Høgli, Sander
Lygre, Jarl Tengesdal
Małecki, Wojciech
Marjara, Avleen Singh

Autoenum

Automatic mapping and exposure analysis of network endpoints

Bachelor's project in IT-Operations and Information Security

Supervisor: Erjon Zoto

May 2021

Bachelor's project

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

NTNU
Kunnskap for en bedre verden

Høgli, Sander
Lygre, Jarl Tengesdal
Małecki, Wojciech
Marjara, Avleen Singh

Autoenum

Automatic mapping and exposure analysis of network endpoints

Bachelor’s project in IT-Operations and Information Security
Supervisor: Erjon Zoto
May 2021

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

NTNU
Norwegian University of
Science and Technology

Abstract

IT-security has never been as important as it is now, with threat agents becoming more sophisticated by the second. This forces us to work hard to keep up and secure our services. The NTNU SOC wanted to implement a service **(Autoenum)** that would help them scan a network periodically from the Internet and/or an internal network. The project group was tasked to create this system, which would contain a scanner, a database and API to help give the system its functionality. The scanner should scan a network to look for hosts and open ports, which would be checked for vulnerabilities. The scan result would be saved in a database and would be made available through an API. The group used Scrum as their software development framework. Scrum helped the group with its agile approach, which includes sprints that were set up throughout the project period. Autoenum has evolved a lot during the project period. It now delivers everything the NTNU SOC asked for and some more as well. The group hopes Autoenum will be of value when NTNU SOC is conducting their work of keeping the NTNU network secure.

Sammendrag

IT-sikkerhet har aldri vært like viktig som det er nå, med trusselagenter som blir mer sofistikerte for hver sekund som går. Vi må jobbe hardt for å holde følge og sikre tjenestene våre. NTNU SOC ønsket å implementere en tjeneste (Autoenum) som skulle hjelpe dem med å skanne et nettverk med jevne mellomrom fra Internett og / eller et internt nettverk. Prosjektgruppen hadde fått i oppgave å lage dette systemet, som skulle inneholde en skanner, en database og API for å gi funksjonaliteten til systemet. Skanneren vil skanne et nettverk for å se etter maskiner, åpne porter, som vil bli sjekket for sårbarheter. Resultatet fra skannene blir lagret i en database og blir gjort tilgjengelig via en API. Gruppen brukte Scrum som rammeverk for programvareutviklingen. Scrum hjalp gruppen med sin smidige tilnærming, som inneholder sprinter som ble satt opp gjennom hele prosjektperioden. Autoenum har utviklet seg mye i løpet av prosjektperioden. Den leverer nå alt NTNU SOC ba om og litt mer. Gruppen håper Autoenum vil være av verdi for NTNU SOC, når de skal utføre arbeidet sitt for å sikre NTNU-nettverket.

Preface

The members of this bachelor project, Wojtek Malecki, Sander Høgli, Avleen Singh Marjara and Jarl Tengesdal Lygre, would like to thank Christoffer Vargtass Hallstensen, representing NTNU SOC for presenting us with an interesting task. Christoffer provided us with many valuable ideas and tips along the way, of which we are grateful. We would also like to thank our supervisor, Erjon Zoto for guiding us through the project and giving us a lot of valuable feedback throughout the project period.

Contents

Abstract . . . . **iii**
Sammendrag . . . . **iv**
Preface . . . . **v**
Contents . . . . **vi**
Figures . . . . **ix**
Tables . . . . **x**
Code Listings . . . . **xi**
Acronyms . . . . **xii**
Glossary . . . . **xiv**
1 Introduction **1**
1.1 Project Background 1
1.2 Purpose 1
1.3 Target groups 2
1.4 Group background and competence 2
1.5 Constraints 3
1.6 Roles 3
1.7 Project goals 4
1.8 About the report 4
2 Theory **6**
2.1 Software Development Life Cycle 6
2.2 Scrum 6
2.3 Infrastructure as code 8
2.4 Automation 9
2.4.1 Ansible 9
2.5 Network mapping 10
2.5.1 Scanning 10
2.5.2 Scan types 11
2.5.3 Vulnerability 11
2.6 Databases 12
2.7 Containers 13
2.8 RESTful API 13
3 Methodology **16**
3.1 Digital workspace 16
3.2 Scrum 18

3.3 Similar projects 19
3.4 Report structure 20
3.5 Technology 21
4 Design 24
4.1 Requirements 24
4.1.1 Scanner 25
4.2 Architecture 26
4.3 Components 27
4.4 Licence 27
5 Implementation 28
5.1 Github repository 28
5.1.1 Open source tools 28
5.2 Overview 29
5.2.1 Scanner 29
5.2.2 Autoenum MongoDB 33
5.2.3 Autoenum API 34
5.2.4 Autoenum screengrabber 36
5.2.5 Autoenum web interface 37
5.2.6 CVE database 39
5.2.7 CVE API and search 39
5.2.8 CVE Redis 40
5.2.9 Volumes 40
5.3 Setup and deployment 41
5.3.1 Recommendations 42
6 Testing 44
6.1 Test environment 44
6.1.1 Overview 44
6.1.2 Topology 44
6.1.3 Services 46
6.1.4 Roles 47
6.1.5 Deployment 47
6.2 Testing 48
6.3 Beta test 48
7 Discussion 50
7.1 Decisions 50
7.1.1 Docker 50
7.1.2 Scanner 50
7.1.3 Database 51
7.1.4 Structure 51
7.2 Purpose and usefulness 52
7.2.1 Detection and incident management 52
7.3 Results 53
7.4 Deviations 53
7.5 Project criticism 54

8 Conclusion . . . 55
8.1 Work ethic . . . 55
8.2 Further work . . . 55
8.3 Conclusion of the work . . . 56
Bibliography . . . 57
A Task description . . . 62
B Scanner code . . . 64
B.1 scanner.py . . . 64
B.2 cve_lookup.py . . . 69
C Docker . . . 70
C.1 Screengrabber . . . 70
C.2 API . . . 71
C.3 Web interface . . . 72
C.4 docker-compose . . . 75
D Playbooks . . . 78
D.1 Setup and deploy Autoenum . . . 78
D.2 Playbooks for test environment . . . 79
E Code for test environment . . . 83
E.1 heat.yaml . . . 83
E.2 env.yaml . . . 89
E.3 manager_boot.bash . . . 90
E.4 windows_boot.ps1 . . . 91
F Sample API response . . . 92
G Database . . . 102
G.1 Schema . . . 102
G.2 Document . . . 104
H Web interface screenshots . . . 107
I Gantt diagram . . . 110
J Repository structure . . . 112
K Sprint reviews . . . 114
L Preliminary report . . . 120
M Time report . . . 139

Figures

2.1 The Scrum Framework[28] 7
2.2 URI example [46] 14

3.1 Snippet of scrum board in Trello 17
3.2 Planned sprints 18
3.3 SDLC illustration 20

4.1 Flowchart for scanner 25
4.2 Architecture detailed 26

5.1 Example of calling MAC endpoint and response 35
5.2 Web site hosted on web server in test environment 36
5.3 Response from screengrabber 37
5.4 Web interface: Home. More screenshots in appendix H 38
5.5 Making request to CVE API and receiving response 39
5.6 CVE search web interface 40

6.1 Topology of test environment 45
6.2 Flowchart deployment of test environment 47

H.1 Web interface: Detailed view 1/3 107
H.2 Web interface: Detailed view 2/3 108
H.3 Web interface: Detailed view 3/3 108
H.4 Web interface: List view 109
H.5 Web interface: Search 109

Tables

1.1 Relevant competence 3

5.1 The different repos for the different parts of the project 28
5.2 Open source tools used in the project 28
5.3 Sample database schema 34
5.4 API endpoints 34

6.1 Instances in OpenStack test environment 45
6.2 Services running on the instances 46
6.3 Roles used in project 47

Code Listings

5.1 perform_host_discovery( ) 29
5.2 perform_portscan( ) 30
5.3 perform_tcp_scan( ) 30
5.4 perform_udp_scan( ) 31
5.5 merge_results( ) 31
5.6 take_screengrab( ) 32
5.7 insert_db( ) 33
5.8 Code for MAC endpoint 35
5.9 Code snippet screengrabber 36
5.10 MVC of web interface 37
5.11 Setup 41
5.12 Running 41
5.13 Possible configuration variables for Autoenum 42
5.14 Adding username and password to mongoDB 42
6.1 Bash-commands to launch test environment 48
B.1 scanner.py 64
B.2 cve_lookup.py 69
C.1 Screengrabber - app.py 70
C.2 API - app.py( ) 71
C.3 index.js/server.js 72
C.4 hosts.js 72
C.5 Docker compose 75
D.1 Main playbook to setup Autoenum 78
D.2 Main playbook to setup Docker 78
D.3 Playbook for Windows machines in test environment 79
D.4 Main playbook Linux based machines in test environment 82
E.1 heat.yaml - HEAT template test environment 83
E.2 env.yaml - Environment variables 89
E.3 manager_boot.bash 90
E.4 windows_boot.ps1 91
F.1 Sample API response 92
G.1 Sample database schema 102
G.2 Sample database document 104
J.1 Directory 112

Acronyms

ACK	Acknowledgement. 11, 29
API	Application Programming Interface. 1–3, 13, 21, 24, 34, 48, 51
ARP	Address Resolution Protocol. 29
Bash	Bourne Again Shell. xi, 2, 21, 48
CPE	Common Platform Enumeration. 24, 30, 45
CVE	Common Vulnerabilities and Exposures. 12, 25, 29, 46
GRC	Governance, Risk and Compliance. 1
HOT	Heat orchestration template. 44
HTML	Hypertext Markup Language. 13, 23
HTTP	Hypertext Transfer Protocol. 13
IaaS	Infrastructure as a service. xvi
IaC	Infrastructure as code. 8, 9
ICMP	Internet Control Message Protocol. 11, 29
IEEE	Institute of Electrical and Electronics Engineers. xvii
IP	Internet Protocol. xiv, xv, 10, 24, 29, 50
JSON	JavaScript Object Notation. 3, 12, 13, 19, 24, 31
MAC	Media access control. xiv, xv, 10, 24, 30, 34
NTNU	Norges teknisk-naturvitenskapelige universitet (Norwegian university of science and technology). 1, 2, 27, 51
OS	Operating system. xiv, xv, xvii, 10, 25, 30, 34, 40, 44

POSIX Portable Operating System Interface. 10

REST Representational state transfer. 1, 3, 13, 24, 34

SDLC Software Development Life Cycle. 4, 6, 20, 51

SOAP Simple Object Access Protocol. xviii

SOC Security Operations Centre. 1, 2, 56

SQL Structured Query Language. 12

SSH Secure Shell. 10, 46

SSL Secure sockets layer. 30

SYN Synchronize Sequence Number. 11, 29

TCP Transmission Control Protocol. 11, 25, 29, 30, 51

UDP User Datagram Protocol. 11, 26, 31

UUID Universally Unique Identifier. 34

VM Virtual machine. 13, 44, 46, 48

XML Extensible Markup Language. 13

YAML YAML Ain’t Markup Language. 10, 54

Glossary

Access control list is the mechanism which provides access control for a system. This can be done by explicitly listing IPs of hosts or networks which are allowed to pass through the access control (whitelisting), or by listing the ones who are denied access (blacklisting) [1]. 43

Address Resolution Protocol is a protocol used to find MAC addresses associated to IP addresses. xii

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. xvii, 3, 9, 24, 41, 47

Application Programming Interface is a computing interface that defines interactions between multiple software intermediaries. xii, 13

Bootstrap is an open source Cascading Style Sheets framework . 27

Bourne Again Shell is a command language often used for Unix based machines. xii

Cascading Style Sheets is a language used to design the visual layout of Hypertext Markup Language. xiv

Cloud is the availability of computer resources over the Internet, especially storage and computing power, without direct active management by the user. The term is generally used to describe massive data centers that is available to users [2]. 8

Cluster is multiple computers that work together, but is viewed as one single unit. 8

Common Platform Enumeration is a structured notation when writing OS' and software [3] . xii

Common Vulnerabilities and Exposures "is a list of publicly disclosed computer security flaws. When someone refers to a CVE, they mean a security flaw that's been assigned a CVE ID number." [4]. xii

Container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another [5]. xv, 13, 22, 29, 33, 48

cron is a utility used to schedule tasks, such as running a command or a script on a Unix-based OS. 41

Discord is a VoIP, instant messaging and digital distribution platform designed for creating communities. 16

Dockerfile is the document that has all the necessary commands to build the desired Image [6]. xv

Dockerize is the process of preparing, deploying and running of applications in a Container. The process includes building a docker Image based on a Dockerfile which then can be deployed. 26, 33, 50

Extensible Markup Language is a language to encode documents in a tree-structure. The format is human and machine readable. xiii

Flavor is a tier used to describe the size of disk, number of CPU cores and RAM-size in OpenStack. 45

GET is a HTTP method used to get data from a specified resource. 34

GitHub is a provider of Internet hosting for software development and version control using Git. 16, 39, 41, 47

Heat is the main project in the OpenStack Orchestration program, it implements an orchestration engine to launch infrastructure based on templates [7]. xv, 18

Heat orchestration template is a template format supported by the Heat. xii

Heat stack is the collection of the resources created when running the HOT template. Resources can include: networks, routers, servers, network interfaces, storage devices and more . 48

Host object is our definition of all the aggregated data on one particular host. Includes IP address, MAC address, Ports etc.. . 27, 31, 33, 34, 51

Hypertext Markup Language is the World Wide Web's core markup language and is widely used for documents designed to be displayed in a web browser [8]. xii, xiv

Hypertext Transfer Protocol is a protocol for transferring text and other media on the World Wide Web. xii

Image is a copy of computer system which can be used to start a new instance of a system [9]. xv, 45

Infrastructure as a service is a cloud computing model that delivers computing, network and storage resources. xii

Infrastructure as code is an approach to infrastructure automation based on practices from software development. It emphasizes consistent, repeatable routines for provisioning and changing systems and their configuration. xii, 8

Internet Control Message Protocol "is a supporting protocol in the Internet protocol suite. It is used by network devices, to send error messages and operational information indicating success or failure when communicating with another IP address" [10]. xii

Internet Protocol is a unique address on a private network or on the Internet. xii, xviii

JavaScript Object Notation is a language independent data-interchange format. xii

Library is "resources used by computer programs, often for software development. These may include configuration data, documentation, help data, message templates, pre-written code and subroutines, classes, values or type specifications" [11]. 29

Media access control is the layer that controls the hardware responsible for interaction with the wired, optical or wireless transmission medium. xii

Microservice is an architectural style for making a distributed application, where each component of the application acts as an independent service[12] . 3, 24, 27

MongoDB is a document database, which means it stores data in key-value pairs as opposed to traditional relational databases which use tables. 12, 51

Nmap is a free and open-source network scanner. 10, 29, 39, 52

OneDrive is a file hosting and sharing service by Microsoft [13] often used organizations. 16

OpenStack is an open standard cloud computing platform, often used to deploy infrastructure as part of IaaS [14]. xvii, 24, 44, 48

Overleaf is a collaborative online LaTeX editor used for writing, editing and publishing scientific documents [15]. 16

Playbook is Ansible's blueprint of automated tasks to preform [16]. 4, 10, 41, 44, 48, 49

Port is a logical construct which is a communication endpoint in a network. Used to distinguish between different network services. xv, 1, 25, 29, 30, 44, 53

Portable Operating System Interface is a family of standards for maintaining compatability between OS' specified by IEEE in 1988 [17]. xiii

Python is an interpreted high-level programming language. The groups language of choice in this project. 2, 27, 29

Representational state transfer is a standard for a software architecture for interactive applications that typically use multiple Web services. xiii, 13

Scrum is an agile software development framework. 3, 16

Secure Shell is a network protocol used to connect from one network device to another. The established connection in encrypted [18]. It works on devices on the same network or over the Internet . xiii

Secure sockets layer is a security standard technology for securing an internet connection between two systems by encrypting the transmitted data.. xiii

Security Operations Centre is a digital security and emergency preparedness function and reception center located under the section for digital security at the IT department and coordinates the operational digital security work at NTNU. xiii, 1

Simple Object Access Protocol is a messaging protocol specification used to exchange structured data [19]. xiii

SkyHiGh is NTNUs OpenStack implementation. 44, 47

Software Development Life Cycle is "a process for planning, creating, testing, and deploying an information system." [20]. xiii, 6

Structured Query Language is a programming language used by nearly all relational databases to query, manipulate, and define data, and to provide access control [21]. xiii

Taskgiver is equivalent of the norwegian "oppdragsgiver". 2, 3, 22, 24, 44, 48, 51, 53, 56

Teams is a communication and file sharing service by Microsoft commonly used by universities and companies. 16

Toggl is a time tracking software created and developed by Toggl OÜ. 16

Transmission Control Protocol "is a transport protocol that is used on top of IP to ensure reliable transmission of packets" [22]. xiii

User Datagram Protocol "is a lightweight data transport protocol that works on top of Internet Protocol" [23]. xiii

Virtual machine is a technology that allows emulation of a computer. One can think if it as running a computer inside another computer [24]. xiii, 13

WinRM is a SOAP-based protocol supported by Windows to allow it to communicate with another server. 10

YAML is a human-readable data-serialization language [25], often used for writing configuration files. xiii

1. *Introduction*

1.1 Project Background

NTNU has a section for digital security based in Gjøvik. This section works proactively, actively and reactively around digital and information security at several levels in the organization. It consists of the Security Operations Centre (SOC) which specializes in detection, security analysis and incident response. It also contains an advisory service Governance, Risk and Compliance (GRC) which works with proactive safety advice, risk management and security architecture.
When operating a large network such as the NTNU network, keeping track of hosts, open Ports, running services and potential vulnerabilities in the network can be a difficult task. This is especially challenging while doing it over time. By having a system that keeps track of the aforementioned hosts, including its open ports and vulnerabilities over time, the section for digital security can easily have access to historical information on each discovered host on the NTNU network. Having a system that simply scans the network without organizing or saving the results is not that challenging or useful. The section for digital security therefore needs a system that organizes the results and saves them, so that it can be used for detection and security analysis over time. Having this data easily available be can be useful in an incident response context as responders then have access to a baseline of the network.

1.2 Purpose

Our group was tasked with creating **Autoenum**. Autoenum is a system which periodically scans a network to discover hosts, open ports and running services. Additionally the scope was expanded to look for potential vulnerabilities linked to the services or the host. The system should be able to perform in both a test environment and the real world. The data gathered from the scans should be stored in a database.
All the data aggregated by Autoenum will have to be exposed through a REST API for integration with NTNU SOC's existing systems. The data will be used in detection and exposure analysis, therefore the data has to be saved in a manner where it is easy to query. The database should also be able to store a lot of data, as the scans will be performed periodically. Furthermore Autoenum should be

integrated with several tools for data collection and analytic purposes.

1.3 Target groups

The primary target group for the project is our Taskgiver, the NTNU SOC. They will have an interest in the practical side of the report, which includes the use of the code base for our scanner. The open source community is also included in the target demographic. Our project might serve as an inspiration to other parties who are trying to implement a system with similar functionalities.
Some secondary target groups that might have an interest in the theoretical aspect of the project are fellow students and researchers. Students might be interested in the project, because it can help them with their own bachelor projects or similar assignments. Researchers that are working on research papers of a similar nature might also be interested in how our approach to the report is. They might also be interested in the result of Autoenum and the report.
The last target group that might be interested in the project are system administrators. Our network scanner can be deployed on any Debian[1]-based machine and used to scan the network and process the data. Since the code is public, anyone can use it or modify it so it fits their needs.

1.4 Group background and competence

We are a group of four students who share the program of study, IT-Operations and Information Security[2] at NTNU Gjøvik. The group was already organized and prepared to work together when we started to prioritize available bachelor projects. When prioritizing the available projects, the group agreed on which project to give the highest priority. We chose to prioritize this project because it was in our area of interest. Additionally the group members have relevant prior knowledge and experience in many of the areas of competence this project requires. The courses listed in table 1.1 are courses with relevance to this project. The group has an adequate understanding of these areas to combine the central concepts from each of the areas into a bigger project.
The group's relevant background competence from the courses listed in 1.1 includes: Scripting in Bash, Python and Powershell, virtualization, Docker, Nmap, networking, NoSQL databases, system development, infrastructure, automation, risk assessment, APIs, Puppet and software development models.

[1]a Linux distribution composed of free and open-source software
[2]`https://www.ntnu.no/studier/bitsec/studiets-oppbygging#year=2018&programmeCode=BITSEC`

Course code	Course name
IDG2001	Cloud Technologies
IMT2006	Computer Networks
IMT2007	Network Security
IMT2008	ITSM, Security and Risk Management
IMT2243	Software Engineering
IMT2282	Operating Systems
IMT2571	Data Modelling and Database Systems
IMT3003	Service Architecture Operations
IMT3004	Incident Response, Ethical Hacking and Forensics
IMT3005	Infrastructure as code

Table 1.1: Relevant competence

1.5 Constraints

The Taskgiver gave us some specific constraints which we have noted below. We have also mentioned some of our own limitations in terms of experience and knowledge.
The configuration has to be done through Ansible. Ansible ensures that Autoenum can easily be integrated into the taskgiver's infrastructure, which already uses Ansible as the preferred automation tool. Autoenum has to use open source code. Open source code is a type of licensing agreement that permit users to modify and use existing code in other projects. The code we produce has to be open source, as it will be published publicly to NTNU open, this will make it available for the public. The service has to support Microservice architecture.
As a proof of concept, the service should include a basic web interface which enables the user to search for the stored data. This is not a major part of the project, but it is an easier way of interacting with the aggregated data. The primary method of extracting data from the system is through a RESTful API, responses of which should be in JSON.
The deadline to finish the project was May 20th of 2021. We chose to divide our time during the project using Scrum. It is an agile development framework which helped us divide the project into several parts, where one has to be finished before the other can begin.

1.6 Roles

The Taskgiver is the NTNU SOC, represented by Christoffer Vargtass Hallstensen. Christoffer is also the product owner. The supervisor is Erjon Zoto, employed as a lector at the institute of Information Security and Communication Technologies at NTNU. Before starting the project, we defined roles for each of the group members. The following roles were to be treated as main area of responsibility and does not mean that the member with the respective role would work exclusively with that area:

- **Avleen Singh Marjara -** Group leader: in charge of the code for Autoenum and communication with taskgiver
- **Sander Høgli -** Group secretary and scrum master: in charge of writing meeting minutes and sprint review documents.
- **Jarl Tengesdal Lygre -** In charge of scheduling meeting and communication with supervisor.
- **Wojciech Małecki -** In charge of code for test environment and writing Playbooks

1.7 Project goals

The desired goals of the project are described below:

- **To provide a usable system that has the functionality the taskgiver needs and has requested.** Autoenum will follow the functionality mentioned in section 1.2.
- **Improving the security of the NTNU network.** We hope that by using Autoenum or taking inspiration from it, the SOC is able to improve their level of security.
- **Provide a report that acts as supporting source for Autoenum.** This means that the report should provide good written documentation about the system and that it should contain a good mix between theoretical knowledge and practicality.
- **The report and Autoenum should be relevant for future work.** The code base should be easily maintainable and modifiable by the taskgiver and other individuals tasked with this activity. The report should provide readers with inspiration, knowledge and support with similar tasks.

1.8 About the report

This report is written in LaTeX, which provides formatting and the ability to link and reference different chapters. In this report, clicking an acronym will show the reader the full word. If the word in the acronym list includes a link, the reader can click and receive the glossary list with an explanation. The glossary includes words that are explained and is linked the first time it is mentioned in the given context. Other links include where glossary words are mentioned in the report, references, tables and figures. In addition to references we are using footnotes as small comments where a link or small description is needed. Below we provide a brief overview related to the structure of the report. The way the report is structured is slightly inspired by the SDLC in terms of how we decided to order the different sections and chapters.

Chapter 1 Introduction

This chapter provides a general overview regarding the background, purpose, target groups, our competence and project goals.

Chapter 2 Theory

This chapter provides information and a abbreviated introduction about the different concepts, theories and expressions used in the report. This should provide readers basic knowledge and understanding to comprehend more of the report.

Chapter 3 Methodology

This chapter provides information related to our methods, how we chose to work, and which framework we used as inspiration for the structure of the report.

Chapter 4 Design

This chapter provides a description of the architectural design and how the different components of Autoenum are connected.

Chapter 5 Implementation

Describes and discusses our thoughts and methods on how we implemented different components throughout Autoenum and which technology we ended up using and how we use them. It also includes a description of how to deploy Autoenum.

Chapter 6 Testing

Describes and gives an overview of our test environment and how we tested Autoenum.

Chapter 7 Discussion

Reflects on the potential usefulness of the technology. This means trying to measure the significance in relation to detection and incident management. This section will also relate to the theory section, previous reflections and project results.

Chapter 8 Conclusion

Concludes the report, discussing if we meet the project goals and explore further work.

2. *Theory*

This chapter will provide general theory about software development frameworks and describe some relevant theory about the technologies we used in the project. It will be an overall overview and the use of the technologies will come in chapter 5 Implementation.

2.1 Software Development Life Cycle

Simply explained, Software Development Life Cycle (SDLC) is the shortened and oversimplified version of different development processes used when going from requirements to delivering a finished product and maintaining it[26]. These processes are very generic and simplified. The different processes are often called and divided into: requirements definition, design, implementation, testing and maintenance. This almost always begins with an idea to solve some kind of problem or make something easier and more efficient. An idea develops to some requirements where different considerations must be discussed and defined in detail. These requirements must be realistic and possible with the technologies available today. In the design phase one must establish a basic overall architecture while still consider all the requirements in the previous phase. In the implementation and testing stages the software is functional and is tested to ensure it meets the requirements.

2.2 Scrum

In "The Scrum Guide™", Scrum is defined like this: "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value"[27]. Scrum consists of the following components:

- **The Scrum Team** Consists of the product owner, the development team and the Scrum master. The product owner's job is to manage the product backlog. The development team are the ones doing the development of the product. The Scrum master follows the Scrum guide and makes decisions regarding Scrum in general during the project.
- **Scrum Events** Consists of the sprint, sprint planning, daily Scrum, sprint

review and sprint retrospective. The sprint is a period of time which lasts a month or less where usable product increments are created. Sprint planning is where the different sprints are planned, how long and what to work on. Daily Scrum is a meeting held every day during the sprint. In this meeting, there will either be questions or discussion on what has been done or what will be done and how. The sprint review is held at the end of each sprint. The Scrum team goes through what was done in the sprint, based on the changes to the product backlog, it is discussed what will be done next. The sprint retrospective is when the Scrum team inspects itself and creates a plan for how they can perform better at the next sprint.

- **Scrum artifacts** Consists of product backlog, sprint backlog and increment. Product backlog is a list of everything that should be in a product. It lists all functions and features of the product, the requirements and everything else that should be recorded. Sprint backlog contains what functionality the development team thinks will be in the next increment and the work that needs to be done to finish it. The Increment is the sum of all the product backlog items which were completed during a sprint. It is also the value of the increments of the previous sprints.

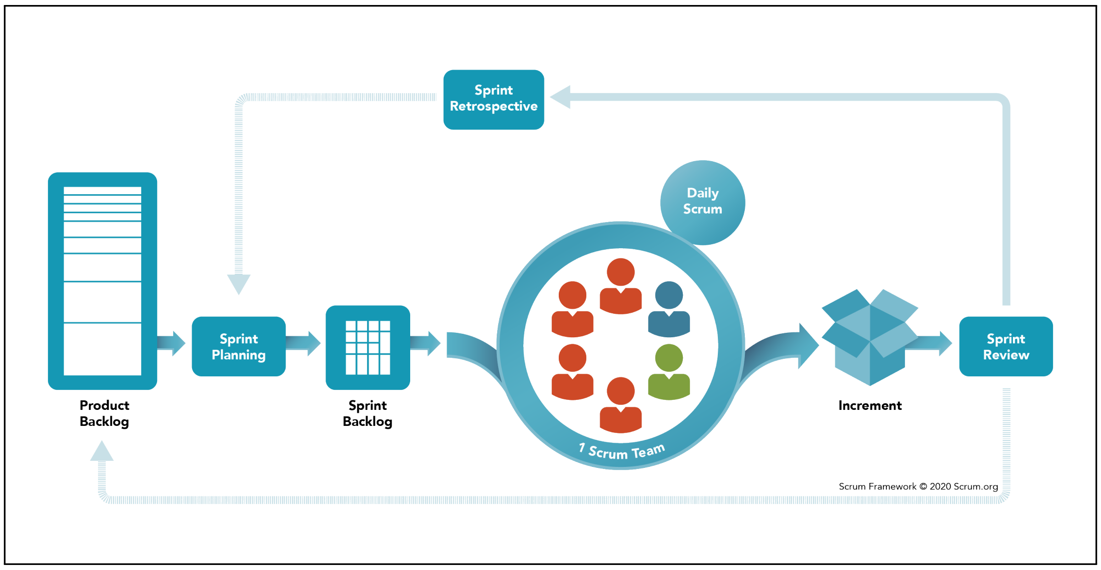


Figure 2.1: The Scrum Framework[28]

The framework has Scrum teams, these teams have different roles, events, artifacts and rules. Every part of the framework serves a purpose as a whole in the Scrum framework. Scrum in practise consists of several "sprints", of which each is a time slots where you work with certain things during the project, i.e. the first sprint is planning and lasts for three weeks. This makes it so that everything in the project, from planing to testing gets an adequate amount of time each.
If we compare Scrum to a non-agile software development strategy such as the

waterfall approach, you can see why Scrum is called agile. The waterfall model is linear sequential, and it is therefore harder to make changes midway throughout the project. When developing software with the waterfall model, each phase can only begin when the previous phase is done. This approach has many disadvantages compared to scrum, such as: not suited for requirement changes in ongoing projects and generally higher risk in terms of failure and delivery of the finished product[26].

2.3 Infrastructure as code

Infrastructure as code (IaC) is useful because in the modern day we need systems and services that can respond fast to scaling and changes over time. We need to have efficient methods to operate different services in the Cloud. By using the cloud and IaC principles we can lower the barrier to update, deploy, scale and configure services faster and more efficient. IaC also helps to keep systems and services consistent and more reliable because of different methods for testing and validation[29].
IaC is a way to go about infrastructure development as one develops software in the modern world. We want to reuse and have consistency in our code. In IaC, having consistent and repeatable code and routines for changing, deploying and configuration on multiple machines is important. Changes are made to definitions and afterwards released to systems through processes that include exhaustive validation[29]. The main goal in using IaC principles is to automate and minimize the additional work with deploying, configuration and testing which otherwise has to be done manually on every machine in a network of servers.
This takes us to the basic principals in IaC. As Kief Morris describes in[29],

1. **Systems needs to be easy to reproduce**. This means at any time the infrastructure can be rebuilt at any time without any risk of failing or making configuration mistakes.
2. **Systems are disposable**. They can easily be deleted, created, moved or scaled.
3. **Systems are consistent**. This means that two servers in a Cluster, should behave almost the same if they are serving a similar service.
4. **Processes are repeatable**. This means that every action you take in your infrastructure should be repeatable. This can be scripts or other tools that keeps you from changing something manually on multiple machines.
5. **Design is always changing**. This emphasizes that when a system or service is developed, it is almost impossible to know everything about how the system or service is going to be used or changed with time or future requirement requests.

2.4 Automation

Automation is in the IT sense of things, "..the use of instructions to create a repeated process that replaces an IT professional's manual work.."[30]. Automation helps realize changes, deployments and configurations for systems as described in the previous section, which leads to the system administrators having to do less manual work. Automation also helps many other aspects, including some of the following:

- **Vulnerability Management** "is the process of identifying, analyzing, triaging and resolving computer security vulnerabilities"[31]. Building an effective vulnerability management program is a five-step process which includes the following steps:

  1. Checking for vulnerabilities
  2. Identifying vulnerabilities
  3. Evaluating the vulnerabilities
  4. Resolving vulnerabilities
  5. Reporting & patching vulnerabilities

  Automation helps realise a small portion of vulnerability management, primarily checking for vulnerabilities and identifying them. This can be done by automating a network scanner which both checks for vulnerabilities in a network and then identifies them with the help of a known vulnerability database. Having these processes automated makes the tasks a lot less tedious to do. It also allows for the processes to be done several times a day which increases the security level of the network.

- **Deployment automation** is strongly related to IaC where it should be possible to take down a server and have it up and running within a short period of time. This short period of time is achieved thanks to deployment automation. Deployment automation makes the application deployment process automated, saving a lot of time and resources if done correctly. Application deployment is used in tandem with IaC.

- **Security and compliance** can both be directly linked to automation. Most of the breaches on IT infrastructures are because of human errors[32]. Automation can help prevent these if done correctly, having a complete configuration that has been tested multiple times ready to be launched. Security monitoring can be realized with automation as well, monitoring such as network scanning and vulnerability scanning.

2.4.1 Ansible

To fulfill the basic principles of IaC discussed in section 2.3 and to keep consistency across deployments and configurations, one should use an IT automation system. Ansible is such an IT automation system, which can be used both for configuration management and application deployment. Ansible is agent-less, as opposed to

many other popular systems such as Puppet[33] and Chef[34]. Instead of agents, Ansible relies on the use of SSH by default to push out commands to the managed nodes. This drastically reduces the overhead compared to an agent based system, which has higher overhead due to agents installed on each managed node. These agents have to periodically pull configurations from the manager or master node rather than having the configuration pushed to the nodes.
Having no agent introduces another advantage. In an agent-less model there is no prior configuration required for the managed nodes, other than ensuring that they have an SSH-server installed and that the manager has the SSH-key to the managed node.
Ansible introduces some concepts that are important to grasp:

- **Playbooks:** When running commands on your managed nodes, Ansible uses a list of multiple commands defined by the user. This collection of commands is referred to as play. The plays can be things like installing software, running an executable file, copying files or restarting the machine. The plays reside in what's called an Ansible Playbook. In other words the playbook is just a list of plays. Playbooks are written in YAML, which is easy for humans to read and understand[35].
- **Roles:** A role is a way of standardising and sharing file structures in Ansible. A role makes it possible to divide Playbooks into logical components, which then can be used to construct more advanced mechanisms. Usually, a role is a set of instructions, used to perform a specific task, i.e. to automatically install and configure a service on a host[36].
- **Compatibility:** In addition to Unix-like systems Ansible also supports Windows. However when using Ansible with Windows it is recommended to use WinRM instead of SSH. The reasoning behind this is that Windows as an OS is non-POSIX-compliant, and that the way Ansible interacts with Windows is fundamentally different from the way it interacts with POSIX based systems. Ansible version 2.8 added support for SSH for Windows, but it is only experimental[37].

2.5 Network mapping

2.5.1 Scanning

When scanning a network we often use tools that have already been developed, are accessible and trusted by many. One example is Nmap. As mentioned in[38] by Gordon Lyon, this is a free and open source utility for network exploration and security auditing. There are many tools that can be used to explore a network. Usually a scanner finds which hosts are on the network. After finding the hosts, the scanner looks for more details, such as: IP addresses, MAC addresses, open ports, services and products running on the ports.

2.5.2 Scan types

There are three basic scan types:

1. **TCP:** This is one of the most used communication protocols that manages the exchange of messages in networks. The main goal of every TCP connection is to provide a reliable flow of data and if something goes wrong, inform the sender about a failed transmission. TCP Connect scans are based on the TCP "Three-way handshake", which is a process of communication between two network-enabled devices. The "Three-way handshake" is the only method that provides real reliability. "Three-way handshake" consists of 3 stages that need to be performed between two counterparts. The first stage starts when a client tries to establish connection with a server, by sending a SYN to inform the server about upcoming communication and to synchronize sequence numbers between devices. When server receives the SYN message, it responds with a message flagged with SYN+ACK signal bits, which means that the server confirmed the synchronization and is ready for upcoming messages. The connection is concluded with another ACK message coming from client, after that the data transfer can begin[39].
2. **TCP SYN "Half-open":** This scan works slightly differently compared to TCP connects scans, but uses the same three-way handshake principle. The only difference between the scan types is that instead of using a full three-way handshake, concluding the process with a "ACK", it sends a message flagged with "RST". The "RST" resets the transmission and prevents it from completing.
   A SYN scan is often also called as a "Half-open" or "Stealth" scan. It is called "Stealth" because SYN scans could bypass some of the older Intrusion Detection systems that were configured to look for completed transmissions. SYN are also significantly faster than standard TCP Connect scans[40].
3. **UDP:** These scans are stateless, it means that there is no reliable communication between the sender and the recipient. When packets are sent to an open port, no response will come back and the port will be marked with "open|filtered" label. If the port is closed, the targeted machine will send a ICMP (ping) with a "Port unreachable" error message. Since there is no way to know for sure if a port is open, UDP scans are way slower than the rest. Therefore they are used to scan a small number of ports, usually the most known ones[40].

2.5.3 Vulnerability

Different kinds of vulnerabilities will always be present, as new vulnerabilities are discovered every day and are growing in numbers. A vulnerability can be defined as follows:

> "*A weakness in the computational logic (e.g., code) found in software and hardware components that, when exploited, results in a negative*

> *impact to confidentiality, integrity, or availability. Mitigation of the vulnerabilities in this context typically involves coding changes, but could also include specification changes or even specification deprecations (e.g., removal of affected protocols or functionality in their entirety)*"[41].

A vulnerability does not indicate that there is some kind of risk. This is because a risk exists when there are some kind of assets that needs to be protected. This can be user data, financial information, personal information, classified data or privilege escalations. If a vulnerability exists but there are no assets of value that must be protected, it is just classified as an exploitable vulnerability.
Almost all discovered vulnerabilities get registered in databases, helping the public and affected organizations. One of these records of common vulnerabilities is called CVE. This is a list of common vulnerabilities and exposures that has been discovered. These get assigned a number based on when they were discovered with a brief description and related references. To become a CVE vulnerability, it must meet some criteria. First of all the vulnerability can be fixed independent of other bugs. It also has to be acknowledged by the vendor or shared through documentation. Finally it can only affect one code base, because each vulnerability gets a separate CVE number[4].

2.6 Databases

Databases are one of the most common components of an IT system. They are used to store and collect different types of information or data. Usually there is a database management system that is used to control the database. A database is usually structured in columns and rows, which makes the data easily accessible to different applications, users and it makes querying more efficient. Some key elements is that the data is easily accessed, managed, modified, updated, controlled and organized. Right now there are many different types of databases and even types that do not use SQL. These are called non-relational databases. Contrary a well-know database is MySQL, which refers to an open source relational database management system based on SQL[21].
Examples of NoSQL databases include MongoDB. These databases do not use relational structure but instead use different kind of methods to organize data. MongoDB is a document based database, which structures data similar to JSON. In a MongoDB database every document has a field for an unique value. This is used to identify a unique entry in the document hierarchy. In relation to SQL databases, this is comparable to the primary key. Other data related to that unique id is embedded within that id, and is not stored in a separate table. This is one of the ways it differs from traditional SQL databases.

2.7 Containers

Kief Morris explains in[42] that Containers are a way to install and run applications in an alternative way. Using containers provides a way to standardize a format to run services and applications on servers. These containers are used to collect and define an environment for a process. Docker is one of these tools and is used to package software. The process can also be called containerization. Docker uses dockerfile to define what kind of dependencies are needed to run the application on a server. This file is used to create a container image, which contains all things needed to run a certain system. By bundling all needed dependencies, containers can be easily repeated to create a runtime environment.
Using containers is an alternative way to run multiple instances of an application and are often compared to Virtual machine (VM). One of the main differences between VMs and containers is that each VM has a guest operating system on top of the host operating system. Container services like Docker share the host operating system, including the kernel. Because of this it is often the most lightweight, compared to virtual machines.

2.8 RESTful API

API is short for Application Programming Interface and are services that facilitates communication between different applications. To understand how an API works, one can compare it to visiting a web site. When a user requests a web site, the HTML for the web site is sent to the users device. The HTML is then rendered into a web page with pictures, videos and other graphic elements. It is common for APIs to run over HTTP, in which case they work similarly as a web site. The difference is that the API is optimized for an application to understand the received data[43]. The request to the API is often made by an application. Since the application has to be able to understand the response of the APIs, the response are in formats which are easy for machines to understand. JSON and XML are examples of formats which are typically used as a response from an API.
REST is short for Representational state transfer, and is an architectural style for distributed hypermedia systems [44]. It allows a service running on the web to represent its resources (e.g. picture or file) as text. In order to be classified as a RESTful API, the service must follow these constraints [44][45]:

- **Stateless:** Interaction between the client and server should be stateless, meaning that everything needed to handle the request should be included in the request itself.
- **Cacheable:** Communication between the client and server should be cacheble, meaning that the server should be able to store copies of frequently used data, in order to speed up the future requests.
- **Layered system:** There can be multiple layers between the client and the server, but the client will only be aware of the immediate layer. This means

that proxies or load balancers can be placed between the client and the server without having to update client-side or server-side code. This also means that API **A** which the user is making requests to can request resources from API **B** without the user knowing it.

- **Client–server:** The system is comprised of servers and clients. The clients and servers are independent of each other. The servers should handle back-end tasks such as databases, while client handle the front-end tasks such as user interfaces.
- **Uniform interface:** All devices should communicate with API the same way. In other words the way they interact with the API should be uniform. This means that using the API with e.g. Ubuntu or Windows based machine should be exactly the same. Uniform interface has its own constraints:
    - **Identification of resources:** Each resource should be identified uniquely with a Uniform Resource Identifier (URI). Example URI in figure 2.2.

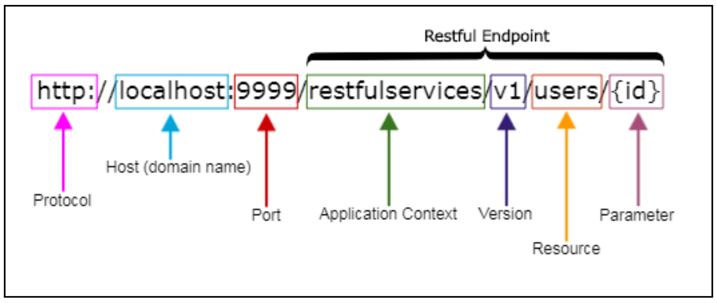


Figure 2.2: URI example [46]

-
    - **Manipulation of resources (through representations):** The resources are manipulated when represented. This means that the user does not interact with i.e. a database directly, but the data (the resource) in the database is represented by the API. This makes the decoupling of clients and server easier, as one can change the implementation without affecting the client. E.g. if moving from SQL database to a NoSQL database, the representation of the resource does not need to change.
    - **Self-descriptive messages:** Request and responses must include adequate information for the receiver to understand it in an isolated context. The message must have the right media type, e.g. *application/json*, so that the receiver knows that the response should be parsed as JSON.
    - **HATEOAS - Hypermedia as the engine of application state:** Hypermedia can be a part of the response object for a resource which the client can traverse or use to request another resource[47]. This is comparable to accessing a web page and using the links on the web page to navigate to other pages or resources.

If an API implements the aforementioned constraints, it can be classified a RESTful

API. Requesting resource from a RESTful API is done by making a HTTP request (figure 2.2) with the following methods being used most frequently[48]:

- **GET:** used to request the specified resource
- **POST:** used to add an entity to the resource
- **PUT:** used to update a resource
- **DELETE:** used to delete a resource

3. *Methodology*

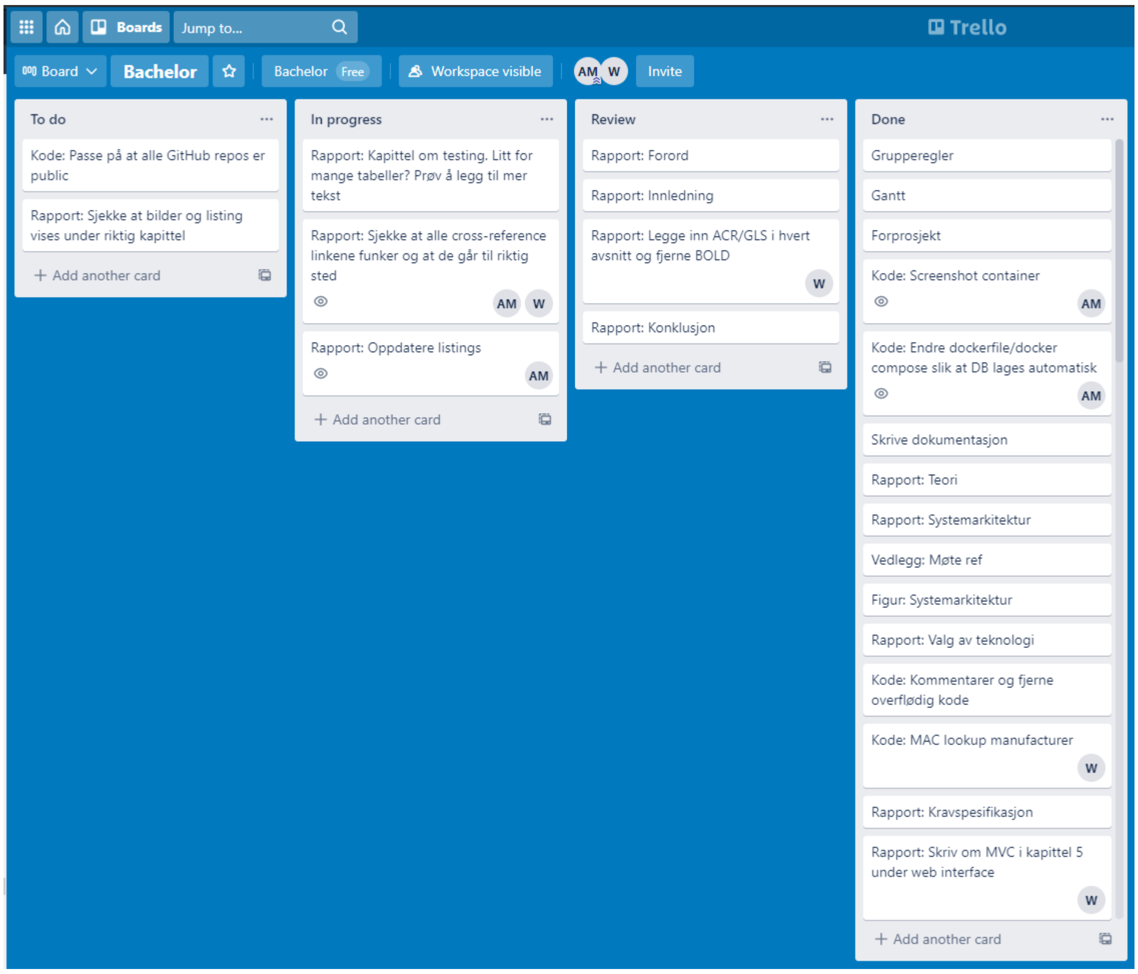
This chapter contains the different types of methods we are using throughout the project, from our software development framework, what research we did, how we decided on our report structure and how we have chosen to work on the project. It also contains what technologies we are using and why we chose to use them.

3.1 Digital workspace

At first our plan was to work as a group physically together at school. This plan did not materialize because of the ongoing pandemic. We chose to rather work on the bachelor project from home, as this is what felt the most safe. During the project we have almost exclusively worked together at fixed times and days in the week while writing the report and working on the code. We decided to do this because it made it possible to discuss, elaborate and get more involved with each others work. We knew from previous experience that nobody would be able to be involved with everything as tasks are delegated throughout the group. With this method we were able to share and hopefully learn from each other throughout the project. Another reason is because it is more fun and sociable than working alone. We experienced that our productivity level rises when we worked together. Before the project began we decided on which tools we were going to use to manage the project:

- **Communication:** We used Discord for our internal meetings/work-sessions and Teams for our meetings with the taskgiver and supervisor
- **File sharing:** For file sharing internally in the group, we used OneDrive. The OneDrive contained notes from meetings, contracts and documents related to scrum.
- **Code repository:** We have set up our code repository on GitHub, which is used to store the code and enables easier collaboration. GitHub is also used to easily deploy all the code in the test environment.
- **Text editor:** We wrote the report using the LaTeXeditor Overleaf.
- **Time tracking:** To keep track of our working hours we used Toggl. By the end of the project, we generated a listing in Toggl which shows how many hours each student has worked.
- **Scrum board:** We used Trello to create a Scrum board. The Scrum board

contained all tasks related to the report and the code. The Scrum board is divided into four sections: **To do**, **In progress**, **Review** and **Done**. This helped us to organize and prioritize the right tasks at the right time. The Scrum board was also used as a final checklist for the project. Figure 3.1 shows our Scrum board.



To do

Kode: Passe på at alle GitHub repos er public

Rapport: Sjekke at bilder og listing vises under riktig kapittel

+ Add another card

In progress

Rapport: Kapittel om testing. Litt for mange tabeller? Prøv å legg til mer tekst

Rapport: Sjekke at alle cross-reference linkene funker og at de går til riktig sted AM W

Rapport: Oppdatere listings AM

+ Add another card

Review

Rapport: Forord

Rapport: Innledning

Rapport: Legge inn ACR/GLS i hvert avsnitt og fjerne BOLD W

Rapport: Konklusjon

+ Add another card

Done

Grupperegler

Gantt

Forprosjekt

Kode: Screenshot container AM

Kode: Endre dockerfile/docker compose slik at DB lages automatisk AM

Skrive dokumentasjon

Rapport: Teori

Rapport: Systemarkitektur

Vedlegg: Møte ref

Figur: Systemarkitektur

Rapport: Valg av teknologi

Kode: Kommentarer og fjerne overflødig kode

Kode: MAC lookup manufacturer W

Rapport: Kravspesifikasjon

Rapport: Skriv om MVC i kapittel 5 under web interface W

+ Add another card

Figure 3.1: Snippet of scrum board in Trello

3.2 Scrum

We identified that our service could easily be broken down into smaller parts, as the taskgiver required that the service should be able to run in a microservice architecture. We chose Scrum, as the service should potentially be ready for deployment at the end of each sprint. The Scrum model also allowed us to adapt more easily to new changes during the project period.

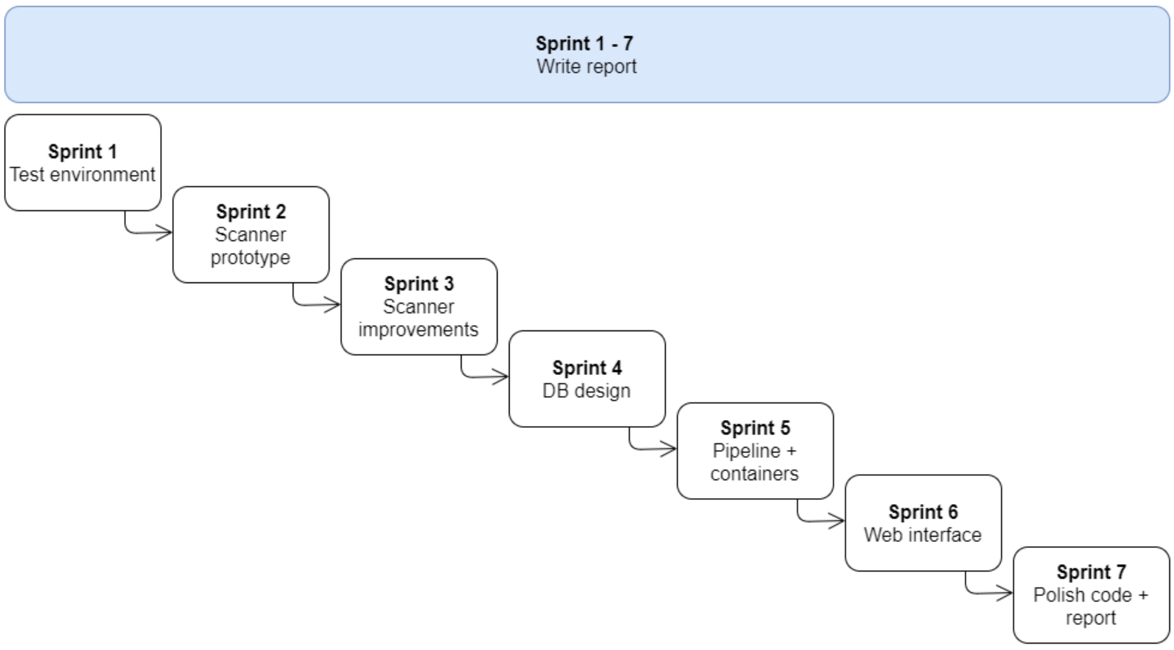


Figure 3.2: Planned sprints

Sprints

All sprints (figure 3.2) started off by having a sprint review of the previous sprint (see Gantt diagram in I). As we had an agile approach to this project, we began testing while coding. This was done to ensure that the product was "shippable" at the end of each sprint. Our sprints began on Mondays and ended on Friday the week after the beginning of the sprint, making each sprint last twelve days. The weekend between sprints were used as a buffer that could be used to extend the sprint if there were any delays or extra time was needed. We decided from the beginning to write a lot of the report while we worked on the technical aspects of the project, and we did stick to that plan throughout the project. This enabled us to better document the choices we made, and have continuous progress and distribute the amount of work.

- **Sprint 1 - Prepare test environment:** This includes making Heat templates for consistent and repeatable deployment of our test infrastructure in SkyHigh, setting up Ansible to install dependencies on machines in infrastructure

- **Sprint 2 - Prototype of the scanner:** Make a prototype of the scanner which includes the most basic functionalities and outputs the result to a file in JSON-format.
- **Sprint 3 - Scanner improvements:** After testing the prototype we will add more features to the scanner.
- **Sprint 4 - Handling output and database design:** After adding additional functionality to the scanner we will know what data the scanner outputs. We will then design the database and implement a JSON parser and insert the data to the database
- **Sprint 5 - Analysis pipeline and containerization:** All gathered data needs to be thoroughly analysed, to make it more scalable, we are going to containerize the process and implement REST API. This sprint is one week longer to account for lost working time due to easter vacation.
- **Sprint 6 - Web interface and final test:** Making a small web interface as a proof of concept. This web interface will primarily include functions for searching the saved data in the database. After making the web interface we will test to see if the whole service works as intended
- **Sprint 7 - Polishing code and report:** The last sprint will be used to polish the code (if needed) and make the finishing touches to the report.

After each sprint, there was a sprint review (appendix K). The sprint review consists of discussing what was done during the previous sprint and we discussed if the increment we created in the previous sprint was adequate. In cases where we completed the sprint before the deadline, the sprint review meeting would be held prematurely to make sure that we did not miss anything important. If it was confirmed that we had completed everything within the sprint early, we would start with the next sprint ahead of time.

3.3 Similar projects

At the start of the project, we tried to find similar projects for inspiration and ideas. We wanted to find tools and solutions that had same or similar functionalities to Autoenum, use them and see how they perform. We found a few tools, and we successfully managed to compare all of them. This allowed us to see what functionalities and features we would like to have in Autoenum. One of the first tools we found was Reconnoitre[1], which is a open-source, multithreaded, information gathering tool. At first, we thought about integrating it in our system, but after reading the documentation and testing, we realised that Reconnoitre was using tools that were causing too much traffic in the network. One of the other tools

[1] `https://github.com/codingo/Reconnoitre`

was DirBuster[2], which is a tool used for brute-forcing directories. Since Reconnaitre was using tools such as DirBuster, it was more of a vulnerability scanner, that is why we decided to not go forward with it. At this point, we knew that Autoenum should be a fast, reliable network scanner and based on Reconnaitre's performance, we knew what we should avoid.
The next tool that we researched was IVRE[3], which is a network recon framework. It is very similar to Autoenum, but much more advanced. All the functionalities that we had to implement in Autoenum, are implemented in IVRE, so we took some inspiration from this software.

3.4 Report structure

The structure of our report has changed a lot during the project. Towards the end of the project we decided on the structure that is present in the report now. It consists of eight chapters and is slightly inspired by the SDLC. We took inspiration from the five stages of the SDLC's life cycle: **requirement definition, design, implementation, testing and maintenance**. The stages are depicted in figure 3.3. We chose not to include maintenance as we will be only be responsible for developing the system and not maintaining it.

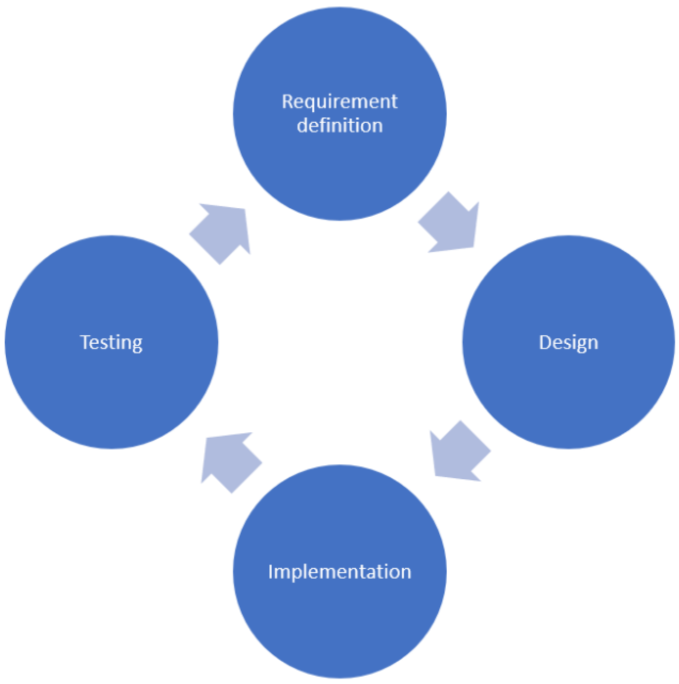


Figure 3.3: SDLC illustration

[2]https://github.com/KajanM/DirBuster
[3]https://ivre.rocks/

Requirement definition is the initial step in every development process based on the SDLC. In our case, this step began with discussing and documenting all software requirements, and then conducting a preliminary project, which was delivered to the taskgiver for approval. This can be found in chapter 4.
Design is the next stage in the SDLC model. The design part started with the group discussing what kind of architecture would suit our system the best. After a couple of tries, we decided on the one we show in the report. This part of the report shows that we are able to plan and look forward with software development in mind.
Implementation is the next chapter (5). Here we go in to detail on the technical aspects of the project. Implementation started early in the project when we began coding Autoenum and setting up the database and the API that it uses. This chapter shows that we are able to discuss and properly show our code and it is functionality in an easy to understand way.
Testing is the last section of the SDLC. Testing was done simultaneously with the implementation, as the code needed to be tested while it was written. Chapter 6 shows the test environment of Autoenum and describes how we tested it.

3.5 Technology

In this part of methodology, we will be looking at the various technologies we chose to use during the development of Autoenum. We will be describing what we use and why we use it.

Python and Flask

When choosing the programming and scripting language for the project, our choice quickly fell on Python largely due to Python's many community made modules. Although we discussed using Bash, some of the mentioned Python modules are doing exactly what we need, and instead of making our own modules from the ground up we have used some modules which have been tried and tested by a large community. Flask is a web framework written in Python and we are using it for our API and screengrabber. Screengrabber is the term we use to describe one of the functionalities required by the task description. Simply explained it captures a image of the relevant website. We chose to use Flask over another web framework because it was easy to use and suited our needs.

Nmap

When we were choosing the technology for our scanner, **Nmap** was the first thing that came to mind. It is the staple of network scanning, it is free, opensource, reliable and easy to use. If we look at Nmap's official website[38], it is described as flexible, powerful, portable, easy, free, well documented, supported, acclaimed and popular. We had some prior experience with Nmap, from which we knew that Nmap would fit our purpose well.

MongoDB

We ended up using the NoSQL MongoDB for our database service. The main reason why we chose MongoDB over a SQL database like i.e. MySQL is that we don't have to worry about relations in the database. MongoDB uses documents in a JSON-like format[49] while a SQL database like MySQL uses tables. Because we are working with documents instead of relations in the database, we do not need to alter the schema if we decide to save more data at a later time. This is also in line with our agile approach, where we have to release a potentially shippable product at the end of each sprint. By using MongoDB we do not have to spend time altering the schema every time we add new functions in Autoenum.

Docker

Docker is a major part of Autoenum. Docker is well suited for a system like this, where it consist of several microservices. The way we have implemented the system, the scanner can partially work without some of the containers. i.e.: the scanner can work without the screengrabber or the CVE database. As an added point of reliability, Docker will automatically restart the containers if they fail. All the previous points contribute to Autoenum becomming more robust and reliable.
In addition to using Containers, we are using volumes. The use of volumes in a Docker environment allows for sharing files between multiple containers and the host they are running on. More importantly they provide data persistency. Without volumes, the data inside the container would be deleted when stopping the container. This would not be acceptable in the case of a database. When using volumes in Docker, the volume is mounted both on the host and in the container. When the container is stopped the data is still saved on the host, and when restarting the container it mounts to the same path. This enables the new container to access data that was saved to the path by the old container.

Ansible

The Taskgiver requires that Ansible must be used to set up Autoenum. In addition to using it for Autoenum, we have chosen to use Ansbile to configure the test environment as well because this will help us achieve consistency across deployments as discussed in section 2.4.

NodeJS

None of the group members had any prior experience in making web applications. Even though the web interface is just supposed to be a proof of concept, we wanted it to have some useful functionality. The main reason our choice fell on NodeJS is that it was easy to learn due to the vast amount of online resources, but at the same time provided everything we needed. Additionally it is easy to set up with MongoDB. We combined NodeJS with *Express*, which is a standard web application server framework made for building websites, in order to write the server-side code[50]. We also used *EJS*, a templating language for generating the HTML for web pages by using JavaScript[51].

Redis

Redis is a data structure store that runs in the memory of a system. It can be used as a stand alone database or it can act as a cache for another database. The advantage of using cache like Redis is that it is much faster to read from memory than from a disk. When a query is made the first time, Redis will retrieve the data from the database and store it in memory. If the same query is made again the response will be sent directly from Redis instead of the database. In larger systems where the same query might be performed several times in a short time period, the workload of the database will be significantly reduced at the same time increasing the response times.

4. *Design*

This chapter describes the requirements for Autoenum. It also gives a general overview of the design and describes some of the different components.

4.1 Requirements

Functional requirements include all the functionalities the service or product must include to enable the end-users tasks. The requirements we mention will hopefully end with a system that behave as wanted under specific conditions.
Technical and operational requirements include how the system is built, what kind of standards, language and operating system is used. Some of our technical requirements include that the service must use Ansible to configure and automate the deployment of the service. This is a strict technical requirement. All the servers should be able to be deployed with the use of OpenStack. The language used to develop code and scripts must be widely used and have a rather large community. It is an advantage if it already have a large database of modules and example code in relation to network scanning and discovery. The service has to support Microservice architecture, meaning that components get separated into smaller parts with everything it needs to function. The requirements have been developed in consultation with the Taskgiver and are listed below:

- **Open source:** All code used in the project must have an open source licence. The finished product must be licensed as open source.
- **Microservice architecture:** The service must support microservice architecture, preferably by using containers.
- **Automation:** The solution should be configurable and deployable using Ansible. This means the service developed must be easy to deploy into existing infrastructure and not require rewriting or manual configuration. It should be easily deployed and automated with the use of Ansible.
- **API:** A RESTful API must be implemented in the service to allow for integration with the taskgivers existing services. The API responses should be in JSON. API must be implemented with CPE, IP and MAC as valid queries.
- **Web interface:** As a proof of concept the service should include a web interface for searching the saved data. When entering the IP address associated with the web view, the user should get an overview over the different hosts from all the previous scans. Some information will be shown in the preview

for each host, but to get a more detailed view including screengrab, the user must click on preferred IP address. On this site the user should be able to do a basic search, which contains IP, MAC, CPE or date. It is important to note that this is only a proof of concept and will most likely not be emphasized as one of the main parts of the functionality of the service.

- **CVE:** Hosts that are found by the scanner should have their open Ports and OS' checked for CVEs in a CVE database.

4.1.1 Scanner

The main function of Autoenum is the scanner. The scanner takes a network address as an input and use it as its target. This should be defined in a separate file for ease of use, and when the network changes. The network scanner runs on a machine within the network and the IP of the machine has to be excluded before every scan. Every scan needs to be registered and saved in the database where it can be used for further analysis. Every scan needs to be saved and not overwritten, this is critical for further use of the collected data. While we will not process or analyse the data, the database needs to be exposed through the REST API as the taskgiver will use it to get information from the database. The REST API is used to get information associated with one object so it can be used for further analysis.

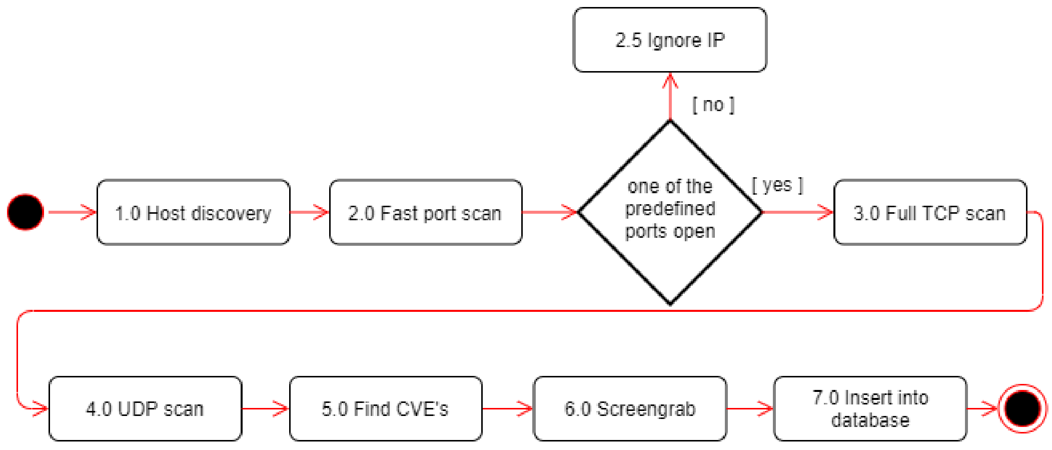


Figure 4.1: Flowchart for scanner

The scanner needs to be separated into different stages (see figure 4.1), described below:

- **Host discovery:** This is where IPs on the network are discovered.
- **Port scan:** All IPs found in stage 1 will be used when the port scan is preformed. This finds open ports.
- **TCP scan:** If predefined ports are open, a TCP scan will be preformed on IPs with open ports.

- **UDP scan:** If predefined ports are open, a UDP scan will be preformed on IPs with open ports.
- **Find CVE:** This stage uses the CPE to find CVE.
- **Screengrab:** If predefined ports are open, a screenshot will be taken.
- **Insert to database:** All the data will then be aggregated and put into the database.

4.2 Architecture

Based on meetings with the taskgiver and his requirements we developed the following architecture as seen in 4.2. Autoenum consists of several components as stated in the task description (appendix A). Some of the components are running locally on the host whilst the majority of the components have been Dockerized to support microservice architecture.

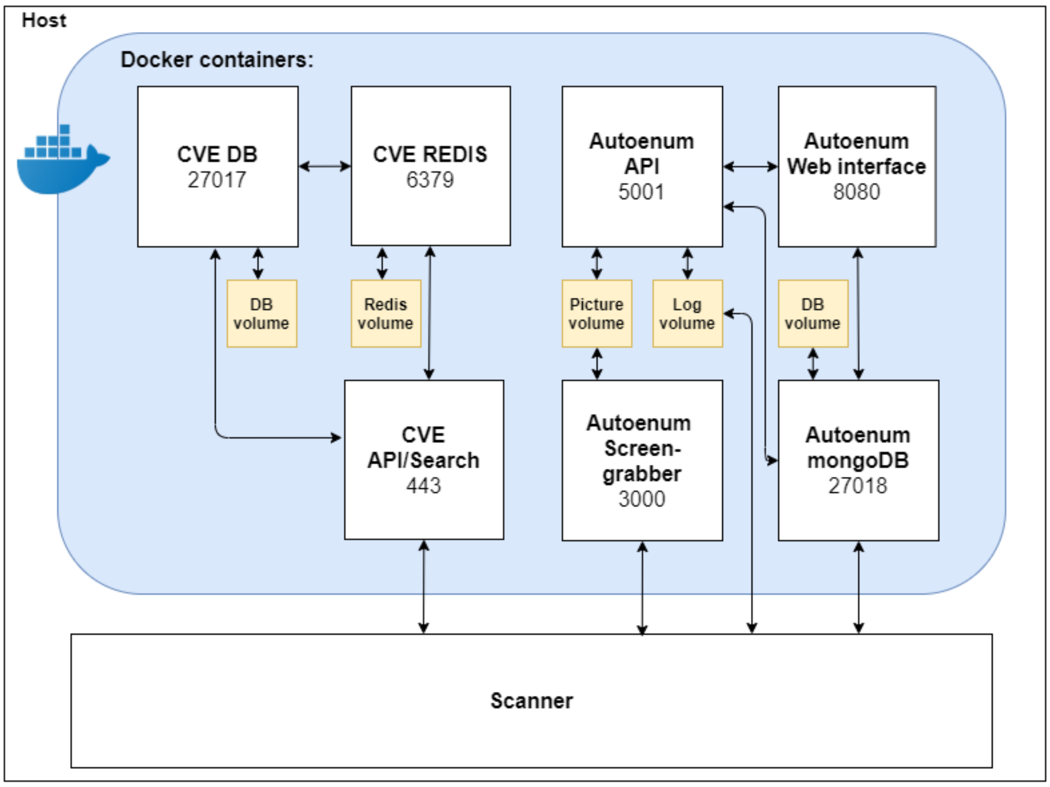


Figure 4.2: Architecture detailed

4.3 Components

Below we describe some of the most notable components in greater detail:

- **Autoenum MongoDB:** The database runs in a container to support Microservice architecture. The container is connected to a volume to ensure data persistency. The data saved in the database is grouped by Host object.
- **Autoenum API:** The API will be the primary way to interact with Autoenum. It is written in Python by the use of the Flask framework, and is Dockerized. As a minimum requirement, it must have an endpoint for finding hosts based on CPE. The API responds to GET-requests only, as there is no need to manually add data to the database. The API responses will be in JSON format, which enables the user to integrate the API in existing systems. In addition to sending data from the database, the API will send the screengrabs when requested by the web interface. In order to send the screengrabs, the respective containers have to share a volume.
- **Autoenum Screengrabber:** The screengrabber runs in a container and is written in Python/Flask. The scanner will send a GET-request containing the IP-address of the host to screengrab, and the screengrabber will perform the grab by using imgkit and wkhtmltopdf. The response will be in JSON and include filename, time and date. The image itself will be saved to the volumed shared with the API and not in the database.
- **Autoenum Web interface:** The web interface provide an easy way of viewing and searching through the data in the database. It is Dockerized, but will be made using NodeJS, Express, EJS and Bootstrap. This is only used to show the actual data and is not a major part of the requirements.
- **CVE Database and API:** Autoenum will have a local instance of a CVE database running to reduce the amount of outbound network traffic generated while the system is scanning. The CVE search Autoenum will use[1], has its own API to facilitate implementation with the scanner. The API will be connected to Redis to speed up the data retrieval. Additionally it comes with a web interface that can be used by the user to read the details of the CVE.

4.4 Licence

One of the requirements in the task description was that the code we produce must have an open source licence. During the development of Autoenum, we have been inspired by many open source projects and we have benefited greatly from the open source community. As a way of showing appreciation to the community we chose to give the project a BSD-3-clause licence. We also feel that the BSD-3 licence is in line with NTNU's motto: "***Knowledge for a better world***". By issuing a copy-left licence like the BSD-3 licence, we hope that the open source community can benefit from Autoenum and continue building on it.

[1]`https://github.com/cve-search/CVE-Search-Docker`

5. *Implementation*

This chapter describes and discusses our thoughts and methods on how we implemented the chosen technologies throughout Autoenum. It contains explanations of some code snippets. It also contains a detailed explanation on how to setup and deploy Autoenum which might be useful for our target groups in section 1.3.

5.1 Github repository

The code for our project is published on different repositories. Table 5.1 describes the different repositories.

Used for	URL
Code for Autoenum	`https://github.com/asm492/autoenum`
Code for deploying test environment	`https://github.com/asm492/auto`
Configuration for test environment	`https://github.com/Monastyr/autoenum-TestENV`

Table 5.1: The different repos for the different parts of the project

5.1.1 Open source tools

Our project relies on other open source code and projects. The projects listed in 5.2 are used in Autoenum.

Name	Licence
Nmap [52]	Based on GNU GPLv2
python3-nmap v1.5.0 [53]	GPL-3.0
wkhtmltopdf [54]	LGPLv3
imgkit [55]	MIT
Flask v1.1.2 [56]	BSD-3-Clause
PyMongo v3.11.3 [57]	Apache-2.0
CVE-Search-Docker [58]	GPL-3.0
pywinrm v0.3.0 [59]	MIT

Table 5.2: Open source tools used in the project

5.2 Overview

When writing the code for Autoenum, we discovered that we had to make some changes to the architecture that was initially planned. The changes mainly consisted of moving the code for the enrichment Container in to the scanner. Additionally some containers were added to support the CVE database. After making the changes we ended up with the architecture shown in figure 4.2.

5.2.1 Scanner

The scanner is the core of Autoenum. It is written in Python and interfaces with the other components. The main functionality of the scanner is implemented using Python Nmap, which is a Python Library which enables Nmap commands to be executed from Python[1]. The scanner relies on the other containers to get additional data about the scanned host, but can potentially run without utilizing the functionality provided by the containers in case one of the containers shuts down. However the data from that specific container would be missing. The output of the scanner might not be saved if the database container shuts down. The containers therefore play an important part in the data gathering process.
As shown in figure 4.1 the scanner works by dividing the scanning process into seven stages, which will be explained in the subsequent paragraphs. The code for the scanner is attached in appendix B.1.

Stage 1.0 - Host discovery

The first stage of the scanner is to find hosts. By using the **-iL** argument, Nmap reads the IPs of the hosts or networks from the **target.txt** file. To reduce the number of hosts that will be port scanned in the later stages we first perform a host-discovery scan in order to determine if a given host responds. If a host does respond, its IP is sent down to stage 2. The host discovery is done by sending a TCP SYN packet to Port 443, TCP ACK to port 80, ICMP timestamp request and ARP request[60] to all the IP addresses specified in the target file.

Code listing 5.1: perform_host_discovery( )

```python
def perform_host_discovery():
    # Stage 1
    logging.debug('[HOST DISCOVERY] started')
    nmap = nmap3.NmapHostDiscovery()
    res = nmap.nmap_no_portscan(
        None, args="-sn --excludefile exclude_ip.txt -iL target.txt")
    res = remove_keys(res)
    logging.debug(res)
    f = open("ips_to_scan.txt", "w")
    for ip in res:
        logging.debug('Found IP: ' + ip)
        if res[ip]['state']['state'] == "up":
            f.write(ip + "\n")
```

[1]https://pypi.org/project/python3-nmap/

```
    f.close()
    logging.debug('[HOST DISCOVERY] done')
```

Stage 2.0 - Fast port scan

This step takes the hosts that responded to the scan from previous stage and scans for common open Ports (22, 443, 80 etc.)[2] [3]. The results are then sent to another function(**find_interesting_ip( )**) which loops through the scan result to check if one of the ports is open. If one of the ports is open then the IP is written to a new file(**ips_to_scan.txt**) which is used by the next stage to scan more thoroughly.

Code listing 5.2: perform_portscan( )

```python
def perform_portscan():
    # Stage 2
    logging.debug('[FAST PORTSCAN] started')
    nmap = nmap3.NmapHostDiscovery()
    res = nmap.scan_top_ports(None, args="-F -iL ips_to_scan.txt")
    res = remove_keys(res)
    logging.debug(res)
    find_interesting_ip(res)
    logging.debug('[FAST PORTSCAN] done')
    return res
```

Stage 3.0 - Full TCP scan

This stage performs a full TCP scan of the IPs in **ips_to_scan.txt**(generated in Stage 2.0). This is arguably the most important stage of the scanner, because it gathers the majority of data about the hosts. The result of this scan includes:

- Port status
- Service name
- Product name and version
- IP address
- MAC address
- Hostname
- OS detection
- CPEs
- SSL-certificates.

Code listing 5.3: perform_tcp_scan( )

```python
def perform_tcp_scan():
    # Stage 3
    logging.debug('[TCP SCAN] started')
    nmap = nmap3.Nmap()
    result = nmap.nmap_version_detection(
        None, "-sV -p- --script ssl-cert -vv -O -iL ips_to_scan.txt")
```

[2] https://pypi.org/project/python3-nmap/
[3] https://nmap.org/book/nmap-services.html

```python
    remove_keys(result)
    print(result)
    logging.debug(result)
    logging.debug('[TCP SCAN] done')
    return result
```

Stage 4.0 - UDP scan

This stage performs a UDP scan on a limited number of ports. It scans the same IP addresses as the TCP scan. Since UDP is a stateless protocol, we had to reduce the number of ports to scan as running a full scan would take a long time. We therefore settled on scanning UDP ports that are often exploited in amplification attacks[61] [62]. Some of these ports include port 123 for NTP and port 445 for SMB. NTP stands for Network Time Protocol which provides time sync between computers and network systems. SMB stands for Server Message Block which is a communication protocol for providing shared access in a network[63] [64].

Code listing 5.4: perform_udp_scan( )

```python
def perform_udp_scan():
    # Stage 4
    logging.debug('[UDP SCAN] started')
    nmap = nmap3.NmapScanTechniques()
    result = nmap.nmap_udp_scan(
        None, "-iL ips_to_scan.txt -p53,67,68,123,137,138,161,445,5000")
    remove_keys(result)
    logging.debug('[UDP SCAN] done')
    return result
```

Stage 5.0 - Find CVEs

This stage merges the results from the scans and enriches the results further with screengrabs and CVEs. This step does not perform any scans. After running the previous scans, the results from the TCP-scan (listing 5.3) and UDP-scan (listing 5.4) have to be merged so we create one single Host object. The results from the Nmap scans are in Python dictionaries, which essentially are key-value pairs and resemble JSON. The results are sent to the **merge_results( )** function (listing 5.5), which creates a separate host object for each of the hosts in the previous scan results.

Code listing 5.5: merge_results( )

```python
    for i in t:
        os = t[i]['osmatch']
        t_ports = t[i]['ports']
        u_ports = u[i]['ports']
        ports = t_ports + u_ports

        # OS CPE :
        for j in t[i]['osmatch']:
            if 'cpe' in j:
                if j['cpe']:
                    oscve = []
                    oscpe = j['cpe']
                    oscve = cve_lookup.find_cve(oscpe)
                    j['cve'] = oscve

        for port in ports:
            cve = []
            for script in port['scripts']:
                s = script['data']
                s.pop(0, None)
            if 'cpe' in port:
                if 'cpe' in port['cpe'][0]:
                    cpe = port['cpe'][0]['cpe']
                    cve = cve_lookup.find_cve(cpe)
                    port['cpe'][0]['cve'] = cve

                screengrab = take_screengrab(i)
                if 'Filename' in screengrab:
                    port['screengrab'] = screengrab
        hostname = t[i]['hostname']
        macaddress = t[i]['macaddress']
        state = t[i]['state']
        stats = {'scandate': startdate, 'scantime': starttime}

        uid = str(uuid.uuid4())
        host = {'uuid': uid, 'ip': i, 'hostname': hostname, 'macaddress': macaddress,
                'osmatch': os, 'ports': ports, 'state': state, 'scanstats': stats}
        insert_db(host)
```

In the process of creating separate host objects, the code tries to find CVEs based on CPEs (5.5 calls find_cve( ) in appendix B.2) and performs a screenshot (5.5 calls 5.6)

Stage 6.0 - Screengrab

The **take_screengrab( )** function (5.6) makes a HTTP-request to the container running the screengrabber (listing C.1). If the screengrab is successful on the requested IP, the screengrabber returns the filename, date and time of the screengrab to the scanner in JSON-format.

Code listing 5.6: take_screengrab( )

```python
def take_screengrab(ip):
    url = 'http://localhost:3000/takescreengrab/'
    url += ip
    urllib3.disable_warnings()
    requests.packages.urllib3.disable_warnings()
    resp = {}
    try:
        resp = requests.get(url, verify=False, timeout=1).json()
    except requests.exceptions.HTTPError as errorHTTP:
        logging.debug("[SCREENGRAB] Http Error: ", errorHTTP)
    except requests.exceptions.ConnectionError as errorConnection:
        logging.debug("[SCREENGRAB] Error Connecting: ", errorConnection)
    except requests.exceptions.Timeout as errorTimeout:
        logging.debug("[SCREENGRAB] Timeout Error: ", errorTimeout)
    except requests.exceptions.RequestException as errorRequest:
        logging.debug("[SCREENGRAB] ERROR: ", errorRequest)

    return resp
```

Stage 7.0 - Database insertion

When all the data is collected, the remaining step is to insert the data into the database. The code-snippet shown in 5.7 connects to the database, and inserts the record. The scanning is now complete and the data can be viewed through the API or the web interface.

Code listing 5.7: insert_db( )

```
def insert_db(res):
    myclient = pymongo.MongoClient(DBLINK)
    mydb = myclient["mydb"]
    mycol = mydb["scans"]
    mycol.insert_one(res)
```

Commandline arguments for scanner

The scanner script can be started with different command line arguments:

- **-v/–verbose** enables output to screen.
- **-t/–test** does not run a scan. Reads scan results of previous scan from JSON. It is used to test the enrichment process and saving to database (Stage 5.0 - 7.0).
- **-w/–write** writes new **JSON** files of scan.
- **-s/–skip** skips host discovery and fast port scan (Stage 1.0 - Stage 2.0). As described in figure 4.1, the scanner by default ignores the hosts that do not have one of the most common ports open. This option ensures that the host will be scanned even if the most common ports are closed. By enabling this option the scanner uses significantly more time, but performs full scan of all hosts.

5.2.2 Autoenum MongoDB

The database itself is also Dockerized. The database relies on a volume to assure data consistency whenever the Container is stopped or restarted. Although the data saved on each Host object varies based on what the scanner finds, we have created a schema. The schema consists of the data listed in 5.3. All host objects have the keys in the table header. The different values associated to the keys will be put in place after the scan is done. The schema mainly consists of nested objects, which are essentially objects inside other objects.
Sample schema and database document are included in appendix G.
Each of the keys in the schema (table 5.3) contain the following:

ObjectId	**UUID**	**IP**	**Hostname**	**macaddress**	**osmatch**	**ports**	**state**	**scanstats**
ObjectId	UUID	String	Object	Objects	Objects	Objects	Object	Object

Table 5.3: Sample database schema

- **ObjectId** The mongoDB object id for the database document
- **UUID**: UUID for the object. Queryable
- **IP**: IP address of the scanned host. Queryable
- **macaddress**: Contains MAC address, vendor and address type
- **Hostname**: Contains the name of the host and which method is used to find the name.
- **osmatch**: Array of objects. The objects contain data regarding OS matches which include: OS name, accuracy, CPEs and CVEs.
- **ports**: Array of objects which include: port protocol, port number, port state, service name, product name, version, CPEs and CVEs
- **state**: State of host and time to live
- **scanstats**: Date and time of scan

5.2.3 Autoenum API

We have made a RESTful API with ten different endpoints (table 5.4).

Method	**Endpoint**	**Returns**
GET	/	a helper message to show that the API is working
GET	**/all**	the whole database
GET	**/log**	scanner log for debug purposes (**Not** JSON)
GET	/<cpe>	all host objects with the given CPE
GET	**/uuid/**<uuid>	the host object with the given UUID
GET	**/ip/**<ip address>	all host objects with given IP address
GET	**/date/**<YYYYMMDD>	all host objects scanned on a given date
GET	**/mac/**<mac address>	all host objects with the given MAC address
GET	**/picture/<filename>**	the requested image for viewing in web interface
GET	**/viewpicture/<filename>**	the requested image for download

Table 5.4: API endpoints

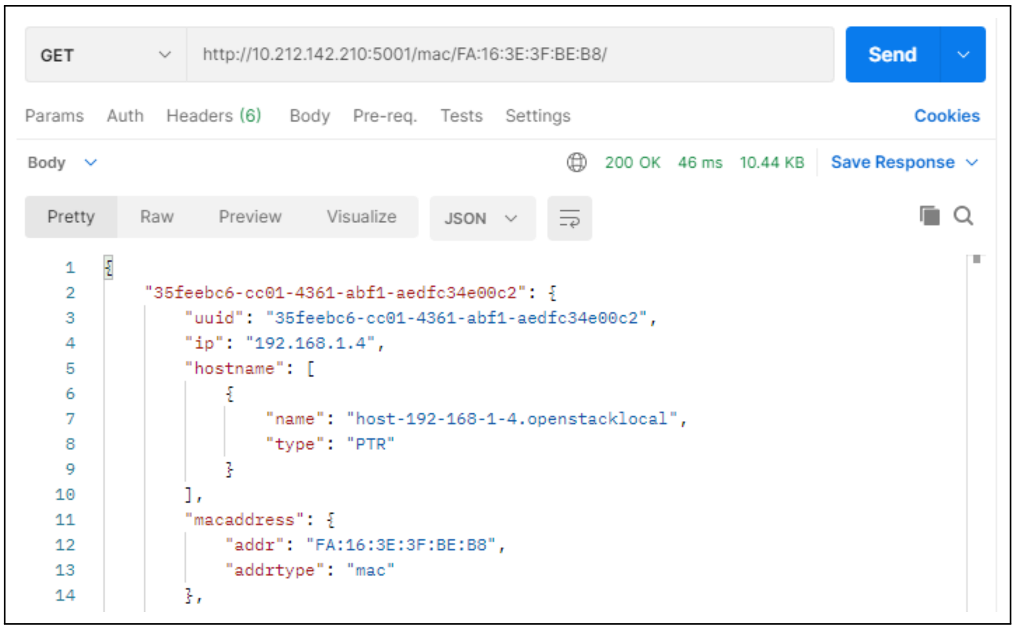
All of the endpoints are called by using the GET method, as there is no need to add or update data by using the **POST** method. The different endpoints enable a user to find Host objects based on data he or she already possesses. In the case where one does not have any data, the whole database can be returned by calling the /**all** endpoint.
When calling an endpoint, a query is designed based on the respective endpoint

and the parameter (listing 5.8). After the query has been designed, the query is passed on to **find_in_db( )** as a parameter. The function then connects to the database, queries the database and finally returns the results.

Code listing 5.8: Code for MAC endpoint

```python
@app.route("/mac/<string:macaddr>/", methods=['GET'])
def mac(macaddr):
  query = { "macaddress.addr" : macaddr }
  return find_in_db(query)
```

Figure 5.1 shows a GET-request being made to the MAC endpoint (5.8). The response from the API contains the results from the database in JSON format along with HTTP status code 200[4], which means that the request made has succeeded.



```
GET    http://10.212.142.210:5001/mac/FA:16:3E:3F:BE:B8/

200 OK   46 ms   10.44 KB

{
    "35feebc6-cc01-4361-abf1-aedfc34e00c2": {
        "uuid": "35feebc6-cc01-4361-abf1-aedfc34e00c2",
        "ip": "192.168.1.4",
        "hostname": [
            {
                "name": "host-192-168-1-4.openstacklocal",
                "type": "PTR"
            }
        ],
        "macaddress": {
            "addr": "FA:16:3E:3F:BE:B8",
            "addrtype": "mac"
        },
```

Figure 5.1: Example of calling MAC endpoint and response

Caching has been implemented in the API to speed up the response time in cases where the same request is made to the API within 5 minutes of the last one. Full code for API can be found in listing C.2.

[4]https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#successful_responses

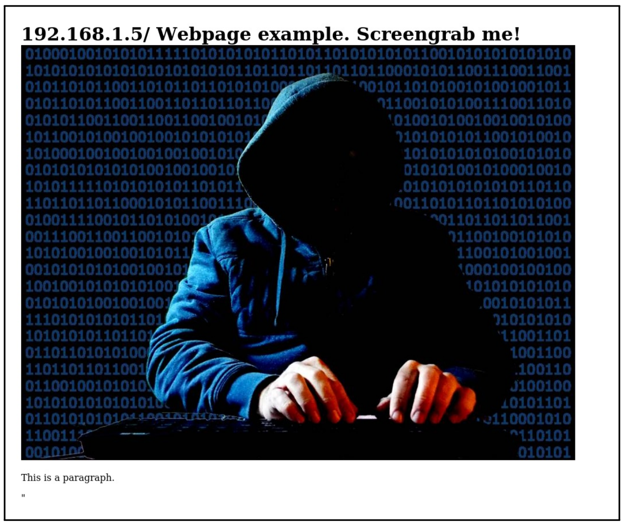
5.2.4 Autoenum screengrabber

The screengrabber (5.9) responds to GET-requests made by the scanner (5.6). The request includes the IP address of the host to screengrab.

Code listing 5.9: Code snippet screengrabber

```
    imgkit_options= { 'quiet' : ''}
    response = {}
    response['date'] =  date
    response['time'] = time

    try:
        imgkit.from_url(ip, path, options=imgkit_options)
    except ConnectionRefusedError:
        response['Message'] = "Connection refused"
    except IOError:
        response['Message'] = "IOError on"
    else:
        response['Filename'] = filename

    return jsonify(response)
```

The screengrabber then proceeds to download the HTML for the requested IP and renders the result into an image. The image is saved onto a volume shared with the API. The filename is generated based on the current time and date, and a random number is added to the filename.



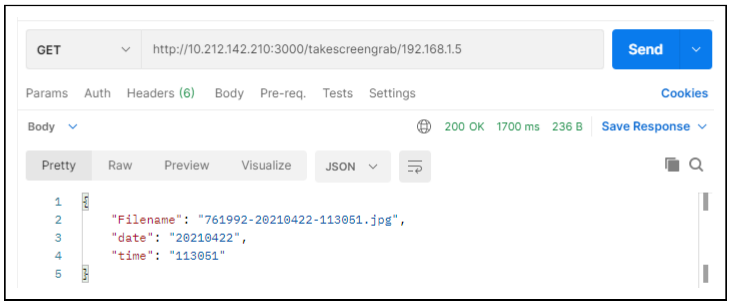
192.168.1.5/ Webpage example. Screengrab me!

This is a paragraph.

"

Figure 5.2: Web site hosted on web server in test environment

The filename, time and date is sent back to the scanner in a JSON response as shown in 5.3. The scanner saves the details about the image in the database along with the scan results for the given host. A sample screengrab is shown in figure 5.2.



```
GET http://10.212.142.210:3000/takescreengrab/192.168.1.5
```

```
{
    "Filename": "761992-20210422-113051.jpg",
    "date": "20210422",
    "time": "113051"
}
```

Figure 5.3: Response from screengrabber

5.2.5 Autoenum web interface

The web interface is inspired by the Model-View-Controller pattern, also known as MVC. The MVC is a software designing pattern, which separates code into three separate sections. Each of those sections has a purpose and depends on the rest. As seen in 5.10:

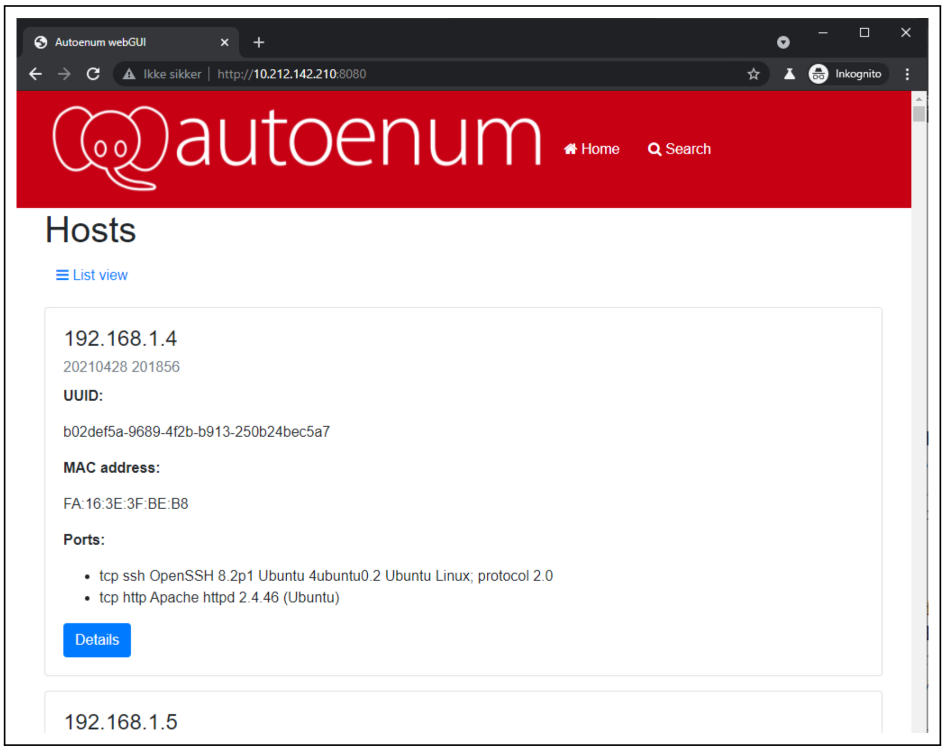
- **Model** is not fully implemented as there is no need to manually add data. *scan.js* is the schema for the database. If it had been properly implemented it would be used to ensure the format and data types before being saved to the database.
- **Views directory:** contains all the views for the web interface and they are written in EJS. These are only templates which will be rendered by the controllers.
- **Controllers:** */server.js* and */routes/hosts.js*. The controllers render the views when requested by the user. When a user wants to access detailed data about a host the controller is responsible for retrieving the data from the database and creating the view that is requested by the user.

Code listing 5.10: MVC of web interface

```
auto/Docker/website/
|-- Dockerfile
|-- models
|   `--  scan.js
|-- node_modules [...]
|-- package.json
|-- package-lock.json
|-- routes
|   `--  hosts.js
|-- server.js
`--  views
    |-- banner_logo.png
    |-- details.ejs
    |-- header.ejs
    |-- index.ejs
    |-- list.ejs
    |-- Logo.png
    |-- search.ejs
    `-- searchresults.ejs
```

The web interface (code in appendix C.3) allows the user to interact with the data aggregated by the system. However the web interface is not the preferred way of interacting with the data. There are some limitations on what data is displayed compared to the API, which essentially returns the whole host object. The web interface is a proof of concept, but with many core functionalities implemented:

- **Home page** (figure 5.4) where basic information such as IP addresses, MAC addresses, Ports about all host objects is displayed as cards
- **Basic information** displayed in list format (figure H.4)
- **A detailed view** of each host object (figure H.1). Displays all results from OS-detection, ports, hostname, state, screengrabs, CVEs with links to the CVE search web interface (figure 5.6) and a button to get the host object from the database in JSON
- **Host object search** based on IP, MAC, UUID, CPE and date (figure H.5).



Hosts

192.168.1.4
20210428 201856
UUID:
b02def5a-9689-4f2b-b913-250b24bec5a7
MAC address:
FA:16:3E:3F:BE:B8
Ports:
- tcp ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 Ubuntu Linux; protocol 2.0
- tcp http Apache httpd 2.4.46 (Ubuntu)

192.168.1.5

Figure 5.4: Web interface: Home. More screenshots in appendix H

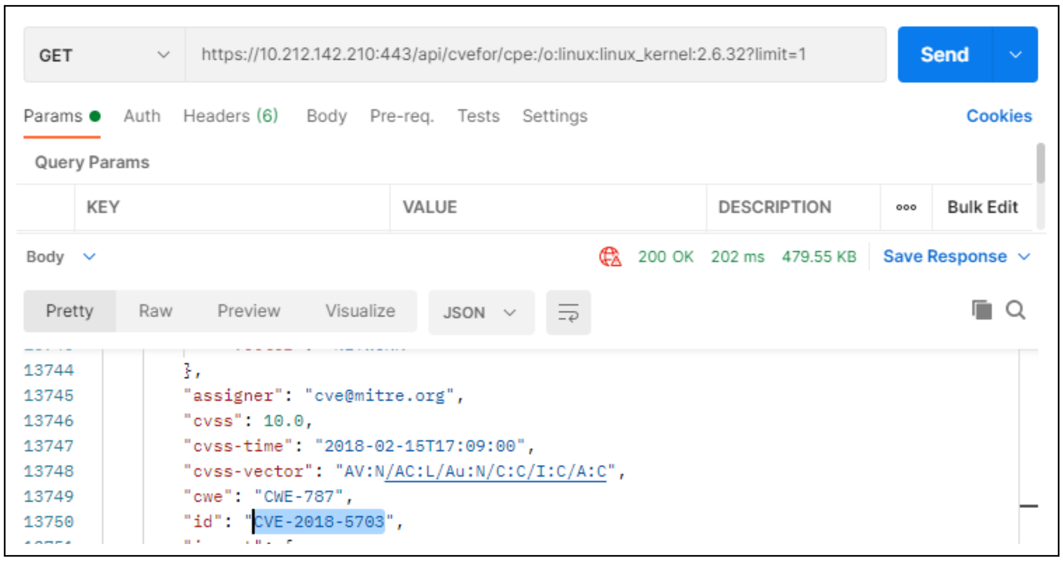
The web interface can not be used to modify or add data. It simply displays some data about each host object in the database.

5.2.6 CVE database

The CVE search relies on a database to store all the CVEs. The GitHub repository used for the CVE search[5] includes database dumps.
When looking for CVEs in the network, we could have easily used a Nmap script. The challenge by doing this is that there would be a lot of unnecessary network traffic in order to get the data. The solution to this problem is to host a local copy of the CVE database. By hosting it on our local system, the responses will likely be quicker. However, the biggest downside to hosting the database ourselves, is that we must ensure that the database is always up to date.

5.2.7 CVE API and search

This container provides both the API and web interface for CVE search. When the scanner calls the API, the CPE is sent in the GET-request. The API then queries the CVE database for the provided CPE.

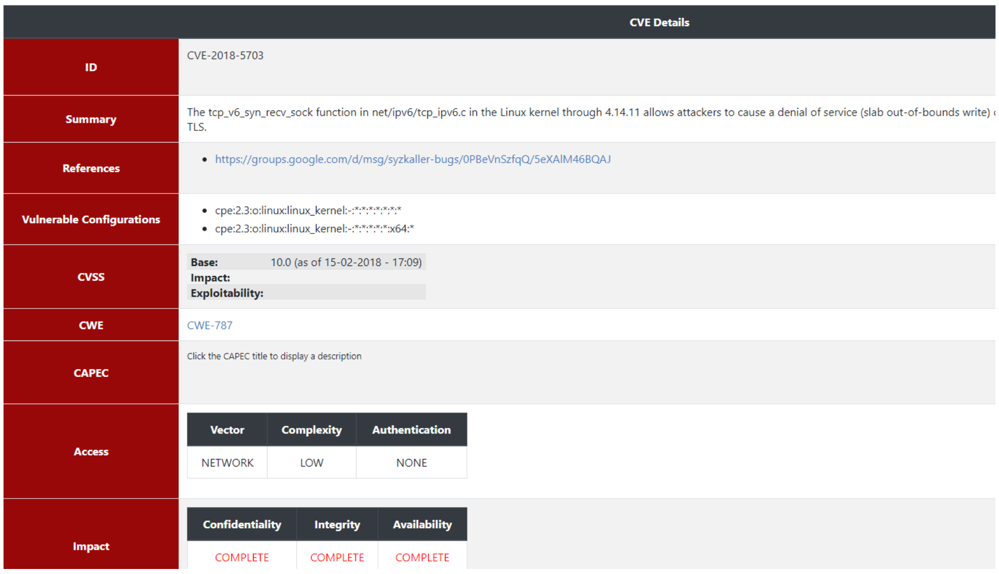


```
GET https://10.212.142.210:443/api/cvefor/cpe:/o:linux:linux_kernel:2.6.32?limit=1

200 OK  202 ms  479.55 KB

    },
    "assigner": "cve@mitre.org",
    "cvss": 10.0,
    "cvss-time": "2018-02-15T17:09:00",
    "cvss-vector": "AV:N/AC:L/Au:N/C:C/I:C/A:C",
    "cwe": "CWE-787",
    "id": "CVE-2018-5703",
```

Figure 5.5: Making request to CVE API and receiving response

The matches in the database are returned to the scanner in JSON. The response includes the CVE itself, along with other details about the CVE. The scanner keeps the CVEs and stores them in the host object. The other details will not be saved, but can be accessed through the CVE search web interface when searching for a CVE.
The CVE search web interface (5.6) provides the user with an easier way of reading the details about a CVE, and the aforementioned ignored details can be read on the web interface.

[5]`https://github.com/cve-search/CVE-Search-Docker`



CVE Details	
ID	CVE-2018-5703
Summary	The tcp_v6_syn_recv_sock function in net/ipv6/tcp_ipv6.c in the Linux kernel through 4.14.11 allows attackers to cause a denial of service (slab out-of-bounds write) TLS.
References	• https://groups.google.com/d/msg/syzkaller-bugs/0PBeVnSzfqQ/5eXAlM46BQAJ
Vulnerable Configurations	• cpe:2.3:o:linux:linux_kernel:-:*:*:*:*:*:*:* • cpe:2.3:o:linux:linux_kernel:-:*:*:*:*:*:x64:*
CVSS	**Base:** 10.0 (as of 15-02-2018 - 17:09) **Impact:** **Exploitability:**
CWE	CWE-787
CAPEC	Click the CAPEC title to display a description
Access	**Vector:** NETWORK; **Complexity:** LOW; **Authentication:** NONE
Impact	**Confidentiality:** COMPLETE; **Integrity:** COMPLETE; **Availability:** COMPLETE

Figure 5.6: CVE search web interface

5.2.8 CVE Redis

The Redis container is used to cache the responses from the database. When the API checks the database for CVEs it checks the Redis cache first to see if there are any entries there. If it does not find anything there it checks the database directly. This is exactly why the CVE search uses Redis. It reduces the latency when the same query is made multiple times, something which is highly likely when looking for CVEs in a large network where multiple servers are running the same OS or applications.

5.2.9 Volumes

Autoenum creates volumes for the different containers to achieve data persistency and allow the containers to share files with each other and the host. Figure 4.2 explains which container interacts with which volume:

1. Autoenum MongoDB, CVE Redis and CVE DB have their own volumes for data persistency
2. Autoenum API and the scanner share a volume. This is simply so that the user can access the scanner log through the API, and doesn't have to log in to the host and find the file
3. Autoenum API and Autoenum Screengrabber share a volume, so that the API can access the pictures the screengrabber takes.

5.3 Setup and deployment

Ansible is the tool that we used to configure the testing environment and to install the application. As mentioned in 2.4.1 we made a set of Ansible roles and Playbook (appendix D.1) to automate the deployment. The playbook was based on the repositories listed in table 6.3.

Setup

Before deploying Autoenum, some prerequisites have to be installed (listing 5.11). After installing the prerequisites and Ansible, the code must be cloned from the GitHub repository and copied to the **roles** directory (directory tree appendix J.1). The last step is to add the network or the IP the user wants to scan.

Code listing 5.11: Setup

```
# Prerequisites for Ansible
sudo apt install git
sudo apt install python3.8
sudo apt install python3-pip -y
sudo apt-get install ansible -y

# Clone code
mkdir -p /etc/ansible/roles && cd /etc/ansible
git clone https://github.com/asm492/autoenum
cp -r autoenum/autoenum /etc/ansible/roles

# Adding targets
echo "192.168.1.0/24" > /etc/ansible/autoenum/scanner/target.txt
```

Running

The next step is to run the Playbook which installs the dependencies for Autoenum and starts the containers. Finally when the containers are running, the scanner can be started. The scanner can be started directly from the command line, but the preferred way of running the scanner is through a job scheduler like cron.

Code listing 5.12: Running

```
# Run the Playbook
ansible-playbook /etc/ansible/autoenum/example-playbook.yml

# Check if containers are running
docker ps

# Output should be similar to this
CONTAINER ID   STATUS      PORTS                        NAMES
c39fd8f11198   Up 5 days   0.0.0.0:443->5000/tcp        files_cve_search_1
e10043e5b54b   Up 5 days   0.0.0.0:8080->8080/tcp       autoenum-webgui
5a41b27408df   Up 5 days   27017/tcp                    files_mongo_1
3cef26891b1e   Up 5 days   0.0.0.0:3000->3000/tcp       autoenum-screengrab
503901e3e48b   Up 5 days   0.0.0.0:27018->27017/tcp     autoenum-mongodb
07c432a1cb83   Up 5 days   6379/tcp                     files_redis_1
8c89cc44cfc8   Up 5 days   0.0.0.0:5001->5001/tcp       autoenum-api
```

```
# Create a cronjob, save and quit. Example runs at midnight every Wed:
crontab -e
0 0 * * 3 python3 /etc/ansible/scanner.py
```

Configuration

Listing 5.13 shows some of the variables of interest in Autoenum. One might like to change them before deploying Autoenum.

Code listing 5.13: Possible configuration variables for Autoenum

```
# Options/settings
    # autoenum/autoenum/files/api/app.py:
        DB_URL = "mongodb://autoenum-mongodb:27017/"
        DB_NAME = "mydb"
        COLLECTION_NAME = "scans"
        CACHE_TIME = 300
    # autoenum/autoenum/scanner/scanner.py
        DB_URL = 'mongodb://localhost:27018/'
        DB_NAME = "mydb"
        COLLECTION_NAME = "scans"
        TARGETFILE = "target.txt"
        LOG_FORMAT = "%(name)s %(asctime)s - %(message)s"
        # Path and name of log:
        FILENAME = "/var/lib/docker/volumes/files_log-volume/_data/Scanner.log"
    # autoenum/autoenum/scanner/cve_lookup.py
        LIMIT = "10"    # Limits the number of CVEs returned by CVE API
```

5.3.1 Recommendations

After running Autoenum, it will effectively produce a complete map of the network or at least the scanned hosts. If someone with the wrong intentions gets access to the data, they will have knowledge of how to attack the different hosts. The following recommendations should be met before deploying Autoenum in a production environment:

- **Secure database:** As a minimum requirement, there should be implemented a user account with a strong password as shown in listing 5.14.

  Code listing 5.14: Adding username and password to mongoDB

  ```
  #Add this to Docker compose file
      environment:
        MONGO_INITDB_ROOT_USERNAME: root
        MONGO_INITDB_ROOT_PASSWORD: databasepassword
  ```

- **Data replication:** To make sure that the data collected by Autoenum stays intact, a database should be redundant. It means that the data should be held in at least two different places simultaneously, so the data is not lost when one of the databases loses its availability. In our case, the easiest way to achieve data redundancy is to replicate the database, using Docker containers spread across different hosts. It would also be wise to frequently back up the database, so the data stays intact if a critical situation occurs. Those steps should provide a persistent data integrity.

- **Access Control List:** There should be an Access control list blocking unauthorized hosts from connecting to the machine where Autoenum is deployed.

6. *Testing*

This chapter gives an overview of the testing environment in OpenStack and how it was used throughout the development process of Autoenum. The last part of this chapter was included to discuss the results from a beta test conducted by the Taskgiver.

6.1 Test environment

In this part of the chapter we go more in depth and give an overview of our testing environment, some tables and figures for visualizing with more detailed information and descriptions. To be able to verify that Autoenum could do what it was supposed to do, we had to create a test environment.

6.1.1 Overview

We created our test environment in SkyHiGh. The testing environment consisted of seven servers in total, which ran different OS or different versions. To deploy the testing environment we had chosen to use HOT-templates. The biggest argument to use HOT-templates is that they provide consistency and re-usability across deployments, which results in the exact same test environment being deployed each time as long as the deployer has access to OpenStack.
When the stack is deployed, it starts up the new instances and runs a startup script on the VM's to prepare them for Ansible. The Playbook is then run by the manager node, provisioning the worker nodes to the desired configuration. This includes setting up user accounts for Ansible, downloading and installing needed software and changing the software configuration to the desired state.

6.1.2 Topology

As seen in figure 6.1 the test environment consists of two networks. This was necessary when testing if the scanner was able to detect open Ports and OS' across the different networks.

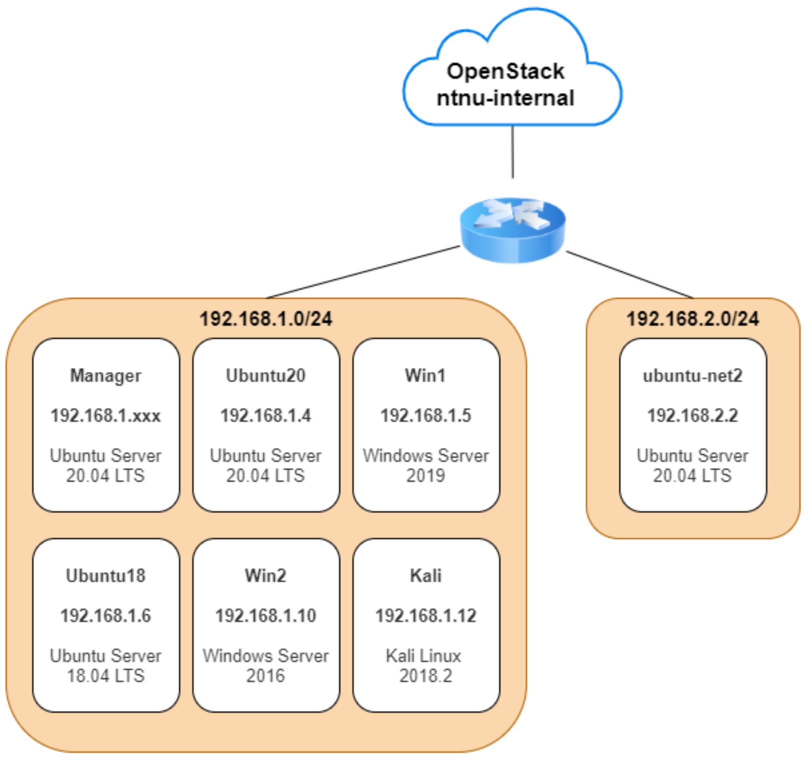


Figure 6.1: Topology of test environment

Table 6.1 lists the different Images we used in the test environment. We included some Debian-based OS' and two different Windows versions. Having different OS' was important to see if the scanner could distinguish between the different OS' and their CPEs. Since the servers were running different applications, we deployed them by using a couple of different Flavors.

Instance name	IP Address	Image	Flavor
Manager	Dynamic	Ubuntu Server 20.04 LTS	m1.large
Ubuntu20	192.168.1.4	Ubuntu Server 20.04 LTS	m1.tiny
Win1	192.168.1.5	Windows Server 2019 Standard	m1.small
Ubuntu18	192.168.1.6	Ubuntu Server 18.04 LTS	m1.tiny
Win2	192.168.1.10	Windows Server 2016 Standard	m1.small
Kali	192.168.1.12	Kali Linux 2018.2 xfce	m1.small
ubuntu-net2	192.168.2.2	Ubuntu Server 20.04 LTS	m1.tiny

Table 6.1: Instances in OpenStack test environment

6.1.3 Services

To be able to verify that the scanner worked as intended, the VMs in the test environment had to be configured with a couple of different services (table 6.2) that could be picked up by the scanner. With the exception of OpenSSH[1] on all Linux based machines, we installed and configured different services on each of the VMs. Some of the installed services in the test environment were deliberately chosen because they had known vulnerabilities. This was crucial, as we would have to test that our service was able to find CVEs in the network.

Instance name	**Port**	**Service/Product**
Ubuntu20	22	OpenSSH 8.2p1 Ubuntu 4
	80	Apache httpd v2.4.46
Win1	80	Microsoft IIS httpd v10.0
	443	Microsoft IIS httpd v10.0
	3389	Microsoft Terminal Services (RDP)
	5985	Microsoft HTTPAPI httpd v2.0 (WinRM)
	5986	Microsoft HTTPAPI httpd v2.0 (WinRM)
Ubuntu18	21	vsftpd v2.3.4
	22	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
	23	Linux telnetd
	139	Samba smbd 3.X - 4.X
	445	Samba smbd 3.X - 4.X
Win2	80	Apache httpd v2.4.46
	135	Microsoft Windows RPC
	445	Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
	3389	Microsoft Terminal Services (RDP)
	5985	Microsoft HTTPAPI httpd v2.0 (WinRM)
	5986	Microsoft HTTPAPI httpd v2.0 (WinRM)
	49669	Microsoft Windows RPC
	49731	Microsoft Windows RPC
Kali	22	OpenSSH 7.7p1 Debian 2
	5901	VNC protocol 3.8
	6001	X11
ubuntu-net2	22	OpenSSH 8.2p1 Ubuntu 4

Table 6.2: Services running on the instances

[1]Utility to enable SSH-access on a machine, `https://www.openssh.com/`

6.1.4 Roles

The roles described in table 6.3 are roles we found on GitHub or Ansible Galaxy. The rest of the roles (listed in table 6.2) were implemented manually. The whole code was based on the official Ansible documentation[2]. The roles we found on GitHub and Galaxy are listed in table 6.3.

Name	Description
dwva-ansible [65]	Installs and configures DVWA, alongside with apache2, mysql and php.
ansible-role-samba [66]	Installs and configures Samba file server. The installed version of Samba is prone to CVE-2017-7494
ansible-role-docker [67]	Installs and configures Docker and Docker-compose

Table 6.3: Roles used in project

6.1.5 Deployment

The test environment can be deployed to SkyHiGh by running the commands listed in listing 6.1. Figure 6.2 illustrates the process that will be explained in the next paragraph.

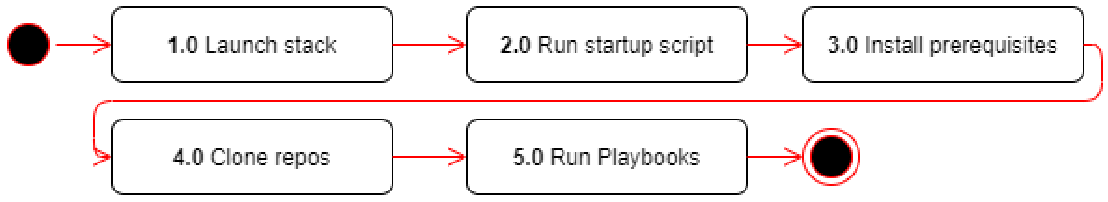


Figure 6.2: Flowchart deployment of test environment

[2]https://docs.ansible.com/

After running the command, OpenStack creates the requested Heat stack based on the provided template (E.1) and environment variables (E.2). During the deployment of the Windows VMs, a startup script (E.4) is injected to create a user account with administrator privileges which is to be used by Ansible. A script is also injected into the Manager node (E.3). This script installs the prerequisites for Docker, Ansible and the scanner itself. Following that, Ansible takes control and runs the Playbooks (D.2, D.2). The required services are installed (table 6.2) and configured into the desired state.

Code listing 6.1: Bash-commands to launch test environment

```
git clone https://github.com/asm492/auto
cd auto/heat/
openstack stack create -t heat.yaml -e env.yaml AutoEnumStack
```

6.2 Testing

We used our test environment a lot during Autoenum's development. The advantage of choosing a microservice architecture and an agile development model, was that we could write a small amount of code that could be tested and changed immediately. Whenever we added functionality or changed the code, we ran a test right away. Testing was done in parallel with the development to catch any errors at an early stage. The tests were comprised of three parts:

- **Unit test:** Whenever we added a new component to Autoenum, we performed a unit test. Since all of Autoenum's components are running in separate Containers, testing each of them individually was simple and effortless. In the case of the API, CVE search and screengrabber we performed the unit tests by making API calls through Postman[3].
- **Integration test:** After verifying that the component worked individually we integrated the respective component with the existing ones. We then tested the components to see if they worked correctly.
- **System test:** If the integration tests went as planned we proceeded with a full system test. We did this by starting the scanner and watching the output as it scanned. When the scan was finished, we checked the data in the database by calling all the API endpoints (section 5.2.3) and ensuring that the data was displayed correctly on the web interface.

6.3 Beta test

By the end of March 2021, we were almost done with **Autoenum**. We had some minor bug fixes to implement, but the main functionalities of the service were done. We therefore arranged a meeting with our Taskgiver. During the meeting we gave the Taskgiver a live demo of the system. After the presentation the taskgiver

[3]Application for building and testing APIs, `https://www.postman.com/`

suggested that they wanted to test **Autoenum** in NTNU SOC's own environment. This would be beneficial for both parties, as the group gets valuable feedback on how the system performs in a larger, real-world environment, and the employer get the opportunity to check if Autoenum fulfills their requirements. We promptly accepted the taskgivers request. After the demo they proposed some changes that had to be implemented before the test. The taskgiver also wanted concise documentation on how to deploy and run Autoenum, along with Playbook and the code.
We made the requested changes, wrote the documentation and Playbook over easter and gave the taskgiver access to the code. Our plan was to improve Autoenum based on the feedback from the beta test, however that fell through. Unfortunately the taskgiver did not have time to perform a beta test before the deadline of report, however he might be able to perform the test before the presentation of the project.

7. *Discussion*

This chapter discusses some of the decisions we made and reflect on the potential usefulness of Autoenum, as in trying to see the significance of Autoenum in relation to detection and incident management. This section will also relate to the theory chapter (2), previous reflections and project results.

7.1 Decisions

As with every project there will be decisions and discussions on what are the best ways to solve problems. In our case they often related to specifying the requirements, what technology to use and what the architectural design should look like.

7.1.1 Docker

Our initial plan was to Dockerize every component, but after a meeting with the taskgiver during the planning phase, we were advised not to do so as that would involve opening many ports in Docker to allow the traffic related to the scanner to pass through. It would introduce some latency as well when redirecting the traffic from the host to the container. We therefore settled on running the scanner script on the host itself, and dockerize everything else as figure 4.2 depicts.

7.1.2 Scanner

Prior to writing the code for the scanner it was clear that Autoenum would be used in large networks. Scanning a large network is time consuming. To reduce the time needed to complete a scan we had to reduce the number of IP addresses that would be scanned. We separated the scanning process into seven stages, where Stage 1.0 is performing a host discovery. The host discovery is done by sending multiple packets as mentioned in chapter 5.2.1. Only the hosts which respond are then written to a file used by the next stage. After the discovery is done, the host that were found are passed on to Stage 2.0 where a port scan of common open ports is performed. At this point we discovered a flaw in Autoenum. The flaw was that hosts might have open ports which are not a part of Nmap's common open port list[1], meaning that the IP would be ignored by the subsequent stages.

[1]https://nmap.org/book/nmap-services.html

To avoid discarding hosts that might have uncommon ports open, we added an optional argument (see chapter 5.2.1) to the scanner. Using this option comes with a significant time penalty, as it skips the host discovery and common port scan. Meaning that every single IP address (network and host) specified in the target file will be scanned for all TCP ports, without checking if the respective host is up or not. However, the scanner will likely find more hosts when using this option.

7.1.3 Database

Before we started writing the code for Autoenum, we decided to use MySQL as our database service. This was mainly because throughout our course of study, we only had experience using and deploying database services using MySQL. But when we added more functionality in each sprint, we spent a lot of time rewriting the code parsing and inserting the scan results into the database. This amounted to a lot of time wasted adapting the parser for each sprint. Another challenge we faced was that we had to perform many complex queries to the database when viewing the data. We would have to join the data from the different tables.
After a meeting with the taskgiver we settled on MongoDB, which had some big advantages over a relational database. The biggest advantage of using MongoDB over a relational database is that there was no need to update the schema and rewrite the code for every sprint. Using MongoDB also allowed us to sort the data by a Host object where all details about a host would be stored in one document rather than spread out in different tables. When retrieving the data from MongoDB, the response is a key-value pair, which makes the data easy to parse both when using it with Python in the API and when using it with NodeJS in the web interface.

7.1.4 Structure

Our report structure was often discussed and changed. Some inspiration was taken from past reports, suggestion document from NTNU and the typical SDLC. The discussion was often related with the chapters requirements, design and implementation. Even though it seems easy to separate and delegate what part fits the different chapters, no report follows a predefined structure perfectly. Based on what Autoenum was going to provide to the Taskgiver, and the given requirements, we decided to merge requirements and design. This was mainly done based on the lack of a end-user and use-case functionality. We felt these elements did not add any useful information as Autoenums functionality was described in the design chapter. This was also based on that Autoenum is a service that does not have a typical end user, but is rather managed by an administrator.

7.2 Purpose and usefulness

Network scanners and other similar technologies are very useful utilities. If used with the wrong intentions, legal issues and concerns may arise. One of the main concerns about network scanning is when a third party scans a network without the owners consent. Using Nmap to port scan is not illegal, because it does not have an official law that forbids it. If port scanning is legal or not, it often depends on the context, i.e using these kind of tools to disrupt systems and scan networks to find vulnerabilities that is not under ones responsibility and control.
To prevent concerns and consequences, it may be wise to follow some rules and develop some precautions before using these kind of tools as mentioned by Lyon[68]:

- **Permission:** It is important to ensure that one has permission. Although what one is doing is not meant to be harmful or done with malicious intentions, it is always best to get written consent and ensure one has permission to do so.
- **Specify:** Always try to specify and limit what information is gathered. Try to be specific with ports and limit full scans. This will both save time, reduce the intrusion of the scan and limit complaints from network owners.
- **Reason:** Make sure one has a reason for using these kind of tools. Try to have a clear and concise justification for the activity one is performing. Doing a port scan might not be a signal of malicious intentions, but if it is followed with the use of an exploit, the intentions becomes very clear.

Network scanning in itself is pretty harmless, but if used with malicious intent, it can lead to catastrophic consequences on the network. If a threat agent gets control over Autoenum, the agent can use the results from the scan to map out the network. This information can be later used to execute an attack, because the attacker has access to the network architecture and addresses. This is why it is important for Autoenum to be secure and to be placed in a controlled and heavily surveilled area of the NTNU SOC.

7.2.1 Detection and incident management

The usefulness by having a network scanner on your own network can be significant. It is important to know this is only a tool in a series of preparations. Managing incidents includes everything from preparing, preventing and having detection systems in place before incidents happen. It is important to find out how your company reacts and responds when a incident occurs.
Having a network scanner such as **Autoenum** can be beneficial in a incident management context. It can establish a baseline[69] of the hosts connected to the network, and more importantly update the baseline over time. The baseline can later be used to compare against when irregularities are detected to verify if something is to be classified as an incident or not. Having a baseline for system and network activity can be found listed under Incident Analysis Resources[70]. This is a list of tools and tips on how to prepare for incidents. Possessing the right data

at the right time can make the subsequent step, detection and analysis easier, as the incident responder can study the deviations between the historical baselines and the current situation to determine if i.e. Ports have been opened or new hosts have appeared on the network since the last scan.

7.3 Results

After working with the project for almost five months, we feel we have been able to achieve the goals for the project (1.7). After writing the report we feel that it provides a good documentation on **Autoenum**.

- **Make a usable system that has the functionality the taskgiver needs and has requested.** We are confident in that Autoenum fulfills all of the requirements stated by the taskgiver.
- **Improving the security of the NTNU network.** If Autoenum is used as intended, we believe it will help improve the level of security of the NTNU network. With the functionalities we have implemented, and if used correctly, it should at the very least improve the level of security slightly.
- **Provide a report that acts as supporting literature for the system.** We feel that the report we provided presents a complete overview of the product. It contains a balanced mix between practical and theoretical knowledge. The report is written in such a way that a non-technical person could easily understand the project and even use Autoenum, using the technical documentation.
- **The report and Autoenum should be relevant for future work.** We feel that our report has documented our thoughts behind Autoenum well, why we did the things we did and how. The report should be relevant for the members of our target group 1.3. Autoenum as a system will also be relevant for those who want to try it in their networks to increase the level of security.

7.4 Deviations

As discussed in the risk assessment part of our preliminary report (5.1 in appendix L) one of our primary concerns were that we might be delayed because of improper planning. Despite our concerns we were able to finish much of coding long before we initially planned to. On the other hand, there have also been moments where we have done some drastic changes to the code (i.e. changing from SQL to MongoDB). Being a bit ahead of the sprints has proven to be a good idea.
In addition to the task (appendix A) given by the Taskgiver we expanded our scope by implementing the CVE search as described in chapter 5.2. We were also able to develop a functional web interface. One of the risks mentions that the test environment does not simulate the a production environment well enough. This might be a limiting factor in our project, as we have not been able to test Autoenum on a network with more than seven hosts. This also leads us to discuss

risk number 6, which was described as decreased level of quality as a result of poor communication with taskgiver. As mentioned in chapter we held a presentation for the taskgiver at the end of March. During the meeting we were told that the taskgiver wanted to run a beta test of Autoenum after easter. This was a great opportunity for us to see if Autoenum could handle what it was made for. Our plan was to spend the last two sprints to improve or fix Autoenum based on the results from the beta test. Unfortunately the taskgiver has not had time to run the test, and therefore been unable to provide us with any feedback. This resulted in submitting Autoenum without improving it.
As mentioned in the methodology chapter 3 we have followed the software development framework Scrum during the project period. We had set up a total of seven sprints 3.2 at the start of the project. We found out after the first and second sprint, we were progressing a lot faster than planned. We made an overall quality check on what we had previously done and found nothing wrong with the work we had done. Then we continued our pace throughout the project, which led to us finishing Autoenum during sprint 4.

7.5 Project criticism

While the group is satisfied with the outcome of the project, there are some things we would do differently if we have had the time.

- The system is handling sensitive data that should be kept away from a potential intruder. The API should have used an API key.
- The system is lacking a way of easily configuring settings and parameters. This could i.e. have been done by using a YAML-file or through the web interface.
- To scale better, the cache in the API should have been implemented by having a separate container running Memcached or Redis, just like the one used in CVE search.
- Autoenum only works with IPv4. We should have implemented support for IPv6.
- Our test environment is very small and limited, and it may not properly imitate a real environment. It should have included other non-Debian based OS'. They were omitted as writing Playbooks would have taken a long time.

8. *Conclusion*

This chapter concludes our project and address some concluding topics. These include our work ethic, further work, concluding our work and closing statement.

8.1 Work ethic

We had a good work ethic throughout the project period. We managed to work from Monday to Friday every week during the writing period. With some exceptions, we accomplished to keep that work schedule. There is a graph of the total hours worked during the project for the whole group, that can be found in appendix M. Throughout the whole process of designing Autoenum and writing the report, we were highly motivated and focused on completing the task. The task itself was interesting and challenged what we had learned throughout our course of study. That is why we chose it out of the many task proposals for our bachelor project.
Our division of labour throughout the project has been quite clear. Two of our group members did most of the technical work, with the coding, setting up **Autoenum** and so on. The other two worked mostly on the report, trying out different structures, adding or removing more theory and so on. This division of labour has worked great for us, as we managed to have a report that followed the system throughout the project, rather than writing the report at the very end of the project.

8.2 Further work

Despite the fact that the project fulfills all of the taskgivers requirements, the group members agree that we could have extended the project to include more functionality:

- Our first priority would be to fix and implement the changes in chapter 7.5.
- Add more functionality to the web interface. i.e.: The functionality to define scan targets and start scans manually from web interface or by making a POST-request to the API.
- Refactor and optimize the code, as it could be too slow when scanning a larger network.

- Add a service that notifies SOC-staff when a new CVE is found in the network.

8.3 Conclusion of the work

Our Taskgiver wanted a service which automatically discovers and enumerates hosts, open ports and running services on a network. We feel confident that we have delivered a product that meets their demands and expectations. As discussed in chapter 7.3, we have achieved and fulfilled the project goals listed in chapter 1.7. The task has provided us with the opportunity to go more in depth into and use technologies and methodologies we were already familiar with, as well as learning new technologies and combining them with our prior knowledge.
After working on this project for 5 months, we can finally say that we are done. The task has provided us with a great learning experience and tested our skills and knowledge. The group is very satisfied with the outcome of both Autoenum and the report, even though there were things that we could have done better. We hope that both Autoenum and the report can be of value to the members of our target groups.

Bibliography

[1] R. Shirey, *Internet security glossary, version 2*, https://tools.ietf.org/html/rfc4949, [Online; accessed 27-April-2021].

[2] Wikipedia contributors, *Cloud computing — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Cloud_computing&oldid=1016481284, [Online; accessed 9-April-2021].

[3] NIST, *Official common platform enumeration (cpe) dictionary*, https://nvd.nist.gov/products/cpe, [Online; accessed 02-March-2021].

[4] Red Hat, *What is a cve?* https://www.redhat.com/en/topics/security/what-is-cve, [Online; accessed 05-March-2021].

[5] Docker, *What is a container?* https://www.docker.com/resources/what-container, [Online; accessed 23-March-2021].

[6] Docker, *Dockerfile reference*, https://docs.docker.com/engine/reference/builder/, [Online; accessed 24-March-2021].

[7] OpenStack, *Heat*, https://wiki.openstack.org/wiki/Heat, [Online; accessed 24-February-2021].

[8] T. Berners-Lee, *Html - living standard*, https://html.spec.whatwg.org/#is-this-html5?, [Online; accessed 28-April-2021].

[9] Wikipedia contributors, *System image — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=System_image&oldid=973253530, [Online; accessed 22-March-2021], 2020.

[10] Wikipedia contributors, *Internet control message protocol — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Internet_Control_Message_Protocol&oldid=1014631093, [Online; accessed 6-April-2021].

[11] Wikipedia contributors, *Library (computing) — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Library_(computing)&oldid=1017253221, [Online; accessed 16-April-2021].

[12] RedHat, *What are microservices?* https://www.redhat.com/en/topics/microservices/what-are-microservices, [Online; accessed 03-May-2021].

[13] Microsoft, *Onedrive*, https://www.microsoft.com/nb-no/microsoft-365/onedrive/online-cloud-storage, [Online; accessed 31-March-2021].

[14] Wikipedia contributors, *Openstack — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=OpenStack&oldid=1006548754, [Online; accessed 24-February-2021].

[15] Overleaf, *Overleaf*, https://www.overleaf.com/, [Online; accessed 31-March-2021].

[16] Red Hat, *What is an ansible playbook?* https://www.redhat.com/en/topics/automation/what-is-an-ansible-playbook, [Online; accessed 24-February-2021].

[17] A. Josey, *Posix™ 1003.1 frequently asked questions (faq version 1.18)*, http://www.opengroup.org/austin/papers/posix_faq.html, [Online; accessed 24-March-2021].

[18] SSH, *Ssh protocol*, https://www.ssh.com/ssh/protocol/, [Online; accessed 23-March-2021].

[19] Wikipedia contributors, *Soap — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=SOAP&oldid=1013240336, [Online; accessed 23-March-2021].

[20] Wikipedia contributors, *Cloud computing — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Systems_development_life_cycle&oldid=1014869490, [Online; accessed 10-April-2021].

[21] Oracle, *What is a database*, https://www.oracle.com/database/what-is-database/, [Online; accessed 24-February-2021].

[22] P. Fox, *Transmission control protocol (tcp)*, https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp, [Online; accessed 16-April-2021].

[23] P. Fox, *User datagram protocol (udp)*, https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/user-datagram-protocol-udp, [Online; accessed 16-April-2021].

[24] Microsoft, *What is a virtual machine?* https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/, [Online; accessed 22-March-2021].

[25] yaml, *What it is*, https://yaml.org/, [Online; accessed 23-March-2021].

[26] I. Sommerville, *Software Engineering*. Pearson, 2016, p. 45.

[27] K. Schwaber and J. Sutherland, *The scrum guide™*, https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf, [Online; accessed 09-March-2021].

[28] *Scrum Framework*, https://scrumorg-website-prod.s3.amazonaws.com/drupal/2021-01/Scrumorg-Scrum-Framework-tabloid.pdf, [Online; accessed 16-May-2021].

[29] K. Morris, *Infrastructure as Code*. O'Reilly Media, 2016, pp. 3–13.

[30] S. J. Bigelow, *It automation*, https://searchitoperations.techtarget.com/definition/IT-automation, [Online; accessed 24-February-2021].

[31] D. Kaplan, *What is Vulnerability Management?* https://www.siemplify.co/blog/vulnerability-management/, [Online; accessed 18-May-2021].

[32] Kaspersky, *The Human Factor in IT Security: How Employees are Making Businesses Vulnerable from Within*, https://www.kaspersky.com/blog/the-human-factor-in-it-security/, [Online; accessed 08-May-2021].

[33] Puppet, *Overview of puppet's architecture*, https://puppet.com/docs/puppet/5.5/architecture.html, [Online; accessed 23-March-2021].

[34] Chef, *Chef infra client (executable)*, https://docs.chef.io/ctl_chef_client/, [Online; accessed 23-March-2021].

[35] Red Hat Ansible, *Overview how ansible works*, https://www.ansible.com/overview/how-ansible-works, [Online; accessed 23-March-2021].

[36] Ansible, *Roles*, https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html, [Online; accessed 23-March-2021].

[37] Ansible, *Setting up a windows host*, https://docs.ansible.com/ansible/latest/user_guide/windows_setup.html/#windows-ssh-setup, [Online; accessed 23-March-2021].

[38] G. Lyon, *Nmap Network Scanning*. Insecure.Com LLC, 2008, p. 1.

[39] Geeksforgeeks, *Tcp 3-way handshake process*, https://www.geeksforgeeks.org/tcp-3-way-handshake-process/, [Online; accessed 24-March-2021].

[40] G. Lyon, *Port scanning techniques*, https://nmap.org/book/man-port-scanning-techniques.html, [Online; accessed 06-April-2021].

[41] NIST, *Vulnerabilities*, https://nvd.nist.gov/vuln, [Online; accessed 24-February-2021].

[42] K. Morris, *Infrastructure as Code*. O'Reilly Media, 2016, pp. 70–79.

[43] K. Lane, *What exactly is an api?* https://blog.postman.com/intro-to-apis-what-is-an-api/, [Online; accessed 27-April-2021].

[44] R. T. Fielding, 'Architectural styles and the design of network-based software architectures,' p. 76, 2000. [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

[45] K. Lange, *THE LITTLE BOOK ON REST SERVICES*. Kenneth Lange, 2016, pp. 5–8. [Online]. Available: https://www.kennethlange.com/books/The-Little-Book-on-REST-Services.pdf.

[46] *Rest url explanined*, https://avaldes.com/wp-content/uploads/2017/08/REST_URL_structure.png, [Online; accessed 28-April-2021].

[47] REST API, *HATEOAS Driven REST APIs*, https://restfulapi.net/hateoas/, [Online; accessed 30-April-2021].

[48] Mozilla Corporation and contributors, *Http request methods*, https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods, [Online; accessed 27-April-2021].

[49] MongoDB, *What is mongodb?* https://www.mongodb.com/what-is-mongodb, [Online; accessed 03-March-2021].

[50] Express, *Express*, https://expressjs.com/, [Online; accessed 03-May-2021].

[51] M. Eernisse, *What is ejs?* https://ejs.co/, [Online; accessed 29-April-2021].

[52] G. Lyon, *Nmap Public Source License Version 0.93*, https://nmap.org/npsl/npsl-annotated.html, [Online; accessed 12-May-2021].

[53] nmmapper, *python3-nmap*, https://pypi.org/project/python3-nmap/, [Online; accessed 12-May-2021].

[54] J. Truelsen and A. Kulkarni, *wkhtmltopdf*, https://github.com/wkhtmltopdf/wkhtmltopdf, [Online; accessed 12-May-2021].

[55] C. Dolphin and O. College, *IMGKit: Python library of HTML to IMG wrapper*, https://pypi.org/project/imgkit/, [Online; accessed 12-May-2021].

[56] D. Lord, *flask*, https://github.com/pallets/flask/tree/0.12.x, [Online; accessed 12-May-2021].

[57] M. Dirolf and B. Hackett, *pymongo*, https://github.com/mongodb/mongo-python-driver/tree/v3.11, [Online; accessed 12-May-2021].

[58] GitHub user P-T-I, *CVE Search Docker*, https://github.com/cve-search/CVE-Search-Docker/tree/v1.4, [Online; accessed 12-May-2021].

[59] A. Diyan, *pywinrm*, https://github.com/diyan/pywinrm/tree/v0.1.x, [Online; accessed 12-May-2021].

[60] Nmap, *Chapter 15. Nmap Reference Guide: Host discovery*, https://nmap.org/book/man-host-discovery.html, [Online; accessed 04-May-2021].

[61] Cybersecurity and infrastructure security agency, *Udp-based amplification attacks*, https://us-cert.cisa.gov/ncas/alerts/TA14-017A, [Online; accessed 28-April-2021].

[62] M. Majkowski, *Reflections on reflection (attacks)*, https://blog.cloudflare.com/reflections-on-reflections/, [Online; accessed 28-April-2021].

[63] ntp.org, http://www.ntp.org/, [Online; accessed 06-May-2021].

[64] smb.org, *Just what is SMB?* https://www.samba.org/cifs/docs/what-is-smb.html, [Online; accessed 06-May-2021].

[65] J. Townsend, *dvwa-ansible*, https://github.com/L1ghtn1ng/dvwa-ansible/tree/master/roles, [Online; accessed 15-May-2021].

[66] B. V. Vreckem, *ansible-role-samba*, https://github.com/bertvv/ansible-role-samba, [Online; accessed 15-May-2021].

[67] J. Geerling, *ansible-role-docker*, https://github.com/geerlingguy/ansible-role-docker, [Online; accessed 15-May-2021].

[68] G. Lyon, *Chapter 1. getting started with nmap*, https://nmap.org/book/legal-issues.html, [Online; accessed 26-April-2021].

[69] C. Christianson, *Baselines and incident handling*, https://www.sans.org/reading-room/whitepapers/incident/baselines-incident-handling-2068, [Online; accessed 20-April-2021].

[70] P. Cichonski, T. Millar, T. Grance and K. Scarfone, 'Computer security incident handling guide,' pp. 21–23, 2012. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf.

A. *Task description*

AUTOENUM: Automatisk kartlegging og ekponeringsanalyse av enheter på nett

Prosjektet går up på å designe og implementere en tjeneste som skanner et nettverk periodisk i fra internett og/eller internt nett og lagrer resultatene samt metadata om tjenester funnet i en database for videre analyse. Databasen må kunne eksponeres via et REST API for integrasjon med andre tjenester. Omfang av oppgaven kan diskuteres med gruppen, da dette er et prosjekt som kan skaleres opp og ut i fra interesseområder innen skanning, kartlegging og penetrasjonstesting.

Oppdragsgiver

Seksjon for Digital sikkerhet ligger under IT-avdelingen og har det helhetlige ansvaret for digital sikkerhet ved NTNU. Seksjonen arbeider proaktivt, aktivt og reaktiv med digital sikkerhet/informasjonssikerhet på flere nivåer i organisasjonen og består av en faggruppe og en rådgivertjeneste. Faggruppen NTNU SOC (Sikkerhetsoperasjonssenter) har ansvaret for deteksjon, sikkerhetsanalyse og hendelseshåndtering mens rådgivertjenesten (Governance, Risk and Compliance - GRC) arbeider med proaktiv sikkerhetsrådgivning, risikostyring og sikkerhetsarkitektur.

Kontaktperson:

Navn: Christoffer Vargtass Hallstensen Tittel: Gruppeleder SOC
Email: christoffer.hallstensen@ntnu.no Tek: 611 35 145 / 481 35 180

Oppgavens mål

Målet med oppgaven er lage en løsning som periodisk kartlegger et nettverk og lagrer dataene i en database (RDBMS eller NoSQL) for videre bruk i deteksjonsanalyse, eksponeringsanalyse og statistisk analyse på tilgjengelige tjenester over tid. Dataene er tenkt benyttet for både operativ sikkerhet og strategisk sikkerhetsarbeid på NTNU, blant annet for sikkerhetsanalyse og måling av etterlevelse av styringssystem for informasjonssikkerhet.

Systemet bør implementeres med flere forskjellige løst integrerte verktøy for datakolleksjon og analyse og å bygge en datakolleksjon og analyse-pipeline. Det bør utvikles støttescript som gjør enkel enrichment av tjenester funnet som for eksempel automatisk screen-grabbing, IP-lookup, MAC-adresse søk osv.

Oppgavens krav

- Tjenesten skal settes opp og konfigureres med Ansible (Automasjon)
- Tjenesten skal benytte åpen kildekode
- Tjenesten skal støtte mikrotjeneste arkitektur, gjerne med kontainere (REST API)
- Kode/Script skal ha åpen kildekode lisens
- Tjenesten bør ha et enkelt webgrensesnitt for søk i dataene (Proof of Concept)

B. *Scanner code*

B.1 scanner.py

Code listing B.1: scanner.py

```python
import nmap3
import json
import socket
import sys
import os
import requests
import requests.packages
import urllib3
from datetime import datetime
import cve_lookup
import time
import argparse
import logging
import pymongo
import uuid
import ast

DB_URL = 'mongodb://localhost:27018/'
DB_NAME = "mydb"
COLLECTION_NAME = "scans"
TARGETFILE = "target.txt"
LOG_FORMAT = "%(name)s %(asctime)s - %(message)s"
FILENAME = "/var/lib/docker/volumes/files_log-volume/_data/Scanner.log"

def exclude_self():
    #https://stackoverflow.com/questions/166506/finding-local-ip-addresses-using-
        pythons-stdlib
    host_ip = ([l for l in ([ip for ip in socket.gethostbyname_ex(socket.
        gethostname())[2] if not ip.startswith("127.")][:1], [
              [(s.connect(('8.8.8.8', 53)), s.getsockname()[0], s.close()) for s
                  in [socket.socket(socket.AF_INET, socket.SOCK_DGRAM)]][0][1]])
                  if l][0][0])
    f = open("exclude_ip.txt", "w")
    f.write(host_ip + "\n")
    f.close()

def has_target():
    # Checks if target.txt exists and is populated
    logging.debug('[TARGETS] checking')
    try:
        f = open("target.txt", "r")
    except FileNotFoundError as e:
        logging.debug(e)
        logging.debug("\t[TARGETS] File missing!")
        sys.exit(1)
    else:
        if os.path.getsize(TARGETFILE):
         # Ergo file is not empty
            logging.debug("\t[TARGETS] OK")
            return 0
        else:
            # File empty
            logging.debug("\t[TARGETS] File empty!")
            sys.exit(1)

        f.close()

def copy_file():
    # Target.txt ok
    if not has_target():
        os.system('cp target.txt ips_to_scan.txt')

def find_interesting_ip(result):
    logging.debug('\t[INTERESTING IP] started')
    output_list = open("ips_to_scan.txt", "w")
    for ip_addr in result:
        logging.debug("\t\t" + ip_addr)
        for i in range(len(result[ip_addr]['ports'])):
            if result[ip_addr]['ports'][i]['state'] == "open" or result[ip_addr]['
                ports'][i]['state'] == "filtered":
                output_list.write(ip_addr + "\n")
                logging.debug("\t\t\t" + result[ip_addr]['ports'][i]['portid'])
                break

    output_list.close()
    logging.debug("\t[INTERESTING IP] done")

def perform_host_discovery():
    # Stage 1
    logging.debug('[HOST DISCOVERY] started')
    nmap = nmap3.NmapHostDiscovery()
    res = nmap.nmap_no_portscan(
        None, args="-sn --excludefile exclude_ip.txt -iL target.txt")
    logging.debug('Output of host discovery: ')
    res = remove_keys(res)
    logging.debug(res)
    f = open("ips_to_scan.txt", "w")
    for ip in res:
        logging.debug('Found IP: ' + ip)
        if res[ip]['state']['state'] == "up":
            f.write(ip + "\n")
    f.close()
    logging.debug('[HOST DISCOVERY] done')

def perform_portscan():
    # Stage 2
    logging.debug('[FAST PORTSCAN] started')
    nmap = nmap3.NmapHostDiscovery()
    res = nmap.scan_top_ports(None, args="-F -iL ips_to_scan.txt")
    res = remove_keys(res)
    logging.debug(res)
    find_interesting_ip(res)
    logging.debug('[FAST PORTSCAN] done')
    return res

def perform_tcp_scan():
    # Stage 3
    logging.debug('[TCP SCAN] started')
    nmap = nmap3.Nmap()
    result = nmap.nmap_version_detection(
        None, "-sV -p- --script ssl-cert -vv -O -iL ips_to_scan.txt")
    remove_keys(result)
    print(result)
    logging.debug(result)

    logging.debug('[TCP SCAN] done')
    return result

def perform_udp_scan():
    # Stage 4
    logging.debug('[UDP SCAN] started')
    nmap = nmap3.NmapScanTechniques()
    result = nmap.nmap_udp_scan(
        None, "-iL ips_to_scan.txt -p53,67,68,123,137,138,161,445,5000")
    remove_keys(result)
    logging.debug('[UDP SCAN] done')
    return result

def remove_keys(res):
    logging.debug('[KEYS] deleting')
    res.pop('stats', None)
    res.pop('runtime', None)
    logging.debug(res)
    logging.debug('[KEYS] done')
    return res

def take_screengrab(ip):
  # Stage 6
    url = 'http://localhost:3000/takescreengrab/'
    url += ip
    urllib3.disable_warnings()
    requests.packages.urllib3.disable_warnings()
    logging.debug(url)
    try:
        resp = requests.get(url, verify=False, timeout=1).json()
    except requests.exceptions.HTTPError as errorHTTP:
        logging.debug("[SCREENGRAB] Http Error: ", errorHTTP)
    except requests.exceptions.ConnectionError as errorConnection:
        logging.debug("[SCREENGRAB] Error Connecting: ", errorConnection)
    except requests.exceptions.Timeout as errorTimeout:
        logging.debug("[SCREENGRAB] Timeout Error: ", errorTimeout)
    except requests.exceptions.RequestException as errorRequest:
        logging.debug("[SCREENGRAB] ERROR: ", errorRequest)

    return resp

def merge_results(t, u, start):
  #Stage 5
    logging.debug("[MERGE RESULTS] start")
    starttime = start.strftime("%H%M%S")
    startdate = start.strftime("%Y%m%d")
    for i in t:
        os = t[i]['osmatch']
        t_ports = t[i]['ports']
        ports = t_ports
        if i in u:
            u_ports = u[i]['ports']
            ports += u_ports

        for j in t[i]['osmatch']:
            if 'cpe' in j:
                if j['cpe']:
                    oscve = []
                    oscpe = j['cpe']
                    oscve = cve_lookup.find_cve(oscpe)
                    logging.debug(oscve)
                    j['cve'] = oscve

        for port in ports:
            cve = []
            for script in port['scripts']:
                s = script['data']
                s.pop(0, None)
                if "pem" in  script['data']:
                  cert = script['data']['pem']
                  script['data']['pem'] = cert.rstrip("\n")
            if 'cpe' in port:
                if 'cpe' in port['cpe'][0]:
                    cpe = port['cpe'][0]['cpe']
                    cve = cve_lookup.find_cve(cpe)
                    port['cpe'][0]['cve'] = cve

                screengrab = take_screengrab(i)
                if 'Filename' in screengrab:
                    port['screengrab'] = screengrab
        hostname = t[i]['hostname']
        macaddress = t[i]['macaddress']
        state = t[i]['state']
        stats = {'scandate': startdate, 'scantime': starttime}

        uid = str(uuid.uuid4())
        host = {'uuid': uid, 'ip': i, 'hostname': hostname, 'macaddress':
             macaddress,
                'osmatch': os, 'ports': ports, 'state': state, 'scanstats': stats}
        insert_db(host)

    logging.debug("[MERGE RESULTS] done")


def insert_db(res):
  # Stage 7
    myclient = pymongo.MongoClient(DB_URL)
    mydb = myclient[DB_NAME]
    mycol = mydb[COLLECTION_NAME]
    mycol.insert_one(res)

if __name__ == "__main__":
    # Get current time
    now = datetime.now()
    start = now.strftime("%H%M%S")
```

```
parser = argparse.ArgumentParser(
    description="USAGE: python3 scanner.py [options]")
parser.add_argument(
    "-v", "--verbose", help="Enable output to screen, no output by default",
        action="store_true")
parser.add_argument(
    "-t", "--test", help="Enable test mode. Does not perform a scan, reads from
         attached file", action="store_true")
parser.add_argument(
    "-w", "--write", help="Writes result to separate tcp.json and udp.json
        files", action="store_true")
parser.add_argument(
    "-s", "--skip", help="Skips host discovery and fast portscan, starts from
        the full scan stage", action="store_true")
args = parser.parse_args()
level = logging.DEBUG
handlers = [logging.FileHandler(FILENAME)]

# Always output to file. To screen if -v
if args.verbose:
    handlers.append(logging.StreamHandler())

logging.basicConfig(level=level, format=LOG_FORMAT, handlers=handlers)
logging.debug('[SCRIPT] started')
# MAIN:

if not args.test:
    if not args.skip:
        has_target()
        exclude_self()
        perform_host_discovery()
        perform_portscan()
    else:
        copy_file()
        logging.debug('[SKIP] host discovery and fast port scan')
    result_tcp = perform_tcp_scan()
    result_udp = perform_udp_scan()

if args.write:
    # For testing:
    f = open("tcp.json", "w")
    f.write(str(result_tcp))
    f.close()

    f = open("udp.json", "w")
    f.write(str(result_udp))
    f.close()

if args.test:
    logging.debug("*****[TEST MODE ENABLED]*****")
    with open('tcp.json') as f:
        data = f.read()
    f.close()
    result_tcp = ast.literal_eval(data)

    with open('udp.json') as f:
        data = f.read()
    f.close()
    result_udp = ast.literal_eval(data)
merge_results(result_tcp, result_udp, now)
logging.debug('[SCRIPT] done')
```

B.2 cve_lookup.py

Code listing B.2: cve_lookup.py

```python
import requests
import requests.packages
import urllib3
import json

# Default LIMIT = 30
LIMIT = "10"
PARAM = "?limit="
MAX_TIMEOUT = 5

def find_cve(cpe):
    url = 'https://localhost:443/api/cvefor/'
    url = url + cpe + PARAM + LIMIT
    urllib3.disable_warnings()
    requests.packages.urllib3.disable_warnings()
    cve = []
    try:
        resp = requests.get(url, verify=False, timeout=MAX_TIMEOUT)
    except requests.exceptions.HTTPError as errorHTTP:
        print("Http Error:", errorHTTP)
    except requests.exceptions.ConnectionError as errorConnection:
        print("Error Connecting:", errorConnection)
    except requests.exceptions.Timeout as errorTimeout:
        print("Timeout Error:", errorTimeout)
    except requests.exceptions.RequestException as errorRequest:
        print("General Error", errorRequest)
    else:
        r = []
        r.append(resp.json())
        result = {}
        for i in range(len(r)):
            result[i] = r[i]
        for i in result[0]:
            if 'id' in i:
                cve.append(i['id'])

    return cve
```

C. *Docker*

C.1 Screengrabber

Code listing C.1: Screengrabber - app.py

```python
from flask import Flask
from flask import jsonify
import datetime
import imgkit
import os
import random

app = Flask(__name__)

@app.route("/")
def hello():
  m = {}
  m['Message:'] = "Screenshot works!"
  m['Port'] = "3000"
  m['Endpoint'] = "/takescreengrab/<ip>/"
  return jsonify(m)

@app.route("/takescreengrab/<string:ip>/", methods=['GET'])
def takescreengrab(ip):
    t = datetime.datetime.now()
    filename=str(random.randint(100000,999999))
    filename += "-"
    date = t.strftime("%Y%m%d")
    filename += date
    filename += "-"
    time = t.strftime("%H%M%S")
    filename += time
    filename += ".jpg"
    path = "/pictures/" + filename

    imgkit_options= { 'quiet' : ''}
    response = {}
    response['date'] =  date
    response['time'] = time

    try:
        imgkit.from_url(ip, path, options=imgkit_options)
    except ConnectionRefusedError:
        response['Message'] = "Connection refused"
    except IOError:
        response['Message'] = "IOError on"
    else:
        response['Filename'] = filename

    return jsonify(response)


if __name__=="__main__":
  app.run(host='0.0.0.0', port=3000)
```

C.2 API

Code listing C.2: API - app.py( )

```
from flask import Flask
from flask import jsonify
import datetime
import imgkit
import os
import random

app = Flask(__name__)

@app.route("/")
def hello():
  m = {}
  m['Message:'] = "Screenshot works!"
  m['Port'] = "3000"
  m['Endpoint'] = "/takescreengrab/<ip>/"
  return jsonify(m)

@app.route("/takescreengrab/<string:ip>/", methods=['GET'])
def takescreengrab(ip):
    t = datetime.datetime.now()
    filename=str(random.randint(100000,999999))
    filename += "-"
    date = t.strftime("%Y%m%d")
    filename += date
    filename += "-"
    time = t.strftime("%H%M%S")
    filename += time
    filename += ".jpg"
    path = "/pictures/" + filename

    imgkit_options= { 'quiet' : ''}
    response = {}
    response['date'] =  date
    response['time'] = time

    try:
        imgkit.from_url(ip, path, options=imgkit_options)
    except ConnectionRefusedError:
        response['Message'] = "Connection refused"
    except IOError:
        response['Message'] = "IOError on"
    else:
        response['Filename'] = filename

    return jsonify(response)


if __name__=="__main__":
  app.run(host='0.0.0.0', port=3000)
```

C.3 Web interface

Code listing C.3: index.js/server.js

```
const hostsRouter = require('./routes/hosts')
const express = require('express')
const app = express()
const Scan = require('./models/scan')
const mongoose = require('mongoose')
const port = 8080;

const MongoClient = require('mongodb').MongoClient;
const { urlencoded } = require('body-parser')
app.use(express.urlencoded({ extended: false }))

const uri = "mongodb://autoenum-mongodb:27017/"
app.set('view engine', 'ejs')
app.get('/', async (req, res) =>{
    //Link til views/index.ejs
    const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology
         : true });
    try{

      await client.connect();
      const database = client.db('mydb');
      const collection = database.collection('scans');
      var hosts = [];
      hosts = await collection.find().toArray();
      console.log(hosts)
      res.render('index', {hosts: hosts});

    }catch(err){
      console.log("Feil" + err)
    }finally{
      await client.close();
    }

});

app.use('/hosts', hostsRouter)

app.listen(port, () =>{
    console.log('Example app listening at http://localhost:${port}')
});
```

Code listing C.4: hosts.js

```
const express = require('express')
const { urlencoded } = require('body-parser')
const router = express.Router()
const MongoClient = require('mongodb').MongoClient;
const uri = "mongodb://autoenum-mongodb:27017/"
const ObjectID = require('mongodb').ObjectID;

require('../models/scan');

router.get('/list_view', async (req, res)  => {

    const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology
        : true });
    try{

      await client.connect();
      const database = client.db('mydb');
      const collection = database.collection('scans');

      var hosts = [];
      hosts = await collection.find().toArray();
      console.log(hosts)
      res.render('./../views/list', {hosts: hosts})

    }catch(err){
      console.log("Feil" + err)
    }finally{
      await client.close();
    }

})

router.get('/details/:id', async (req, res)  => {
    var img = ''
    var req_url = req.headers.referer
    var cve_url = req_url

    req_url = req_url.replace("8080/","5001/")

    cve_url = cve_url.replace("http://", "https://")
    cve_url = cve_url.replace("8080/", "443/cve/")

    var hostId = req.params.id
    console.log(hostId)
    const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology
        : true });
    try{

      await client.connect();
      const database = client.db('mydb');
      const collection = database.collection('scans');
      var query = {'uuid': hostId}
      console.log(query)

      var host = await collection.findOne(query);


      for(var i = 0; i < host['ports'].length; i++){
        if('screengrab' in host['ports'][i]){
          var view_img = req_url
          view_img += "viewpicture/"
          img = req_url
          img += "picture/"
          img += host['ports'][i]['screengrab']['Filename']
          view_img += host['ports'][i]['screengrab']['Filename']
        }
      }

      res.render('./../views/details', {host: host, image: img, viewimage: view_img
          , cve_url :cve_url})
    }catch(err){
      res.send(err)
    }finally{
      await client.close();
    }


})

router.get('/getjson/:id', async (req, res)  => {


    var hostId = req.params.id
    console.log(hostId)
    const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology
        : true });
    try{

      await client.connect();
      const database = client.db('mydb');
      const collection = database.collection('scans');

      var query = {'uuid': hostId}
      console.log(query)

      var host = []
      host = await collection.findOne(query);
      console.log(host)
      return res.json(host)

    }catch(err){
      console.log("Feil" + err)
      return res(err)
    }finally{
      await client.close();
    }


})


router.post('/search', async function(req, res) {

  const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology:
      true });

    var query = req.body.all_query
    var ip_q = {ip : query};
    var mac_q = {"macaddress.addr" : query}
    var date_q = {"scanstats.scandate" : query};
    var os_cpe_q = {"osmatch.cpe" : query };
    var ports_cpe_q = {"ports.cpe.cpe" : query };
    var uuid_q = {"uuid" : query}
    var q = {"$or": [ip_q, mac_q, date_q, os_cpe_q, ports_cpe_q, uuid_q]}

    try{

      await client.connect();
      const database = client.db('mydb');
      const collection = database.collection('scans');

      var hosts = []
      hosts = await collection.find(q).toArray();
      res.render('./../views/searchresults', {hosts: hosts, query: query});

    }catch(err){
      console.log("Feil" + err)
      return res(err)
    }finally{
      await client.close();
    }

})

router.get('/search', async (req, res)  => {

    res.render('./../views/search')

})

module.exports = router
```

C.4 docker-compose

Code listing C.5: Docker compose

```
version: "3.2"
services:
  autoenum-api:
    container_name: autoenum-api
    hostname: autoenum-api
    restart: always
    build: ./api
    ports:
      - '5001:5001'
    networks:
      - autoenum-network
    volumes:
      - "screengrab-volume:/pictures"
      - "log-volume:/log"
  autoenum-mongodb:
    container_name: autoenum-mongodb
    restart: always
    image: mongo
    ports:
      - '27018:27017'
    volumes:
      - "mongo-docker-volume:/data/db"
    networks:
      - autoenum-network
```

```
autoenum-screengrab:
  container_name: autoenum-screengrab
  restart: always
  build: ./screengrab
  ports:
    - '3000:3000'
  networks:
    - autoenum-network
  volumes:
    - "screengrab-volume:/pictures"
autoenum-webgui:
  container_name: autoenum-webgui
  restart: always
  build: ./website
  depends_on:
    - autoenum-api
  ports:
    - '8080:8080'
  networks:
    - autoenum-network
cve_search:
  image: cve_search
  build:
    context: .
    dockerfile: docker/images/cve_search/dockerfile-cve_search
    args:
      - REPO=cve-search/cve-search
      - BRANCH=master
  hostname: cve_search
  depends_on:
    - redis
    - mongo
  restart: always
  environment:
    - PYTHONUNBUFFERED=TRUE
  ports:
    - 443:5000
  networks:
    - autoenum-network
redis:
  image: cve_search-redis
  hostname: redis
  restart: always
  build:
    context: .
    dockerfile: docker/images/redis/dockerfile-redis
  volumes:
    - "./.cve_search_data/cve_search_redis:/data"
  expose:
    - 6379
  networks:
    - autoenum-network
mongo:
  image: cve_search-mongo
  hostname: mongo
  restart: always
  build:
    context: .
    dockerfile: docker/images/mongodb/dockerfile-mongo
  volumes:
      - "./.cve_search_data/cve_search_mongodb:/data/db"
    expose:
      - 27017
    networks:
      - autoenum-network
volumes:
  mongo-docker-volume:
    external: false
  screengrab-volume:
    external: false
  log-volume:
    external: false
networks:
  autoenum-network:
    driver: bridge
```

D. *Playbooks*

D.1 Setup and deploy Autoenum

Code listing D.1: Main playbook to setup Autoenum

```
- name: Installing autoenum locally
  hosts: localhost
  connection: local
  become: true
  become_method: sudo

  roles:
    - autoenum
```

Code listing D.2: Main playbook to setup Docker

```
---
- include_tasks: setup-RedHat.yml
  when: ansible_os_family == 'RedHat'

- include_tasks: setup-Debian.yml
  when: ansible_os_family == 'Debian'

- name: Install Docker.
  package:
    name: "{{ docker_package }}"
    state: "{{ docker_package_state }}"
  notify: restart docker

- name: Ensure Docker is started and enabled at boot.
  service:
    name: docker
    state: "{{ docker_service_state }}"
    enabled: "{{ docker_service_enabled }}"

- name: Ensure handlers are notified now to avoid firewall conflicts.
  meta: flush_handlers

- include_tasks: docker-compose.yml
  when: docker_install_compose | bool

- include_tasks: docker-users.yml
  when: docker_users | length > 0

- git:
    repo: https://github.com/cve-search/CVE-Search-Docker.git
    dest: /etc/ansible/roles/autoenum/files/cve-search
    clone: yes

- name: Install the package "nmap"
  apt:
    name: nmap
    state: present

- pip:
    requirements: /etc/ansible/roles/autoenum/files/scanner/requirements.txt

- name: running docker compose
  shell: docker-compose -f /etc/ansible/roles/autoenum/files/docker-compose.yml up
       -d
```

D.2 Playbooks for test environment

Code listing D.3: Playbook for Windows machines in test environment

```
---
- hosts: Win1
  gather_facts: no
  tasks:
  - name: Install IIS Web-Server with sub features and management tools
    ansible.windows.win_feature:
      name: Web-Server
      state: present
    register: win_feature
  - name: Reboot if installing Web-Server feature requires it
    ansible.windows.win_reboot:
    when: win_feature.reboot_required
  - name: Install rest
    ansible.windows.win_feature:
      name: Web-WebServer
      name: Web-Mgmt-Tools
      name: Web-Mgmt-Console
      state: present
    register: win_feature
  - name: Reboot if installing Web-Server feature requires it again
    ansible.windows.win_reboot:
    when: win_feature.reboot_required
  - name: Create website
    win_copy:
      src: websites/index1.html
      dest: C:\Inetpub\wwwroot\index.html
      remote_src: no
  - name: Run IIS web server
    win_service:
      name: W3Svc
      state: started
  - name: Copy SSL cert to Windows
    win_copy:
      src: websites/cert.pfx
      dest: C:\Users\ansibleuser\Desktop\cert.pfx
      remote_src: no
  - name: Import cert to IIS
    ansible.windows.win_certificate_store:
      path: C:\Users\ansibleuser\Desktop\cert.pfx
      state: present
      password: "@nsib1epaSsw0rd"
    become: yes
    become_method: runas
    become_user: ansibleuser
  - name: Add a HTTPS binding
    community.windows.win_iis_webbinding:
      name: Default Web Site
      protocol: https
      port: 443
      ip: "*"
      certificate_hash: 8978622c9ddcf0e9f6e1f6b3955295359cab7915
      state: present
  - name: Install RPC
    ansible.windows.win_feature:
      name: RPC-over-HTTP-Proxy
      state: present
  - name: Install SMB 1
    ansible.windows.win_feature:
      name: FS-SMB1
      name: FS-SMB1-SERVER
      state: present
  - name: SNMP-Service
    ansible.windows.win_feature:
      name: SNMP-Service
      state: present
- hosts: Win2
  gather_facts: no
  tasks:
  - name: Install DNS and DHCP
    ansible.windows.win_feature:
      name: DNS
      name: DHCP
      state: present
  - name: Install IIS
    ansible.windows.win_feature:
      name: Web-Server
      state: present
    register: win_feature
  - name: Reboot if installing Web-Server feature requires it
    ansible.windows.win_reboot:
    when: win_feature.reboot_required
  - name: Install rest
    ansible.windows.win_feature:
      name: Web-Mgmt-Tools
      name: Web-Mgmt-Console
      name: Web-Ftp-Server
      name: Web-Ftp-Service
      state: present
    register: win_feature
  - name: Reboot win2
    ansible.windows.win_reboot:
    when: win_feature.reboot_required
  - name: Configure and start FTP
    win_shell: |
      Import-Module WebAdministration
      $ftpsitename = "FTPfolder"
      $path = "C:\Users\ansibleuser\Desktop\FTPfolder"
      mkdir $path
      New-WebFtpSite -Name $ftpsitename -Port 21 -PhysicalPath $path
      $param = @{
             Path    = 'C:\Users\ansibleuser\Desktop\FTPfolder'
             Name    = 'ftpserver.security.authentication.basicauthentication.
                 disabled'
             Value   = $true
             Verbose = $True
      }
  - name: Make sure IIS webserver not running
    win_service:
      name: W3Svc
      state: stopped
  - name: Run FTP
    win_service:
      name: ftpsvc
      state: started
  - name: Download Apache 2.4.46
    win_get_url:
      url: https://www.apachelounge.com/download/VS16/binaries/httpd-2.4.46-win64-
          VS16.zip
      dest: C:\Users\ansibleuser\Desktop\Apache.zip
  - name: Unzipping Apache
    community.windows.win_unzip:
      src: C:\Users\ansibleuser\Desktop\Apache.zip
      dest: C:\
      delete_archive: yes
  - name: Install Apache
    script: InstallApacheOnWindows.bat
  - name: Deleting Apache default index.html
    ansible.windows.win_file:
      path: C:\Apache24\htdocs\index.html
      state: absent
  - name: Create new website
    win_copy:
      src: websites/index2.html
      dest: C:\Apache24\htdocs\index.html
      remote_src: no
  - name: Open firewall
    community.windows.win_firewall_rule:
      name: Allow Apache (by Ansible)
      description: Created by Ansible.
      program: C:\Apache24\bin\httpd.exe
      localport: 80
      action: allow
      direction: in
      protocol: tcp
      state: present
      enabled: yes
  - name: Run Apache service
    win_service:
      name: Apache2.4
      state: started
```

This is only the main playbook for Linux. It uses roles defined in other files which are not included in the report. All files are available in the repository for test environment configuration, see table 5.1.

Code listing D.4: Main playbook Linux based machines in test environment

```
---
- name: Installing DVWA
  hosts: 192.168.1.4
  remote_user: ansible
  become: true
  become_method: sudo

  roles:
    - common
    - web
    - db
    - php

- name: Install Additonal services and samba
  hosts: 192.168.1.6
  remote_user: ansible
  become: true
  become_method: sudo

  roles:
    - additional
    - samba

#- name: Installing autoenum locally
#  hosts: localhost
#  connection: local
#  become: true
#  become_method: sudo

  #roles:
    #- autoenum
```

E. *Code for test environment*

E.1 heat.yaml

Code listing E.1: heat.yaml - HEAT template test environment

```
heat_template_version: 2013-05-23

description: >
  HOT template to deploy the test environment needed for Autoenum.
  Creates 2 new neutron networks plus one router to the public
  network, and deploys 7 servers into the new network.
  Quota needed: 7 instances, 14 VCPUs, 36 GB RAM, 1 FloatIP, 3 security-groups.
parameters:
  key_name:
    type: string
    description: Name of keypair to assign to servers
  image:
    type: string
    description: Name of image to use for servers
  flavor:
    type: string
    description: Flavor to use for servers
  public_net:
    type: string
    description: >
      ID or name of public network for which floating IP addresses will be
          allocated
  private_net_name:
    type: string
    description: Name of private network to be created
  private_net_cidr:
    type: string
    description: Private network address (CIDR notation)
  private_net_gateway:
    type: string
    description: Private network gateway address
  private_net_pool_start:
    type: string
    description: Start of private network IP address allocation pool
  private_net_pool_end:
    type: string
    description: End of private network IP address allocation pool
  private_subnet_name:
    type: string
  private_net_name2:
    type: string
    description: Name of private network to be created
  private_net_cidr2:
    type: string
    description: Private network address (CIDR notation)
  private_subnet_name2:
    type: string
    description: Name of private network to be created
  private_net_gateway2:
    type: string
    description: Private network gateway address
  private_net_pool_start2:
    type: string
    description: Start of private network IP address allocation pool
  private_net_pool_end2:
    type: string
    description: End of private network IP address allocation pool
  REMOTE_IP:
    type: string
    default: 0.0.0.0/0

resources:
  private_net:
    type: OS::Neutron::Net
    properties:
      name: { get_param: private_net_name }

  private_net2:
    type: OS::Neutron::Net
    properties:
      name: { get_param: private_net_name2 }

  private_subnet:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: private_net }
      cidr: { get_param: private_net_cidr }
      gateway_ip: { get_param: private_net_gateway }
      allocation_pools:
        - start: { get_param: private_net_pool_start }
          end: { get_param: private_net_pool_end }

  private_subnet2:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: private_net2 }
      cidr: { get_param: private_net_cidr2 }
      gateway_ip: { get_param: private_net_gateway2 }
      allocation_pools:
        - start: { get_param: private_net_pool_start2 }
          end: { get_param: private_net_pool_end2 }

  router:
    type: OS::Neutron::Router
    properties:
      external_gateway_info:
        network: { get_param: public_net }

  router_interface:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: { get_resource: router }
      subnet_id: { get_resource: private_subnet }

  router_interface2:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: { get_resource: router }
      subnet_id: { get_resource: private_subnet2 }

  server1:
    type: OS::Nova::Server
    properties:
      name: Manager
      image: Ubuntu Server 20.04 LTS (Focal Fossa) amd64
      flavor: m1.large
      key_name: { get_param: key_name }
      user_data_format: RAW
      user_data: { get_file: lib/manager_boot.bash }
      networks:
        - port: { get_resource: server1_public_port }

  server1_public_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: private_net }
      fixed_ips:
        - subnet_id: { get_resource: private_subnet }
      security_groups:
        - default
        - { get_resource: manager_security_group }

  server1_floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: { get_param: public_net }
      port_id: { get_resource: server1_public_port }

  server2:
    type: OS::Nova::Server
    properties:
      name: Ubuntu20
      image: Ubuntu Server 20.04 LTS (Focal Fossa) amd64
      flavor: m1.tiny
      key_name: { get_param: key_name }
      networks:
        - port: { get_resource: server2_port }

  server2_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: private_net }
      fixed_ips:
        - ip_address: 192.168.1.4
      security_groups:
        - default
        - { get_resource: manager_security_group }

  server3:
    type: OS::Nova::Server
    properties:
      name: Win1
      image: Windows Server 2019 Standard [Unlicensed]
      flavor: m1.small
      key_name: { get_param: key_name }
      user_data_format: RAW
      user_data: { get_file: lib/windows_boot.ps1 }
      networks:
        - port: { get_resource: server3_port }

  server3_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: private_net }
      fixed_ips:
        - ip_address: 192.168.1.5
      security_groups:
        - default
        - { get_resource: windows_security_group }

  server4:
    type: OS::Nova::Server
    properties:
      name: Ubuntu18
      image: Ubuntu Server 18.04 LTS (Bionic Beaver) amd64
      flavor: m1.tiny
      key_name: { get_param: key_name }
      networks:
        - port: { get_resource: server4_port }

  server4_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: private_net }
      fixed_ips:
        - ip_address: 192.168.1.6
      security_groups:
        - default
        - { get_resource: manager_security_group }

  server8:
    type: OS::Nova::Server
    properties:
      name: Win2
      image: Windows Server 2016 Standard [Unlicensed]
      flavor: m1.small
      key_name: { get_param: key_name }
      user_data_format: RAW
      user_data: { get_file: lib/windows_boot.ps1 }
      networks:
        - port: { get_resource: server8_port }

  server8_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: private_net }
      fixed_ips:
        - ip_address: 192.168.1.10
      security_groups:
        - default
        - { get_resource: windows_security_group }
```

```
server10:
  type: OS::Nova::Server
  properties:
    name: Kali
    image: Kali Linux 2018.2 xfce amd64
    flavor: m1.small
    key_name: { get_param: key_name }
    networks:
      - port: { get_resource: server10_port }

server10_port:
  type: OS::Neutron::Port
  properties:
    network_id: { get_resource: private_net }
    fixed_ips:
      - ip_address: 192.168.1.12
    security_groups:
      - default
      - { get_resource: manager_security_group }

server20:
  type: OS::Nova::Server
  properties:
    name: ubuntu-net2
    image: Ubuntu Server 20.04 LTS (Focal Fossa) amd64
    flavor: m1.tiny
    key_name: { get_param: key_name }
    networks:
      - port: { get_resource: server20_port }

server20_port:
  type: OS::Neutron::Port
  properties:
    network_id: { get_resource: private_net2 }
    fixed_ips:
      - ip_address: 192.168.2.2
    security_groups:
      - default
      - { get_resource: manager_security_group }

manager_security_group:
  type: OS::Neutron::SecurityGroup
  properties:
    name: manager-security-group
    description: >
      SG for manager and other Linux based machines.
      Allows tcp: 21-25, 80, 8080, 443, 3000, 3389, 5000-5001, 67-68, 53
    rules:
       - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
           protocol: icmp }
       - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
           protocol: tcp, port_range_min: 21, port_range_max: 25}
       - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
           protocol: tcp, port_range_min: 80, port_range_max: 80}
       - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: egress,
           protocol: tcp, port_range_min: 80, port_range_max: 80}
       - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
           protocol: tcp, port_range_min: 8080, port_range_max: 8080}
```

```
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 443, port_range_max: 443}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 3000, port_range_max: 3000}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 3389, port_range_max: 3389}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 5000, port_range_max: 5001}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 1688, port_range_max: 1688}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: egress,
            protocol: udp, port_range_min: 67, port_range_max: 68}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: udp, port_range_min: 67, port_range_max: 68}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 53, port_range_max: 53}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: udp, port_range_min: 53, port_range_max: 53}

  windows_security_group:
    type: OS::Neutron::SecurityGroup
    properties:
      name: windows-security-group
      description: >
        SG for Windows servers
      rules:
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: icmp }
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 21, port_range_max: 22}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 80, port_range_max: 80}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: egress,
            protocol: tcp, port_range_min: 80, port_range_max: 80}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 443, port_range_max: 443}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 3306, port_range_max: 3306}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 3306, port_range_max: 3306}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 3389, port_range_max: 3389}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: udp, port_range_min: 3389, port_range_max: 3389}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: egress,
            protocol: tcp, port_range_min: 3389, port_range_max: 3389}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: egress,
            protocol: udp, port_range_min: 3389, port_range_max: 3389}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: tcp, port_range_min: 5985, port_range_max: 5986}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: udp, port_range_min: 5985, port_range_max: 5986}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: egress,
            protocol: tcp, port_range_min: 5985, port_range_max: 5986}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: egress,
            protocol: udp, port_range_min: 5985, port_range_max: 5986}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: egress,
            protocol: udp, port_range_min: 67, port_range_max: 68}
        - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
            protocol: udp, port_range_min: 67, port_range_max: 68}
```

```
         - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
             protocol: tcp, port_range_min: 1688, port_range_max: 1688}
         - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
             protocol: tcp, port_range_min: 53, port_range_max: 53}
         - { remote_ip_prefix: { get_param: REMOTE_IP }, direction: ingress,
             protocol: udp, port_range_min: 53, port_range_max: 53}

outputs:
  server1_private_ip:
    description: IP address of server1 in private network
    value: { get_attr: [ server1, first_address ] }
  server1_public_ip:
    description: Floating IP address of server1 in public network
    value: { get_attr: [ server1_floating_ip, floating_ip_address ] }
  server2_private_ip:
    description: IP address of server2 in private network
    value: { get_attr: [ server2, first_address ] }
  server3_private_ip:
    description: IP address of server3 in private network
    value: { get_attr: [ server3, first_address ] }
  server4_private_ip:
    description: IP address of server4 in private network
    value: { get_attr: [ server4, first_address ] }
  server8_private_ip:
    description: IP address of server8 in private network
    value: { get_attr: [ server8, first_address ] }
  server10_private_ip:
    description: IP address of server10 in private network
    value: { get_attr: [ server10, first_address ] }
  server20_private_ip:
    description: IP address of server20 in private network
    value: { get_attr: [ server20, first_address ] }
```

E.2 env.yaml

Code listing E.2: env.yaml - Environment variables

```
parameters:
  key_name: newkey
  image: Ubuntu Server 18.04 LTS (Bionic Beaver) amd64
  flavor: m1.small
  public_net: ntnu-internal
  private_net_name: net1
  private_net_cidr: 192.168.1.0/24
  private_net_gateway: 192.168.1.1
  private_net_pool_start: 192.168.1.20
  private_net_pool_end: 192.168.1.100
  private_subnet_name: subnet1
  private_net_name2: net2
  private_net_cidr2: 192.168.2.0/24
  private_subnet_name2: subnet2
  private_net_gateway2: 192.168.2.1
  private_net_pool_start2: 192.168.2.20
  private_net_pool_end2: 192.168.2.100
```

E.3 manager_boot.bash

Code listing E.3: manager_boot.bash

```bash
#!/bin/bash -v
export DEBIAN_FRONTEND=noninteractive

sudo apt update -y

# Prerequisites
sudo apt-get install jq -y
sudo apt install nmap -y
sudo apt install python3-pip -y
sudo apt-get install ansible -y

sudo pip3 install python3-nmap
sudo pip3 install "pywinrm>=0.3.0"
sudo pip3 install pymongo

ansible-galaxy collection install ansible.windows
ansible-galaxy collection install community.windows

#Docker and Docker compose
sudo apt-get update -y
sudo apt-get install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
sudo apt-get update -y
sudo apt-get install -y docker-ce
sudo apt install docker-compose -y

sudo timedatectl set-timezone Europe/Oslo

cat <<EOF > /etc/ansible/key
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAxeNN8Qmou5ar72deuhQcew50FSuW2wwiv7jVzJOXmQV5O8lS
9HxSiyxVtN3VIFjw37bUL9rmY2/QePlRYS3K1mJA4EeYv1mqdW+8lGi2Z8LuroTT
r9lgypfwnrY1JAuLOlHXY/Tmpd574aryw6QmSpsdl4TI5S13M7d4+aVTiTBISd2C
93ZILunkf/duT1pUfUhc2bivfszeU0JAYy1Vy2qeWhRNxiTbeDzXBVPL8wio62Of
O2PiVZ7LIDr14fuI7ZCPTwS1ogz/U+GIej473gE2HHvpDiK3TnCYlVyihAHtfMZ2
k7mTdOKXH72FA//5OFU2yx0P2j5cnagyotnGjwIDAQABAoIBAQCSNNgwX8eYGcGc
104Iw7UrQkmIHrWN0BCYgJMOXHnkaEPjZWLyGizOgQot4LyH8s69K5LobJ5OF536
05JJ75BvBxcR3jRAJJqpu82kBR3H2iGJNcBFq6E07j+ss8jdgd3zT+aJBrenE5OJ
70kAPXbBJowdl9DqasYoosUyBfGLaKDhPjukEkqotspUhWYhOIHHcaHrR5vFO9nL
SaTHDOcGmnoUvp9TYEIyzEBk3ArUin39g+hawH31fGlj+2tQinOasGt0IQrCFh5j
EfkaBz8FrCoK/HcIkcvly8Q8MSNVBOBqwynNGsV+PvUp+voGC5vgj9hrypuD8mAK
RRfNTHSJAoGBAO1OrMC8yB/x1U+ta2WuclRuhFnR0BezE2KIA5mZi5goJHAKP31G
UH/xAyVFirWEFdeGa9DT9GiYKX+M8Oxba/uerf3waPGYovyvfYhcuR2JcMN/F2yo
Cd9pQztxKQ8GpLEUOWostvqW9rQjvXD/X9Mhvl+jUkQr/wKWo3IPg9HVAoGBANV5
u7ok0wkA0TWXj9IuHMOcsoE50eq6WSphFz39tyGkruAADB2eyXE0WDWslLjPDEQ+
OGs+W0fnwH4JjeTQLB/mJRfK/8ei8rkvIRJnJPbXqgBIkNlGUsPL3XDPIzcq/gKb
```

```
osUEMvbihMnp9zpRE7YsRGlY3vKp7biWLEWb+oTTAoGAXwoeR8aTg6+v1YxHsd5u
rX/hg7Ny2rr+bXy5rF+BN7wD89c23C43+TWGI/w49D9lG/8a2PS6MtWV8R56Mr7e
fVRsrIIHFZMi235RETbJcJnlznXs5Lhb09ztbzX/0qO/e6f04p/r3GpvfW++5C1y
rDUccGMRhHn2VIwOA5VRHs0CgYEAtg7/vxywrjj4M1By47lX5qu4wOTi1eDfMnlj
LQc4K4UbbwYbTxegjN8ra3snywUpXPoDe9LOXmCTlenoDYBMYVgRwlzqDwQ1JSHA
fsVgjPQYk+1POz3yT/GJhS/ixKXxw5+gDY4rOMqunNTgd+e1e+P85Cta2HF7v7Sz
RRplaOkCgYEAwjW0V04MwMUDFc+/r265iPJ3EWNQXJ76RYO6+XZ868Khyin8sXAm
kRQAt8YObHEm6NNrlnxvaGC7Zb5TgvKfuJ8P0WiopgjAY1zoLGvtFTqNJL4Jj+jF
ZnYmdO3AJMQfWW65mFje5CFxfsaeHJd6cnQAaw0BT3br5ljDIF3ZL5M=
-----END RSA PRIVATE KEY-----
EOF
chmod 500 /etc/ansible/key

# Configure test environment
git clone https://github.com/Monastyr/autoenum-TestENV.git
mv autoenum-TestENV/* /etc/ansible/
cd /etc/ansible
ansible-playbook main.yml
ansible-playbook windows.yaml

# Setup Autoenum
git clone https://github.com/asm492/autoenum
cp -r autoenum/autoenum /etc/ansible/roles
ansible-playbook /etc/ansible/autoenum/example-playbook.yml
```

E.4 windows_boot.ps1

Code listing E.4: windows_boot.ps1

```
#ps1_sysnative

#Make new user:
$usrname = "ansibleuser"
$password = ConvertTo-SecureString "@nsib1epaSsw0rd" -AsPlainText -Force
New-LocalUser $usrname -Password $password -FullName "Ansible User" -Description "
    Ansible account with admin privileges"
Add-LocalGroupMember -Group "Administrators" -Member $usrname

#Bypass SSL
[Net.ServicePointManager]::SecurityProtocol = [System.Net.SecurityProtocolType]::
    Tls12

#Fixes winRM memory bug in Powershell v3.0
$url = "https://raw.githubusercontent.com/jborean93/ansible-windows/master/scripts/
    Install-WMF3Hotfix.ps1"
$file = "$env:temp\Install-WMF3Hotfix.ps1"
(New-Object -TypeName System.Net.WebClient).DownloadFile($url, $file)
powershell.exe -ExecutionPolicy ByPass -File $file -Verbose

#WinRM setup
$url = "https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/
    ConfigureRemotingForAnsible.ps1"
$file = "$env:temp\ConfigureRemotingForAnsible.ps1"
(New-Object -TypeName System.Net.WebClient).DownloadFile($url, $file)
powershell.exe -ExecutionPolicy ByPass -File $file
```

F. *Sample API response*

Requested API endpoint: /ip/192.168.1.4/

Code listing F.1: Sample API response

```
{
    "35feebc6-cc01-4361-abf1-aedfc34e00c2": {
      "uuid": "35feebc6-cc01-4361-abf1-aedfc34e00c2",
      "ip": "192.168.1.4",
      "hostname": [
        {
          "name": "host-192-168-1-4.openstacklocal",
          "type": "PTR"
        }
      ],
      "macaddress": {
        "addr": "FA:16:3E:3F:BE:B8",
        "addrtype": "mac"
      },
      "osmatch": [
        {
          "name": "Linux 2.6.32",
          "accuracy": "96",
          "line": "55173",
          "osclass": {
            "type": "general purpose",
            "vendor": "Linux",
            "osfamily": "Linux",
            "osgen": "2.6.X",
            "accuracy": "96"
          },
          "cpe": "cpe:/o:linux:linux_kernel:2.6.32",
          "cve": [
            "CVE-2018-5703",
            "CVE-2018-20961",
            "CVE-2017-7895",
            "CVE-2019-15505",
          "CVE-2019-15504",
          "CVE-2016-9555",
          "CVE-2019-14896",
          "CVE-2019-15292",
          "CVE-2016-7117",
          "CVE-2015-8812"
        ]
      },
      {
        "name": "Linux 3.2 - 4.9",
        "accuracy": "96",
        "line": "65105",
        "osclass": {
          "type": "general purpose",
          "vendor": "Linux",
          "osfamily": "Linux",
          "osgen": "4.X",
          "accuracy": "96"
        },
        "cpe": "cpe:/o:linux:linux_kernel:4",
        "cve": [
          "CVE-2016-10150",
          "CVE-2016-7117",
          "CVE-2019-15292",
          "CVE-2016-10229",
          "CVE-2018-12714",
          "CVE-2016-3955",
          "CVE-2016-9555",
          "CVE-2019-14901",
          "CVE-2015-8812",
          "CVE-2019-10125"
        ]
      },
      {
        "name": "Linux 2.6.32 - 3.10",
        "accuracy": "96",
        "line": "56381",
        "osclass": {
          "type": "general purpose",
          "vendor": "Linux",
          "osfamily": "Linux",
          "osgen": "3.X",
          "accuracy": "96"
        },
          "cpe": "cpe:/o:linux:linux_kernel:3",
          "cve": []
        },
        {
          "name": "Linux 3.4 - 3.10",
          "accuracy": "95",
          "line": "65366",
          "osclass": {
            "type": "general purpose",
            "vendor": "Linux",
            "osfamily": "Linux",
            "osgen": "3.X",
            "accuracy": "95"
          },
          "cpe": "cpe:/o:linux:linux_kernel:3",
          "cve": []
        },
        {
          "name": "Synology DiskStation Manager 5.2-5644",
          "accuracy": "95",
          "line": "101159",
          "osclass": {
            "type": "storage-misc",
            "vendor": "Synology",
            "osfamily": "DiskStation Manager",
            "osgen": "5.X",
            "accuracy": "95"
          },
          "cpe": "cpe:/a:synology:diskstation_manager:5.2",
          "cve": [
            "CVE-2018-1160",
            "CVE-2018-13284",
            "CVE-2017-12075",
            "CVE-2017-15889",
            "CVE-2018-8920",
            "CVE-2018-8919",
            "CVE-2018-7184",
            "CVE-2018-7185",
            "CVE-2017-9554",
            "CVE-2017-5753"
          ]
        },
        {
          "name": "Linux 3.1",
          "accuracy": "95",
          "line": "62708",
          "osclass": {
            "type": "general purpose",
            "vendor": "Linux",
            "osfamily": "Linux",
            "osgen": "3.X",
            "accuracy": "95"
          },
          "cpe": "cpe:/o:linux:linux_kernel:3.1",
          "cve": [
            "CVE-2018-5703",
            "CVE-2018-20961",
            "CVE-2017-7895",
            "CVE-2019-15505",
            "CVE-2019-15504",
            "CVE-2016-9555",
            "CVE-2019-14901",
            "CVE-2019-15292",
            "CVE-2016-7117",
            "CVE-2015-8812"
          ]
        },
        {
          "name": "Linux 3.2",
          "accuracy": "95",
          "line": "64455",
          "osclass": {
            "type": "general purpose",
            "vendor": "Linux",
            "osfamily": "Linux",
            "osgen": "3.X",
            "accuracy": "95"
          },
          "cpe": "cpe:/o:linux:linux_kernel:3.2",
          "cve": [
            "CVE-2018-5703",
            "CVE-2018-20961",
            "CVE-2017-7895",
            "CVE-2019-15505",
            "CVE-2019-15504",
            "CVE-2016-9555",
            "CVE-2019-14901",
            "CVE-2019-15292",
          "CVE-2016-7117",
          "CVE-2015-8812"
        ]
      },
      {
        "name": "AXIS 210A or 211 Network Camera (Linux 2.6.17)",
        "accuracy": "94",
        "line": "61606",
        "osclass": {
          "type": "webcam",
          "vendor": "AXIS",
          "osfamily": "embedded",
          "accuracy": "94"
        },
        "cpe": "cpe:/h:axis:211_network_camera",
        "cve": []
      },
      {
        "name": "Linux 2.6.32 - 2.6.35",
        "accuracy": "94",
        "line": "56153",
        "osclass": {
          "type": "general purpose",
          "vendor": "Linux",
          "osfamily": "Linux",
          "osgen": "2.6.X",
          "accuracy": "94"
        },
        "cpe": "cpe:/o:linux:linux_kernel:2.6",
        "cve": [
          "CVE-2009-0065",
          "CVE-2008-4395",
          "CVE-2009-1385",
          "CVE-2009-3613",
          "CVE-2009-2844",
          "CVE-2009-1439",
          "CVE-2009-3726",
          "CVE-2008-4576",
          "CVE-2009-1389",
          "CVE-2010-0008"
        ]
      },
      {
        "name": "Linux 2.6.32 - 3.5",
          "accuracy": "94",
          "line": "56585",
          "osclass": {
            "type": "general purpose",
            "vendor": "Linux",
            "osfamily": "Linux",
            "osgen": "3.X",
            "accuracy": "94"
          },
          "cpe": "cpe:/o:linux:linux_kernel:3",
          "cve": []
        }
      ],
      "ports": [
        {
          "protocol": "tcp",
          "portid": "22",
          "state": "open",
          "reason": "syn-ack",
          "reason_ttl": "64",
          "service": {
            "name": "ssh",
            "product": "OpenSSH",
            "version": "8.2p1 Ubuntu 4",
            "extrainfo": "Ubuntu Linux; protocol 2.0",
            "ostype": "Linux",
            "method": "probed",
            "conf": "10"
          },
          "cpe": [
            {
              "cpe": "cpe:/o:linux:linux_kernel",
              "cve": [
                "CVE-2019-8069",
                "CVE-2019-7096",
                "CVE-2020-9633",
                "CVE-2019-8255",
                "CVE-2019-8070",
                "CVE-2018-4944",
                "CVE-2018-4937",
                "CVE-2018-4935",
                "CVE-2018-5703",
                "CVE-2018-4920"
              ]
            }
          ],
          "scripts": [],
          "screengrab": {
            "Filename": "968575-20210406-104744.jpg",
            "date": "20210406",
            "time": "104744"
          }
        },
        {
          "protocol": "tcp",
          "portid": "80",
          "state": "open",
          "reason": "syn-ack",
          "reason_ttl": "64",
          "service": {
            "name": "http",
            "product": "Apache httpd",
            "version": "2.4.46",
            "extrainfo": "(Ubuntu)",
            "method": "probed",
            "conf": "10"
          },
          "cpe": [
            {
              "cpe": "cpe:/a:apache:http_server:2.4.46",
              "cve": []
            }
          ],
          "scripts": [
            {
              "name": "http-server-header",
              "raw": "Apache/2.4.46 (Ubuntu)",
              "data": {}
            }
          ],
          "screengrab": {
            "Filename": "194452-20210406-104745.jpg",
            "date": "20210406",
            "time": "104745"
          }
        },
        {
          "protocol": "udp",
          "portid": "53",
          "state": "closed",
          "reason": "port-unreach",
          "reason_ttl": "64",
          "service": {
            "name": "domain",
            "method": "table",
            "conf": "3"
          },
          "scripts": []
        },
        {
          "protocol": "udp",
          "portid": "67",
          "state": "closed",
          "reason": "port-unreach",
          "reason_ttl": "64",
          "service": {
            "name": "dhcps",
            "method": "table",
            "conf": "3"
          },
          "scripts": []
        },
        {
          "protocol": "udp",
          "portid": "68",
          "state": "open|filtered",
          "reason": "no-response",
          "reason_ttl": "0",
          "service": {
            "name": "dhcpc",
            "method": "table",
            "conf": "3"
          },
          "scripts": []
        },
        {
          "protocol": "udp",
          "portid": "123",
          "state": "closed",
          "reason": "port-unreach",
          "reason_ttl": "64",
          "service": {
            "name": "ntp",
            "method": "table",
            "conf": "3"
          },
          "scripts": []
        },
        {
          "protocol": "udp",
          "portid": "137",
          "state": "closed",
          "reason": "port-unreach",
          "reason_ttl": "64",
          "service": {
            "name": "netbios-ns",
            "method": "table",
            "conf": "3"
          },
          "scripts": []
        },
        {
          "protocol": "udp",
          "portid": "138",
          "state": "closed",
          "reason": "port-unreach",
          "reason_ttl": "64",
          "service": {
            "name": "netbios-dgm",
            "method": "table",
            "conf": "3"
          },
          "scripts": []
        },
        {
          "protocol": "udp",
          "portid": "161",
          "state": "closed",
          "reason": "port-unreach",
          "reason_ttl": "64",
          "service": {
            "name": "snmp",
            "method": "table",
            "conf": "3"
          },
          "scripts": []
```

```
      },
      {
        "protocol": "udp",
        "portid": "445",
        "state": "closed",
        "reason": "port-unreach",
        "reason_ttl": "64",
        "service": {
          "name": "microsoft-ds",
          "method": "table",
          "conf": "3"
        },
        "scripts": []
      },
      {
        "protocol": "udp",
        "portid": "5000",
        "state": "closed",
        "reason": "port-unreach",
        "reason_ttl": "64",
        "service": {
          "name": "upnp",
          "method": "table",
          "conf": "3"
        },
        "scripts": []
      }
    ],
    "state": {
      "state": "up",
      "reason": "arp-response",
      "reason_ttl": "0"
    },
    "scanstats": {
      "scandate": "20210406",
      "scantime": "124408"
    }
  }
}
```

G. *Database*

G.1 Schema

'{ }' indicates an object/subdocument. An object can consist of other objects, numbers, strings and arrays. '[ ]' indicates an array. An array can consist of objects, numbers and strings.

Code listing G.1: Sample database schema

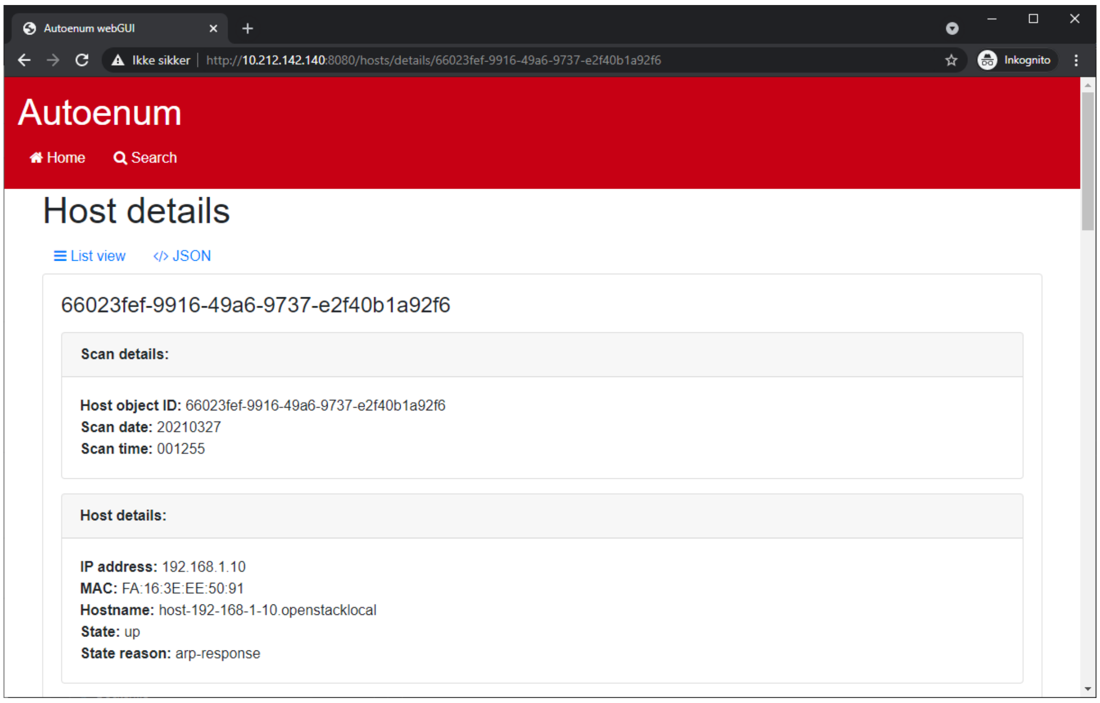
```
{
        "_id" : "ObjectId",
        "uuid" : "String/UUID v4",
        "ip" : "String",
        "hostname" : [
                {
                        "name": "String",
                        "type": "String"
                }
        ],
        "macaddress" : {
                "addr" : "String",
                "addrtype" : "String",
                "vendor": "String"
        },
        "osmatch" : [
                {
                        "name" : "String",
                        "accuracy" : "String",
                        "line" : "String",
                        "osclass" : {
                                "type" : "String",
                                "vendor" : "String",
                                "osfamily" : "String",
                                "osgen" : "String",
                                "accuracy" : "String"
                        },
                        "cpe" : "String",
                        "cve" : []
                }
        ],
        "ports" : [
                {
                        "protocol" : "String",
                        "portid" : "String",
                        "state" : "String",
                        "reason" : "String",
                        "reason_ttl" : "String",
                        "service" : {
                                "name" : "String",
                                "product" : "String",
                                "version" : "String",
                                "extrainfo" : "String",
                                "ostype" : "String",
                                "method" : "String",
                                "conf" : "String"
                        },
                        "cpe" : [
                                {
                                        "cpe" : "String",
                                        "cve" : []
                                }
                        ],
                        "scripts" : [ ]
                }
        ],
        "state" : {
                "state" : "String",
                "reason" : "String",
                "reason_ttl" : "String"
        },
        "scanstats" : {
                "scandate" : "String",
                "scantime" : "String"
        }
}
```

G.2 Document

Code listing G.2: Sample database document

```
{
    "_id": ObjectId("606c3c578e1b559ea9930e2e"),
    "uuid": "145b6ae9-835c-4b6d-9aad-d3bf8066bd6b",
    "ip": "192.168.1.6",
    "hostname": [
        {
            "name": "host-192-168-1-6.openstacklocal",
            "type": "PTR"
        }
    ],
    "macaddress": {
        "addr": "FA:16:3E:32:7B:C7",
        "addrtype": "mac"
    },
    "osmatch": [
        {
            "name": "Linux 2.6.32",
            "accuracy": "96",
            "line": "55173",
            "osclass": {
                "type": "general purpose",
                "vendor": "Linux",
                "osfamily": "Linux",
                "osgen": "2.6.X",
                "accuracy": "96"
            },
            "cpe": "cpe:/o:linux:linux_kernel:2.6.32",
            "cve": [
                "CVE-2018-5703",
                "CVE-2018-20961",
                "CVE-2017-7895",
                "CVE-2019-15505",
                "CVE-2019-15504",
                "CVE-2016-9555",
                "CVE-2019-14896",
                "CVE-2019-15292",
                "CVE-2016-7117",
                "CVE-2015-8812"
            ]
        }
    ],
    "ports": [
        {
            "protocol": "tcp",
            "portid": "21",
            "state": "open",
            "reason": "syn-ack",
            "reason_ttl": "64",
            "service": {
                "name": "ftp",
                "product": "vsftpd",
                "version": "2.3.4",
                "ostype": "Unix",
                "method": "probed",
                "conf": "10"
            },
            "cpe": [
                {
                    "cpe": "cpe:/a:vsftpd:vsftpd:2.3.4",
                    "cve": []
                }
            ],
            "scripts": []
        },
        {
            "protocol": "tcp",
            "portid": "22",
            "state": "open",
            "reason": "syn-ack",
            "reason_ttl": "64",
            "service": {
                "name": "ssh",
                "product": "OpenSSH",
                "version": "7.6p1 Ubuntu 4ubuntu0.3",
                "extrainfo": "Ubuntu Linux; protocol 2.0",
                "ostype": "Linux",
                "method": "probed",
                "conf": "10"
            },
            "cpe": [
                {
                    "cpe": "cpe:/o:linux:linux_kernel",
                    "cve": [
                        "CVE-2019-8069",
                        "CVE-2019-7096",
                        "CVE-2020-9633",
                        "CVE-2019-8255",
                        "CVE-2019-8070",
                        "CVE-2018-4944",
                        "CVE-2018-4937",
                        "CVE-2018-4935",
                        "CVE-2018-5703",
                        "CVE-2018-4920"
                    ]
                }
            ],
            "scripts": []
        }
    ],
    "state": {
        "state": "up",
        "reason": "arp-response",
        "reason_ttl": "0"
    },
    "scanstats": {
        "scandate": "20210406",
        "scantime": "124408"
    }
}
```

H. *Web interface screenshots*


Autoenum webGUI
Ikke sikker
http://10.212.142.140:8080/hosts/details/66023fef-9916-49a6-9737-e2f40b1a92f6
Inkognito
Autoenum
Home
Search
List view
JSON

Host details

66023fef-9916-49a6-9737-e2f40b1a92f6

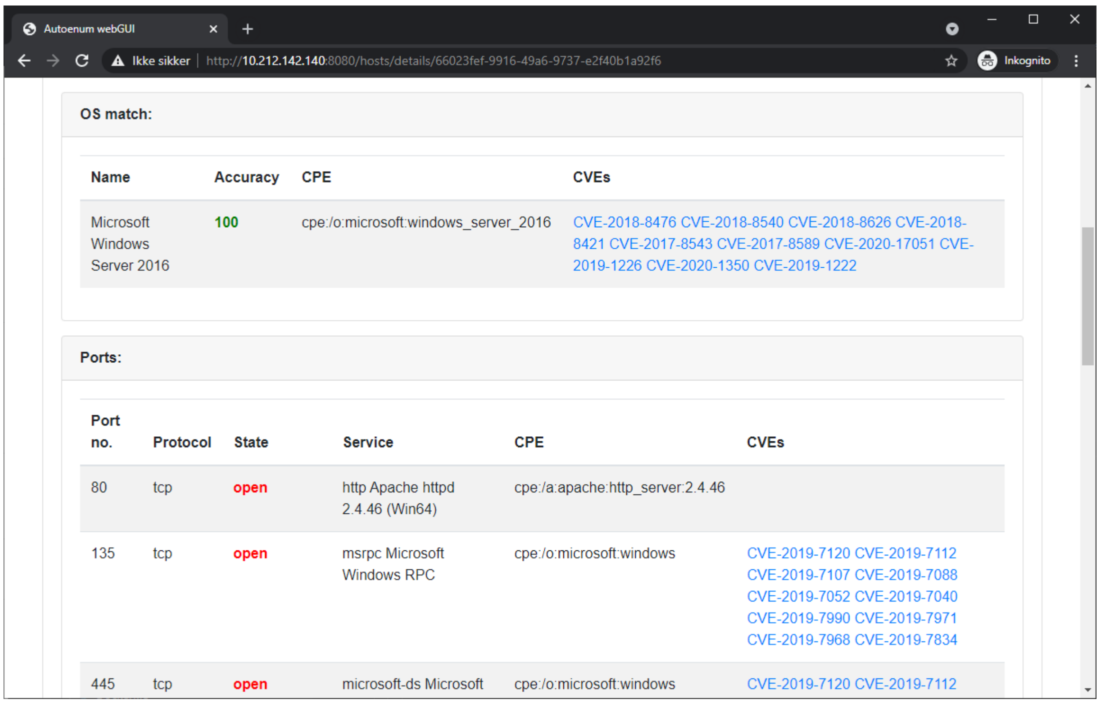
Scan details:

Host object ID: 66023fef-9916-49a6-9737-e2f40b1a92f6
Scan date: 20210327
Scan time: 001255

Host details:

IP address: 192.168.1.10
MAC: FA:16:3E:EE:50:91
Hostname: host-192-168-1-10.openstacklocal
State: up
State reason: arp-response

Figure H.1: Web interface: Detailed view 1/3


Autoenum webGUI
Ikke sikker | http://10.212.142.140:8080/hosts/details/66023fef-9916-49a6-9737-e2f40b1a92f6
Inkognito

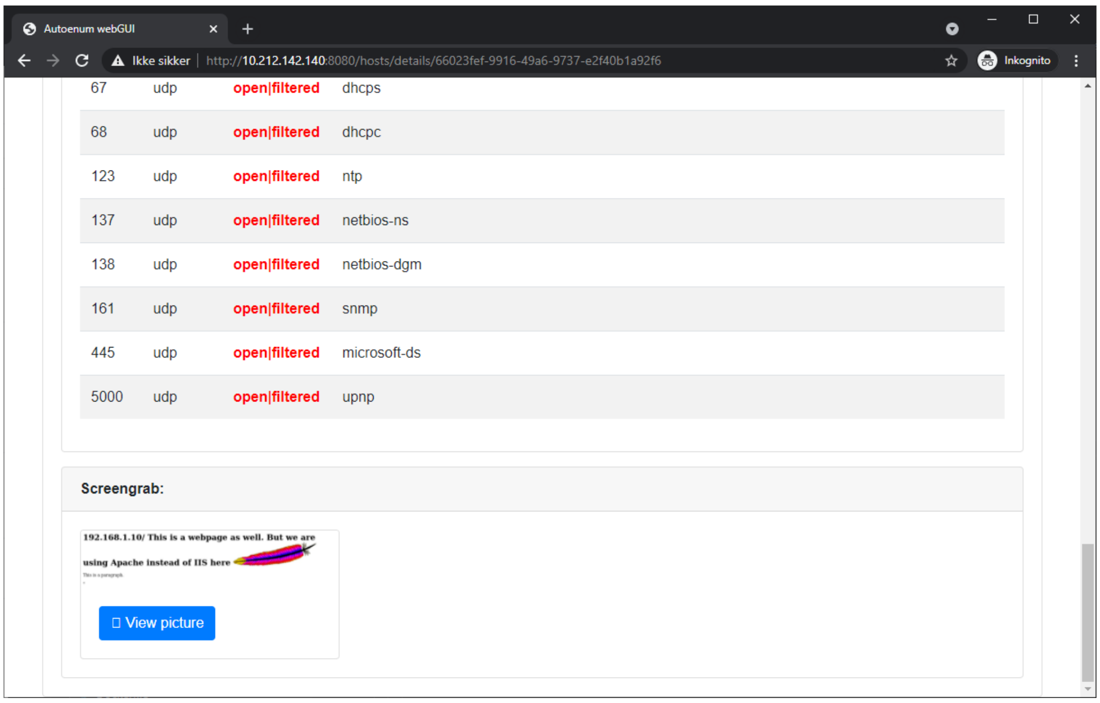
OS match:

Name	Accuracy	CPE	CVEs
Microsoft Windows Server 2016	100	cpe:/o:microsoft:windows_server_2016	CVE-2018-8476 CVE-2018-8540 CVE-2018-8626 CVE-2018-8421 CVE-2017-8543 CVE-2017-8589 CVE-2020-17051 CVE-2019-1226 CVE-2020-1350 CVE-2019-1222

Ports:

Port no.	Protocol	State	Service	CPE	CVEs
80	tcp	open	http Apache httpd 2.4.46 (Win64)	cpe:/a:apache:http_server:2.4.46	
135	tcp	open	msrpc Microsoft Windows RPC	cpe:/o:microsoft:windows	CVE-2019-7120 CVE-2019-7112 CVE-2019-7107 CVE-2019-7088 CVE-2019-7052 CVE-2019-7040 CVE-2019-7990 CVE-2019-7971 CVE-2019-7968 CVE-2019-7834
445	tcp	open	microsoft-ds Microsoft	cpe:/o:microsoft:windows	CVE-2019-7120 CVE-2019-7112

Figure H.2: Web interface: Detailed view 2/3


Autoenum webGUI
Ikke sikker | http://10.212.142.140:8080/hosts/details/66023fef-9916-49a6-9737-e2f40b1a92f6
Inkognito

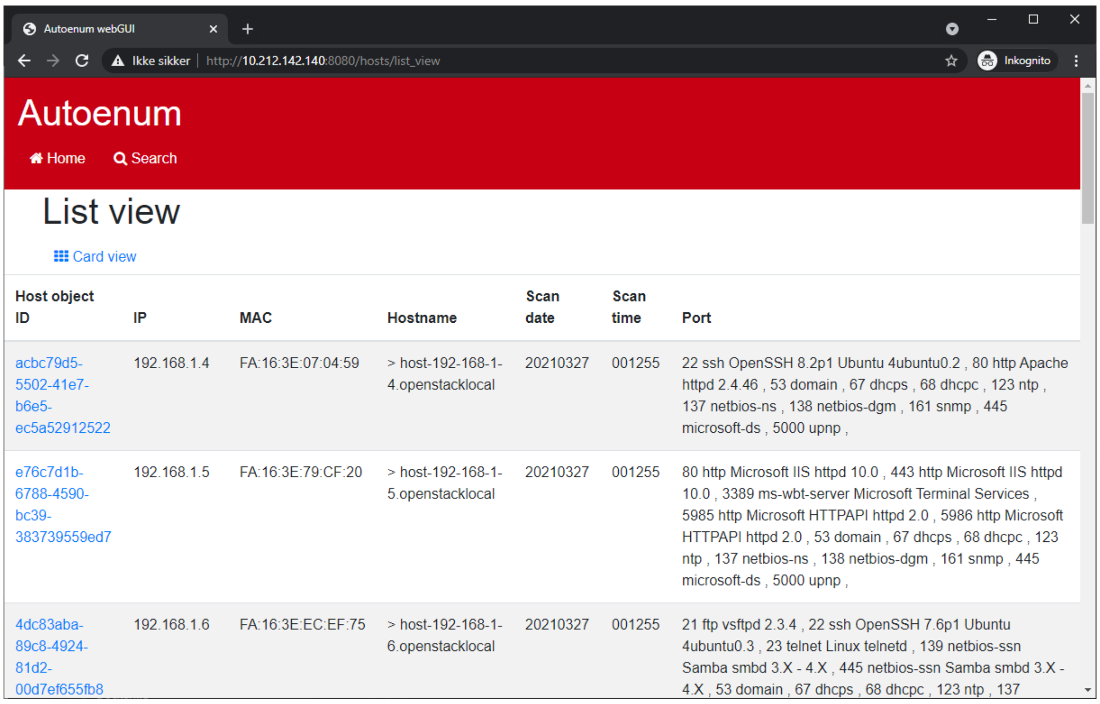
67	udp	open|filtered	dhcps
68	udp	open|filtered	dhcpc
123	udp	open|filtered	ntp
137	udp	open|filtered	netbios-ns
138	udp	open|filtered	netbios-dgm
161	udp	open|filtered	snmp
445	udp	open|filtered	microsoft-ds
5000	udp	open|filtered	upnp

Screengrab:

192.168.1.10/ This is a webpage as well. But we are using Apache instead of IIS here

This is a paragraph

Figure H.3: Web interface: Detailed view 3/3



Autoenum

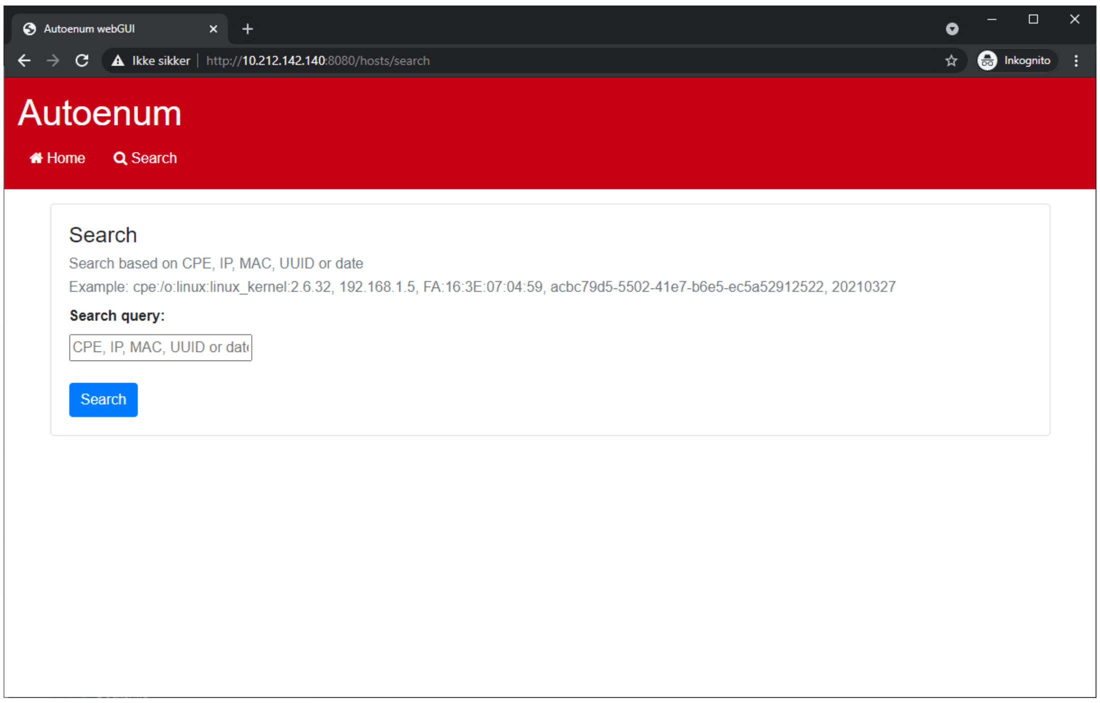
Home Search

List view

Card view

Host object ID	IP	MAC	Hostname	Scan date	Scan time	Port
acbc79d5-5502-41e7-b6e5-ec5a52912522	192.168.1.4	FA:16:3E:07:04:59	> host-192-168-1-4.openstacklocal	20210327	001255	22 ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 , 80 http Apache httpd 2.4.46 , 53 domain , 67 dhcps , 68 dhcpc , 123 ntp , 137 netbios-ns , 138 netbios-dgm , 161 snmp , 445 microsoft-ds , 5000 upnp ,
e76c7d1b-6788-4590-bc39-383739559ed7	192.168.1.5	FA:16:3E:79:CF:20	> host-192-168-1-5.openstacklocal	20210327	001255	80 http Microsoft IIS httpd 10.0 , 443 http Microsoft IIS httpd 10.0 , 3389 ms-wbt-server Microsoft Terminal Services , 5985 http Microsoft HTTPAPI httpd 2.0 , 5986 http Microsoft HTTPAPI httpd 2.0 , 53 domain , 67 dhcps , 68 dhcpc , 123 ntp , 137 netbios-ns , 138 netbios-dgm , 161 snmp , 445 microsoft-ds , 5000 upnp ,
4dc83aba-89c8-4924-81d2-00d7ef655fb8	192.168.1.6	FA:16:3E:EC:EF:75	> host-192-168-1-6.openstacklocal	20210327	001255	21 ftp vsftpd 2.3.4 , 22 ssh OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 , 23 telnet Linux telnetd , 139 netbios-ssn Samba smbd 3.X - 4.X , 445 netbios-ssn Samba smbd 3.X - 4.X , 53 domain , 67 dhcps , 68 dhcpc , 123 ntp , 137

Figure H.4: Web interface: List view


Autoenum webGUI
Ikke sikker | http://10.212.142.140:8080/hosts/search
Inkognito

Autoenum

Home Search

Search

Search based on CPE, IP, MAC, UUID or date

Example: cpe:/o:linux:linux_kernel:2.6.32, 192.168.1.5, FA:16:3E:07:04:59, acbc79d5-5502-41e7-b6e5-ec5a52912522, 20210327

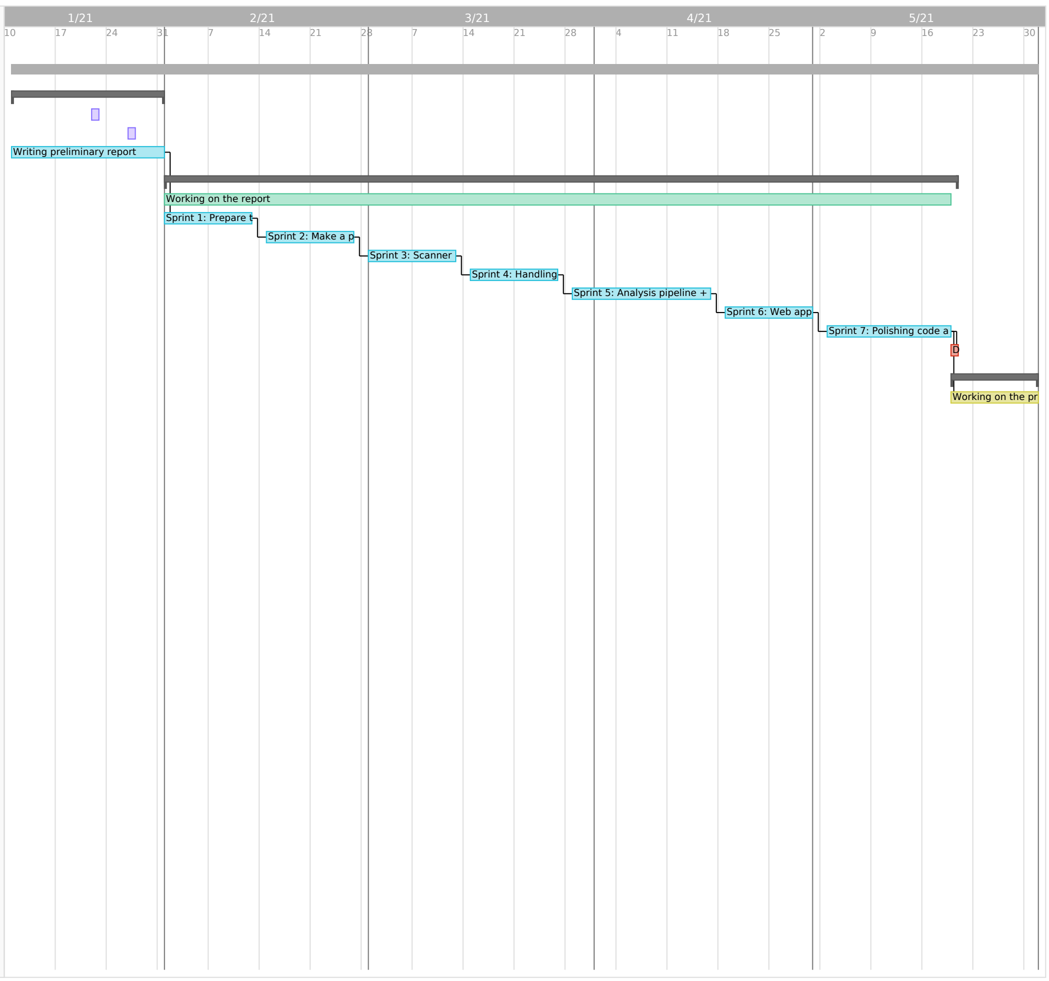
Search query:

CPE, IP, MAC, UUID or date
Search

Figure H.5: Web interface: Search

I. *Gantt diagram*

Bachelor	start	end
Preliminary project	**11/01/21**	**31/01/21**
Meeting with employer	22/01	22/01
Meeting with supervisor	27/01	27/01
Writing preliminary report	11/01	31/01
Main project	**01/02/21**	**20/05/21**
Working on the report	01/02	19/05
Sprint 1: Prepare test environment	01/02	12/02
Sprint 2: Make a prototype of scanner	15/02	26/02
Sprint 3: Scanner improvements	01/03	12/03
Sprint 4: Handling output, DB design	15/03	26/03
Sprint 5: Analysis pipeline + containe...	29/03	16/04
Sprint 6: Web app/GUI and final test	19/04	30/04
Sprint 7: Polishing code and report	03/05	19/05
DEADLINE	20/05	20/05
Presentation	**20/05/21**	**31/05/21**
Working on the presentation	20/05	31/05



J. *Repository structure*

Omitted output of **/autoenum/autoenum/files/website/node_modules**

Code listing J.1: Directory

```
.
`-- autoenum
    |-- CHANGES.md
    |-- LICENSE
    |-- README.md
    |-- autoenum
    |   |-- defaults
    |   |   `-- main.yml
    |   |-- files
    |   |   |-- api
    |   |   |   |-- Dockerfile
    |   |   |   |-- app.py
    |   |   |   |-- display.html
    |   |   |   `-- requirements.txt
    |   |   |-- docker-compose.yml
    |   |   |-- scanner
    |   |   |   `-- requirements.txt
    |   |   |-- screengrab
    |   |   |   |-- Dockerfile
    |   |   |   |-- app.py
    |   |   |   `-- requirements.txt
    |   |   `-- website
    |   |       |-- Dockerfile
    |   |       |-- models
    |   |       |   `-- scan.js
    |   |       |-- node_modules [...]
    |   |       |-- package-lock.json
    |   |       |-- package.json
    |   |       |-- routes
    |   |       |   `-- hosts.js
    |   |       |-- server.js
    |   |       `-- views
    |   |           |-- banner.png
    |   |           |-- banner_red.jpg
    |   |           |-- banner_white.jpg
    |   |           |-- details.ejs
    |   |           |-- head_white.png
    |   |           |-- header.ejs
    |   |           |-- index.ejs
    |   |           |-- list.ejs
    |   |           |-- search.ejs
    |   |           `-- searchresults.ejs
    |   |-- handlers
    |   |   `-- main.yml
    |   `-- tasks
    |       |-- docker-compose.yml
    |       |-- docker-users.yml
    |       |-- main.yml
    |       |-- setup-Debian.yml
    |       `-- setup-RedHat.yml
    |-- checklist.md
    |-- example-playbook.yml
    `-- scanner
        |-- cve_lookup.py
        |-- requirements.txt
        |-- scanner.py
        `-- target.txt
```

K. *Sprint reviews*

Sprint Review 1

Dato
01.02.2021 - 12.02.2021

Hva vi skulle gjøre
- Rapport: Begynne på rapporten
- Kode: Sette opp testmiljø, første versjon

Hva som er blitt gjort
- Rapport: Sette opp dokument i LaTeX
- Rapport: Begynne på teori
- Rapport: Begynne på introduksjon
- Rapport: Begynne på testmiljø diagram
- Kode: Sette opp testmiljø, første versjon

Hva som IKKE har blitt gjort
- …

Diskusjon rundt sprint
Gruppen føler at vi har jobbet bra under Sprint 1. Alle møter opp til de daglige arbeidsøktene/møtene våre og vi føler at vi har fått en god start på bacheloren.

Sprint Review 2

Dato
15.02.2021 - 26.02.2021

Hva vi skulle gjøre
- Kode: Lage prototype av scanner
- Rapport: Fortsette på rapporten

Hva som er blitt gjort
- Kode: Lage prototype av skanner
- Kode: Lagde en SQL database
- Rapport: Fortsatte på punktene fra Sprint 1
- Rapport: Begynte på kapittel implementation
- Rapport: Begynte på kapittel Technology

Hva som IKKE har blitt gjort

-

Diskusjon rundt sprint

Vi føler vi har fått en god start på bacheloroppgaven etter de to første sprintene, da vi allerede ligger litt foran skjema. Samtidig har vi forsikret oss om at hastigheten vi jobber i ikke har påvirket kvaliteten på produktet så langt.

Sprint Review 3

Dato

01.03.2021 – 12.03.2021

Hva vi skulle gjøre

- Kode: Legge til mer funksjonalitet på scanneren

Hva som er blitt gjort

- Kode: Lagt til mer funksjonalitet på scanneren (SSL-grab)
- Kode: Gikk fra MySQL database til mongoDB database
- Kode: Laget Docker compose
- Kode: Delvis implementert CVE søk
- Kode: Lagde dockerisert API
- Kode: Flyttet database til docker
- Kode: Begynt arbeidet på web grensesnitt
- Rapport: La til methodology kapittel
- Rapport: Flyttet innholdet fra technology kapittel til methodology.

Hva som IKKE har blitt gjort

-

Diskusjon rundt sprint

Vi har klart å gjøre alt vi skulle ha gjort hittil. Etter sprint 3 ligger vi nesten en hel sprint foran det vi egentlig skulle. Ettersom vi ligger foran skjema er arbeidsmoralen og motivasjonen på topp!

Sprint Review 4

Dato

15.03.2021 – 26.03.2021

Hva vi skulle gjøre

- Kode: Håndtere output fra nmap
- Kode: Designe databasen

Hva som er blitt gjort

- Kode: Lagt til flere endpoints på API
- Kode: Endret litt på API response (fortsatt JSON)
- Kode: Web grensesnitt funker slik det skal
- Rapport: Begynte på kapittelet conclusion

Hva som IKKE har blitt gjort

-

Diskusjon rundt sprint

På dagen da sprinten sluttet, hadde vi møte med oppdragsgiveren vår Christoffer. Vi presenterte det tekniske vi hadde gjort i løpet av sprintene 1-4 og han ville at vi skulle ha en beta-test av Autoenum på NTNU-nettverket i sprint 5 en gang. Sprinten gikk fint på koden og det tekniske, men vi begynner å slite litt med strukturen av rapporten vår. Det er mye informasjon som vi føler enten ikke hører hjemme i noen kapittler eller hører hjemme i for mange.

Sprint Review 5

Dato

29.03.2021 – 16.04.2021

Hva vi skulle gjøre

- Kode: Analyse pipeline
- Kode: Containerization

Hva som er blitt gjort

- Kode: Testest og klargjort kode for oppdragsgivers beta test
- Kode: Laget Ansible Playbook og sjekket at denne har fungert
- Kode: Feilsøke problemer som dukket opp under deployment.
- Kode: Lagt til CVE, linker til CVE DB og CPE på web grensesnitt

- Kode: Endret søkefunksjonen på web grensesnitt
- Kode: Lagt til UUID etter konkret ønske fra oppdragsgiver
- Kode: Lagt til UUID som endpoint på API
- Kode: Laget eget repo for beta test
- Rapport: Utarbeidet dokumentasjon for oppdragsgiver
- Rapport: Endret strukturen på rapporten
- Rapport: Er i mål med kapittel 1 og 2 i rapporten
- Rapport: Tokk inspirasjon fra SDLC da vi endret strukturen til rapporten
- Rapport: Skrevet mye på implementation og methodology
-

Hva som IKKE har blitt gjort

-

Diskusjon rundt sprint

Denne sprinten har vært en uke lengre enn de andre sprintene, grunnet påsken. Vi jobbet med oppgaven i virkedagene i påsken. Vi misforsto oppgaven før vi satte i gang, og trodde at **analyse pipelinen** det referreres til i oppgaveteksten skulle skje etter selve scannen var utført. Dette punktet falt bort, da oppdragsgiver avklarte for oss at scannen vår oppfylte kravene i analyse pipelinen. Vi var også ferdige med å containerize alt som skulle være det.

Da vi nesten var ferdig med koden kontaktet vi oppdragsgiver for å vise han en live demo. Oppdragsgiver ønsket å teste systemet i eget miljø, og vi ble bedt om å utarbeide dokumentasjon og en Playbook. Da vi allerede var ferdige med alt vi skulle gjøre i løpet av sprinten før sprinten begynte, valgte vi å takke ja til beta testen.

Sprint Review 6

Dato

19.04.2021 – 30.04.2021

Hva vi skulle gjøre

- Kode: Lage web interface
- Kode: Test av Autoenum

Hva som er blitt gjort

- Rapport: Diskusjon
- Rapport: Omstrukturering
- Rapport: Lagt til flere figurer
- Rapport: Fjernet overflødig kode i rapporten

Hva som IKKE har blitt gjort

-

Diskusjon rundt sprint

- Vi var ferdige med web interface før vi begynte på sprinten, og har for det meste jobbet med rapporten og tilbakemeldigene fra veileder. I skrivende stund (30.04) venter vi fremdeles på resultater fra beta test

Sprint Review 7

Dato

03.05.2021 – 19.05.2021

Hva vi skulle gjøre

- Finpusse kode og rapport

Hva som er blitt gjort

- Kode: Lagt til cache på API
- Kode: Endringer i web interface for feilhåndtering
- Rapport: Gramatikk, skrivefeil
- Rapport: Jobbet med kommentarer fra Erjon

Hva som IKKE har blitt gjort

-

Diskusjon rundt sprint

- Vi har brukt siste sprint på å gjøre ferdig rapporten og koden.

L. *Preliminary report*

Preliminary report bachelor thesis DCSG2900

Avleen Singh Marjara, 505092
Jarl Tengesdal Lygre, 505100
Sander Høgli, 484191
Wojciech Malecki, 505075

January 2021

Contents

1 Goals **2**
1.1 Background 2
1.2 Limitations 2
1.3 Project goals 2

2 Scope **3**
2.1 Subject area 3
2.2 Task description 3
2.3 Clarification of scope 3

3 Participants and roles **3**
3.1 Group roles 3
3.2 Taskgiver and supervisor 4

4 Resources **4**
4.1 Needed resources 4

5 Risk assessment **5**
5.1 Risks 6
5.2 Risk matrix 6
5.3 Counter measures 7
5.4 Risk matrix with measures 7

6 Progress plan **8**
6.1 Scrum 8
6.2 Sprints 8

A Project agreement **10**

B Group rules **15**

1 Goals

1.1 Background

The group is tasked with creating and implementing a solution which periodically scans a network and saves the results and metadata about running services. The data is stored in a database. The database will have to be exposed through a REST API for integration with their existing services. The data is to be used in detection and exposure analysis. Furthermore the system should be integrated with several tools for data collection and a pipeline for analytics.

1.2 Limitations

The taskgiver have given us some specific constraints which we have noted below. We also mention some of our own limitations in terms of experience, knowledge and deadlines.

- Configuration is to be done through Ansible
- The code we produce has to be open source
- The service has to use open source code
- The service has to support microservice architecture
- As a proof of concept, the service should include a basic webGUI which enables the user to search for the stored data
- Output of script should be in JSON
- The given time to finish the project
- Our ability to acquire new knowledge in the give time periode

1.3 Project goals

Our goal is to produce a product in line with the specification and constraints given by the client of our bachelor thesis.

The desired effects of the project is to:

- Make a usable system that has the functionality the taskgiver needs.
- The code base should be easily maintainable and modifiable by the taskgiver.
- The data collected by the system should be able to be used by the taskgiver to improve operational security. The data should also be useful for strategic security work.
- Improving security of the NTNU network

2 Scope

2.1 Subject area

This project touches on several subject areas. Our group is familiar with some of areas from previous courses. These include automation, microservices, programming, databases and systems engineering. Although we are familiar with the areas, this project forces us combine the knowledge obtained in the previous courses. Even though we are familiar with some of the areas, we still have to learn new technologies and languages such as Python and Ansible, so that the final product is in line with the taskgivers requirements.

2.2 Task description

The goal of the thesis is to create a solution that periodically maps the network and stores the data in a database (RDBMS or NoSQL) for further use in detection analysis, exposure analysis and statistical analysis of available services over time. The data is intended to be used for both operational security and strategic security work at NTNU, including for security analysis and measuring compliance with management systems for information security. The system should be implemented with several different integrated tools for data collection and analysis and to build a data collection and analysis pipeline. There should be development support script that makes easy enrichment of services found such as automatic screen-grabbing, IP lookup, MAC address search etc.

2.3 Clarification of scope

After a session with the taskgiver we have extended and clearified the scope. In addition to what the taskgiver required we are going to add a vulnerability scan. The scan will be divided into three levels:

- Level 1: Host discovery
- Level 2: Port scan, screen-/bannergrabs
- Level 3: Vulnerability scan on a subset of the scanned machines.

3 Participants and roles

3.1 Group roles

We have agreed on giving each group member their distinct role. However, in the event where a member is absent, his role will be fulfilled by his deputy.

Name	Role(s)
Avleen	Group leader
Sander	Secretary
Jarl	Deputy leader and secretary
Wojtek	Scrum master

Table 1: Roles

3.2 Taskgiver and supervisor

The taskgiver is NTNU SOC. Our contact person is Christoffer Vargtass Hallstensen. We aren't planning to have meetings regularly, but we will have meetings as required. Erjon Zoto is our supervisor. We have planned weekly meetings with him. If there is no need for a meeting, it will be pushed back another week or two.

4 Resources

The group members have agreed to use the following tools/resources:

- Toggl: Time tracking
- Trello: Light project organization, scrum board and to-do list
- TeamGantt: Project organization
- OneDrive: Sharing and saving files between group
- Teams: Meetings with taskgiver and supervisor
- Discord: Group meetings and work sessions
- GitHub: Code sharing and version control

4.1 Needed resources

The group needs access to resources in SkyHigh (NTNU's OpenStack implementation) to be able develop and test the service. At the time of writing we have been allocated adequate resources in SkyHigh.

5 Risk assessment

As part of the preliminary project, we will find out what risks we face when carrying out the main project. Here we will briefly and concretely describe the main risks we have found. The completion of the project depends on the collaboration between participants and their capacity to work. Each team member has a unique set of skills, that as a team, we value and depend on. Based on the group and project characteristics, we have managed to outline the risks we face. We will be using the ROS veiledning with some changes that makes it more suitable for our assignment. It will be used for the risk assessment in the preliminary project:

Likelihood

Level of possibility	**Description**	**Description of likelihood**	**Frequency Interval (P)**
1	UNLIKELY	One time every other year	0,9/365 to 0,5/365
2	LESS LIKELY	One time every other month	11.9/365 to 6/365
3	LIKELY	More than once per month	P>12/365
4	VERY LIKELY	Once a week	52/365

Table 2: Likelihood table

Impact

Impact will be assessed from a scale from 1 to 4, where 1 is insignificant and 4 is critical. Table 3 describes these intervals.

Level of impact	**Description of level of impact**	**Description of impact**
1	INSIGNIFICANT	No damage done to the project
2	SMALL	Small damage done to the project
3	SERIOUS	Serious damage done to the project
4	CRITICAL	Critical damage done to the project

Table 3: Impact table

5.1 Risks

The group has identified the following risks:

Risk no.	**Description**	**Likelihood**	**Impact**	**Countermeasure**
1	Run out of time due to poor planning	1	4	Yes
2	Scope of the project is too comprehensive which causes delays or setbacks	2	2	Yes
3	Absence due to mild disease	2	1	No
4	Misunderstanding the project	2	3	Yes
5	Loss of availability/data (Overleaf, GitHub)	1	3	Yes
6	Poor communication between the group and taskgiver leads to decreasing level of quality	1	3	Yes
7	Focus on/prioritize things that are not essential for the project	4	2	Yes
8	Test environment doesn't simulate the real world environment well enough	2	3	No
9	The project is too complex for the group	2	3	Yes
10	Running out of time during a sprint	2	2	Yes

5.2 Risk matrix

Impact/Likelihood	**1**	**2**	**3**	**4**
1	🟩	🟩 3	🟩	🟩
2	🟩	🟩 2, 10	🟨	🟨 7
3	🟩 5, 6	🟨 4, 8, 9	🟨	🟥
4	🟩 1	🟨	🟥	⬛

Table 4: Risk matrix before implementing counter measures

5.3 Counter measures

We have defined counter measures as a measure that decreases the likelihood or the impact of the risks. In cases where we are not able to decrease the likelihood or the impact, we have not made any countermeasures as they simply wouldn't reduce the risk.

Risk no.	Description	Likelihood	Impact
1	Detailed planning in advance (Gantt) and ensure that product is potentially shippable at the end of each sprint	1	3
2	Detailed planning in preliminary project so that supervisor/consultant can share his opinion about the scope.	1	2
4	Frequent communication with taskgiver and supervisor	1	3
5	Local backup of code and report	1	1
6	Arrange regular meetings with taskgiver and discuss ongoing matters	1	2
7	Have a common understanding of what the task demands. Create good user stories in the scrum process	2	2
9	Define project requirements and limitations, so it is at a reasonable level of complexity	1	3
10	Having a two extra days between each sprint which can be used for catch up	2	1

5.4 Risk matrix with measures

Impact/Likelihood	**1**	**2**	**3**	**4**
1	🟩 5	🟩 3*, 10	🟩	🟩
2	🟩 2,6	🟩 7	🟨	🟨
3	🟩 1,4,9	🟨 8*	🟨	🟥
4	🟩	🟨	🟥	⬛

Table 5: Risk matrix after implementing counter measures. *No counter measure

6 Progress plan

6.1 Scrum

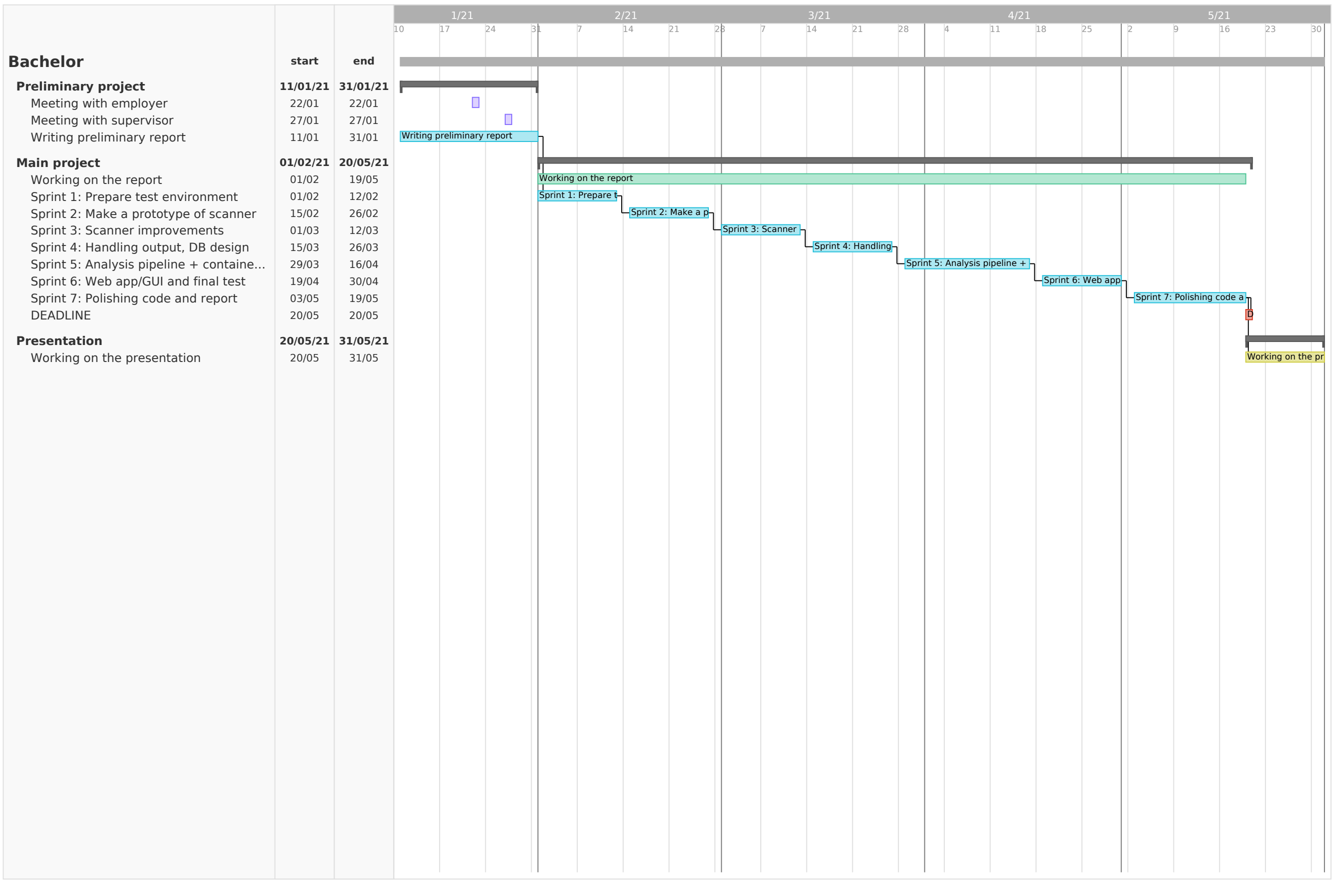
We have identified that our service can easily be broken down into smaller parts, as the taskgiver is requiring that the service should be able to run in a microservice architecture. As the service can easily be "potentially shippable" at the end of each sprint, we have chosen scrum. The scrum model will also allow us easier adapt to new changes during the project period.

6.2 Sprints

All sprints start of by having a sprint review of the previous sprint. As we have an agile approach to this project, we will be testing while coding. This is done to ensure that the product is shippable at the end of each sprint. Our sprints begin on Mondays and end on Friday the week after the beginning, making each sprint 12 days. The weekend between sprints is used as a buffer that can be used to extend the sprint if there are any delays or extra time is needed.

Our plan is to write much of the report while we work on the technical part. This enables us to better document the choices we make.

- **Sprint 1:** Prepare test environment: This includes making HEAT templates for consistent and repeatable deployment of our test infrastructure in SkyHigh, setting up Ansible to install dependencies on machines in infrastructure
- **Sprint 2:** Make a prototype of scanner which includes the most basic functionalities and outputs the result to a file in JSON format.
- **Sprint 3:** Scanner improvements: after testing the prototype we will add more features to the scanner.
- **Sprint 4:** Handling output, DB design: After adding additional functionality to the scanner we will know what data the scanner outputs. We will then design the database and implement a JSON parser and insert the data to the database
- **Sprint 5:** Analysis pipeline + containerization: All gathered data need to be thoroughly analysed, to make it more scalable, we are going to containerize the process and implement REST API. This sprint is one week longer to account for lost working time due to easter.
- **Sprint 6:** Web app/GUI and final test: Making a small web app as a proof of concept. This webGUI will primarily includes functions for searching the saved data in the database. After making the webGUI we will test to see if the whole service works as intended
- **Sprint 7:** Polishing code and report: The last sprint will be used to polish the code (if needed) and make the finishing touches to the report.



Bachelor	start	end
Preliminary project	**11/01/21**	**31/01/21**
Meeting with employer	22/01	22/01
Meeting with supervisor	27/01	27/01
Writing preliminary report	11/01	31/01
Main project	**01/02/21**	**20/05/21**
Working on the report	01/02	19/05
Sprint 1: Prepare test environment	01/02	12/02
Sprint 2: Make a prototype of scanner	15/02	26/02
Sprint 3: Scanner improvements	01/03	12/03
Sprint 4: Handling output, DB design	15/03	26/03
Sprint 5: Analysis pipeline + containe...	29/03	16/04
Sprint 6: Web app/GUI and final test	19/04	30/04
Sprint 7: Polishing code and report	03/05	19/05
DEADLINE	20/05	20/05
Presentation	**20/05/21**	**31/05/21**
Working on the presentation	20/05	31/05

A Project agreement

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Christoffer Vargtass Hallstensen (NTNU IT SOC)
(oppdragsgiver), og

Avleen Singh Marjara, Sander Høgli, Jarl Tengesdal Lygre, Wojciech Malecki
(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra **11.01.2021** til **20.05.2021**.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle beståtte bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): **Erjon Zoto**

Oppdragsgivers kontaktperson (navn): **Christoffer Vargtass Hallstensen**

Student(er) (signatur):

[illegible] dato **12.01.2021**

[illegible] dato **12.01.2021**

[illegible] dato **12.01.2021**

[illegible] dato **12.01.2021**

Oppdragsgiver (signatur): [illegible] dato 28.01.2021

NTNU
NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): ______________________________ dato __________

B Group rules

Gruppekontrakt for DSCS2900

Denne kontrakten regulerer samarbeidet mellom gruppemedlemmene i DSCS2900 våren 2021. Alle gruppemedlemmer forplikter seg til å overholde følgende bestemmelser:

1. Gruppemedlemmer

- Avleen Singh Marjara
- Jarl Tengesdal Lygre
- Sander Høgli
- Wojtek Malecki

2. Mål for prosjektet

1. Gjøre en god faglig innsats
2. Oppnå tverrfaglighet
3. Lære å jobbe godt som et team

3. Orden

1. Det forventes at alle medlemmer møter til avtalt tidspunkt.
2. Dersom et medlem er forsinket, skal han varsle de andre.
3. Hver arbeidsøkt/gruppemøte starter med en oppsummering av hva som er gjort siden sist, dersom medlemmene har jobbet hver for mellom øktene og gå igjennom dagens agenda.
4. Hver arbeidsøkt/gruppemøte avsluttes med å skrive møtereferat. Her skal følgende noteres: Hvem som var til stede, hva som ble gjort, hva som skal gjøres (til) neste møte.
5. Gruppen har avtalt faste arbeidstider. Disse er Mandag-fredag: 11-16, med noen unntak pga forelesninger. Dette er kun et minimum, og alle er innstilt på å øke arbeidstiden utover semesteret.
6. Som gruppemedlem er man forpliktet til å holde seg oppdatert på, og skaffe seg generell forståelse for hva de andre arbeider med.

4. Beslutningsregler

1. Alle avgjørelser skal være oppe til diskusjon og besluttes i fellesskap
2. Ved ugyldig fravær mister man sin rett til innflytelse og må rette seg etter det som vedtas
3. Dersom det etter gjentatte forsøk ikke oppnås enighet, skal det søkes råd hos veileder.

5. Arbeidsprosess

1. Gruppen skal sette konkrete delmål med klare tidsrammer underveis i arbeidet. Dette gjøres som en del av forprosjektet.
2. Gruppen skal dele oppgavene slik at det blir en rettferdig fordeling. Et medlem skal ikke sitte med flere oppgaver enn det han klarer å håndtere, mens andre gjør mye mindre eller ingen ting.
3. Rapporten skal skrives underveis.
4. Gruppemedlemmene må loggføre arbeidstiden.

6. Miljø

1. Gruppemedlemmene skal opprettholde en god tone internt i gruppen.
2. Det er lov å komme med kritikk, men den skal være konstruktiv og saklig.

3. Det forventes at alle har like mye eierskap til oppgaven, slik at hvert enkelt medlem føler at det er deres ansvar å ta tak i problemer som dukker opp underveis.

Signatur: 19.01.2021

[illegible] [illegible] [illegible]

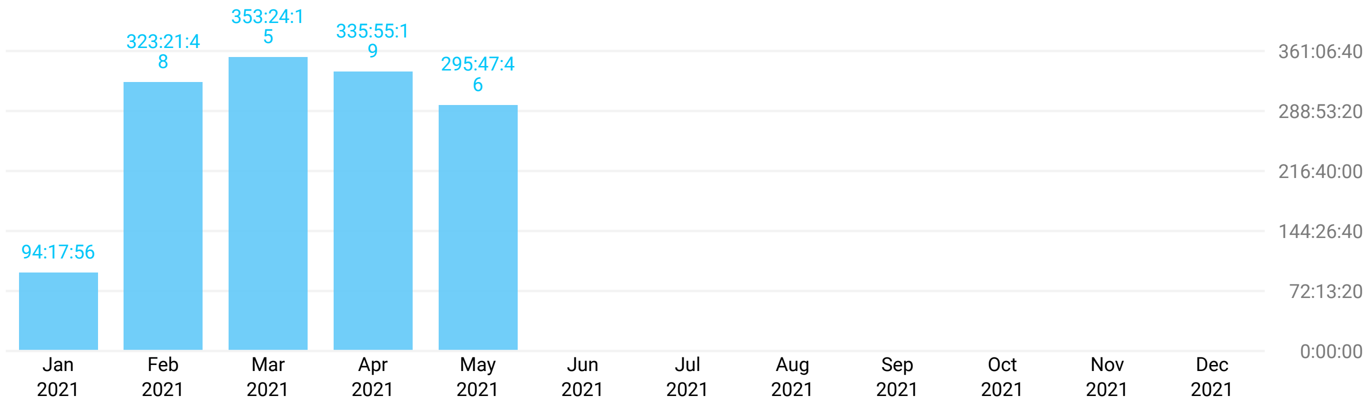
[illegible]

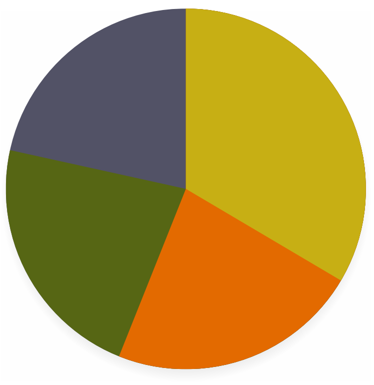
M. *Time report*

Summary Report

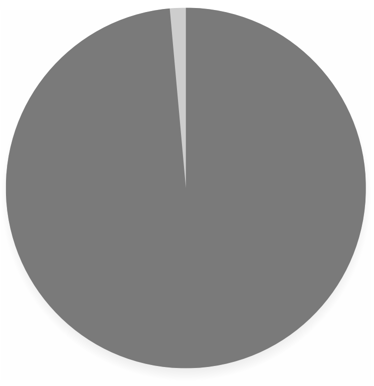
01/01/2021 – 12/31/2021

TOTAL HOURS: 1402:47:04





USER	DURATION
AS Avleen S. Marjara	470:05:42
WM Wojciech Małecki	316:15:48
JA Jarltl	314:06:00
SH Sander Høgli	302:19:34



TIME ENTRY	DURATION
● Without description	1383:23:40
● Other time entries	19:23:24

USER - TIME ENTRY	DURATION
AS Avleen S. Marjara	470:05:42
b	3:15:00
bachel	1:31:00
dokumentasjon	0:19:03
Dokumentasjon for beta test	5:14:34
Gjøre klar til beta test	1:01:47
Utarbeide dokumentasjon og teste kode før beta test	1:02:00
Without description	457:42:18
JA Jarltl	314:06:00
bac	5:00:00
Without description	309:06:00
SH Sander Høgli	302:19:34
Lese gjennom	2:00:00
Without description	300:19:34
WM Wojciech Małecki	316:15:48
Without description	316:15:48

Høgli, Lygre, Małecki, Marjara

Autoenum

NTNU
Kunnskap for en bedre verden