Gymnasium Nepomucenum Coesfeld computer science project course

Design and implementation of a graphical network scan and incident response tool

Project work

submitted by
Lutz Pfannenschmidt
on June 9, 2024

Supervisor: Achim Willenbring

Declaration of independence

I declare with my signature that I have written this thesis independently and have not used any sources other than those indicated. I have fully acknowledged all passages in this thesis that are taken from other works (including the World Wide Web and other electronic text and data collections), stating the sources. Any use of AI tools in connection with my work has been documented by screenshot and made available to the reviewer in the appendix or as a separate PDF file.

Place, date **Signature**

Summary

This project deals with the development of incident response software for network analysis. The software enables security teams to perform network scans, identify security vulnerabilities and take appropriate countermeasures. Based on the objectives of the thesis, functional and non-functional requirements are analyzed and implemented. The implementation includes the integration of Nmap for network analysis, the development of a user-friendly user interface and the integration of WebSSH for efficient remote maintenance. By evaluating the software, its user-friendliness and effectiveness are assessed, identifying potential for improvement and deriving recommendations for adaptation. The result is a powerful and customizable incident response software that supports security teams in the effective analysis and management of security incidents.

Table of contents

1 Introduction

1.1 Motivation and background

Given the increasing sophistication and frequency of cyberattacks, a fast and precise response to security incidents is crucial for the protection of corporate networks and sensitive data. As a possible solution to this problem, I developed an incident response tool for the graphical display of network scans as part of my computer science project course.

The main goal of this project is to develop an intuitive platform that makes it easier for Incident Response (IR) teams to perform network scans and graphically visualize the data obtained. By using Nmap, comprehensive network scans can be created with information about connected devices, open ports and potential security risks.

The development of a user-friendly visual interface based on the results of Nmap enables IR teams to quickly analyze complex network data and make informed decisions on how to respond to security incidents. This app is designed to improve the efficiency and effectiveness of incident response processes and thus sustainably strengthen the security of corporate networks.

In the course of this project, I will focus on the specific requirements of incident response teams and describe the development and implementation of the visual network scanning interface in detail. In addition, possible use cases and potential extensions are discussed in order to further optimize the functionality and relevance of this solution for incident response teams.

1.2 Goals

The objectives of this study can be summarized as follows:

1. **Development of a user-friendly visual representation of network scans to improve the detection and analysis of networks:**
   - The aim is to create an intuitive and appealing user interface that makes it possible to display network scans visually.
   - The graphical representation of the networks should make it easier and quicker for users to gain a comprehensive understanding of the network structure.
   - This includes the visualization of network components and connections.
2. **Implementation of clear archiving and display of all past scans:**
   - A function is to be developed that archives all network cans performed in a structured and easily accessible manner.
   - This archiving enables administrators to retrieve and compare previous scans quickly and easily.
   - The clarity and structure of the archiving should ensure efficient tracking and analysis of network changes.
3. **Creation of a Web-based Secure Shell (WebSSH) interface for direct and fast problem solving in the network to enable administrators to carry out efficient remote maintenance:**
   - The aim is to develop a web interface that makes it possible to establish SSH connections directly via a web browser.
   - This WebSSH interface is designed to provide users with the ability to quickly and efficiently access network devices and perform maintenance work without having to rely on additional SSH clients.

- The integration into the existing user interface aims to provide a seamless and user-friendly solution that significantly reduces the response time in the event of security incidents and network problems.

2 Basics

2.1 The OSI model

The Open Systems Interconnection (OSI) model is a reference model that defines the conventions and tasks of the various hardware and software components to help developers design compatible systems[1] . It consists of seven layers, each of which fulfills specific functions to ensure the smooth flow of data transmission.

The OSI model helps by:

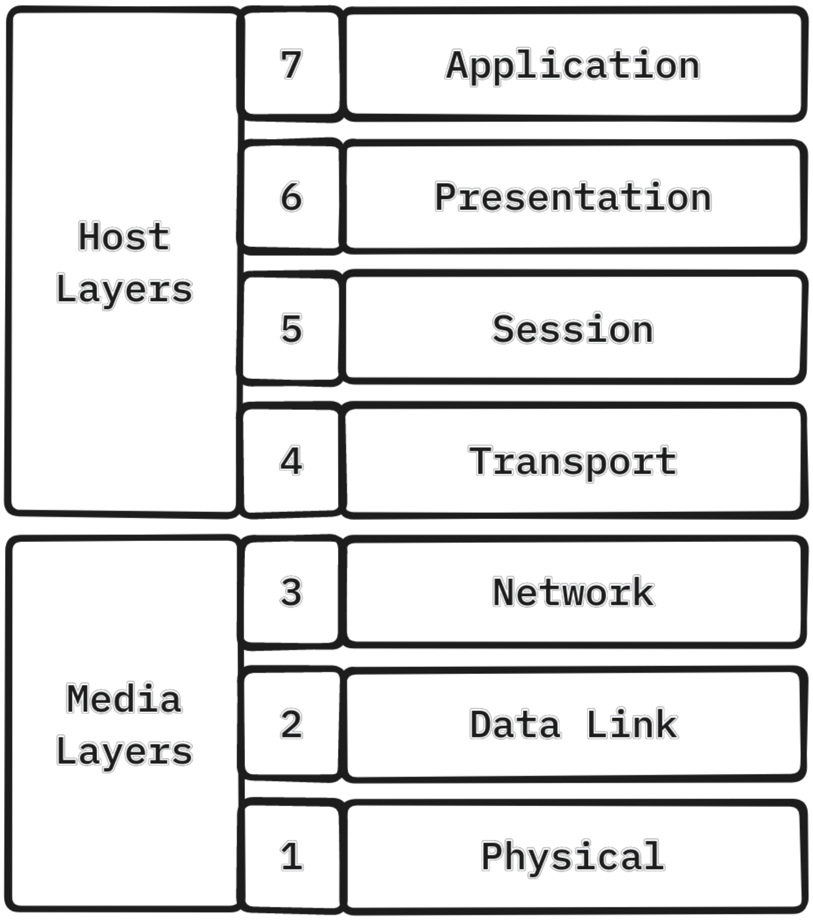
1. Enables standardization and compatibility between different network systems
2. offers developers a structured approach to designing network systems
3. Troubleshooting and problem solving in networks simplified, as each level has clearly defined tasks and interfaces

The structure of the OSI model according to recommendation X.200[2] is shown in Figure 2.1.

[1]K. Sumit, D. Sumit and Vivek, "THE OSI MODEL: OVERVIEW ON THE SEVEN LAYERS OF COMPUTER NETWORKS".

[2]International Telecommunication Union (ITU), *Recommendation X.200 (07/94)*.

Figure 2.1: The 7 layer OSI model



The 7 levels are defined as follows:

1. **Physical level** The physical level describes the electrical and physical specifications. Among other things, voltage, cable type, pin layout and host bus adapter (HBA) are defined here.

2. **Data link level** The data link level is responsible for error detection and correction at the bit level. It ensures the reliability of data transmission at the physical level.

3. **Network level** The network level takes care of the forwarding of

data packets between different networks. Routing and addressing are carried out here.

4. **Transport level** The transport level is responsible for end-to-end (E2E) communication between applications. It ensures that data is transferred reliably and in the correct order.

5. **Session level** The session level terminates, manages and establishes sessions between applications. It enables data exchange and synchronization between the communication partners.

6. **Presentation level** The presentation takes care of the display and conversion of data formats to ensure compatibility between different systems.

7. **Software/applications level** The software/applications level contains the applications and services that interact directly with the user. Application protocols such as HTTP, FTP, SMTP etc. are implemented here.

2.1.1 The data transmission process according to the OSI model

The data transmission process in the OSI model is as follows[3] :

On the transmitter side:

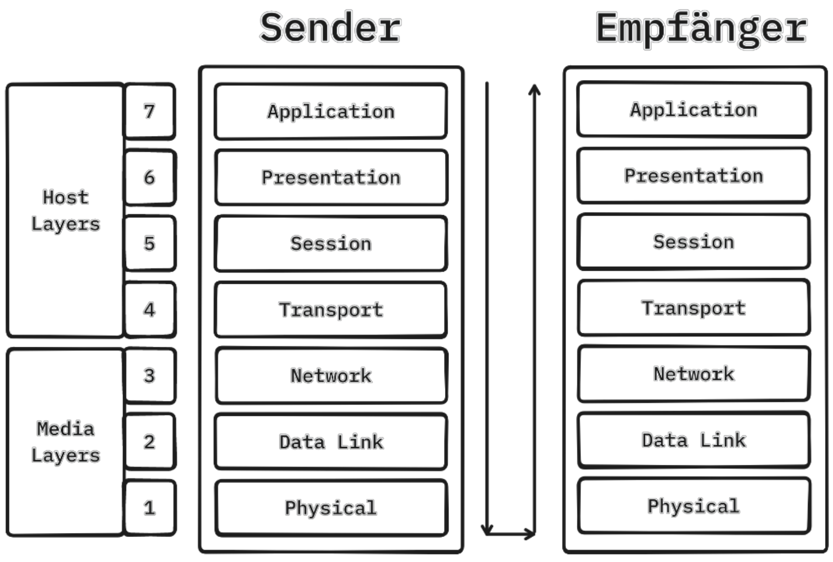
1. The application on layer 7 generates the data to be transmitted and transfers it to the underlying layers.
2. Layer 6 (presentation) converts the data into a standardized format to ensure compatibility with the receiver.
3. Layer 5 (session) establishes and manages the communication session between sender and receiver.
4. Layer 4 (TCP or UDP) divides the data into packets, numbers them and adds additional information to ensure reliable and correct transmission.
5. Layer 3 (IP) addresses the packets and routes them to the recipient via the Local Area Network (LAN) and then the Wide Area Network (WAN).
6. Layer 2 (Media Access Control (MAC)) packs the packets into frames, adds sums and sends them via the physical medium.
7. Layer 1 (Physical) converts the digital signals into electrical or optical signals and transmits them via the network cable or other transmission media.

[3]K. Sumit, D. Sumit and Vivek, "THE OSI MODEL: OVERVIEW ON THE SEVEN LAYERS OF COMPUTER NETWORKS".

On the receiver side:

1. Layer 1 (Physical) receives the signals from the transmission medium and converts them into data that can be used by the hardware.
2. Layer 2 (MAC) checks the integrity of the frames and removes checksums.
3. Layer 3 (IP) checks the addressing and removes routing information.
4. Layer 4 (TCP or UDP) reconstructs the original data from the packets and checks the sequence and completeness.
5. Layer 5 (session) synchronizes the data exchange.
6. Layer 6 (presentation) converts the data into a format that can be read by the application.
7. Layer 7 (application) receives the data and makes it available to the user.

Figure 2.2: The data transmission process according to the OSI model



2.2 Introduction to network security and incident response

Today, network security is critical as it is becoming increasingly difficult to protect these networks in today's digital world. This chapter contains the basic concepts and practices of network security and incident response (IR).

2.2.1 Network security

Network security is concerned with ensuring the confidentiality, integrity and availability of data and resources in a network.[4] These are the main objectives of this thesis:

1. **Confidentiality**: The data should only be able to be viewed by authorized users.
2. **Integrity**: The data should be protected against unauthorized changes to ensure that it remains correct and unchanged.
3. **Availability**: The data should be provided as quickly and easily as possible.

2.3 Introduction to Nmap

Nmap, short for „Network Mapper", is an indispensable open source tool used by network administrators and security experts worldwide to analyze networks and identify security vulnerabilities. In the context of my project, Nmap plays a central role in performing network scans and collecting detailed information about the network.

2.3.1 Functionality of Nmap

The main functions of Nmap include:

[4]Peterson and Davie, *Computer Networks: A Systems Approach*.

- **Port scanning**: Nmap identifies open ports on a host, which is essential to detect running services and potential security vulnerabilities. This function makes it possible to obtain an overview of which applications and services are active on the network devices.
- **Service detection**: In addition to detecting open ports, Nmap can identify the services running behind them and their versions. This is particularly useful for detecting outdated or obsolete services that could pose a security risk.
- **Operating system detection**: By analyzing network traffic patterns, Nmap can determine the operating system of a target host. This information is valuable for carrying out targeted security measures and vulnerability analyses.
- **Nmap Scripting Engine (NSE)**: With the NSE, specific checks and tests can be automated by creating user-defined scripts or using existing scripts.

2.3.2 Interaction of Nmap with the OSI model

Nmap works on several layers of the OSI model to perform comprehensive network scans:

- **Data Link Layer**: Nmap can use Address Resolution Protocol (ARP) scans to resolve IP addresses and find out which devices are active in the network. These scans are particularly useful in local networks.
- **Network Layer**: Nmap uses Internet Control Message Protocol (ICMP) packets for ping scans to find out whether hosts are active and reachable.
- **Transport Layer**: Nmap performs port scans by sending TCP and UDP packets to identify open ports and determine which services are running on those ports.

- **Application Layer**: Nmap can use NSE scripts and service discovery techniques to work at the application level to identify specific services and find vulnerabilities in those services.

2.3.3 Use of Nmap for incident response teams

The use of Nmap offers significant advantages for incident response teams:

- **Comprehensive network analysis**: With the ability to collect detailed information about networks and their hosts, Nmap enables a thorough analysis of the network status. This is crucial for the early detection and elimination of security vulnerabilities.

- **Early detection of security vulnerabilities**: Nmap helps to identify vulnerabilities, such as outdated or faulty services, early. This allows preventive measures to be taken before potential attackers exploit these vulnerabilities.

- **Flexibility and versatility**: Nmap supports various scanning technologies and protocols and can therefore be used in different network scenarios. This versatility makes it a valuable tool for network security.

2.3.4 Summary

Nmap is an essential tool for network security that contributes to network analysis and vulnerability identification through its comprehensive functions. The detailed information that Nmap provides about networks and their hosts helps incident response teams to efficiently detect and resolve security incidents. By interacting with different layers of the OSI model, Nmap can perform a wide range of network scans and thus make an important contribution to the proactive and effective security strategy of organizations.

2.4 Introduction to enterprise networks

Enterprise networks are complex infrastructures designed to support the communication and data processing requirements of large organizations. These networks comprise various components and security mechanisms that work together to ensure smooth and secure operation.

2.4.1 Main components of an enterprise network

A typical enterprise network consists of several important components, each of which fulfills a specific role in the network[5] :

- **Firewalls**: Firewalls are essential for protecting the network from unauthorized access. They filter incoming and outgoing network traffic based on predefined security rules and thus prevent harmful data from entering the network or confidential information from leaving the network.

- **Clients**: These are the end devices such as computers, laptops, tablets and smartphones that are used by users to access network resources and services. Each client on the network must be correctly configured and secured to ensure the overall security of the network.

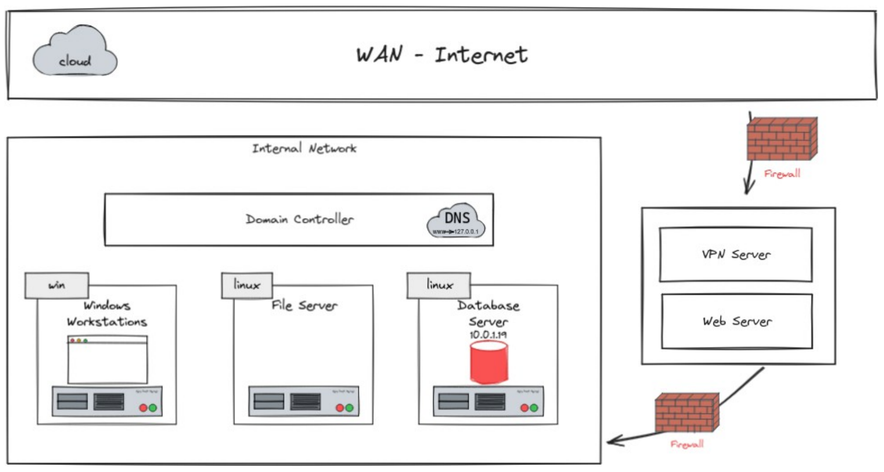
- **Demilitarized Zones (DMZs)**: A DMZ is an isolated network area that serves as a buffer zone between the internal network and external networks, such as the Internet. Publicly accessible services such as web servers, mail servers and DNS servers are placed in the DMZ to prevent attackers from gaining direct access to the internal network.

- **Virtual Private Networks (VPNs)**: VPNs enable secure connections over the Internet by encrypting data and protecting communication between remote users and the corporate network. VPNs are particularly important for connecting remote employees and branch offices to the main network.

[5]Peterson and Davie, *Computer Networks: A Systems Approach*.

- **Switches and routers**: Switches and routers direct data traffic within the network and connect different network segments with each other. Switches work on the data link layer of the OSI model and connect devices within the same network, while routers work on the network layer and forward data traffic between different networks.
- **Servers**: Servers are central service providers in the network that provide various services and applications. These can be file and print servers, database servers, e-mail servers, web servers and application servers. The availability and security of these servers is crucial for the operation of the network.

Figure 2.3: Structure of an enterprise network



2.4.2 Summary

An enterprise network is a complex and multi-layered structure consisting of various components that work together to enable secure and efficient communication and data processing. The implementation of security measures and adherence to best practices are essential to protect the network from threats and ensure smooth operation. By integrating technologies such as firewalls and VPNs, companies can improve their network security.

3 Requirements analysis

This chapter describes the functional and non-functional requirements based on the objectives. Functional requirements relate to specific functions and behaviors of the system, while non-functional requirements relate to the quality and performance of the system.

3.1 Functional requirements

No.	Requirement	Cat.	Description
01	Server-based network scan	MUST	The server should connect to the network at the request of the Scan clients (website).
02	Binary/Executable package	MUST	Client and server should be packed into an executable file that can be opened with a single click. command can be executed.
03	Configurable scan	MUST	The scan should be configurable in order to to fulfill different requirements.
04	Manual entry	TARGET	Possibility, machines manually add- that were not recognized.
05	Automatic data request	TARGET	Automatic requests with from data Firewall loggers (if possible).
06	WebSSH	MUST	Possibility to establish an SSH connection to a machine with an open SSH port directly from the browser. SSH port.
07	Graphical view	MUST	Clear display of the scanned image Network.
08	Scan History	MUST	An overview of all previous scans.

Table 3.1: Functional requirements for the network scanning interface

3.2 Non-functional requirements

No.	Requirement	Cat.	Description
09	Notes on machines	TARGET	Possibility to add notes to each machine. to add.
10	Performance	MUST	The scan must be carried out within acceptable time frames, in particular for fast scans.
11	User friendliness	TARGET	The user interface must be intuitive and be easy to operate.
12	Security	TARGET	All Data transmissions must be verified be carried out conclusively.
13	Robustness	TARGET	The system must be robust and error-resistant to remain stable even under high loads.
04	Documentation	TARGET	Instructions for using the tool.

Table 3.2: Non-functional requirements

4 Concept and design

4.1 Architecture of the application

The architecture of the developed application defines the basic framework on which all functions are based. It determines the structure of how the various components interact and communicate with each other.

An easily expandable structure was chosen for the architecture in order to ensure flexibility in development. The application consists of the following main components:

- **Backend**: The backend of the application comprises the core logic and the data processing functions. This is where network scans are carried out, data is processed and prepared for the user interface and stored for further use.
- **Frontend**: The user interface is the interface through which the user interacts with the application. This is where the results of the network scans are presented visually.
- **Data storage**: The data from all network scans is stored in compressed form in the application folder and read in when the application is restarted.

4.2 Selection of technologies and frameworks

The following criteria were [illegible]nto [illegible]electing the technologies and frameworks for the development of the application: Performance, efficiency and compatibility with the project objectives. The following technologies and frameworks were selected:

- **Go**: Go(-lang) was chosen as a programming language due to its high performance, simple parallelism and efficiency. Golang offers very simple parallelization and easy-to-understand syntax.
- **htmx**: htmx is a powerful front-end framework that makes it possible to change the content of a website without writing JavaScript or reloading the page.
- **Tailwind CSS**: Tailwind CSS was selected as a CSS framework due to its simplicity and flexibility in the design of user interfaces.
- **httprouter by Julien Schmidt**: httprouter is an easy-to-implement HTTP router for Golang that has been optimized for high speed and minimal overhead.

4.3 Design of the user interface and visualization concepts

The design of the user interface was developed with user-friendliness, aesthetics and functionality in mind.

The user interface offers various functions and options for visualizing and analyzing network scans, including

- **Visualization of networks**: The user interface displays networks in the form of diagrams to visualize the relationships between the various network devices and components.
- **Filter function**: Users can filter data according to specific criteria to quickly find and analyze relevant information.
- **Interactive elements**: The user interface provides interactive elements such as buttons, drop-down menus and sliders to help users interact with the data and perform customized analysis.

5 Implementation

5.1 Implementation of the functionalities based on the requirements

The implementation of the application is based on the previously defined requirements and follows a clear structure to ensure efficiency and extensibility. This section describes the most important aspects of the implementation, including the technologies used and the integration of the various components.

5.1.1 Bundling of all components with Go

A central goal of the implementation was to bundle all necessary components into one executable file. External Go libraries are bundled into the same executable file to simplify installation. This makes it easy to deploy and use the application without additional installation steps.

The only feature that is not included as standard is WebSSH. To use this, the external program tty2web[1] is required. With tty2web, it is possible to create a website that can be used to access other computers using the SSH protocol.

- **Integration of all dependencies**: All required libraries and packages are integrated into the project.
- **Compilation**: The application is compiled in such a way that all packages are contained in the executable file.

[1]kost, *kost/ny2web*.

- **Deployment**: The executable file created can be executed directly on the target systems without the need for additional installations (with the exception of the optional tty2web).

5.1.2 Integration of the HTTP router and the user interface

For the web interface and the user interactions I used the HTTP router by Julien Schmidt[2] . The main steps to integrate the router are:

- **Routing configuration**: Definition of the different routes and endpoints through which users can interact with the application.
- **Handler implementation**: Implementation of the handler functions that respond to user requests and perform the corresponding actions.
- **Integration of the user interface**: Integration of the front-end components to provide an interactive and user-friendly interface.

5.1.3 Visualization and storage of network scans

The results of the network scans are saved in compressed form and read in when the application is restarted. This enables efficient use of memory resources and fast availability of the data. The most important steps for implementing this functionality are

- **Data compression**: The results of the network scans are compressed and saved in a special directory in the application.
- **Data recovery**: When the application is started, the saved data is read in and prepared for the user interface.
- **Visualization**: The data is displayed visually in the user interface to enable easy analysis and interpretation.

[2]Schmidt, *julienschmidt/hnprouter*.

5.1.4 Summary

The application was implemented taking into account the defined requirements and using efficient and powerful technologies. By integrating the Go Nmap package, bundling all components and using the httprouter, it was possible to develop a flexible, expandable and user-friendly solution that supports incident response teams in the fast and precise analysis of network scans.

5.2 Integration of Nmap into the application

5.2.1 Using the Go Nmap package

I used the Go Nmap package[3] for the network scans. This package offers a simple way to use Nmap functions in Go. This allowed comprehensive network scans to be implemented efficiently and reliably. The Go Nmap package provides a wrapper for all functions that can normally be used in the console, e.g.

```
/usr/local/bin/nmap -p 80,443,843 google.com youtube.com
```

to

```
scanner, err := nmap.NewScanner(
    ctx,
    nmap.WithTargets("google.com", "youtube.com"),
    nmap.WithPorts("80,443,843"),
)
```

5.3 Integration of WebSSH into the application

To provide a WebSSH console in the application, I have integrated tty2web[4] , which makes it possible to make terminal sessions accessible via a website. This

[3]Le Glaunec, *Ullaakut/nmap*.
[4]kost, *kost/ny2web*.

is particularly useful for quickly accessing remote machines without having to open a separate console.

5.3.1 Why ty2web?

tty2web is easy to use and can be installed externally, as it is intended to be an optional feature.
The command tty2web --permit-write --once ssh root@127.0.0.1, for example, starts a website that can be called up once to establish an SSH connection to the local host, which only works once.

5.3.2 Implementation steps

The integration of tty2web into the application involved several steps to ensure seamless and efficient use. Here are the detailed implementation steps I went through:

Creation of the wrapper libtty2web

Since the original tty2web cannot be used as a package, I have written a wrapper called libtty2web[5] . This wrapper can generate and execute the command to run tty2web without the user having to interact with the os/exec standard library.
The library basically works by creating a Tty2Web object and then adding options:

```
type Tty2Web struct{
        options []option // options to be passed to the binary
        binary  string   // binary to be executed
        process *exec.Cmd // process to be executed
        command []string // command to be executed
}

// AddOptions adds options to the Tty2Web instance
```

[5]Pfannenschmidt, *Lutz-Pfannenschmidt/libny2web*.

```go
func (t *Tty2Web) AddOptions(option ...option){
        t.options = append(t.options, option...)
}

// Run executes the command
func (t *Tty2Web) Run() error {
        t.process = exec.Command(t.binary, t.buildCommand()...)
        err := t.process.Run()
        if err != nil{
                return err
        }
        return nil
}
```

Options are used with the helper functions such as

libtty2web.With<Option>(<Configuration>)

and [illegible]ccount when executing.

```go
type option struct{
        name     string
        value    string
        hasValue bool
}

// IP address to listen (default: "0.0.0.0")
func WithAddress(address string) option{
        return option{
        name: "address",
        value: address,
        hasValue: true,
    }
}

// Port number to listen (default: "8080")
func WithPort(port string) option{
        return option{
        name: "port",
        value: port,
        hasValue: true,
    }
}
```

The use of my package then looks like this:

```
package main

import "github.com/Lutz-Pfannenschmidt/libtty2web"

func main() {
        test := libtty2web.NewTty2Web("bash")
        test.SetBinary("tty2web")
        test.AddOptions(
                libtty2web.WithPermitWrite(),
                libtty2web.WithOnce(),
        )
        err := test.Run()
        if err != nil{
                panic(err)
        }
        test.Wait()
}
```

This program hosts a website that can be called up once and emulates a terminal with bash.

5.4 Development of the visual interface

The visual interface of the application was designed to be user-friendly and intuitive. The most important aspects of the development of the user interface are described below, including the technologies used.

5.4.1 Technologies used

For the development of the user interface, I used modern web technologies to ensure a responsive and engaging user experience. This included:

- **HTML5**: For structuring content and designing the basic page architecture.
- **HTMX**: A framework that makes it possible to change website content without writing JavaScript or reloading the page. This facilitates the creation of dynamic and interactive user interfaces.
- **Tailwind CSS**: A CSS framework that provides fast and flexible styling. It enables simple customization of the user interface through utility-first CSS classes.[6]

5.4.2 Components of the user interface

The user interface consists of several separate components that are hosted by my backend:

- **Analytics**: The central dashboard provides an overview of all important information and functions of the application. Here, users can get an overview of the network.
- **Graph**: This component visualizes the network structure and the results of the network scans in the form of diagrams and graphs. This enables users to quickly and easily understand the relationships between the various network devices and components.
- **History**: A special view for displaying the historical data of network scans. Users can view, compare and analyze past scans to identify trends and patterns in network behavior.

[6]*Tailwind CSS*.

5.4.3 Integration with the backend

The user interface communicates with the backend, which is implemented in Go, via a RESTful API. This API makes it possible to exchange data between the frontend and the backend and to perform actions such as starting network scans or retrieving scan results. By using HTMX, data can be retrieved and updated asynchronously without having to reload the entire page.

6 Evaluation and examples

6.1 Execution of tests

Testing of the application was carried out extensively to ensure that all functions worked as expected. I used the following test methods for this:

- **Unit tests**: Golang has a built-in testing framework that is easy to use. This is used to test individual functions and features.
- **Manual tests**: End-to-end tests were carried out manually to check the user-friendliness and overall functionality of the application.

6.2 Identification of potential for improvement and adaptation recommendations

During the tests and evaluations, several areas were identified that could be improved:

- **Expansion of the filter functions**: The existing filter options could be extended to enable users to analyze network scans even more precisely.
- **Improvement of visualizations**: Additional visualization tools and operations could be added to better represent complex network structures, e.g. Lucidchart[1] - similar tools.
- **Documentation**: There is only a rough documentation on the homepage of my software[2] , which only explains how to use the binary and shows all features, but does not explain them. Here it would be possible to explain how to interact with the individual features.

6.3 Conclusion

As part of this work, a comprehensive network scan analysis and visualization tool was developed to improve both usability and efficiency in network security. The implementation was based on the previously defined requirements and included several key components, including the integration of Nmap, the development of a user-friendly web interface and the provision of a Web-based Secure Shell (WebSSH) console.

6.3.1 Goals achieved

The main objectives of the work were successfully implemented:

- **User-friendly visual representation of network scans**: The application provides a clear and intuitive visual interface to display the network scans.

[1] *Lucidchart*.
[2] *ResponsePlan*.

topology and the scan results. This visual representation makes it much easier to record and analyze network information.

- **Efficient archiving and display of past scans**: All network scans performed are archived in compressed form and can be retrieved at any time. This enables efficient tracking and analysis of network development over time.
- **WebSSH interface for remote maintenance**: The integration of a WebSSH interface enables administrators to quickly and easily access remote machines and resolve problems directly. This improves response time and efficiency when managing and maintaining the network.

6.3.2 Evaluation of the application

The application was tested in various scenarios to verify its functionality and performance. The tests have shown that the application is stable and reliable and fulfills the defined requirements. The user interface proved to be intuitive and user-friendly, and the integration of the various components went smoothly.

6.3.3 Identified potential for improvement

Despite the successful implementation, there are some areas where future work can begin:

- **Extension of the visualization options**: The introduction of additional visualization options could further improve the analysis of the network topology and scan results.[3]
- **Extended security functions**: The implementation of additional security functions, such as automatic detection and notification of suspicious activities, could increase the usefulness of the application.

[3] *Lucidchart*.

- **Performance optimization**: The performance of the application, especially with very large networks, could be further optimized to achieve even faster scanning and processing times.

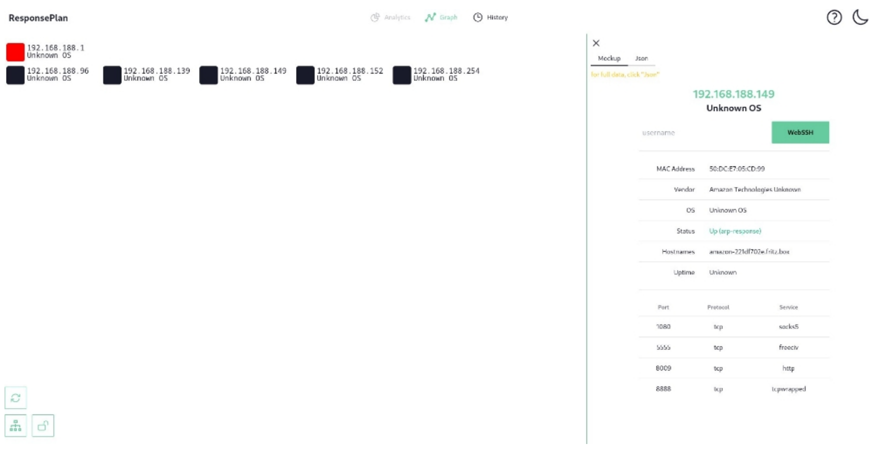
6.3.4 Practical application cases

The developed application offers numerous practical use cases for incident response teams:

- **Fast identification of security vulnerabilities**: The detailed analysis of network scans enables security vulnerabilities to be identified and rectified at an early stage.
- **Efficient network maintenance**: The WebSSH interface enables fast and efficient remote maintenance, which shortens the response time in the event of network problems.
- **Monitoring of network development**: By archiving and analyzing past scans, the development of the network can be monitored and analyzed over time, which supports proactive network security.

Overall, this work has shown that the developed application can contribute to the improvement of network security and maintenance. The combination of powerful network scans, a user-friendly interface, and the ability to manage network security issues effectively and proactively provides incident response teams with a tool to do so.

Figure 6.1: Scan of my private network without operating system identification.



192.168.188.1
Unknown OS

192.168.188.96
Unknown OS

192.168.188.139
Unknown OS

192.168.188.149
Unknown OS

192.168.188.152
Unknown OS

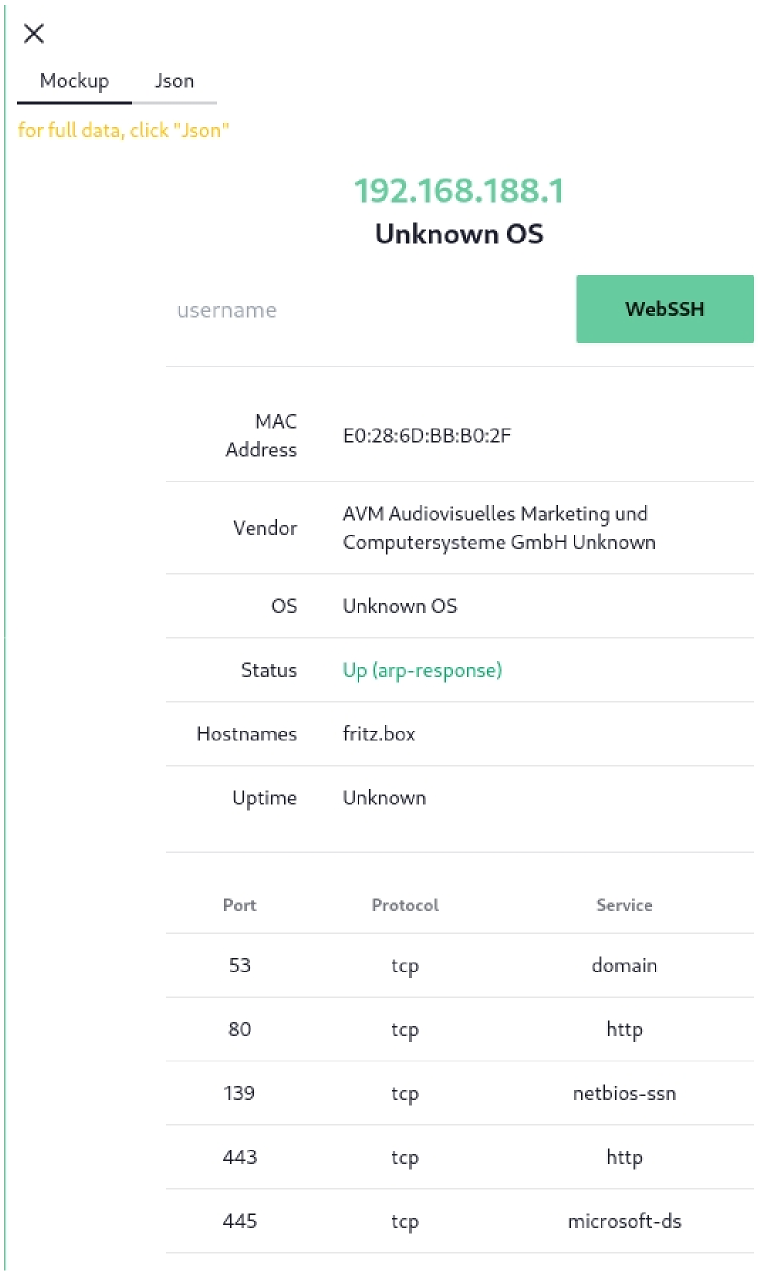
192.168.188.254
Unknown OS

for full data, click "Json"

192.168.188.149
Unknown OS

MAC Address	50:DC:E7:05:CD:99
Vendor	Amazon Technologies Unknown
OS	Unknown OS
Status	Up (arp-response)
Hostnames	amazon-221df702e.fritz.box
Uptime	Unknown

Port	Protocol	Service
1080	tcp	socks5
5555	tcp	freeciv
8009	tcp	http
8888	tcp	tcpwrapped



for full data, click "Json"

192.168.188.1
Unknown OS

MAC Address	E0:28:6D:BB:B0:2F
Vendor	AVM Audiovisuelles Marketing und Computersysteme GmbH Unknown
OS	Unknown OS
Status	Up (arp-response)
Hostnames	fritz.box
Uptime	Unknown

Port	Protocol	Service
53	tcp	domain
80	tcp	http
139	tcp	netbios-ssn
443	tcp	http
445	tcp	microsoft-ds

Figure 6.2: Overview of a FritzBox!

A Appendix

A.1 Source code examples

A.1.1 Tty2Web Wrapper

The following source code shows the use of the libtty2web wrapper, which facilitates the use of the tty2web binary:

```
package main

import "github.com/Lutz-Pfannenschmidt/libtty2web"

func main ( )   {
        test := libtty2web.NewTty2Web
        ("bash") test.SetBinary("tty2web")
        test.Add Options (
                libtty2web.With Permit Write (
                ),libtty2web.WithOnce(),
        )
        err := test.Run()
        if err != nil{
                panic(err)
        }
        test.Wait()
}
```

A.1.2 YAGLL - Yet Another Go Logging Library

I also wrote my own logging library because I needed a customized solution for logging in my project. This library, which I called YAGLL - Yet Another Go Logging Library, is based on the requirements of my software and is strongly inspired by Distillog, but offers some additional functions and customization options.

```go
package main

import "github.com/Lutz-Pfannenschmidt/yagll"

func main () {
	yagll.Toggle (yagll.DEBUG,true)
	yagll.Debugln ("Debug mode enabled")
	yagll.Infof("This is a %s message" , "Info")
}
```

Functions

- **Customizable log output**: The log output can be customized with the SetOutput function. Any os.file instance can be used as output.
- **Switch each protocol level on and off**: Each protocol level can be switched on and off with the Toggle function.
- **Customizable colors**: The colors used for the protocol levels can be adjusted using the SetColor function.
- **Customizable template with text/template**: The template for formatting the log messages is fully customizable with text/template. The following variables are available:
  - {{ . Time }} - The time at which the log message was created
  - {{ .Level }} - The protocol level of the message
  - {{ .Message }} - The message itself

- {{ .Color }} - The color of the protocol level
- {{ .Reset }} - The code to reset the color

A.2 Extended configuration options

The following options can be passed the program to change its behavior:

- -m / --memory - Memory mode only. This deactivates file storage.
- -d / --debug - Activate debug mode. This prints additional information to the console.
- -h / --help - Display the help message.
- -p / --port <port>- Change the port on which the server hosts. By default, this is 1337.
- -e / --expose - Release the server in the LAN. This allows other devices to establish a connection to the server. [Caution]: This may pose a security risk.
- -o / --out <filename>- The file in which the data is to be saved. By default, this is data.responseplan.
- -i / --in <filename>- The file from which the data is to be loaded. By default, this is data.responseplan.
- --tty2web *<path>-* Use a user-defined path for the tty2web binary. By default, this is tty2web.

A.3 Documentation

Detailed documentation and an overview of the project's features can be found on the website https://responseplan.de.

A.4 Repositories

Here are the links to my GitHub repositories:

- **ResponsePlan**: My main application, ResponsePlan, is available on GitHub at the following link:
https://github.com/Lutz-Pfannenschmidt/ResponsePlan
- **YAGLL - Yet Another Go Logging Library**: The logging library YAGLL developed by me is available at the following link:
https://github.com/Lutz-Pfannenschmidt/yagll
- **libtty2web Wrapper**: A wrapper for the tty2web tool, which I have developed, is available under the following link:
https://github.com/Lutz-Pfannenschmidt/libtty2web
- **responseplan.de Documentation**: The documentation for ResponsePlan is available at the following link:
https://github.com/Lutz-Pfannenschmidt/responseplan. de

B List of illustrations

C List of tables

D Literature

International Telecommunication Union (ITU): Recommendation X.200 (07/94) ITU-X.200

International Telecommunication Union (ITU). *Recommendation X.200 (07/94)*. English. Recommendation X.200. ITU, July 1994. URL: https://www.itu.int/rec/T-REC-X.200-199407-I/en.

K. Sumit et al: THE OSI MODEL: OVERVIEW ON THE SEVEN LAYERS OF COMPUTER NETWORKS OSI MODEL

Kumar Sumit, Dalal Sumit and Dixit Vivek. "THE OSI MODEL: OVERVIEW ON THE SEVEN LAYERS OF COMPUTER NETWORKS". In: *Computer Science* 2 (2005), pp. 68-72.

Peterson et al: Computer Networks: A Systems Approach ComputerNetworks

Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann, 2012.

cost: cost/tty2web tty2web

kost. *kost/ny2web*. 2024. URL: %5C%5Chttps://github.com/kost/tty2web (visited on 05. 06. 2024).

Le Glaunec: Ullaakut/nmap GoNmap

Brendan Le Glaunec. *Ullaakut/nmap*. 2024. url: https://github.com/Ullaakut/nmap (visited on 05. 06. 2024).

Lucidchart lucidchart

Lucidchart. 2024. url: https://www.lucidchart.com/ (visited on 08. 06. 2024).

Pfannenschmidt: Lutz-Pfannenschmidt/libtty2web **libtty2web**

Lutz Pfannenschmidt. *Lutz-Pfannenschmidt/libny2web*. 2024. URL: https://github. com/Lutz-Pfannenschmidt/libtty2web (visited on 08. 06. 2024).

ResponsePlan **ResponsePlan**

ResponsePlan. 2024. URL: https : / / www . responseplan . de/ (visited on 08. 06. 2024).

Schmidt: julienschmidt/httprouter **httprouter**

Julien Schmidt. *julienschmidt/hnprouter*. 2024. url: https://github.com/julienschmidt/ httprouter (visited on 05. 06. 2024).

Tailwind CSS **tailwind**

Tailwind CSS. 2024. url: https://tailwindcss.com/ (visited on 08. 06. 2024).

E List of abbreviations

ARP Address Resolution Protocol. 10

E2E End-to-End. 6

HBA Host Bus Adapter. 5

ICMP Internet Control Message Protocol. 10

IP Internet Protocol. 7, 8

IR Incident Response. 1, 9

LAN Local Area Network. 7 **MAC**

Media Access Control. 7, 8 NSE

Nmap Scripting Engine. 10

OSI Open Systems Interconnection. 4, 5, 7, 10, 37

SSH Secure Shell. 19

TCP Transmission Control Protocol. 7, 8, 10

UDP User Datagram Protocol. 7, 8, 10

WAN Wide Area Network. 7

WebSSH Web-based Secure Shell. 2, 19, 21, 28-30, III, V