. Based on these discoveries, we compiled a list that included the ports exhibiting the greatest frequency of positive responses.

Utilizing the compiled list, a subsequent scan was initiated to investigate the top 1000 ports that were previously identified. By focusing our efforts on these ports, we can significantly reduce the time and resources required to conduct a comprehensive scan of the entire country. Furthermore, this targeted approach contributes to a more efficient scanning process and enables us to obtain accurate and relevant insights into the current state of network security in Norway. In order to carry out this evaluation, the Masscan tool was employed due to its demonstrated superiority in terms of speed.

During a span of 2 days, the Masscan tool was able to find 11,584,412 open ports. After investigating the results, an abnormal pattern was detected in some devices. These devices reported a large number of open ports. The devices are all contained within the same /13 network, a subset of 16,000 IP addresses reporting exactly 535 open ports. This particular behavior is commonly linked to the implementation of a SYN cookie technique used by firewalls to prevent attacks. Further discussion of this behavior will be discussed in Section 9.4.1. To address this anomaly, IP addresses with over 500 open ports were removed from the stored results records, resulting in the identification of 2,071,968 open ports.

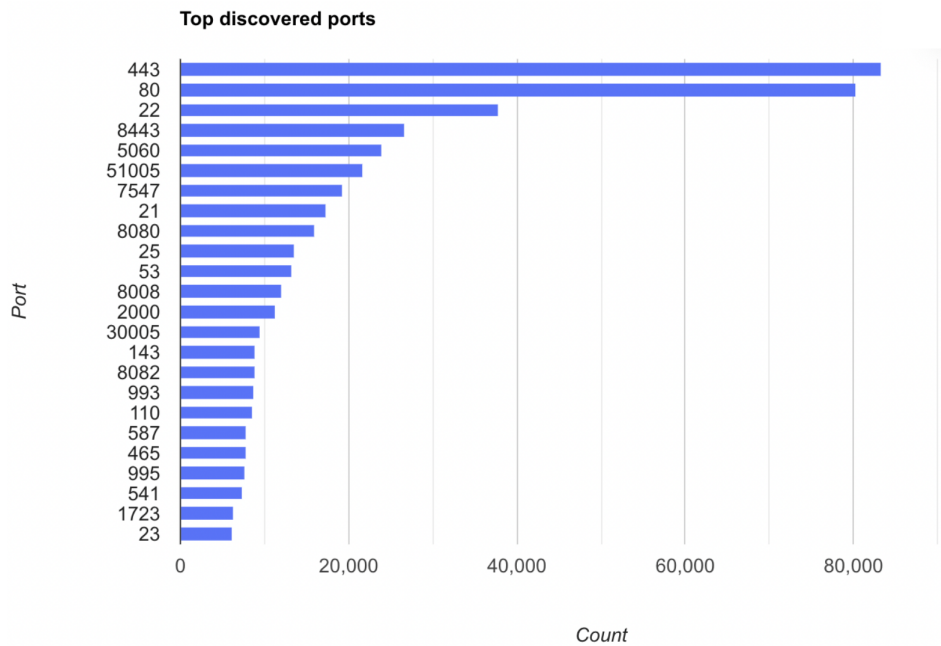


Figure 8.2: Chart for the top most responding ports

The findings reveal that web servers are the most commonly exposed devices on the Norwegian network, with ports 443, 80, 8443, 8080, and 8008 all associated with HTTP and HTTPS web traffic. This is a predictable outcome, given the widespread use of the internet for browsing websites, searching for information, and using social media platforms. These findings underscore the importance of protecting web servers and implementing security measures for remote management services, as they are highly susceptible to cyberattacks. The results will contribute to a better understanding of the nation’s exposed internet network infrastructure.

The second most frequently exposed services are related to the remote management of hosts. Specifically, ports 7547, 30005, and 51005 are associated with TR-069, which was previously discussed in the last chapter as a protocol used by ISPs to manage their customers’ internet routers remotely.

### 8.3 Results for Service and Vulnerability Scanning

The subsequent section presents the outcomes of a comprehensive Nmap service and vulnerability search. The data is collected by scanning hosts at their corresponding open ports as reported in Section 8.2.

After operating for a one-week period, the NREC cloud hosting provider was contacted due to receiving multiple reactions regarding the outgoing network activity. Within this timeframe, the UiO robot successfully performed banner grabbing on the top 35 most frequently reported open ports detected in the previous scan. The decision was made to conclude the vulnerability assessment after receiving the complaint message, to avoid causing any further issues. As a result, the data presented in this section comprises information extracted from banners associated with 598,228 ports. Various bash, grep, and awk commands were employed to extract the relevant information from the XML output files generated by Nmap. The figure illustrates the most frequently occurring product value derived from the extracted banner information.

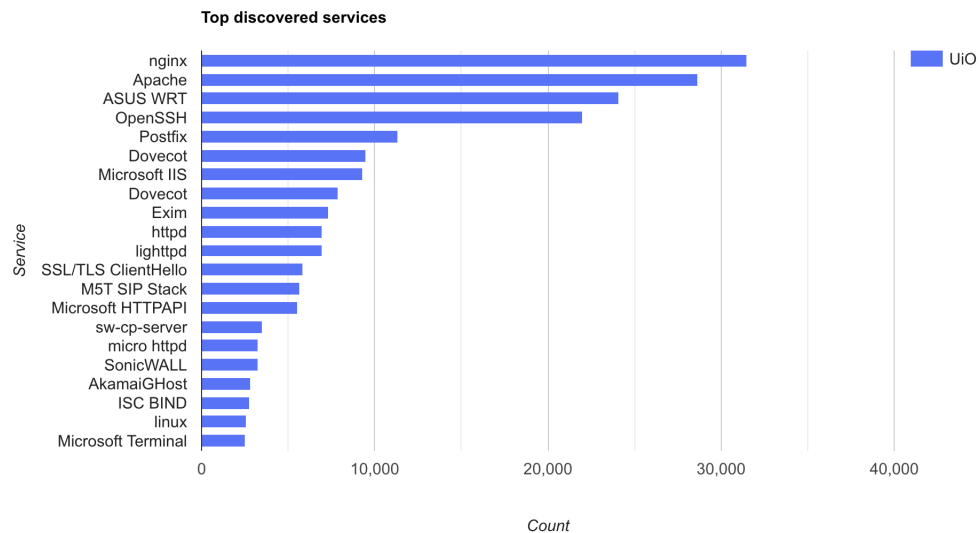


Figure 8.3: Most frequent service detected from banner grabbing

The figure illustrates the software identified on ports that closely align with the popularity findings presented in Figure 8.2. Among the detected services, web services are the most frequently encountered, with the Nginx web service being the most common. It is important to note that the product field may not be consistently present in a banner, resulting in less detected software.

### 8.3.1 Amount of Identified Vulnerabilities

From the collected banner grabbing, the Nmap *Vulners* NSE script allows us to perform vulnerability lookups using the vulnerability database described in more detail in Section 5.3.

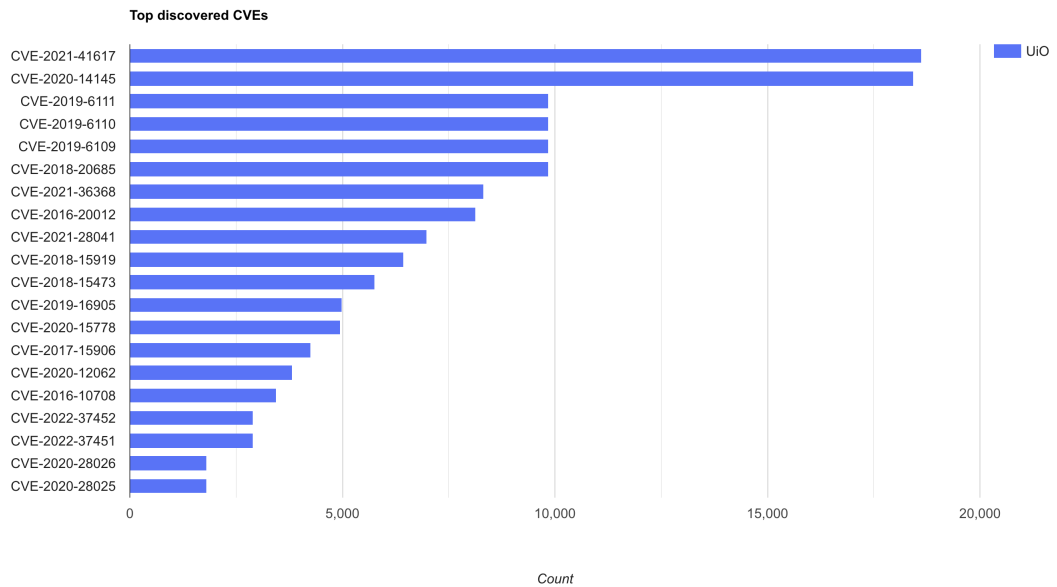


Figure 8.4: Most frequent CVE detected from banner information

The overall number of detected vulnerabilities in the identified software is 400,882 CVE records. Out of all the detected CVEs, 1546 unique ones are detected. The most frequent CVE vulnerability detected was CVE-2021-41617, seen on 18,630 different IP addresses and involved a vulnerability in the OpenSSH protocol.

A device can possess several CVE vulnerabilities that can create an illusion of a higher number of infected devices than the actual count. It is common for software that consists of multiple components or features to have multiple CVEs associated with each of these distinct elements.

### 8.3.2 Exploitable against Non-Exploitable vulnerabilities

Identified vulnerabilities may not always be known to be exploitable. An analysis of the "is\_exploit" stored value in the output of the Nmap vulnerability scan enables the identification of whether the vulnerability has been labeled as exploitable or not.

```
<elem key="is_exploit">true</elem>
```

The "is\_exploit" value returns true or false for a given vulnerability. The difference is crucial in evaluating the potential impact of a vulnerability on a system and determining the appropriate response. A non-exploitable CVE vulnerability identifies a security weakness. However, there are no known methods for an attacker to exploit the vulnerability. While the classification may offer some relief for security personnel, the identified non-exploitable vulnerability is valuable and important not to disregard. It is crucial to address the issue before a potential exploit or proof of concept exploit becomes available. In addition, the exploit state may change over time as researchers discover new tools and techniques. The allocation of vulnerabilities, categorized as exploitable and non-exploitable, can be represented as follows:

Label	Count
True	164,092
False	236,790

Table 8.1: Exploitable vs Non-Exploitable data classification

Looking at the results, we can see that more than half of the detected vulnerabilities are not classified as exploitable, which is information assessed by the vulners database, described in Section 5.3.

### 8.3.3 CVSS scoring

Various CVEs have different levels of severity. Each CVE is allocated a CVE score that represents its severity. Additionally, some CVEs lack public proof of concept (POC), which hinders adversaries from exploiting the vulnerability. The Common Vulnerability Scoring System (CVSS) score ranges from 0-10, where 10 is the highest severity. The CVSS system was described in greater detail in Section 5.4. From our previous vulnerability assessment, we can extract the CVSS identified scores, and plot the numbers in a table.

CVSS Score	Count	Percentage
0-1	36098	9.0%
1-2	0	0.0%
2-3	0	0.0%
3-4	1752	0.44%
4-5	49649	12.38%
5-6	120151	29.97%
6-7	46369	11.57%
7-8	111270	27.76%
8-9	0	0.0%
9-10	25255	6.3%
<b>Total</b>	<b>400882</b>	<b>100%</b>

Table 8.2: Distribution of vulnerabilities by CVSS Scores

The presented table showcases the distribution of CVSS scores associated with the detected CVEs. Analyzing the distribution of CVSS scores reveals that the identified vulnerabilities show a broad range of severity levels, with a median score of 6-7.

Vulnerabilities with a high CVSS score of 9 to 10 are considered critical and require immediate attention. Knowing the CVSS score of a CVE can help security teams to prioritize the vulnerability by assessing the most critical vulnerabilities that require immediate attention. On the other hand, vulnerabilities with scores of 5 or below are generally regarded as less severe and can be addressed at a later stage, allowing security teams to allocate resources appropriately and focus on mitigating the most pressing risks first. This prioritization approach enables organizations to enhance their overall security posture by efficiently addressing vulnerabilities based on their severity levels.

## Chapter 9

# Discussion

This chapter will compare, analyze and discuss the results presented in the previous chapter, along with the challenges encountered during the implementation of a software and vulnerability scanning service. Through this discussion, the intention is to provide a wider context of the findings in the results chapter.

The discussion chapter will also cover potential areas for future work that can support the project. By exploring additional possibilities for investigation and development, this section will contribute to the overall progress and enhancement of the project's results.

### 9.1 Port Detection Compared to Shodan

From the data gathered in Section 8.2, we can compare the results against data collected from the Shodan search engine platform. A thorough evaluation of the developed scanning service can be achieved by comparing it to a well-established service and examining the similarities and differences between the two datasets. This will provide valuable insight into its effectiveness.

We can compare the occurrences for the most common ports against ports reported by Shodan, as mentioned in the source referenced in [57], obtained in March 2023.

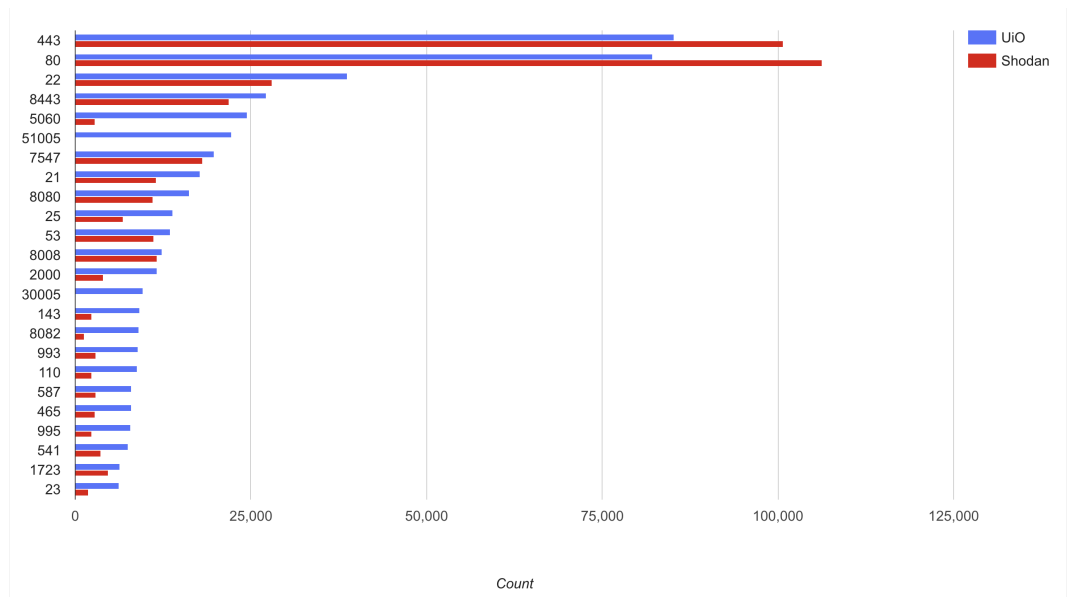


Figure 9.1: List of the 20 most frequent open ports as of March 24, 2023.

An analysis of reported open ports from both the UiO scanner and Shodan indicates similar patterns. The results are sorted based on those reported by UiO rather than Shodan. As previously discussed in Section 8.2, the most occurring open ports are associated with web traffic. The identification of web ports 80 and 443, which are commonly used for HTTP and HTTPS protocols, was observed by both Shodan and the locally hosted scanner at UiO. Notably, Shodan’s detection of these ports exceeded the locally developed scanner’s detection rate by approximately 15 percent.

When comparing the remaining scanning result data with the Facet Analysis tool of the Shodan database, the most notable difference in port occurrence is observed for the lack of Shodan identifying the ports 51005 and 30005, known to be associated with the TR-069 application protocol. A potential reason for this is explained later in Section 9.4.

## 9.2 Service Detection Compared to Shodan

The primary objective of this section is to perform a comparative analysis of the detection capabilities concerning services running on Norwegian hosts. This analysis is based on observations made by the UiO scanner and the data collected from the Shodan search engine. The aim is to assess the effectiveness of the UiO Robot in detecting services in comparison to the well-established Shodan service.

To accomplish this, data was gathered from both sources in March 2023. The detected services by the UiO robot are previously presented in Figure 8.3. The analysis will focus on identifying the most frequently detected



services in Norway by Shodan and compare them with the number of times the UiO robot detected them. The following figure was generated to represent the findings.

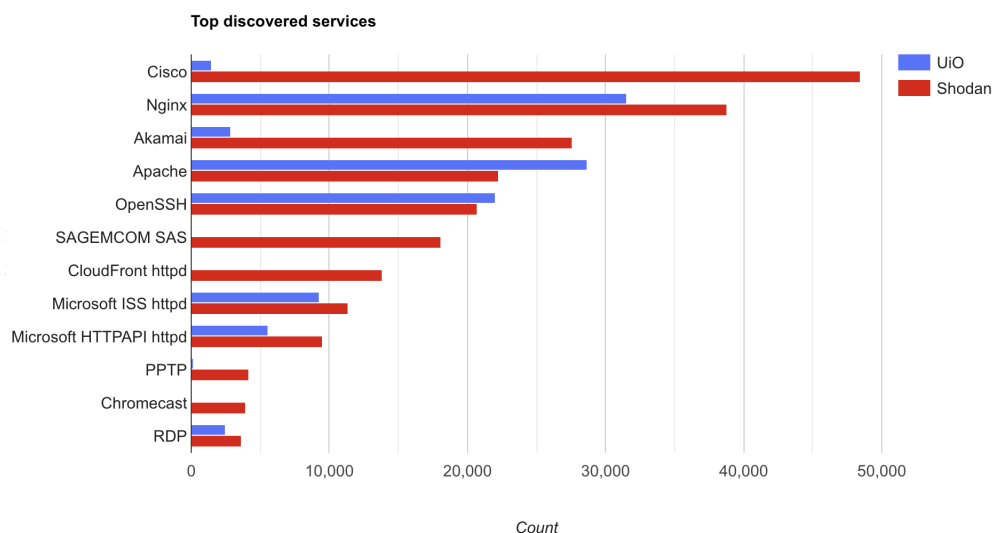


Figure 9.2: Comparing Shodan and UiO robot service detection results

Upon examining the comparison, it becomes evident that the UiO robot fails to identify all of the most commonly detected services identified by Shodan. One possible reason for this could be that a considerable number of detected ports lack accurate product descriptions. Consequently, it becomes crucial to explore alternative parameters that can facilitate accurate identification of the product description. Shodan utilizes TLS fingerprinting as a mechanism to identify more specific services, thereby enabling the determination of the product even in situations where the product field is not present. This approach is particularly useful in cases such as identifying the Chromecast product, displayed second to last in Figure 9.2.

The cause of the insufficient detection of Cisco services can be traced back to the compilation of the frequently opened port list, as outlined in Section 6.4. According to the Shodan search engine, these services are commonly found on TCP port 161. The data obtained in Section 8.2 reveals that port 161 has only been identified 2851 times by the UiO scanner, ranking it as the 339th most frequently opened port. Ideally, performing banner grabbing and version detection on this port would have been preferable. However, unfortunately, as explained in Section 8.3, the detailed banner grabbing had to be terminated due to numerous complaints received towards UiOCERT regarding the network activity generated by the scanner.

Nonetheless, before being stopped, the scanner was able to perform detailed banner grabbing and vulnerability identification on the top 35 identified services towards 616,405 ports. This accounts for over 30 percent of the 2 million open ports detected in Norway, indicating a moderate level of

coverage.

### 9.3 Vulnerabilities Detected Compared to Shodan

This section aims to compare the detection of software vulnerabilities classified with a CVE number detected by the UiO SecurityLab Robot against the public internet search engine Shodan. The results will be analyzed to determine how well the UiO Robot compares to the well-established Shodan service. The data gathered from both the Shodan service and UiO Robot was retrieved in March 2023. To compare the data, the following figure was generated to represent the findings of the most frequently detected CVEs by Shodan in Norway, compared with the number of times the UiO robot detected them.

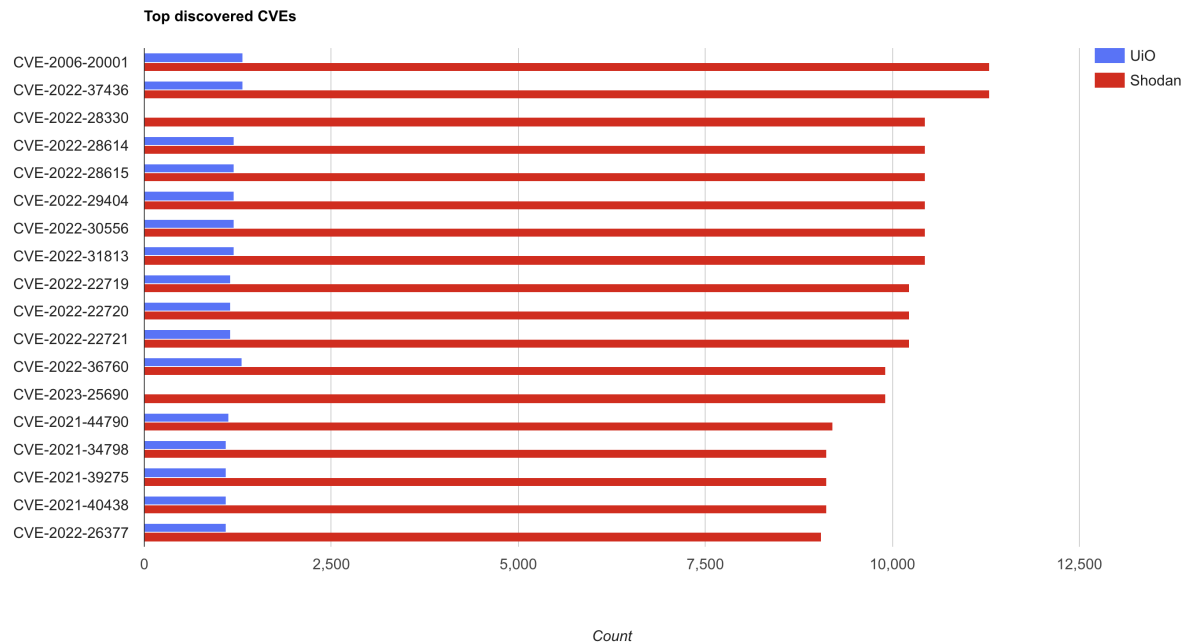


Figure 9.3: Comparison of Shodan vs. UiO Robot CVE detection

The graph presents a comparison of findings, indicating that the number of CVEs identified by Shodan is greater than those detected by the UiO scanner. Additionally, the data reveal that certain CVEs, which were frequently identified by the UiO Robot scanner in Figure 8.4, were either less frequently detected or not detected at all by the Shodan services.

The presence of the top 41 CVE vulnerabilities detected by Shodan correlates with Apache web server vulnerabilities, which is not surprising considering it is one of the most commonly detected services. [58]. The findings differ from those obtained by the UiO scanner, where OpenSSH

emerges as the most frequently identified vulnerability. Thus, there is a disparity in the identification of vulnerabilities between the two approaches.

As discussed in the previous section, some of the missing identified vulnerabilities could be the result of the scanner having to stop performing vulnerability identification, due to complaints received about the network activity generated by the scanner machine.

## 9.4 Analyzing the Detected Results

The results obtained from the network scanning experiments reveal comparable detection to the Shodan search engine. Analysis of the service and vulnerability detection activities conducted earlier revealed that web services on ports 443, 80, 8443, 8080, and 8008 are the most commonly exposed and vulnerable services. These findings align with the results obtained from the Shodan search engine service. This highlights the importance of protecting web services, as they are highly exposed to incoming attacks, especially since the top 41 CVEs identified by Shodan are related to Apache web servers.

In terms of web server product security, Figure 9.2 of the product banner information illustrates that the Nginx web server is more widely used than Apache. However, by looking at the identified CVEs, Apache exhibits a higher vulnerability detection than Nginx, indicating that it is generally more prone to vulnerabilities. While web servers ranked highest in vulnerability detection according to Shodan, it is worth noting that among the top vulnerabilities detected by UiO, there are multiple identified vulnerabilities specific to devices running the OpenSSH service.

The reason for the high amount of identified CVE vulnerabilities in Section 8.3.1 is due to that multiple identified vulnerable services have multiple CVEs assigned for each identified vulnerability.

The findings from performing a selected scan of the home provider internet service provider (ISP) in Section 6.4 revealed a substantial number of devices with open ports related to TR-069. These devices listening for traffic on ports 30005, and 51005 indicate a relation to the TR-069 protocol, a protocol used for applications enabling remote management and provision access of the customer's ISP network router. Even though TR-069 was detected most amount of times, the port was not identified to the same extent when conducting a port scan toward the entire nation in Section 8.2. One possible explanation for this anomaly is that the service is specific for router devices hosted on Internet Service Providers (ISPs) networks, resulting in a higher rate of devices supporting the TR-069 service compared to other enterprise networks. Even though TR-069 is a commonly open port, the vulners database engine did only detect a few CVE vulnerabilities related to the service from the gathered data.

The TR-069 protocol is not detected within the Shodan search engine for the country of Norway, however, detection for the service is largely present in the United States. In the USA, the protocol is commonly on port 7547, however, in Norway, TR-069 is more present on ports 30005 and 51005. A reason for this can be that Shodan only apparently scans for about 1225 ports, mentioned earlier in section 4.5, and the ports 30005 and 51005 do not appear to be in one of them.

The TR-069 protocol does not appear to be detected within the Shodan search engine for the country of Norway. However, considerable detection of this service is observed in the United States. In the USA, the protocol is commonly found on port 7547, whereas in Norway, TR-069 is more frequently encountered on ports 30005 and 51005. One possible explanation for the lack of detection could be that Shodan's scanning coverage is limited to approximately 1225 ports, as previously mentioned in Section 4.5, and ports 30005 and 51005 do not appear to be included among them. This observation highlights a constraint of Shodan's scanning capabilities, as it focuses on a relatively limited range of approximately 1225 ports, thereby restricting the detection of potential services.

The analysis comparing the most common ports listed by Nmap with the results obtained from the test in Section 8.1 shows that the usage of specific ports has changed significantly over the past 15 years. This finding suggests that the currently stored popularity data by Nmap may not be up-to-day for the services which are currently utilized today and may require a change. When conducting network reconnaissance using the Nmap tool, it is typical to specify parameters such as *-F*, *-top-ports*, or no port parameter at all. These commands indicate the target port number, utilizing either a fast scan or a "top-ports" scan that emphasizes selecting the most frequently used ports. By default, Nmap uses the registry list from "Nmap-services" to conduct scanning activities, when no specific ports are specified by the user. It should be noted that if this list is outdated, it can significantly impact the efficiency of users' deployed scans, requiring them to scan a larger number of ports to achieve optimal results.

Considering that Nmap is an open-source tool, there exists the possibility of contributing to the project by creating pull requests in the associated GitHub repository and providing a more updated popularity number of utilized ports and services. This initiative would enable users to conduct network scans that are more effective, accurate, and likely to identify a greater range of services.

### 9.4.1 Unveiling False Positives: SYN Cookies

During the comprehensive scan conducted across all possible TCP ports (Section 8.2), an abnormal pattern was detected on some devices, where they reported a large number of open ports. All these devices were contained within the same /13 network, including over 16,000 IP addresses, each reporting on exactly 535 open ports. Consequently, this scanning activity led to the identification of over 8.5 million additional open ports on a national scale.

After some investigations suggest that this behavior can be attributed to the utilization of an SYN cookie technique implemented by firewalls. SYN cookies are employed as a defensive measure against SYN Flood Denial of Service (DoS) attacks. This technique limits the total number of half-open connections, in order to prevent attackers from overloading the server with connections. The firewall responds with an SYN-ACK packet to the client before receiving the final ACK packet to complete the three-way handshake. The technique mitigates the server having to keep a backlog of half-open connections, which can be used to overload the server with connection requests.

While the use of SYN cookies helps to prevent DoS attacks, it can also prevent the detection of network reconnaissance activities, by falsely presenting a port as open, as the result of an incoming SYN packer will always respond with an SYN-ACK packet, regarding of the ports state. This will result in the port status falsely classifying the port as open and will result in an overestimation of the number of open ports on networks. In light of this issue, the NSM Allvis Nord service recommends that its users should whitelist their scanner's IP addresses, to avoid false positive detection of open ports [59].

## 9.5 Complains and Detection for the Conducted Scanning Activity

As mentioned briefly in Section 8.3, UiO-CERT reached out, informing of several reactions received regarding activities from the IP address used for collecting information. The cause or source of the commendations received by the actors involved was not explicitly specified, but it is presumed to originate from individuals or organizations within the Norwegian IP range infrastructure.

It is likely that the reactions were due to a large number of port requests sent to internet-exposed infrastructures within a short period. Large-scale reconnaissance as performed by the UiO SecurityLab robot could also be implemented by threat actors, seeking simple ways to intrude into an actor's network. The received traffic could also in some cases act as a low-scale SYN flood DoS attack, based on the number of SYN packets sent.

## 9.6 Choosing Appropriate Scan Intervals for Network Assessments

Selecting the appropriate scan frequency is a crucial aspect for effectively detecting trends in the threat landscape. The assessments would need to be performed at regular intervals, in order to detect changes in activity from internet-connected devices. However, frequent network assessments can result in network noise and produce only slight variations in the results obtained. Regarding assessment frequency, the Shodan search engine's feature "*Trends*" displays a monthly interval. This approach allows for detecting new trends in online internet devices over time.

In our study from Section 6.3, we determined that performing detection towards all ports would take 242 days, making large-scale scanning activities at this scope an impractical task. As discovered in Section 4.5, Shodan only performs requests towards about 1225 ports, according to a Twitter post by their official account as of 2020 [30], making a monthly approach more feasible. In our study from Section 6.5, we determined that conducting a vulnerability assessment on all ports identified through the use of Masscan on the top 1000 ports would require approximately 40 days when performed from a single host. This makes a monthly interval frequency impractical for the UiO Robot scanner. In Section 9.8, methods to reduce the scan time will be discussed.

In the scenario where the UiO scanning Robot is hosted by a provider that permits constant network reconnaissance, it could be set up to conduct scans every two months. This will enable students and researchers to compare the outcomes of the scans conducted during alternate months.

In conclusion, selecting an appropriate scanning frequency is a crucial factor in network scanning that necessitates thoughtful consideration. Choosing a bimonthly scanning frequency strikes a balance between conducting regular scans and preventing excessive network noise and complaints.

## 9.7 Limitations

The results obtained in this study have some limitations that must be taken into consideration. One significant limitation is the slow performance of the scanning process, which restricts the frequency of scans that can be conducted. This can affect the comprehensiveness of the scan results and limit the ability to detect changes in the threat landscape, as discussed in the previous section. Additionally, the lack of detailed service descriptions for detected ports may limit the ability to identify all potential threats.

In addition to these limitations, several challenges emerged during the project's development. One major challenge was the occurrence of database indexing bugs, which resulted in less accurate mapping of certain CVEs than expected. Furthermore, not all records were successfully added to the database during indexing, impacting the completeness of the data. The GitHub repositories used for data ingestion, as explained in Section 7.3.1, did not appear to support the ingestion of such large amounts of data as collected in this project. These challenges highlight the need for ongoing development and refinement to ensure the effectiveness and accuracy of the scanning process.

Furthermore, it is important to note that the various tests were conducted at different times, which may have led to variations in the results. Certain services might have been inaccessible during one scan but available during another. Furthermore, some of the previously scanned IPs could be behind a firewall is configured to block incoming traffic from IP addresses previously known to perform reconnaissance activities. To mitigate this issue, future research could consider conducting each testing stage from different machines with different IP addresses.

## 9.8 Further Research and Work

In this section, we discuss possible directions for further research and development to address the limitations of the scanning process and enhance its effectiveness. One area that requires attention is the optimization of the scanning process to improve its performance and enable more frequent scans. To achieve this, it would be necessary to explore ways to reduce the time required for port and banner grabbing, by further analyzing the overhead and latency introduced by virtual machines and hypervisors, which can impact the speed of Nmap scans. One possible solution could be to use a bare metal system, which can offer more direct access to hardware resources, allowing Nmap to run faster and more efficiently. Additionally, the scanner could be set up with multiple computers on different IP addresses scanning a section of their own, which would reduce the scanning time and noise detection by ISP or firewalls. This technique is reportedly used by the Shodan search engine, having multiple hosts with each its own public IP address, configured to perform network reconnaissance activities

towards separate parts of the internet.

Furthermore, implementing a way of notifying about the intent of the scanning activity could also be implemented. Deploying a website with information, and linking the website as a custom user agent when conducting scanning activities could be a helpful step in avoiding complaints for the scanning activity. Security personnel could then be informed about the activity, and a form could also be created in order for companies to fill out and send in, to avoid being scanned further. This feature would help to reduce the incoming complaints mentioned in Section 9.5.

Another area for further development is usability which allows users to launch network scan activities. For instance, implementing additional features on the webpage implemented in Section 7.4, allowing for more fine control of the scanner, without having to utilize a Linux terminal, would make it easier for users to customize the scanning process to their needs and preferences.

The scanning of UDP ports is also an important step to detect further services. As mentioned in Section 4.3, UDP services are much harder to detect than TCP and also require much more time. This is why the protocol was cut out in this project. Further research into methods for detecting services on the protocol could be conducted, which would help in allowing for the detection of UDP services.

The implementation of selecting target IP addresses at random would reduce concurrent traffic towards the same network infrastructure at once, reducing the probability of performing a denial of service attack towards switches and routers not capable of handling large amounts of incoming SYN packets.

Lastly, some improvements could be made to the storage and management of scan results. For instance, adding better timestamps for the results stored in Elasticsearch would make it easier to analyze and interpret the data. Furthermore, adding a button or function for the user to easily empty the stored data in Elasticsearch would improve the user experience and help manage the storage space.



## 9.9 Legal and Ethical Implications

Port scanning, although not necessarily illegal, can result in severe consequences. Since it is often the first step attackers use to scope out a target, which makes the intent of the scanning activity malicious. However, the objective of this project is not to facilitate attacks, but rather to pinpoint devices that are vulnerable due to open ports.

From a legal perspective, port scanning is a bit unclear because the internet is open and not controlled by one authority. In order to successfully prosecute individuals for port scanning, it is necessary to provide evidence that demonstrates their intent to gain unauthorized access or infiltrate a system, rather than simply performing a port scan.

Although it is uncommon, there have been instances of legal cases involving port scanning without subsequent hacking attacks. One notable case involves Scott Moulton, who was contracted to connect a router between the Georgia Police Department with the 911 call center [60]. Scott, concerned about potential risks to the security of the E911 Center, took the initiative to conduct preliminary port scanning on the involved networks. During this process, he scanned the Cherokee County web server. This web server was maintained by the consulting firm VC3, which notified the police about Scott's scanning activity. This caused Scott to lose his contract with the 911 call center, and was arrested for violating the "*Computer Fraud and Abuse Act of America*". The case against Scott was dismissed before trial, indicating that the case lacked sufficient legal grounds or valid arguments to proceed.

Port scanning's legality is unclear due to the decentralized internet. Differentiating between malicious intent and legitimate security assessments is crucial. For a more in-depth discussion of legal issues and similar cases, please refer to the source mentioned above for Scott's case at [60].

# Chapter 10

## Conclusion

The concluding chapter of this thesis will summarize the key findings from the results, addressing the research questions outlined in the introduction. The central research question proposed initially was to assess the number of devices in Norway that are accessible via the Internet, and the subsequent questions aimed to investigate the vulnerability status of these devices, as well as the challenges involved in creating a platform capable of scanning for vulnerabilities.

The initial research question (**Q.1**) aimed to assess the number of internet-accessible devices in Norway. The results demonstrated that the developed platform scanner, as presented in the preceding chapters, can facilitate nationwide network scan activities, providing a reliable answer to this question.

The second question (**Q.2**) further explored how we could identify device vulnerabilities and how our solution compares to other available alternatives. Our scanner could identify vulnerabilities for internet-exposed outdated software, resulting in 400,882 identified CVE vulnerabilities as of March 2023. This finding shows that the scanner can conduct detailed vulnerability assessments, directly answering our second research question.

The final question (**Q.3**) relate to the obstacles encountered when creating a platform capable of identifying service and vulnerabilities. During the development process, we overcame challenges such as slow packet rates, outdated frequent port lists, complaints, and legal issues. Despite these obstacles, we successfully developed the first demo version of the UiO SecurityLab robot to scan specific parts of the Internet.

We experimented with different ways of addressing the port status for network devices, and the results provided insights into their comparable effectiveness. In our comparison, Masscan emerged as a faster option than Nmap due to its different scanning approach, speed optimization, smaller packet sizes, and absence of service discovery. However, the lack of certain features in Masscan suggests that Nmap may be more suitable for more complex detailed types of scans.

Further, our study revealed that Nmap's most commonly used port list appears outdated, indicating a significant shift in online services since 2008. This discovery highlights the need for updated port lists to accurately and effectively scan today's internet landscape.

The Shodan search engine provided a noteworthy representation of the Norwegian service landscape, demonstrating its ability to capture and present valuable insights into the diverse range of services present in Norway. We also conclude that the solution often fails to scan all meaningful ports. This oversight can result in missing service detection, resulting in the need for a more comprehensive scanning solution.

In summation, this thesis contributes valuable insights into the landscape of internet-accessible devices in Norway, their vulnerability status, and the development of a platform to conduct nationwide assessments.



# Chapter 11

## Appendix

The full code implemented for this project can be found at the following GitHub repository: <https://github.com/haraldaarz/MasterThesis>.

### List of Selected Ports from Nmap's Services File

The following displays the 14 most popular services on the internet according to Nmap's "Well Known Port list".

To generate the output, the following Linux command was used:

```
awk '$2~/tcp$/' /usr/share/nmap/nmap-services | sort -r -k3 | head -n 14
```

#### Results:

http	80/tcp	0.484143	# World Wide Web HTTP
telnet	23/tcp	0.221265	
https	443/tcp	0.208669	# secure http (SSL)
ftp	21/tcp	0.197667	# File Transfer [Control]
ssh	22/tcp	0.182286	# Secure Shell Login
smtp	25/tcp	0.131314	# Simple Mail Transfer
rdp	3389/tcp	0.083904	# Microsoft Remote Display Protocol
pop3	110/tcp	0.077142	# PostOffice V.3
microsoft-ds	445/tcp	0.056944	# SMB directly over IP
netbios-ssn	139/tcp	0.050809	# NETBIOS Session Service
imap	143/tcp	0.050420	# Interim Mail Access Protocol v2
domain	53/tcp	0.048463	# Domain Name Server
msrpc	135/tcp	0.047798	# epmap   Microsoft RPC services
mysql	3306/tcp	0.045390	

## 11.1 Code for the Main Scanning Function

```
import os
import time
import sys
import argparse

# print current date
currentDate = time.strftime("%d-%m-%Y-%H:%M:%S")
scanfile = "masscanOUT3.txt"
portsandIP = "portsandIP.txt"
sortedPorts = "sortedPorts.txt"
onlyPorts = "onlyPorts.txt"
uPorts = "uniqPorts.txt"

def inputIps(ips): # Stores provided IP addresses in a file
    os.system("rm hosts2.txt")

    with open ("hosts2.txt", "a+") as file:
        for ip in ips:
            file.write(ip + "\n")
        else :
            print("No IP addresses entered")

def discoveryScan():
    # Discovery Scan
    # masscan all ports on all hosts
    # Output number of open ports and IP addresses to a file
    date = time.strftime("%d-%m-%Y-%H:%M:%S")
    os.system('masscan -iL hosts.txt -p1-65535 --max-rate 100000
-oX discoveryScan' + date + ' --wait 20')
    os.system('cp hosts.txt ' + "discoveryScan_" + date)

def masscanExecute2(ports, rate):
    print("Starting masscan")
    os.system('masscan ' + '-iL hosts.txt -p' + ports
+ ' --rate ' + rate + ' -oL ' + ' masscanOUT3.txt --wait 20')

# Create a file for each open port
def uniquePorts():
    print("Creating files for each port")
    os.system("awk '{ print $3 }' " + scanfile +
" | sort -u -n | grep '\S' | awk 'NF' >" + uPorts + "")
    print("Check 1")

    if not os.path.exists('ports'):
        os.makedirs('ports')
    with open(uPorts, "r+") as f:
        for line in f:
            filename = "ports/" + line.strip() + ".txt" # port filename
```

```

        if filename == "0.txt":
            print("A file with the name 0.txt is created")
            os.system("touch " + filename) # create a file for each port
            print("Created the file: " + filename)
            # if the file is empty, remove it
            if os.stat(filename).st_size == 0:
                os.remove(filename)

def parsefile(): # Takes input from masscan -oL file
    print("Parsing ports and IP addresses to corresponding files")
    with open(scanfile, "r+") as f:
        if f.read(1):
            os.system("awk '{print $3 \" \" $4}' " + scanfile +
                " | grep '\S' > " + portsandIP + "")
        else:
            print("File is empty 2")
            sys.exit()
    with open(portsandIP, "r+") as f:
        if f.read(1):
            os.system("sort -k1 -n -t ' ' " + portsandIP +
                " | awk 'NF' > " + sortedPorts + "")
        else:
            print("File is empty 3")
            sys.exit()

    with open(sortedPorts, "r+") as f:
        for line in f:
            port = line.split()[0] # Get the port
            ip = line.split()[1] # Get the IP

            #store the port and ip in a file named after the port
            filename = "ports/" + port + ".txt"
            with open(filename, "a+") as f:
                f.write(ip + "\n")

    print("Done parsing ports and IP addresses to corresponding files")

def mostUsedPortOrder(): # Start nmap on the most used ports first
    ports = sortedPorts
    os.system("cat " + scanfile + " | awk '{print $3}' |
        sort | uniq -c | sort -nr | awk '{print $2}' | grep '\S' > " + onlyPorts + "")

def nmapExecute():
    # run nmap on each of the port files. Starting with the most used port
    # nmap on the ports with banners found in the
    # masscan output file to separate files for each port
    if not os.path.exists('outputs'):
        os.makedirs('outputs')

```

```

print("Starting Nmap")

with open (onlyPorts, "r+") as f:
    for line in f:
        print("Ports:", line)

        port = line.strip()
        hosts = "ports/" + port + ".txt"
        outputFile = "outputs/nmapOutput-" + port + ".xml"

        os.system("nmap -sV -T5 -Pn -n --open --script=vulners
        -iL " + hosts + " -p " + port + " -oX " + outputFile) #+ " >/dev/null")

print("Done with Nmap")

if __name__ == "__main__":

    if len(sys.argv) == 2:
        port = sys.argv[1]
        masscanExecute2(port, "10000")

    if len(sys.argv) == 3:
        port = sys.argv[1]
        ips = sys.argv[2]
        with open("hosts.txt", "w+") as f:
            f.write(ips)
        masscanExecute2(port, "10000")

    if len(sys.argv) == 4:
        port = sys.argv[1]
        ips = sys.argv[2]
        rate = sys.argv[3]
        with open("hosts.txt", "w+") as f:
            f.write(ips)
        masscanExecute2(port, rate)

uniquePorts()
parsefile()
mostUsedPortOrder()
nmapExecute()
#discoveryScan()

```



## 11.2 Script to detect Favicon Hash

```
import mmh3
import sys
import codecs
import requests

if len(sys.argv) != 2:
    print(f"Usage: {sys.argv[0]} [Favicon URL]")
    sys.exit(0)

try:
    response = requests.get(sys.argv[1])
    favicon = codecs.encode(response.content, 'base64')
    hash = mmh3.hash(favicon)
    print(f"Favicon Hash: {hash}")
except Exception as e:
    print(f"Error occurred as: {e}", file=sys.stderr)
```

## 11.3 Masscan Service Discovery Test

### Terminal Output

```
Starting masscan 1.0.5 (http://bit.ly/14GZzcT) at 2023-02-09 21:11:59 GMT
Forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 16,099,072 hosts [1 port/host]
Rate: 0.00-kpps, 100.00% done, waiting 2 seconds, found 97,646
```

## 11.4 Nmap banner XML results

```
<host starttime="1676085874" endtime="1676088263">
<status state="up" reason="user-set" reason_ttl="0"/>
<address addr="78.26.22.37" addrtype="ipv4"/>
<hostnames></hostnames>
<ports>
<port protocol="tcp" portid="80">
<state state="open" reason="syn-ack" reason_ttl="48"/>
<service name="http" product="Apache httpd" version="2.0.65"
extrainfo="(Unix) mod_ssl/2.0.65 OpenSSL/0.9.8zf PHP/4.3.11" method="probed"
conf="10">
<cpe>cpe:/a:apache:http_server:2.0.65</cpe>
</service>
<script id="http-server-header" output="Apache/2.0.65 (Unix)
mod_ssl/2.0.65 OpenSSL/0.9.8zf PHP/4.3.11">
<elem>Apache/2.0.65 (Unix) mod_ssl/2.0.65 OpenSSL/0.9.8zf PHP/4.3.11</elem>
</script>
```

```

<script id="vulners" output="&#xa; cpe:/a:apache:http_server:2.0.65: &#xa;
&#x9;CVE-2011-3192&#x9;7.8&#x9;https://vulners.com/cve/CVE-2011-3192&#xa;
&#x9;CVE-2013-1862&#x9;5.1&#x9;https://vulners.com/cve/CVE-2013-1862&#xa;
&#x9;CVE-2012-0031&#x9;4.6&#x9;https://vulners.com/cve/CVE-2012-0031&#xa;
&#x9;SSV:20555&#x9;4.3&#x9;https://vulners.com/seebug/SSV:20555&#x9;*EXPLOIT*&#xa;
&#x9;EXPLOITPACK:FDCB3D93694E48CD5EE27CE55D6801DE&#x9;4.3&#x9;
https://vulners.com/exploitpack/EXPLOITPACK:FDCB3D93694E48CD5EE27CE55D6801DE&#x9;
*EXPLOIT*&#xa; &#x9;CVE-2012-0053&#x9;4.3&#x9;https://vulners.com/cve/CVE-2012-0
<table key="cpe:/a:apache:http_server:2.0.65"></table>
</script>
</port>
</ports>
<times srtt="69155" rttvar="69155" to="345775"/>
</host>

```

## 11.5 Logstash-codec-nmap

Sample of code mapping

```

def hashify_service(service)
  return unless service

  protocol = service.protocol rescue nil
  {
    'name' => service.name,
    'ssl' => service.ssl?,
    'protocol' => protocol,
    'product' => service.product,
    'version' => service.version,
    'hostname' => service.hostname, # This is just a string
    'device_type' => service.device_type,
    'fingerprint_method' => service.fingerprint_method.to_s,
    'fingerprint' => service.fingerprint,
    'confidence' => service.confidence
  }
end

```

## 11.6 Webpage

Screenshots of the webservers features

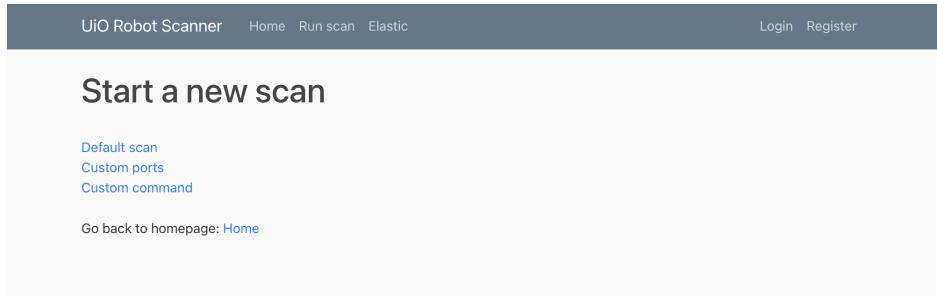


Figure 11.1: Web interface for starting a new scan

### 11.6.1 Scantypes

Default scan against the top ports

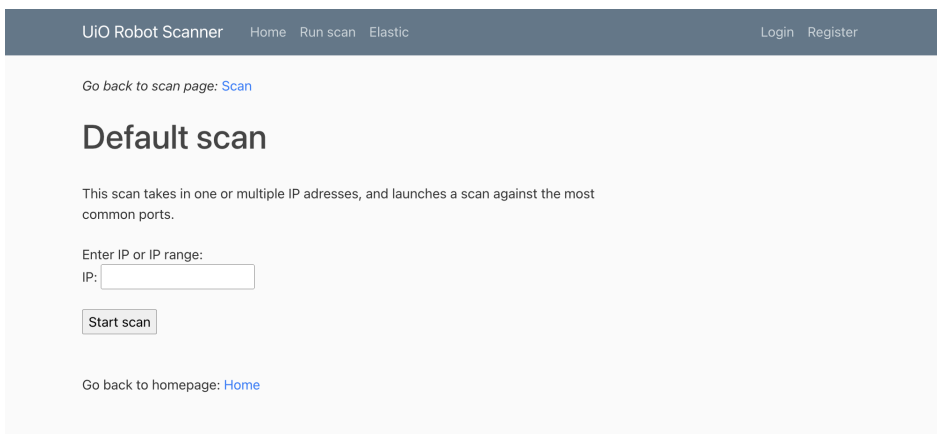


Figure 11.2: Web interface for default scan type

## 11.6.2 Custom port scan

UiO Robot Scanner Home Run scan Elastic Login Register

Go back to scan page: [Scan](#)

### Custom port scan:

Launch a towards one or multiple IP addresses, and specify the ports you want to include.  
Type the ports in the format "1,2,3." or "1-5"

IP:

Port:

Rate:

After the scan is done, you can watch the results here: [Elastic](#)

View database for the specified IPs:

Go back to homepage: [Home](#)

Figure 11.3: Web interface for custom port scan type

## 11.6.3 Custom Nmap or Masscan command scan

UiO Robot Scanner Home Run scan Elastic Login Register

Go back to scan page: [Scan](#)

Type custom Nmap or Masscan command:

Scan type:

Command:

Go back to homepage: [Home](#)

Figure 11.4: Web interface for custom command scan type

# Bibliography

- [1] Statista. *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025*. 2023. URL: <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/> (visited on 05/10/2023).
- [2] *CVE Details - Browse Vulnerabilities By Date*. URL: <https://www.cvedetails.com/browse-by-date.php> (visited on 14/05/2023).
- [3] Torjus Kleng Dahle. 'Large scale vulnerability scanning'. MA thesis. 2020. URL: <https://www.duo.uio.no/bitstream/handle/10852/79552/torjuskd-master-2020v14.pdf?sequence=8>.
- [4] Kristian Helgesen Torkveen. 'Internet Security Scanner'. MA thesis. 2021. URL: <https://www.duo.uio.no/bitstream/handle/10852/87009/Master.pdf?sequence=1>.
- [5] Rodney R Rohrmann, Vincent J Ercolani and Mark W Patton. 'Large scale port scanning through tor using parallel Nmap scans to scan large portions of the IPv4 range'. In: *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. 2017, pp. 185–187. DOI: 10.1109/ISI.2017.8004906. URL: <https://ieeexplore.ieee.org/document/8004906>.
- [6] Chao Yuan et al. 'The Design of Large Scale IP Address and Port Scanning Tool'. In: *Sensors* 20.16 (2020). ISSN: 1424-8220. DOI: 10.3390/s20164423. URL: <https://www.mdpi.com/1424-8220/20/16/4423>.
- [7] Roman Trapickin, Oliver Gasser and Johannes Naab. 'Who Is Scanning the Internet'. In: 2015. URL: [https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2015-09-1/NET-2015-09-1\\_11.pdf](https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2015-09-1/NET-2015-09-1_11.pdf).
- [8] Andrea Tundis, Wojciech Mazurczyk and Max Mühlhäuser. 'A Review of Network Vulnerabilities Scanning Tools: Types, Capabilities and Functioning'. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. Hamburg, Germany: Association for Computing Machinery, 2018. ISBN: 9781450364485. DOI: 10.1145/3230833.3233287. URL: <https://doi.org/10.1145/3230833.3233287>.
- [9] Hwankuk Kim, Taeun Kim and Daeil Jang. 'An Intelligent Improvement of Internet-Wide Scan Engine for Fast Discovery of Vulnerable IoT Devices'. In: *Symmetry* 10.5 (2018). ISSN: 2073-8994. DOI: 10.3390/sym10050151. URL: <https://www.mdpi.com/2073-8994/10/5/151>.

- [10] Ryan Jicha, Mark W Patton and Hsinchun Chen. 'Identifying devices across the IPv4 address space'. In: *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*. 2016, pp. 199–201. DOI: 10.1109/ISI.2016.7745469. URL: <https://ieeexplore.ieee.org/document/7745469>.
- [11] Capt. Meelo. *Finding the Balance Between Speed & Accuracy During an Internet-wide Port Scanning*. 29th July 2019. URL: <https://captmeelo.com/pentest/2019/07/29/port-scanning.html> (visited on 05/10/2023).
- [12] Waheed Ali H. M. Ghanem and Bahari Belaton. 'Improving accuracy of applications fingerprinting on local networks using NMAP-AMAP-ETTERCAP as a hybrid framework'. In: *2013 IEEE International Conference on Control System, Computing and Engineering*. 2013, pp. 403–407. DOI: 10.1109/ICCSCE.2013.6719998.
- [13] Unknown researchers. 'Internet Census 2012'. In: 2012. URL: <https://census2012.sourceforge.net/paper.html>.
- [14] OpenVAS, *Open Vulnerability Assessment Scanner*. URL: <https://www.openvas.org> (visited on 05/10/2023).
- [15] IVRE, *Instrument de veille sur les réseaux extérieurs*. URL: <https://ivre.rocks/> (visited on 05/10/2023).
- [16] Tenable. *Nessus*. URL: <https://www.tenable.com/products/nessus> (visited on 05/10/2023).
- [17] Norwegian National Security Authority (NSM). *Allvis NOR*. URL: <https://nsm.no/tjenester/allvis-nor/> (visited on 05/10/2023).
- [18] Shodan. *Shodan*. URL: <https://www.shodan.io/> (visited on 05/10/2023).
- [19] John Matherly. *Complete Guide to Shodan*. Last updated on 2017-08-23. Leanpub, Oct. 2016. URL: <http://leanpub.com/shodan>.
- [20] ZoomEye. Accessed: 2023-05-15. 2023. URL: <https://www.zoomeye.org/>.
- [21] Censys Search. Accessed: 2023-05-15. 2023. URL: <https://search.censys.io/>.
- [22] *Research Access to Censys Data*. Accessed: 2023-05-15. 2023. URL: <https://support.censys.io/hc/en-us/articles/360038761891-Research-Access-to-Censys-Data>.
- [23] ipinfo.io. *IPinfo*. 2023. URL: <https://ipinfo.io/countries/no> (visited on 03/04/2023).
- [24] Norkart. *Datainnbrudd Q&A*. <https://www.norkart.no/2022/05/13/datainnbrudd-q-a/>. May 2022. (Visited on 05/10/2023).
- [25] Mattis Vaaland. *Norkart-hacking skyldtes trolig en åpen port: – Viktig å støtte utviklerne som har tabbet seg ut*. May 2022. URL: <https://www.kode24.no/artikkel/norkart-hacking-skyldtes-trolig-en-åpen-port-viktig-a-stotte-utviklerne-som-har-tabbet-seg-ut/76072937> (visited on 05/10/2023).

- [26] Wikipedia. *List of TCP and UDP port numbers*. May 2023. URL: [https://en.wikipedia.org/wiki/List\\_of\\_TCP\\_and\\_UDP\\_port\\_numbers](https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers) (visited on 11/05/2023).
- [27] Internet Assigned Numbers Authority (IANA). *Service Name and Transport Protocol Port Number Registry*. URL: <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml> (visited on 11/05/2023).
- [28] Nmap Project. *Port Selection Data and Strategies*. 2009. URL: <https://nmap.org/book/performance-port-selection.html> (visited on 11/05/2023).
- [29] Gordon "Fyodor" Lyon. *Nmap Network Scanning*. City: Publisher Name, Jan. 2009. ISBN: 978-0-9799587-1-7.
- [30] ShodanHQ. *Tweet*. May 2020. URL: <https://twitter.com/shodanhq/status/1263329574525468672?s=20> (visited on 11/05/2023).
- [31] Wikipedia. *Banner grabbing*. June 2022. URL: [https://en.wikipedia.org/wiki/Banner\\_grabbing](https://en.wikipedia.org/wiki/Banner_grabbing) (visited on 11/05/2023).
- [32] Gordon "Fyodor" Lyon. 2009. URL: <https://nmap.org/book/scan-methods-connect-scan.html> (visited on 11/05/2023).
- [33] Gordon "Fyodor" Lyon. *Chapter 9. Nmap Scripting Engine, Usage and Examples*. 2009. URL: <https://nmap.org/book/nse-usage.html#nse-categories> (visited on 11/05/2023).
- [34] Gordon "Fyodor" Lyon. *Chapter 4. Port Scanning Overview, What Are the Most Popular Ports?* 2009. URL: <https://nmap.org/book/port-scanning.html#most-popular-ports> (visited on 13/05/2023).
- [35] *Scan Time Reduction Techniques*. URL: <https://nmap.org/book/reduce-scantime.html> (visited on 11/05/2023).
- [36] Robert David Graham. *Masscan*. <https://github.com/robertdavidgraham/masscan>. Released: September 2013 (Pre-release).
- [37] Gordon "Fyodor" Lyon. *Nmap OS Detection Database*. 2009. URL: <https://nmap.org/book/nmap-os-db.html> (visited on 11/05/2023).
- [38] Trisul Network Analytics. *ja3prints*. <https://github.com/trisulnsm/ja3prints/blob/master/ja3fingerprint.json>. Added: March 1, 2018.
- [39] John Sansatart. *Shodan's Stored Favicon Hashes*. 2019. URL: <https://github.com/sansatart/scripts/blob/master/shodan-favicon-hashes.csv> (visited on 12/04/2023).
- [40] *Shodan Images*. URL: <https://images.shodan.io/> (visited on 11/05/2023).
- [41] FortyNorth Security. *EyeWitness*. <https://github.com/FortyNorthSecurity/EyeWitness>. Created by FortyNorth Security. (Visited on 11/05/2023).
- [42] Mitre Corporation. *Common Weakness Enumeration*. Page Last Updated: May 02, 2023. 2023. URL: <https://cwe.mitre.org/> (visited on 11/05/2023).

- [43] Mitre Corporation. *CWE VIEW: Weaknesses in the 2022 CWE Top 25 Most Dangerous Software Weaknesses*. Page Last Updated: May 02, 2023. 2022. URL: <https://cwe.mitre.org/data/views/2022.html> (visited on 11/05/2023).
- [44] Nmap Project. *Vulners NSE Script*. URL: <https://nmap.org/nsedoc/scripts/vulners.html> (visited on 11/05/2023).
- [45] Vulners. *Vulners Database*. URL: <https://vulners.com/> (visited on 11/05/2023).
- [46] Forum of Incident Response and Security Teams. *Common Vulnerability Scoring System (CVSS)*. 2023. URL: <https://www.first.org/cvss/> (visited on 11/05/2023).
- [47] *Shodan Search: Port 80 in Norway*. URL: <https://www.shodan.io/search?query=country%5C%3A%5C%22NO%5C%22+port%5C%3A80> (visited on 09/02/2023).
- [48] *Shodan - Open Ports in Norway*. <https://www.shodan.io/search?query=country%3A%22NO%22>. Accessed on [Insert Date].
- [49] Ollie N. *Introducing Scanning Made Easy*. National Cyber Security Centre. Jan. 2022. URL: <https://www.ncsc.gov.uk/blog-post/introducing-scanning-made-easy> (visited on 11/05/2023).
- [50] *Nmap Codec Plugin*. Elastic. Nov. 2022. URL: <https://www.elastic.co/guide/en/logstash/current/plugins-codecs-nmap.html> (visited on 11/05/2023).
- [51] Bennett Warner. *elk\_nmap*. [https://github.com/bennettwarner/elk\\_nmap](https://github.com/bennettwarner/elk_nmap). May 2021. (Visited on 11/05/2023).
- [52] Anthony Lapenna. *docker-elk*. <https://github.com/deviantony/docker-elk>. Accessed on [date]. Jan. 2015.
- [53] Chris Rimondi. *VulntoES*. <https://github.com/ChrisRimondi/VulntoES>. May 2014. (Visited on 11/05/2023).
- [54] Marco Lancini. *Offensive ELK: Elasticsearch for Offensive Security*. [https://github.com/marco-lancini/docker\\_offensive\\_elk](https://github.com/marco-lancini/docker_offensive_elk). July 2018. (Visited on 11/05/2023).
- [55] happyc0ding. *scan2elk*. <https://github.com/happyc0ding/scan2elk>. Mar. 2019. (Visited on 11/05/2023).
- [56] happyc0ding. *vulnscan-parser*. <https://github.com/happyc0ding/vulnscan-parser>. June 2019. (Visited on 11/05/2023).
- [57] *Shodan Facet Analysis for Ports in Norway*. URL: <https://www.shodan.io/search/facet?query=country%5C%3A%5C%22NO%5C%22%5C&facet=port> (visited on 24/03/2023).
- [58] *Shodan Facet Analysis for CVEs in Norway*. URL: <https://www.shodan.io/search/facet?query=country%5C%3A%5C%22NO%5C%22%5C&facet=vuln> (visited on 24/03/2023).
- [59] *Allvis Nord, Teknisk informasjon*. URL: <https://doc.allvis.no/#/technical?id=hvor-kommer-trafikken-fra> (visited on 13/05/2023).



[60] The Nmap Project. *Nmap Legal Issues*. 2008. URL: <https://nmap.org/book/legal-issues.html> (visited on 13/05/2023).