Gymnasium Nepomucenum Coesfeld
Projektkurs Informatik

Entwurf und Implementierung eines grafischen Netzwerkscan- und Incident-Response-Werkzeuges

Projektarbeit

vorgelegt von
Lutz Pfannenschmidt
am 9. Juni 2024

Betreuer: Achim Willenbring

Eigenständigkeitserklärung

Ich erkläre mit meiner Unterschrift, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen dieser Arbeit, die dem Wortlaut, dem Sinn oder der Argumentation nach anderen Werken entnommen sind (einschließlich des World Wide Web und anderer elektronischer Text- und Datensammlungen), habe ich unter Angabe der Quellen vollständig kenntlich gemacht. Jede Nutzung von KI-Tools, die im Zusammenhang mit meiner Arbeit stand, habe ich per Screenshot dokumentiert und der bzw. dem Prüfenden im Anhang oder als separate PDF-Datei zur Verfügung gestellt.

Ort, Datum **Unterschrift**

Zusammenfassung

Diese Projektarbeit behandelt die Entwicklung einer Incident-Response-Software zur Netzwerkanalyse. Die Software ermöglicht es Sicherheitsteams, Netzwerkscans durchzuführen, Sicherheitslücken zu identifizieren und geeignete Gegenmaßnahmen zu ergreifen. Basierend auf den Zielen der Arbeit werden funktionale und nicht-funktionale Anforderungen analysiert und umgesetzt. Die Implementierung umfasst die Integration von Nmap für die Netzwerkanalyse, die Entwicklung einer benutzerfreundlichen Benutzeroberfläche und die Einbindung von WebSSH für eine effiziente Fernwartung. Durch die Evaluierung der Software werden ihre Benutzerfreundlichkeit und Effektivität bewertet, wodurch Verbesserungspotenziale identifiziert und Anpassungsempfehlungen abgeleitet werden. Das Ergebnis ist eine leistungsfähige und anpassungsfähige Incident-Response-Software, die Sicherheitsteams bei der effektiven Analyse und Bewältigung von Sicherheitsvorfällen unterstützt.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation und Hintergrund 1

1.2 Ziele 2

2 Grundlagen **4**

2.1 Das OSI Modell 4

2.1.1 Der Ablauf einer Datenübertragung nach dem OSI Modell 7

2.2 Einführung in Netzwerksicherheit und Incident Response 9

2.2.1 Netzwerksicherheit 9

2.3 Einführung in Nmap 9

2.3.1 Funktionalität von Nmap 9

2.3.2 Interaktion von Nmap mit dem OSI-Modell 10

2.3.3 Nutzen von Nmap für Incident-Response-Teams 11

2.3.4 Zusammenfassung 11

2.4 Einführung in Enterprise Netzwerke 12

2.4.1 Hauptkomponenten eines Enterprise Netzwerks 12

2.4.2 Zusammenfassung 13

3 Anforderungsanalyse **14**

3.1 Funktionale Anforderungen 15

3.2 Nicht-funktionale Anforderungen 16

4 Konzept und Design **17**

4.1 Architektur der Anwendung 17

4.2 Auswahl der Technologien und Frameworks 17

4.3 Design der Benutzeroberfläche und Visualisierungskonzepte 18

5 Implementierung **19**
5.1 Umsetzung der Funktionalitäten basierend auf den Anforderungen . . . 19
5.1.1 Bundling aller Komponenten mit Go . . . 19
5.1.2 Integration des HTTP-Routers und der Benutzeroberfläche . . . 20
5.1.3 Visualisierung und Speicherung der Netzwerkscans . . . 20
5.1.4 Zusammenfassung . . . 21
5.2 Integration von Nmap in die Anwendung . . . 21
5.2.1 Verwendung des Go Nmap Packages . . . 21
5.3 Integration von WebSSH in die Anwendung . . . 21
5.3.1 Warum tty2web? . . . 22
5.3.2 Implementierungsschritte . . . 22
5.4 Entwicklung der visuellen Oberfläche . . . 24
5.4.1 Verwendete Technologien . . . 25
5.4.2 Komponenten der Benutzeroberfläche . . . 25
5.4.3 Integration mit dem Backend . . . 26

6 Evaluation und Beispiele **27**
6.1 Durchführung von Tests . . . 27
6.2 Identifizierung von Verbesserungspotenzialen und Anpassungsempfehlungen . . . 28
6.3 Fazit . . . 28
6.3.1 Erreichte Ziele . . . 28
6.3.2 Evaluation der Anwendung . . . 29
6.3.3 Identifizierte Verbesserungspotenziale . . . 29
6.3.4 Praktische Anwendungsfälle . . . 30

A Anhang **33**
A.1 Quellcode-Beispiele . . . 33
A.1.1 Tty2Web Wrapper . . . 33
A.1.2 YAGLL - Yet Another Go Logging Library . . . 34
A.2 Erweiterte Konfigurationsmöglichkeiten . . . 35
A.3 Dokumentation . . . 35
A.4 Repositorys . . . 36

B Abbildungsverzeichnis **37**

C Tabellenverzeichnis **38**

D Literatur **39**

E Abkürzungsverzeichnis **41**

1 Einleitung

1.1 Motivation und Hintergrund

Angesichts der zunehmenden Raffinesse und Häufigkeit von Cyberangriffen ist eine schnelle und präzise Reaktion auf Sicherheitsvorfälle entscheidend für den Schutz von Unternehmensnetzwerken und sensiblen Daten. Als mögliche Lösung dieses Problems habe ich im Rahmen meines Projektkurses Informatik eine Incident-Response Tool zur grafischen Darstellung von Netzwerkscans entwickelt.

Das Hauptziel dieses Projektes besteht darin, eine intuitive Plattform zu entwickeln, die es Incident Response (IR) Teams erleichtert, Netzwerkscans durchzuführen und die gewonnenen Daten grafisch anschaulich darzustellen. Durch die Verwendung von Nmap können umfangende Netzwerkscans mit Informationen über verbundene Geräte, offene Ports und potenzielle Sicherheitsrisiken erstellt werden.

Die Entwicklung einer benutzerfreundlichen visuellen Oberfläche, die auf den Ergebnissen von Nmap basiert, ermöglicht es IR Teams, komplexe Netzwerkdaten schnell zu analysieren und fundierte Entscheidungen zur Reaktion auf Sicherheitsvorfälle zu treffen. Diese App ist darauf ausgerichtet, die Effizienz und Effektivität von Incident-Response-Prozessen zu verbessern und damit die Sicherheit von Unternehmensnetzwerken nachhaltig zu stärken.

Im Verlauf dieser Projektarbeit werde ich den Fokus auf die spezifischen Anforderungen von Incident-Response-Teams legen und die Entwicklung sowie Implementierung der visuellen Netzwerk-Scanning-Oberfläche detailliert beschreiben. Zudem werden mögliche Anwendungsfälle und potenzielle Erweiterungen diskutiert, um die Funktionalität und Relevanz dieser Lösung für Incident-Response-Teams weiter zu optimieren.

1.2 Ziele

Die Ziele der vorliegenden Arbeit lassen sich wie folgt gliedern:

1. **Entwicklung einer benutzerfreundlichen visuellen Darstellung von Netzwerkscans zur Verbesserung der Erfassung und Analyse von Netzwerken:**
    - Ziel ist es, eine intuitive und ansprechende Benutzeroberfläche zu schaffen, die es ermöglicht, Netzwerkscans visuell darzustellen.
    - Durch die grafische Darstellung der Netzwerke sollen Benutzer leichter und schneller ein umfassendes Verständnis der Netzwerkstruktur gewinnen können.
    - Dies umfasst die Visualisierung von Netzwerkkomponenten und Verbindungen.
2. **Implementierung einer übersichtlichen Archivierung und Darstellung aller vergangenen Scans:**
    - Es soll eine Funktion entwickelt werden, die alle durchgeführten Netzwerkscans in einer strukturierten und leicht zugänglichen Weise archiviert.
    - Diese Archivierung ermöglicht es Administratoren, frühere Scans schnell und einfach abzurufen und zu vergleichen.
    - Durch die Übersichtlichkeit und Struktur der Archivierung soll eine effiziente Nachverfolgung und Analyse von Netzwerkveränderungen gewährleistet werden.
3. **Erstellung einer Web-based Secure Shell (WebSSH) Oberfläche zur direkten und schnellen Problemlösung im Netzwerk, um den Administratoren eine effiziente Fernwartung zu ermöglichen:**
    - Ziel ist es, eine Web-Oberfläche zu entwickeln, die es ermöglicht, SSH-Verbindungen direkt über einen Webbrowser herzustellen.
    - Diese WebSSH Oberfläche soll Benutzern die Möglichkeit bieten, schnell und effizient auf Netzwerkgeräte zuzugreifen und Wartungsarbeiten durchzuführen, ohne auf zusätzliche SSH-Clients angewiesen zu sein.

- Durch die Integration in die vorhandene Benutzeroberfläche wird eine nahtlose und benutzerfreundliche Lösung angestrebt, die die Reaktionszeit bei Sicherheitsvorfällen und Netzwerkproblemen erheblich reduziert.

2 Grundlagen

2.1 Das OSI Modell

Das Open Systems Interconnection (OSI) Modell ist ein Referenzmodell, das Konventionen und Aufgaben der verschiedenen Hardware- und Softwarekomponenten definiert, um Entwicklern zu helfen, kompatible Systeme zu entwerfen[1]. Es besteht aus sieben Schichten, die jeweils spezifische Funktionen erfüllen, um den reibungslosen Ablauf der Datenübertragung zu gewährleisten.

Das OSI Modell hilft, indem es:

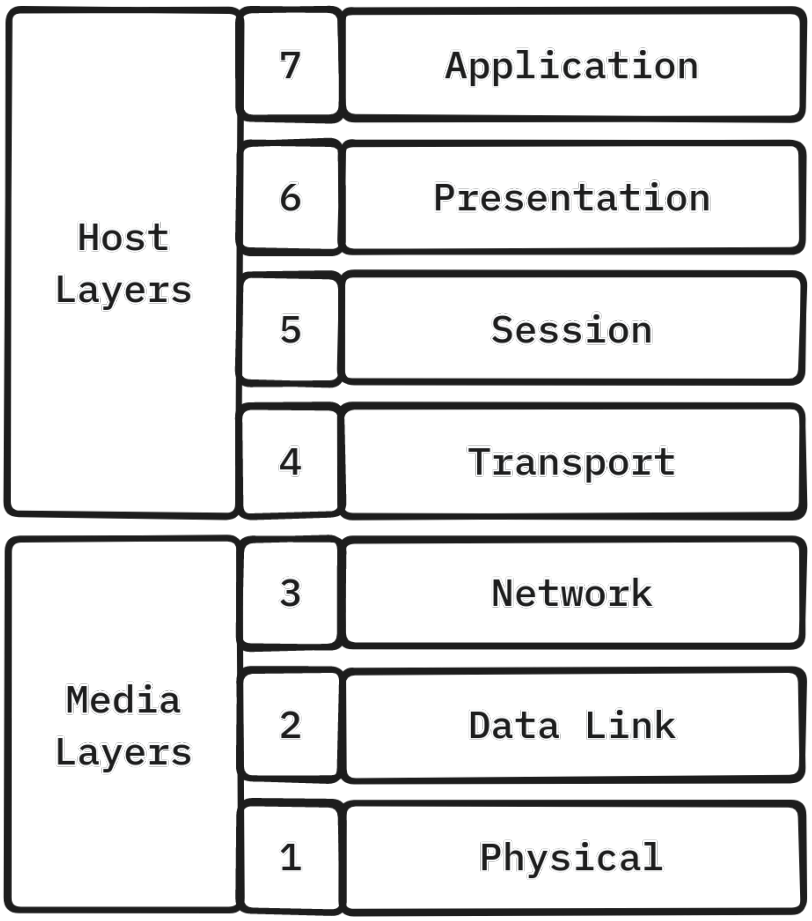
1. Standardisierung und Kompatibilität zwischen verschiedenen Netzwerksystemen ermöglicht
2. Entwickelnden einen strukturierten Ansatz zum Entwurf von Netzwerksystemen bietet
3. Fehlersuche und Problemlösung in Netzwerken vereinfacht, da jede Ebene klar definierte Aufgaben und Schnittstellen hat

Der Aufbau des OSI Modells nach Empfehlung X.200[2] wird in der Darstellung 2.1 gezeigt.

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

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

Abbildung 2.1: Das 7 Layer OSI Modell



Die 7 Ebenen sind wie folgt definiert:

1. **Physische Ebene** Die physische Ebene beschreibt die elektrischen und physischen Spezifikationen. Unter anderen sind hier Spannung, Kabelart, Pin-Layout und Host Bus Adapter (HBA) definiert.

2. **Data Link Ebene** Die Data Link Ebene ist für die Fehlererkennung und -korrektur auf der Bit-Ebene zuständig. Sie sorgt für die Zuverlässigkeit der Datenübertragung auf der physischen Ebene.

3. **Netzwerk Ebene** Die Netzwerkebene kümmert sich um die Weiterleitung von

Datenpaketen zwischen verschiedenen Netzwerken. Hier werden Routing und Adressierung durchgeführt.

4. **Transport Ebene** Die Transportebene ist für die End-to-End (E2E)-Kommunikation zwischen Anwendungen zuständig. Sie stellt sicher, dass Daten zuverlässig und in der richtigen Reihenfolge übertragen werden.

5. **Sitzungs Ebene** Die Sitzungsebene beendet, verwaltet und baut Sitzungen auf zwischen Anwendungen. Sie ermöglicht den Datenaustausch und die Synchronisierung zwischen den Kommunikationspartnern.

6. **Präsentations Ebene** Die Präsentationsebene kümmert sich um die Darstellung und Umwandlung von Datenformaten, um die Kompatibilität zwischen verschiedenen Systemen sicherzustellen.

7. **Software/Applications Ebene** Die Software/Applications Ebene beinhaltet die Anwendungen und Dienste, die direkt mit dem Benutzer interagieren. Hier werden Anwendungsprotokolle wie HTTP, FTP, SMTP usw. implementiert.

2.1.1 Der Ablauf einer Datenübertragung nach dem OSI Modell

Der Ablauf einer Datenübertragung im OSI Modell sieht wie folgt aus[3]:

Auf der Senderseite:

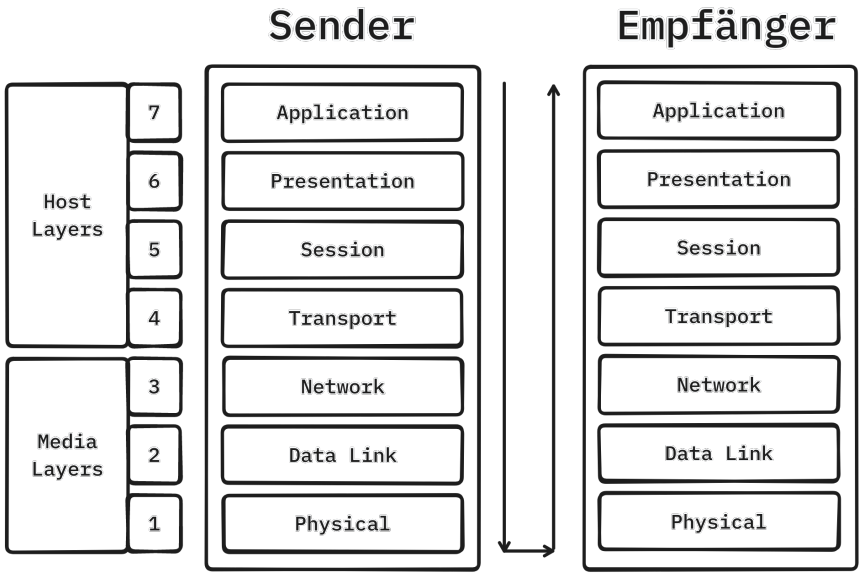
1. Die Anwendung auf Layer 7 erzeugt die zu übertragenden Daten und übergibt sie an die darunterliegenden Schichten.
2. Layer 6 (Präsentation) wandelt die Daten in ein standardisiertes Format um, um die Kompatibilität zum Empfänger sicherzustellen.
3. Layer 5 (Sitzung) etabliert und verwaltet die Kommunikationssitzung zwischen Sender und Empfänger.
4. Layer 4 (TCP oder UDP) teilt die Daten in Pakete auf, nummeriert sie und fügt zusätzliche Informationen hinzu, um eine zuverlässige und korrekte Übertragung zu gewährleisten.
5. Layer 3 (IP) adressiert die Pakete und leitet sie über das Local Area Network (LAN) und folgend Wide Area Network (WAN) zum Empfänger.
6. Layer 2 (Media Access Control (MAC)) verpackt die Pakete in Rahmen, fügt Prüfsummen hinzu und sendet sie über das physische Medium.
7. Layer 1 (Physisch) wandelt die digitalen Signale in elektrische oder optische Signale um und überträgt sie über das Netzwerkkabel oder andere Übertragungsmedien.

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

Auf der Empfängerseite:

1. Layer 1 (Physisch) empfängt die Signale vom Übertragungsmedium und wandelt sie in für die Hardware verwertbare Daten um.
2. Layer 2 (MAC) überprüft die Integrität der Rahmen und entfernt Prüfsummen.
3. Layer 3 (IP) überprüft die Adressierung und entfernt Routing-Informationen.
4. Layer 4 (TCP oder UDP) rekonstruiert die ursprünglichen Daten aus den Paketen und überprüft die Reihenfolge und Vollständigkeit.
5. Layer 5 (Sitzung) synchronisiert den Datenaustausch.
6. Layer 6 (Präsentation) wandelt die Daten in ein für die Anwendung lesbares Format um.
7. Layer 7 (Anwendung) empfängt die Daten und stellt sie dem Benutzer zur Verfügung.

Abbildung 2.2: Der Ablauf einer Datenübertragung nach dem OSI Modell



2.2 Einführung in Netzwerksicherheit und Incident Response

Heutzutage ist die Netzwerksicherheit von entscheidender Bedeutung, da es in der heutigen digitalen Welt immer schwieriger wird, diese Netzwerke zu schützen. Dieses Kapitel enthält die grundlegenden Konzepte und Praktiken zur Netzwerksicherheit und Incident Response (IR).

2.2.1 Netzwerksicherheit

Netzwerksicherheit befasst sich mit der Gewährleistung der Vertraulichkeit, Integrität und Verfügbarkeit von Daten und Ressourcen in einem Netzwerk.[4] Diese Hauptziele soll das Ergebnis dieser Arbeit erreichen:

1. **Vertraulichkeit**: Die Daten sollen nur von autorisierten Benutzern betrachtet werden können.
2. **Integrität**: Die Daten sollten vor unbefugten Änderungen geschützt sein, um sicherzustellen, dass sie korrekt und unverändert bleiben.
3. **Verfügbarkeit**: Die Daten sollen so schnell und einfach zu verstehen wie möglich bereitgestellt werden.

2.3 Einführung in Nmap

Nmap, kurz für „Network Mapper“, ist ein unverzichtbares Open-Source-Tool, das weltweit von Netzwerkadministratoren und Sicherheitsexperten eingesetzt wird, um Netzwerke zu analysieren und Sicherheitslücken zu identifizieren. Im Rahmen meines Projektes spielt Nmap eine zentrale Rolle bei der Durchführung von Netzwerkscans und der Erfassung detaillierter Informationen über das Netzwerk.

2.3.1 Funktionalität von Nmap

Die Hauptfunktionen von Nmap umfassen:

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

- **Port-Scanning**: Nmap identifiziert offene Ports auf einem Host, was essenziell ist, um laufende Dienste und potenzielle Sicherheitslücken zu erkennen. Diese Funktion ermöglicht es, einen Überblick darüber zu erhalten, welche Anwendungen und Dienste auf den Netzwerkgeräten aktiv sind.
- **Service-Erkennung**: Neben der Erkennung offener Ports kann Nmap die dahinter laufenden Dienste und deren Versionen identifizieren. Dies ist besonders nützlich, um veraltete oder anfällige Dienste zu erkennen, die ein Sicherheitsrisiko darstellen könnten.
- **Betriebssystemerkennung**: Durch die Analyse von Netzwerkverkehrsmustern kann Nmap das Betriebssystem eines Zielhosts ermitteln. Diese Information ist wertvoll, um gezielte Sicherheitsmaßnahmen und Schwachstellenanalysen durchzuführen.
- **Nmap Scripting Engine (NSE)**: Mit der NSE können spezifische Prüfungen und Tests automatisiert werden, indem benutzerdefinierte Skripte erstellt oder vorhandene Skripte genutzt werden.

2.3.2 Interaktion von Nmap mit dem OSI-Modell

Nmap arbeitet auf mehreren Schichten des OSI-Modells, um umfassende Netzwerkscans durchzuführen:

- **Data Link Layer**: Nmap kann Address Resolution Protocol (ARP)-Scans nutzen, um IP-Adressen aufzulösen und herauszufinden, welche Geräte im Netzwerk aktiv sind. Diese Scans sind besonders in lokalen Netzwerken nützlich.
- **Network Layer**: Nmap verwendet Internet Control Message Protocol (ICMP)-Pakete für Ping-Scans, um herauszufinden, ob Hosts aktiv sind und erreichbar sind.
- **Transport Layer**: Nmap führt Port-Scans durch, indem es TCP- und UDP-Pakete sendet, um offene Ports zu identifizieren und festzustellen, welche Dienste auf diesen Ports laufen.

- **Application Layer**: Nmap kann mithilfe von NSE-Skripten und Service-Erkennungstechniken auf Anwendungsebene arbeiten, um spezifische Dienste zu identifizieren und Schwachstellen in diesen Diensten zu finden.

2.3.3 Nutzen von Nmap für Incident-Response-Teams

Die Verwendung von Nmap bietet erhebliche Vorteile für Incident-Response-Teams:

- **Umfassende Netzwerkanalyse**: Durch die Fähigkeit, detaillierte Informationen über Netzwerke und deren Hosts zu sammeln, ermöglicht Nmap eine gründliche Analyse des Netzwerkstatus. Dies ist entscheidend für die frühzeitige Erkennung und Behebung von Sicherheitslücken.
- **Früherkennung von Sicherheitslücken**: Nmap hilft dabei, Schwachstellen wie veraltete oder anfällige Dienste frühzeitig zu identifizieren. So können präventive Maßnahmen ergriffen werden, bevor potenzielle Angreifer diese Lücken ausnutzen.
- **Flexibilität und Vielseitigkeit**: Nmap unterstützt verschiedene Scanning-Techniken und -Protokolle und kann somit in unterschiedlichen Netzwerkszenarien eingesetzt werden. Diese Vielseitigkeit macht es zu einem wertvollen Werkzeug für die Netzwerksicherheit.

2.3.4 Zusammenfassung

Nmap ist ein essenzielles Werkzeug für die Netzwerksicherheit, das durch seine umfassenden Funktionen zur Netzwerkanalyse und Schwachstellenidentifikation beiträgt. Die detaillierten Informationen, die Nmap über Netzwerke und deren Hosts liefert, unterstützen Incident-Response-Teams dabei, Sicherheitsvorfälle effizient zu erkennen und zu beheben. Durch die Interaktion mit verschiedenen Schichten des OSI-Modells kann Nmap eine breite Palette von Netzwerkscans durchführen und somit einen wichtigen Beitrag zur proaktiven und effektiven Sicherheitsstrategie von Unternehmen leisten.

2.4 Einführung in Enterprise Netzwerke

Enterprise Netzwerke sind komplexe Infrastrukturen, die darauf ausgelegt sind, die Kommunikations - und Datenverarbeitungsanforderungen großer Organisationen zu unterstützen. Diese Netzwerke umfassen verschiedene Komponenten und Sicherheitsmechanismen, die gemeinsam für einen reibungslosen und sicheren Betrieb sorgen.

2.4.1 Hauptkomponenten eines Enterprise Netzwerks

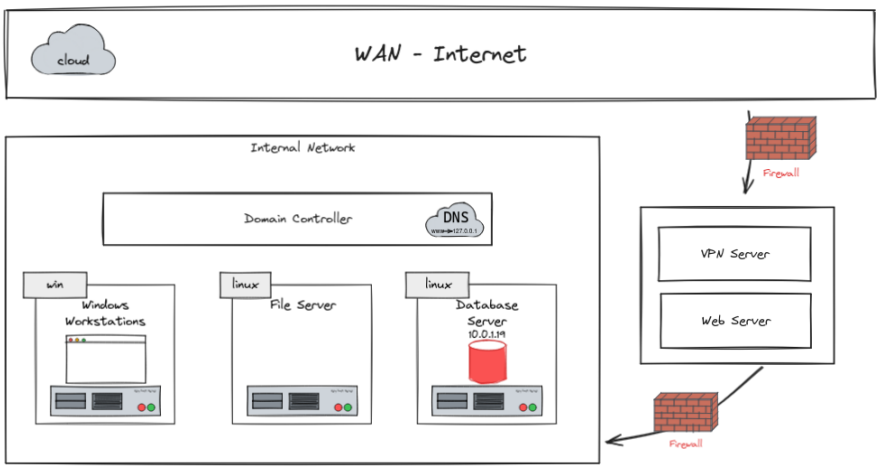
Ein typisches Enterprise Netzwerk besteht aus mehreren wichtigen Komponenten, die jeweils eine spezifische Rolle im Netzwerk erfüllen[5]:

- **Firewalls**: Firewalls sind essenziell für den Schutz des Netzwerks vor unbefugtem Zugriff. Sie filtern den ein- und ausgehenden Netzwerkverkehr basierend auf vordefinierten Sicherheitsregeln und verhindern so, dass schädliche Daten in das Netzwerk gelangen oder vertrauliche Informationen das Netzwerk verlassen.
- **Clients**: Dies sind die Endgeräte wie Computer, Laptops, Tablets und Smartphones, die von den Benutzern genutzt werden, um auf Netzwerkressourcen und -dienste zuzugreifen. Jeder Client im Netzwerk muss korrekt konfiguriert und gesichert sein, um die Gesamtsicherheit des Netzwerks zu gewährleisten.
- **Demilitarized Zones (DMZs)**: Eine DMZ ist ein isolierter Netzwerkbereich, der als Pufferzone zwischen dem internen Netzwerk und externen Netzwerken, wie dem Internet, dient. In der DMZ werden öffentlich zugängliche Dienste wie Webserver, Mailserver und DNS-Server platziert, um zu verhindern, dass Angreifer direkten Zugriff auf das interne Netzwerk erhalten.
- **Virtual Private Networks (VPNs)**: VPNs ermöglichen sichere Verbindungen über das Internet, indem sie Daten verschlüsseln und die Kommunikation zwischen entfernten Benutzern und dem Unternehmensnetzwerk schützen. VPNs sind besonders wichtig für die Verbindung von Remote-Mitarbeitern und Außenstellen mit dem Hauptnetzwerk.

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

- **Switches und Router**: Switches und Router leiten den Datenverkehr innerhalb des Netzwerks und verbinden verschiedene Netzwerksegmente miteinander. Switches arbeiten auf der Data-Link-Schicht des OSI-Modells und verbinden Geräte innerhalb desselben Netzwerks, während Router auf der Netzwerkschicht arbeiten und den Datenverkehr zwischen unterschiedlichen Netzwerken weiterleiten.
- **Server**: Server sind zentrale Dienstleister im Netzwerk, die verschiedene Dienste und Anwendungen bereitstellen. Dies können Datei- und Druckserver, Datenbankserver, E-Mail-Server, Webserver und Anwendungsserver sein. Die Verfügbarkeit und Sicherheit dieser Server ist entscheidend für den Betrieb des Netzwerks.

Abbildung 2.3: Aufbau eines Enterprise Netzwerkes



2.4.2 Zusammenfassung

Ein Enterprise Netzwerk ist eine komplexe und vielschichtige Struktur, die aus verschiedenen Komponenten besteht, die zusammenarbeiten, um eine sichere und effiziente Kommunikation und Datenverarbeitung zu ermöglichen. Die Implementierung von Sicherheitsmaßnahmen und die Einhaltung von Best Practices sind unerlässlich, um das Netzwerk vor Bedrohungen zu schützen und einen reibungslosen Betrieb zu gewährleisten. Durch die Integration von Technologien wie Firewalls und VPNs können Unternehmen ihre Netzwerksicherheit verbessern.

3 Anforderungsanalyse

In diesem Kapitel werden die funktionalen und nicht-funktionalen Anforderungen basierend auf den Zielen beschrieben. Funktionale Anforderungen beziehen sich auf spezifische Funktionen und Verhaltensweisen des Systems, während nicht-funktionale Anforderungen die Qualität und Leistung des Systems betreffen.

3.1 Funktionale Anforderungen

Nr.	Anforderung	Kat.	Beschreibung
01	Server-basierter Netzwerkscan	MUSS	Der Server soll das Netzwerk auf Anfrage des Clients (Website) scannen.
02	Binary/Executable Paket	MUSS	Client und Server sollen in eine ausführbare Datei gepackt werden, die mit einem einzigen Befehl ausgeführt werden kann.
03	Konfigurierbarer Scan	MUSS	Der Scan soll konfigurierbar sein, um unterschiedliche Anforderungen zu erfüllen.
04	Manuelle Eintragung	SOLL	Möglichkeit, Maschinen manuell hinzuzufügen, die nicht erkannt wurden.
05	Automatische Datenanfrage	SOLL	Automatisches Anfragen von Daten bei Firewall-Loggern (wenn möglich).
06	WebSSH	MUSS	Möglichkeit, direkt aus dem Browser eine SSH-Verbindung zu einer Maschine mit offenem SSH-Port herzustellen.
07	Grafische Ansicht	MUSS	Übersichtliche Darstellung des gescannten Netzwerks.
08	Scan History	MUSS	Eine Übersicht zu allen früheren Scans.

Tabelle 3.1: Funktionale Anforderungen an die Netzwerk-Scanning-Oberfläche

3.2 Nicht-funktionale Anforderungen

Nr.	Anforderung	Kat.	Beschreibung
09	Notizen zu Maschinen	SOLL	Möglichkeit, Notizen zu jeder Maschine hinzuzufügen.
10	Performance	MUSS	Der Scan muss innerhalb von akzeptablen Zeiträumen durchgeführt werden, insbesondere bei schnellen Scans.
11	Benutzerfreundlichkeit	SOLL	Die Benutzeroberfläche muss intuitiv und leicht bedienbar sein.
12	Sicherheit	SOLL	Alle Datenübertragungen müssen verschlüsselt erfolgen.
13	Robustheit	SOLL	Das System muss robust und fehlerresistent sein, um auch bei hoher Last stabil zu bleiben.
04	Dokumentation	SOLL	Anleitung zur Nutzung des Tools.

Tabelle 3.2: Nicht-funktionale Anforderungen

4 Konzept und Design

4.1 Architektur der Anwendung

Die Architektur der entwickelten Anwendung legt das grundlegende Gerüst fest, auf dem alle Funktionen aufbauen. Sie bestimmt die Struktur, wie die verschiedenen Komponenten miteinander interagieren und kommunizieren.

Für die Architektur wurde eine einfach zu erweiternde Struktur gewählt, um Flexibilität in der Entwicklung sicherzustellen. Die Anwendung besteht aus folgenden Hauptkomponenten:

- **Backend**: Das Backend der Anwendung umfasst die Kernlogik sowie die Datenverarbeitungsfunktionen. Hier werden Netzwerkscans durchgeführt, Daten verarbeitet und für die Benutzeroberfläche aufbereitet sowie für weitere Benutzung gespeichert.
- **Frontend**: Die Benutzeroberfläche ist die Schnittstelle, über die die Benutzer mit der Anwendung interagieren. Hier werden die Ergebnisse der Netzwerkscans visuell präsentiert.
- **Datenspeicher**: Die Daten der aller Netzwerkscans werden komprimiert in dem Ordner der Anwendung gespeichert und beim erneuten Starten eingelesen.

4.2 Auswahl der Technologien und Frameworks

Bei der Auswahl der Technologien und Frameworks für die Entwicklung der Anwendung wurden folgende Kriterien berücksichtigt: Leistung, Effizienz und Kompatibilität mit den Projektzielen. Die folgenden Technologien und Frameworks wurden ausgewählt:

- **Go**: Go(-lang) wurde als aufgrund seiner hohen Leistung, seiner einfachen Parallelität und seiner Effizienz als Programmiersprache ausgewählt. Golang bietet eine sehr einfache Parallelisierung und leicht verständlichen Syntax.
- **htmx**: htmx ist ein leistungsstarkes Frontend-Framework, dass es ermöglicht den Inhalt einer Website zu ändern, ohne JavaScript zu schreiben oder die Seite neu zu laden.
- **Tailwind CSS**: Tailwind CSS wurde als CSS-Framework ausgewählt, aufgrund seiner Einfachheit und Flexibilität bei der Gestaltung von Benutzeroberflächen.
- **httprouter von Julien Schmidt**: httprouter ist ein einfach zu implementierender HTTP-Router für Golang, der für hohe Geschwindigkeit und minimale Overhead optimiert wurde.

4.3 Design der Benutzeroberfläche und Visualisierungskonzepte

Das Design der Benutzeroberfläche wurde unter Berücksichtigung von Benutzerfreundlichkeit, Ästhetik und Funktionalität entwickelt.

Die Benutzeroberfläche bietet verschiedene Funktionen und Optionen zur Visualisierung und Analyse von Netzwerkscans, darunter:

- **Visualisierung von Netzwerken**: Die Benutzeroberfläche stellt Netzwerke in Form von Diagrammen dar, um die Beziehungen zwischen den verschiedenen Netzwerkgeräten und -komponenten zu veranschaulichen.
- **Filterfunktion**: Benutzer können Daten nach bestimmten Kriterien filtern, um relevante Informationen schnell zu finden und zu analysieren.
- **Interaktive Elemente**: Die Benutzeroberfläche bietet interaktive Elemente wie Schaltflächen, Dropdown-Menüs und Schieberegler, um Benutzern die Interaktion mit den Daten zu erleichtern und benutzerdefinierte Analysen durchzuführen.

5 Implementierung

5.1 Umsetzung der Funktionalitäten basierend auf den Anforderungen

Die Implementierung der Anwendung basiert auf den zuvor definierten Anforderungen und folgt einer klaren Struktur, um die Effizienz und Erweiterbarkeit sicherzustellen. In diesem Abschnitt werden die wichtigsten Aspekte der Implementierung beschrieben, einschließlich der verwendeten Technologien und der Integration der verschiedenen Komponenten.

5.1.1 Bundling aller Komponenten mit Go

Ein zentrales Ziel der Implementierung war es, alle notwendigen Komponenten in eine ausführbare Datei zu bündeln. So werden externe Go-Bibliotheken mit in dieselbe ausführbare Datei gebündelt, um die Installation zu vereinfachen. Dies ermöglicht eine einfache Bereitstellung und Nutzung der Anwendung ohne zusätzliche Installationsschritte.

Das einzige Feature, das nicht standardmäßig enthalten ist, ist WebSSH. Um dies zu nutzen, wird das externe Programm tty2web[1] benötigt. Mit tty2web ist es möglich, eine Website zu erstellen, über die man mittels des SSH-Protokolls auf andere Computer zugreifen kann.

- **Einbindung aller Abhängigkeiten**: Alle benötigten Bibliotheken und Pakete werden in das Projekt integriert.
- **Kompilierung**: Die Anwendung wird so kompiliert, dass alle Pakete in der ausführbaren Datei enthalten sind.

[1] kost, *kost/tty2web*.

- **Bereitstellung**: Die erstellte ausführbare Datei kann direkt auf den Zielsystemen ausgeführt werden, ohne dass zusätzliche Installationen erforderlich sind (mit Ausnahme des optionalen tty2web).

5.1.2 Integration des HTTP-Routers und der Benutzeroberfläche

Für die Webschnittstelle und die Benutzerinteraktionen habe ich den HTTP-Router von Julien Schmidt verwendet[2]. Die wichtigsten Schritte zur Integration des Routers sind:

- **Routing-Konfiguration**: Definition der verschiedenen Routen und Endpunkte, über die Benutzer mit der Anwendung interagieren können.
- **Handler-Implementierung**: Implementierung der Handler-Funktionen, die auf die Anfragen der Benutzer reagieren und die entsprechenden Aktionen ausführen.
- **Einbindung der Benutzeroberfläche**: Integration der Frontend-Komponenten, um eine interaktive und benutzerfreundliche Oberfläche bereitzustellen.

5.1.3 Visualisierung und Speicherung der Netzwerkscans

Die Ergebnisse der Netzwerkscans werden in komprimierter Form gespeichert und beim erneuten Starten der Anwendung eingelesen. Dies ermöglicht eine effiziente Nutzung der Speicherressourcen und eine schnelle Verfügbarkeit der Daten. Die wichtigsten Schritte zur Implementierung dieser Funktionalität sind:

- **Datenkomprimierung**: Die Ergebnisse der Netzwerkscans werden komprimiert und in einem speziellen Verzeichnis der Anwendung gespeichert.
- **Datenwiederherstellung**: Beim Starten der Anwendung werden die gespeicherten Daten eingelesen und für die Benutzeroberfläche aufbereitet.
- **Visualisierung**: Die Daten werden in der Benutzeroberfläche visuell dargestellt, um eine einfache Analyse und Interpretation zu ermöglichen.

[2]Schmidt, *julienschmidt/httprouter*.

5.1.4 Zusammenfassung

Die Implementierung der Anwendung erfolgte unter Berücksichtigung der definierten Anforderungen und unter Verwendung effizienter und leistungsfähiger Technologien. Durch die Integration des Go Nmap Packages, das Bundling aller Komponenten und die Verwendung des `httprouter` konnte eine flexible, erweiterbare und benutzerfreundliche Lösung entwickelt werden, die Incident-Response-Teams bei der schnellen und präzisen Analyse von Netzwerkscans unterstützt.

5.2 Integration von Nmap in die Anwendung

5.2.1 Verwendung des Go Nmap Packages

Für die Netzwerkscans habe ich das Go Nmap Package[3] verwendet. Dieses Paket bietet eine einfache Möglichkeit, Nmap-Funktionen in Go zu nutzen. Dadurch konnten umfassende Netzwerkscans effizient und zuverlässig umgesetzt werden. Das Go Nmap Package bietet einen Wrapper für alle Funktionen, die man normalerweise in der Konsole benutzen kann, z.B. wird

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

zu

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

5.3 Integration von WebSSH in die Anwendung

Um eine WebSSH-Konsole in der Anwendung anzubieten, habe ich tty2web[4] integriert, was es ermöglicht, Terminal-Sitzungen über eine Website zugänglich zu machen. Dies

[3] Le Glaunec, *Ullaakut/nmap*.
[4] kost, *kost/tty2web*.

ist besonders nützlich, um schnell auf entfernte Maschinen zugreifen zu können, ohne eine separate Konsole öffnen zu müssen.

5.3.1 Warum tty2web?

`tty2web` ist einfach zu benutzen und kann extern installiert werden, da es ein optionales Feature sein soll.
Der Befehl `tty2web --permit-write --once ssh root@127.0.0.1` startet z. B. eine Website, die man einmal aufrufen kann, um eine SSH-Verbindung zum lokalen Host herzustellen, die nur einmal funktioniert.

5.3.2 Implementierungsschritte

Die Integration von `tty2web` in die Anwendung umfasste mehrere Schritte, um eine nahtlose und effiziente Nutzung zu gewährleisten. Hier sind die detaillierten Implementierungsschritte, die ich durchgeführt habe:

Erstellung des Wrappers `libtty2web`

Da das originale `tty2web` nicht als Paket verwendet werden kann, habe ich einen Wrapper namens `libtty2web`[5] geschrieben. Dieser Wrapper kann den Befehl zum Ausführen von `tty2web` generieren und ausführen, ohne dass der Benutzer selbst mit der `os/exec` Standard-Library interagieren muss.

Die Library funktioniert grundlegend so, dass man ein Tty2Web-Objekt erstellt und dann Optionen hinzufügt:

```
1 type Tty2Web struct {
2         options []option  // options to be passed to the binary
3         binary  string    // binary to be executed
4         process *exec.Cmd // process to be executed
5         command []string  // command to be executed
6 }
7
8 // AddOptions adds options to the Tty2Web instance
```

[5] Pfannenschmidt, *Lutz-Pfannenschmidt/libtty2web*.

```
9  func (t *Tty2Web) AddOptions(option ...option) {
10         t.options = append(t.options, option...)
11 }
12
13 // Run executes the command
14 func (t *Tty2Web) Run() error {
15         t.process = exec.Command(t.binary, t.buildCommand()...)
16         err := t.process.Run()
17         if err != nil {
18                 return err
19         }
20         return nil
21 }
```

Optionen werden mit den Helferfunktionen wie

`libtty2web.With<Option>(<Configuration>)`

hinzugefügt und beim Ausführen berücksichtigt.

```
1  type option struct {
2          name     string
3          value    string
4          hasValue bool
5  }
6
7  // IP address to listen (default: "0.0.0.0")
8  func WithAddress(address string) option {
9          return option{
10         name: "address",
11         value: address,
12         hasValue: true,
13     }
14 }
15
16 // Port number to listen (default: "8080")
17 func WithPort(port string) option {
```

```
18          return option{
19          name: "port",
20          value: port,
21          hasValue: true,
22      }
23  }
```

Die Benutzung meines Pakets sieht dann folgendermaßen aus:

```
1  package main
2
3  import "github.com/Lutz-Pfannenschmidt/libtty2web"
4
5  func main() {
6           test := libtty2web.NewTty2Web("bash")
7           test.SetBinary("tty2web")
8           test.AddOptions(
9                   libtty2web.WithPermitWrite(),
10                  libtty2web.WithOnce(),
11          )
12          err := test.Run()
13          if err != nil {
14                  panic(err)
15          }
16          test.Wait()
17 }
```

Dieses Programm hostet eine Website, die man einmal aufrufen kann und ein Terminal mit bash emuliert.

5.4 Entwicklung der visuellen Oberfläche

Die visuelle Oberfläche der Anwendung wurde so gestaltet, dass sie benutzerfreundlich und intuitiv ist. Im Folgenden werden die wichtigsten Aspekte der Entwicklung der Benutzeroberfläche beschrieben, einschließlich der verwendeten Technologien.

5.4.1 Verwendete Technologien

Für die Entwicklung der Benutzeroberfläche habe ich moderne Webtechnologien verwendet, um eine reaktionsschnelle und ansprechende Benutzererfahrung zu gewährleisten. Dazu gehören:

- **HTML5**: Zur Strukturierung der Inhalte und zur Gestaltung der grundlegenden Seitenarchitektur.
- **HTMX**: Ein Framework, das es ermöglicht, Inhalte der Website zu ändern, ohne JavaScript zu schreiben oder die Seite neu zu laden. Dies erleichtert die Erstellung dynamischer und interaktiver Benutzeroberflächen.
- **Tailwind CSS**: Ein CSS-Framework, das für schnelles und flexibles Styling sorgt. Es ermöglicht eine einfache Anpassung der Benutzeroberfläche durch utility-first CSS-Klassen.[6]

5.4.2 Komponenten der Benutzeroberfläche

Die Benutzeroberfläche besteht aus mehreren separaten Komponenten, die von meinem Backend gehostet werden:

- **Analytics**: Das zentrale Dashboard bietet eine Übersicht über alle wichtigen Informationen und Funktionen der Anwendung. Hier können Benutzer eine Übersicht über das Netzwerk machen.
- **Graph**: Diese Komponente visualisiert die Netzwerkstruktur und die Ergebnisse der Netzwerkscans in Form von Diagrammen und Graphen. Dies ermöglicht es Benutzern, schnell und einfach die Beziehungen zwischen den verschiedenen Netzwerkgeräten und -komponenten zu verstehen.
- **History**: Eine spezielle Ansicht zur Darstellung der historischen Daten der Netzwerkscans. Benutzer können vergangene Scans einsehen, vergleichen und analysieren, um Trends und Muster im Netzwerkverhalten zu erkennen.

[6] *Tailwind CSS.*

5.4.3 Integration mit dem Backend

Die Benutzeroberfläche kommuniziert über eine RESTful API mit dem Backend, das in Go implementiert ist. Diese API ermöglicht es, Daten zwischen dem Frontend und dem Backend auszutauschen und Aktionen wie das Starten von Netzwerkscans oder das Abrufen von Scan-Resultaten durchzuführen. Durch den Einsatz von HTMX können Daten asynchron abgerufen und aktualisiert werden, ohne dass die gesamte Seite neu geladen werden muss.

6 Evaluation und Beispiele

6.1 Durchführung von Tests

Die Tests der Anwendung wurden umfassend durchgeführt, um sicherzustellen, dass alle Funktionen wie erwartet arbeiten. Hierzu habe ich folgende Testmethoden verwendet:

- **Unit-Tests**: Golang hat ein eingebautes Testing-Framework, welches einfach zu benutzen ist. Hiermit werden einzelne Funktionen und Features getestet.
- **Manuelle Tests**: End-to-End-Tests wurden manuell durchgeführt, um die Benutzerfreundlichkeit und die gesamte Funktionalität der Anwendung zu überprüfen.

6.2 Identifizierung von Verbesserungspotenzialen und Anpassungsempfehlungen

Während der Tests und Bewertungen wurden mehrere Bereiche identifiziert, die verbessert werden können:

- **Erweiterung der Filterfunktionen**: Die bestehenden Filteroptionen könnten erweitert werden, um Benutzern eine noch präzisere Analyse der Netzwerkscans zu ermöglichen.
- **Verbesserung der Visualisierungen**: Zusätzliche Visualisierungstools und -optionen könnten hinzugefügt werden, um komplexe Netzwerkstrukturen noch besser darzustellen, z.B. Lucidchart[1]-ähnliche Tools.
- **Dokumentation**: Es gibt nur eine grobe Dokumentation auf der Homepage meiner Software[2], welche nur die Benutzung der Binary erklärt sowie alle Features darstellt, aber nicht erklärt. Hier könnte man noch erklären, wie mit den einzelnen Features interagiert werden kann.

6.3 Fazit

Im Rahmen dieser Arbeit wurde ein umfassendes Tool zur Analyse und Visualisierung von Netzwerkscans entwickelt, das sowohl die Benutzerfreundlichkeit als auch die Effizienz in der Netzwerksicherheit verbessert. Die Implementierung basierte auf den zuvor definierten Anforderungen und umfasste mehrere zentrale Komponenten, darunter die Integration von Nmap, die Entwicklung einer benutzerfreundlichen Web-Oberfläche und die Bereitstellung einer Web-based Secure Shell (WebSSH) Konsole.

6.3.1 Erreichte Ziele

Die wesentlichen Ziele der Arbeit wurden erfolgreich umgesetzt:

- **Benutzerfreundliche visuelle Darstellung von Netzwerkscans**: Die Anwendung bietet eine klare und intuitive visuelle Oberfläche zur Anzeige der Netz-

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

werktopologie und der Scanergebnisse. Diese visuelle Darstellung erleichtert die Erfassung und Analyse der Netzwerkinformationen erheblich.

- **Effiziente Archivierung und Darstellung vergangener Scans**: Alle durchgeführten Netzwerkscans werden komprimiert archiviert und können jederzeit wieder abgerufen werden. Dies ermöglicht eine effiziente Nachverfolgung und Analyse der Netzwerkentwicklung über die Zeit.
- **WebSSH Oberfläche zur Fernwartung**: Die Integration einer WebSSH Oberfläche ermöglicht es Administratoren, schnell und einfach auf entfernte Maschinen zuzugreifen und Probleme direkt zu beheben. Dies verbessert die Reaktionszeit und Effizienz bei der Verwaltung und Wartung des Netzwerks.

6.3.2 Evaluation der Anwendung

Die Anwendung wurde in verschiedenen Szenarien getestet, um ihre Funktionalität und Leistung zu überprüfen. Die Tests haben gezeigt, dass die Anwendung stabil und zuverlässig arbeitet und die definierten Anforderungen erfüllt. Die Benutzeroberfläche erwies sich als intuitiv und benutzerfreundlich, und die Integration der verschiedenen Komponenten verlief reibungslos.

6.3.3 Identifizierte Verbesserungspotenziale

Trotz der erfolgreichen Umsetzung gibt es einige Bereiche, in denen zukünftige Arbeiten ansetzen können:

- **Erweiterung der Visualisierungsoptionen**: Die Einführung zusätzlicher Visualisierungsoptionen könnte die Analyse der Netzwerktopologie und der Scanergebnisse weiter verbessern.[3]
- **Erweiterte Sicherheitsfunktionen**: Die Implementierung zusätzlicher Sicherheitsfunktionen, wie z.B. die automatische Erkennung und Benachrichtigung bei verdächtigen Aktivitäten, könnte den Nutzen der Anwendung erhöhen.

[3] *Lucidchart*.

- **Optimierung der Performance**: Die Performance der Anwendung, insbesondere bei sehr großen Netzwerken, könnte weiter optimiert werden, um noch schnellere Scan- und Verarbeitungszeiten zu erreichen.

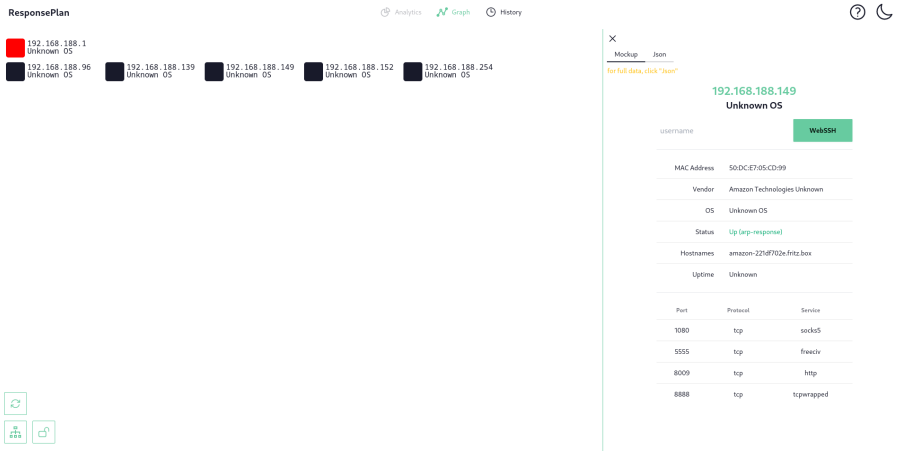
6.3.4 Praktische Anwendungsfälle

Die entwickelte Anwendung bietet zahlreiche praktische Anwendungsfälle für Incident-Response-Teams:

- **Schnelle Identifikation von Sicherheitslücken**: Durch die detaillierte Analyse der Netzwerkscans können Sicherheitslücken frühzeitig erkannt und behoben werden.
- **Effiziente Netzwerkwartung**: Die WebSSH Oberfläche ermöglicht eine schnelle und effiziente Fernwartung, was die Reaktionszeit bei Netzwerkproblemen verkürzt.
- **Überwachung der Netzwerkentwicklung**: Durch die Archivierung und Analyse vergangener Scans kann die Entwicklung des Netzwerks über die Zeit überwacht und analysiert werden, was eine proaktive Netzwerksicherheit unterstützt.

Insgesamt hat diese Arbeit gezeigt, dass die entwickelte Anwendung einen Beitrag zur Verbesserung der Netzwerksicherheit und -wartung leisten kann. Die Kombination aus leistungsfähigen Netzwerkscans, einer benutzerfreundlichen Oberfläche bietet Incident-Response-Teams ein Werkzeug zur effektiven und proaktiven Verwaltung von Netzwerksicherheitsproblemen.

Abbildung 6.1: Scan meines privaten Netzwerkes ohne Betriebssystemerkennung.



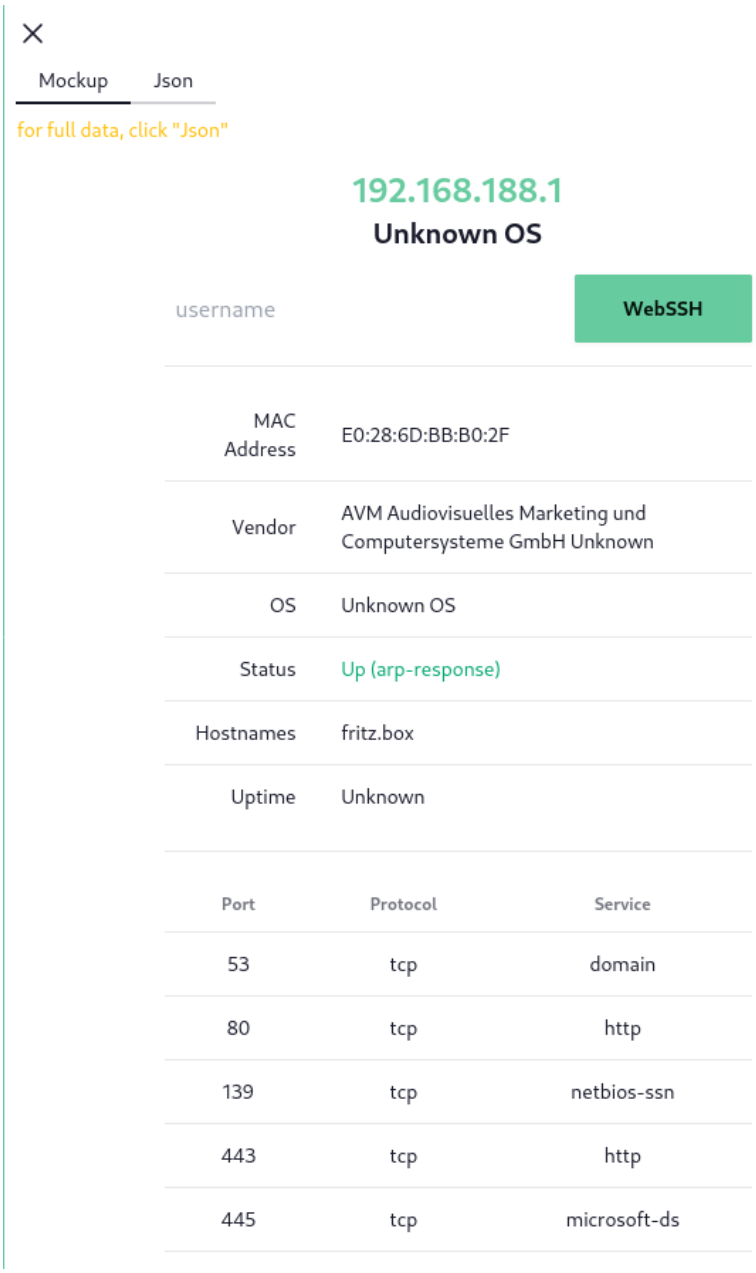


Abbildung 6.2: Übersicht einer FritzBox!

A Anhang

A.1 Quellcode-Beispiele

A.1.1 Tty2Web Wrapper

Der folgende Quellcode zeigt die Benutzung des `libtty2web` Wrappers, der die Nutzung des `tty2web` Binaries erleichtert:

```
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()
}
```

A.1.2 YAGLL - Yet Another Go Logging Library

Außerdem habe ich meine eigene Logging-Bibliothek geschrieben, da ich eine individuelle Lösung für die Protokollierung in meinem Projekt benötigte. Diese Bibliothek, die ich YAGLL - Yet Another Go Logging Library genannt habe, basiert auf den Anforderungen meiner Software und ist stark von Distillog inspiriert, bietet jedoch einige zusätzliche Funktionen und Anpassungsmöglichkeiten.

```
package main

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

func main() {
    yagll.Toggle(yagll.DEBUG, true)
    yagll.Debugln("Debug mode enabled")
    yagll.Infof("Dies ist eine %s Nachricht", "Info")
}
```

Funktionen

- **Anpassbare Protokollausgabe**: Die Protokollausgabe kann mit der Funktion `SetOutput` angepasst werden. Jede `os.File`-Instanz kann als Ausgabe verwendet werden.
- **Jedes Protokolllevel ein- und ausschalten**: Jedes Protokolllevel kann mit der Funktion `Toggle` ein- und ausgeschaltet werden.
- **Anpassbare Farben**: Die Farben, die für die Protokolllevel verwendet werden, können mit der Funktion `SetColor` angepasst werden.
- **Anpassbares Template mit text/template**: Das Template zur Formatierung der Protokollnachrichten ist vollständig anpassbar mit `text/template`. Die folgenden Variablen stehen zur Verfügung:
  - {{ `.Time` }} - Die Zeit, zu der die Protokollnachricht erstellt wurde
  - {{ `.Level` }} - Das Protokolllevel der Nachricht
  - {{ `.Message` }} - Die Nachricht selbst

  - {{ .Color }} - Die Farbe des Protokolllevels
  - {{ .Reset }} - Der Rücksetzcode, um die Farbe zurückzusetzen

A.2 Erweiterte Konfigurationsmöglichkeiten

Folgende Optionen können dem Programm übergeben werden, um sein Verhalten zu ändern:

- `-m` / `--memory` - Nur Speichermodus. Dies deaktiviert die Dateispeicherung.
- `-d` / `--debug` - Debug-Modus aktivieren. Dies druckt zusätzliche Informationen in die Konsole.
- `-h` / `--help` - Die Hilfe-Nachricht anzeigen.
- `-p` / `--port` <port>- Ändern Sie den Port, auf dem der Server hört. Standardmäßig ist dies 1337.
- `-e` / `--expose` - Den Server im LAN freigeben. Dies ermöglicht es anderen Geräten, eine Verbindung zum Server herzustellen. `[Achtung]`: Dies kann ein Sicherheitsrisiko darstellen.
- `-o` / `--out` <Dateiname>- Die Datei, in die die Daten gespeichert werden sollen. Standardmäßig ist dies `data.responseplan`.
- `-i` / `--in` <Dateiname>- Die Datei, aus der die Daten geladen werden sollen. Standardmäßig ist dies `data.responseplan`.
- `--tty2web` <Pfad>- Verwenden Sie einen benutzerdefinierten Pfad für das tty2web-Binary. Standardmäßig ist dies `tty2web`.

A.3 Dokumentation

Eine ausführliche Dokumentation und Übersicht der Features des Projekts finden Sie auf der Website `https://responseplan.de`.

A.4 Repositorys

Hier sind die Links zu meinen GitHub-Repositorys:

- **ResponsePlan**: Meine Hauptanwendung, ResponsePlan, ist auf GitHub unter folgendem Link verfügbar:
  `https://github.com/Lutz-Pfannenschmidt/ResponsePlan`
- **YAGLL - Yet Another Go Logging Library**: Die von mir entwickelte Logging-Bibliothek YAGLL ist unter folgendem Link verfügbar:
  `https://github.com/Lutz-Pfannenschmidt/yagll`
- **libtty2web Wrapper**: Ein Wrapper für das tty2web-Tool, den ich entwickelt habe, ist unter folgendem Link verfügbar:
  `https://github.com/Lutz-Pfannenschmidt/libtty2web`
- **responseplan.de Dokumentation**: Die Dokumentation für ResponsePlan ist unter folgendem Link verfügbar:
  `https://github.com/Lutz-Pfannenschmidt/responseplan.de`

B Abbildungsverzeichnis

2.1 Das 7 Layer OSI Modell 5
2.2 Der Ablauf einer Datenübertragung nach dem OSI Modell 8
2.3 Aufbau eines Enterprise Netzwerkes 13

6.1 Scan meines privaten Netzwerkes ohne Betriebssystemerkennung. 31
6.2 Übersicht einer FritzBox! 32

C Tabellenverzeichnis

3.1 Funktionale Anforderungen an die Netzwerk-Scanning-Oberfläche . . . 15
3.2 Nicht-funktionale Anforderungen 16

D Literatur

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, Juli 1994. URL: https://www.itu.int/rec/T-REC-X.200-199407-I/en.

K. Sumit u. a.: THE OSI MODEL: OVERVIEW ON THE SEVEN LAYERS OF COMPUTER NETWORKS **OSI-MODEL**

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

Peterson u. a.: Computer Networks: A Systems Approach **ComputerNetworks**

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

kost: kost/tty2web **tty2web**

kost. *kost/tty2web*. 2024. URL: %5C%5Chttps://github.com/kost/tty2web (besucht am 05. 06. 2024).

Le Glaunec: Ullaakut/nmap **GoNmap**

Brendan Le Glaunec. *Ullaakut/nmap*. 2024. URL: https://github.com/Ullaakut/nmap (besucht am 05. 06. 2024).

Lucidchart **lucidchart**

Lucidchart. 2024. URL: https://www.lucidchart.com/ (besucht am 08. 06. 2024).

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

Lutz Pfannenschmidt. *Lutz-Pfannenschmidt/libtty2web*. 2024. URL: https://github.com/Lutz-Pfannenschmidt/libtty2web (besucht am 08. 06. 2024).

ResponsePlan **ResponsePlan**

ResponsePlan. 2024. URL: https://www.responseplan.de/ (besucht am 08. 06. 2024).

Schmidt: julienschmidt/httprouter **httprouter**

Julien Schmidt. *julienschmidt/httprouter*. 2024. URL: https://github.com/julienschmidt/httprouter (besucht am 05. 06. 2024).

Tailwind CSS **tailwind**

Tailwind CSS. 2024. URL: https://tailwindcss.com/ (besucht am 08. 06. 2024).

E Abkürzungsverzeichnis

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