Technische Hochschule Ingolstadt

Fakultät für Informatik

Studiengang Wirtschaftsinformatik

Bachelorarbeit

Entwurf und Implementierung eines Sicherheitsmonitors für Heimnetzwerke

Vorgelegt von: Alexander Horn

Erstprüfer: Prof. Dr.-Ing. Hans-Joachim Hof

Zweitprüfer: Prof. Dr.-Ing. Ernst-Heinrich Göldner

Ausgegeben: 18. Oktober 2019

Abgegeben: 16. Januar 2020

Inhaltsverzeichnis

Abkürzungsverzeichnis

ARP Address Resolution Protocol. 8, 22, 23, 34–36

BSI Bundesamt für Sicherheit in der Informationstechnik. 4, 10, 12, 15

CPE Customer Premises Equipment. 23

DDoS Distributed Denial of Service. 1, 10, 13

DHCP Dynamic Host Configuration Protocol. 27

DNS-SD DNS Service Discovery. 5, 29

ENISA European Union Agency for Cybersecurity. 11

IANA Internet Assigned Numbers Authority. 28

ICMP Internet Control Message Protocol. 21, 22

IDS Intrusion Detection System. 3

IEEE Institute of Electrical and Electronics Engineers. 4, 24

IoT Internet of Things. 1, 2, 4, 12, 13, 29, 47

IPS Intrusion Prevention System. 3

mDNS Multicast DNS. 5, 29

NAS Network Attached Storage. 14

NIST National Institute of Standards and Technology. 3

NSE Nmap Scripting Engine. 7

NTLM NT LAN Manager. 36, 37

OWASP Open Web Application Security Project. 14–16, 55

SMB Server Message Block. 7, 25, 36, 37, 43, 47

SNMP Simple Network Management Protocol. 8

SSDP Simple Service Discovery Protocol. 4, 23, 28

SSH Secure Shell. 30, 40, 47

ST Search Target. 28

TTL Time to Live. 26

UDA Universal Plug and Play (UPnP) Device Architecture. 4, 5, 23

UPnP Universal Plug and Play. V, 4, 5, 18, 23, 28, 47

USN Unique Service Name. 28

1 Einleitung

Das Internet verbreitet sich in deutschen Haushalten kontinuierlich immer weiter. Die Anzahl der Haushalte mit Internetzugang ist im Zeitraum von 2008 bis 2018 von 69% auf 90% gestiegen. Dabei nutzten 87% der deutschen Bevölkerung das Internet innerhalb eines Zeitraums von drei Monaten. [Sta19d, S. 54]

Gleichzeitig werden die klassischen Endgeräte, wie Laptops und Mobiltelefone, zunehmend durch intelligente Smart Home-Geräte ergänzt. Laut einem Bericht von Statista wird die Durchdringung deutscher Haushalte mit Smart Home Geräten von 20% im Jahre 2020 auf 37% im Jahre 2024 ansteigen [Sta]. Wie in Abschnitt 2.4 genauer erläutert wird, überschneidet sich der Begriff des Smart Home wesentlich mit dem des Internet of Things (IoT). Dabei stellen IoT-Geräte eine signifikante Angriffsfläche auf die Sicherheit von Netzwerken dar. Dies wird beispielsweise daran deutlich, dass rekordverdächtige Distributed Denial of Service (DDoS)-Angriffe mithilfe von Botnets, bestehend aus IoT-Geräten, durchgeführt wurden. [BI17, S. 76]

In einem Unternehmensumfeld wird versucht, die Gefahren durch den Einsatz von Netzwerkmonitoring zu kontrollieren. In einer für Cisco angefertigten Studie gaben 91% der IT- und IT-Sicherheits-Fachkräfte an, die Investitionen in Netzwerkmonitoring innerhalb der nächsten zwei Jahre erhöhen zu wollen [Olt16, S. 3, 11]. Vor diesem Hintergrund wird im Rahmen dieser Arbeit versucht, das Konzept des Netzwerkmonitorings auf die Anwendung in Heimnetzwerken zu übertragen. Dieses System soll speziell auf die Bedürfnisse und Eigenschaften von Heimnutzern ausgerichtet sein. Es wird in den folgenden Abschnitten zuerst entworfen und danach als Prototyp implementiert.

Zunächst werden in **Abschnitt 2** einige Grundlagen erläutert, die zum Verständnis der anderen Abschnitte notwendig sind. Dazu gehören Begriffsdefinitionen und Erläuterungen zu Technologien. In **Abschnitt 3** werden vergleichbare Soft- und Hardwarelösungen vorgestellt, die bereits als freie oder kommerzielle Software verfügbar sind. In **Abschnitt 4** wird die aktuelle Bedrohungslage in der Bundesrepublik Deutschland analysiert um eine fundierte Basis für die Anforderungen an das zu entwerfende System zu bilden. Im anschließenden **Abschnitt 5** werden die Anforderungen ausgearbeitet und formuliert. In **Abschnitt 6** werden die Anforderungen zu einer konkreten technischen Vorgehensweise umgesetzt, um im abschließenden **Abschnitt 7** als Prototyp implementiert zu werden. Im **Abschnitt 8** wird der Prototyp bezüglich der Erfüllung der aufgestellten Anforderungen analysiert, um die Ergebnisse im letzten **Abschnitt 9** zusammenzufassen und einen Ausblick auf die Zukunft und eine mögliche Weiterentwicklung zu ermöglichen.

2 Grundlagen

In diesem Abschnitt werden einige Grundlagen, die zum Verständnis der restlichen Arbeit benötigt werden, erläutert. Dabei werden nur Themen behandelt, die typischerweise nicht zu den Fachkenntnissen eines Wirtschaftsinformatikers gehören.

2.1 Verständnis eines Heimnetzwerks

Für den Begriff Heimnetzwerk lässt sich in einschlägiger Literatur keine einheitliche Definition finden. Das Oxford-Wörterbuch (auf Lexico.com) definiert den Begriff wie folgt:

> „A network of interconnected electronic devices set up within a home; especially a residential local area network, typically used to share an Internet connection and resources such as printers and files between several computers.“ [Lex]

Das Verständnis der PC Magazine Encyclopedia ist dagegen konkreter und versteht darunter ein (W)LAN, welches für File Sharing und Internetzugriff verwendet wird:

> „A home network comprises the computers and mobile devices that are connected for file sharing and Internet access. A wireless router is all the hardware that is necessary. A home network is a ‚local area network‘ (LAN) combined with a Wi-Fi hotspot.“ [The19]

Aus dem Begriff „Heimnetzwerk“ lässt sich ableiten, dass es sich um ein Netzwerk, welches sich zu Hause befindet, handelt. Ob dieses Netzwerk eine spezifische Technologie einsetzt oder einem spezifischen Einsatzzweck dient, hängt dagegen davon ab, welche Definition verwendet wird. Zu den Technologien, die zur Implementierung eines Heimnetzwerks genutzt werden können, gehören beispielsweise Ethernet, Wi-Fi (auch als WLAN bezeichnet), Bluetooth oder ZigBee[1].

Gemäß einer Studie des Statistischen Bundesamts nutzen 77% der deutschen Bevölkerung das Internet täglich oder fast täglich[2] [Sta19d, S. 54]. Aus diesem Grund lässt sich vermuten, dass der Zugriff auf das Internet einen zentralen Nutzen eines Heimnetzwerks darstellt. Dabei erfolgt die Anbindung dieser Netzwerke an das Internet in der Regel über Ethernet und Wi-Fi [KR17, S. 44 f.].

Für diese Arbeit wird der Begriff des Heimnetzwerks folglich als eine Menge von miteinander verbundenen Geräten definiert, die über das Ethernet- oder das Wi-Fi-Protokoll miteinander kommunizieren. Andere Protokolle werden nicht beachtet. Diese Geräte sind über ein Gateway mit dem Internet verbunden.

[1] ZigBee ist ein Standard für Mesh-Netzwerke mit geringen Datenraten, welcher beispielsweise für IoT-Geräte verwendet wird [KR17, S. 578; Bun]

[2] 87% nutzten das Internet in den letzten drei Monaten, davon 89% täglich oder fast täglich [Sta19d, S. 54]

2.2 Netzwerkmonitoring

Unter einem Netzwerkmonitoring-System versteht man ein System zur Überwachung der „verschiedenen Komponenten, Ereignisse und Protokolle eines Netzwerks sowie seiner bereitgestellten Services." Hierbei kann zwischen passivem und aktivem Monitoring unterschieden werden. Beim passiven Monitoring wird der im Netzwerk fließende Verkehr aus einer Beobachterperspektive analysiert, ohne dass aktiv in ihn eingegriffen wird. Bei aktivem Monitoring werden hingegen Pakete versendet und mit anderen Geräten interagiert, um zusätzliche Informationen zu gewinnen. [LD18]

Zu den Daten, die durch das Netzwerkmonitoring erfasst werden können, gehören zum einen die Topologie des Netzwerks, mit allen Hosts, ihren Standorten und ihren Adressen. Zum anderen können auch weitergehende Daten zu bestimmten Geräten erfasst werden, wie beispielsweise die Art des Gerätes oder die auf dem Gerät laufenden Anwendungen und Dienste. In einem Unternehmensumfeld dient das Monitoring der Prävention von und der Reaktion auf technische Probleme, der Optimierung des Netzwerks, der Erkennung von Sicherheitsproblemen sowie der Einhaltung von gesetzlichen Vorgaben. [NB09, S. 1]

Obwohl beim Netzwerkmonitoring auch einige Aspekte der IT-Sicherheit mit einfließen, sind diese von Intrusion Detection Systemen (IDS) zu unterscheiden, wie im folgenden Abschnitt 2.3 verdeutlicht wird. [NB09, S. 1]

2.3 Intrusion Detection Systeme (IDS)

Das National Institute of Standards and Technology (NIST) definiert ein IDS folgendermaßen:

> „A security service that monitors and analyzes network or system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner." [Sto+15, S. B-9]

Dabei ist ein IDS von einem Intrusion Prevention System (IPS) abzugrenzen. Ein IPS wird von der NIST im selben Bericht folgendermaßen definiert:

> „A system that can detect an intrusive activity and can also attempt to stop the activity, ideally before it reaches its targets." [Sto+15, S. B-9]

Ein IPS baut somit auf der Funktionalität eines IDS auf und erweitert diese um eine automatische Prävention von Angriffen.

Die Grenze zwischen einem IDS (beziehungsweise IPS) und einem Netzwerkmonitoring-System besteht dabei darin, dass ein Netzwerkmonitoring-System lediglich den aktuellen Status eines Netzwerks widerspiegeln soll, ohne einen besonderen Fokus auf sicherheitsrelevante Vorfälle zu legen. Dagegen soll ein IPS speziell der Erkennung und der Abwehr von Angriffen dienen. Nichtsdestotrotz können die Informationen aus einem Netzwerkmonitoring auch sicherheitsrelevante Informationen liefern, etwa wenn unbekannte Geräte oder Dienste erkannt werden. [NB09, S. 1]

2.4 Internet of Things (IoT)

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) versteht unter IoT-Geräten „‚intelligente' Gegenstände, die zusätzliche ‚smarte' Funktionen enthalten." [Bun] Diese sind oft (drahtlos) an das Internet angebunden und können über dieses erreicht werden, was nach dieser Definition allerdings kein zwingender Bestandteil des IoT ist. Dagegen setzt die Definition des Institute of Electrical and Electronics Engineers (IEEE) eine Verbindung ins Internet als integralen Bestandteil des IoT voraus. [Bun; IEE15, S. 72]

Dem IoT steht der Begriff des **Smart Home** nahe. Dieser beschreibt ein „informations- und sensortechnisch aufgerüstete[s], in sich selbst und nach außen hin vernetzte[s] Zuhause" [Ben19]. Das Smart Home setzt allerdings nicht zwingend eine Verbindung ins Internet voraus. Zur Kommunikation können auch Standards wie ZigBee oder andere Protokolle verwendet werden. Abhängig davon, welche Definition des IoT verwendet wird, kann also davon gesprochen werden, dass das Smart Home auf dem IoT aufbaut. [Ben19]

Ein Beispiel für ein Smart Home-Szenario wäre eine Wohnung, welche mit intelligenten, mit dem Internet vernetzten Elementen wie einer automatischen Heizungssteuerung, fernsteuerbaren Glühbirnen oder Überwachungskameras ausgestattet ist. [Ben19]

2.5 Universal Plug and Play (UPnP)

UPnP ist eine Gruppe von auf gängigen Technologien aufbauenden Standards zur Kommunikation zwischen Computern und anderen intelligenten Geräten in unadministrierten Netzwerken. [UPn15, S. 10]

Die UPnP Device Architecture (UDA) beschreibt eine Menge von Technologien für ein konfigurationsloses Netzwerk. UDA ermöglicht es, dass Geräte dynamisch Netzwerken beitreten, IP-Adressen beziehen, ihre eigenen Ressourcen im Netzwerk ankündigen und die Ressourcen anderer Netzwerkteilnehmer konsumieren können, ohne dass eine Konfiguration durch den Benutzer notwendig ist. [UPn15, S. 10]

Der UDA-Standard teilt das Vorgehen der Protokolle in sechs Schritte auf:

Schritt 0 (Addressing): Zuweisung von IP-Adressen. Dies geschieht wenn möglich über DHCP, ansonsten wird auf AutoIP (wird im UDA-Standard definiert) zurückgegriffen. [UPn15, S. 16]

Schritt 1 (Discovery): Ankündigung der im Netzwerk verfügbaren Dienste. Ein Gerät kündigt seine Dienste immer beim Beitritt eines Netzwerks in diesem an. Zusätzlich kann jederzeit eine erneute Auflistung der Dienste eines Gerätes oder eines Ressourcentypes angefordert werden. Hierfür dient das Simple Service Discovery Protocol (SSDP), welches in Abschnitt 6.4 genauer erläutert wird. [UPn15, S. 19, 22]

Schritt 2 (Description): Abruf der Eigenschaften und Fähigkeiten (Capabilities) eines Gerätes [UPn15, S. 43].

Schritt 3 (Control): Beschreibung des Aufrufs von Funktionen und des Abrufs von Werten auf den Geräten [UPn15, S. 72].

Schritt 4 (Eventing): Abonnieren und Bekanntgabe von Werten, wenn sie verändert werden [UPn15, S. 88].

Schritt 5 (Presentation): Bereitstellung einer HTML-basierten Benutzeroberfläche zur Kontrolle und Steuerung eines Gerätes [UPn15, S. 108].

2.6 Bonjour

Bonjour ist ein Standard von Apple, welcher zum Teil eine Alternative zu UPnP darstellt. Er ermöglicht, ähnlich zu UPnP, ein automatisches, konfigurationsloses Entdecken von Geräten und Diensten in einem Netzwerk. [App19]

Bonjour setzt sich aus einer Kombination von drei verschiedenen Technologien zusammen:

Link-local Adressing ermöglicht eine automatische Zuweisung von IP-Adressen, wenn kein DHCP-Server verfügbar ist (analog zu UDA Schritt 0).

Multicast DNS (mDNS) ermöglicht den Abruf und das Veröffentlichen von DNS-Records (beispielsweise Hostnamen) ohne einen lokalen DNS-Server. Hierfür werden DNS-Anfragen über Multicast an alle Netzwerkteilnehmer versendet und von diesen beantwortet.

DNS Service Discovery (DNS-SD) definiert ein Schema zur Entdeckung von Diensten in einem Netzwerk über das DNS (siehe Auflistung 2.1, analog zu UDA Schritt 1). Dies ist gemäß dem DNS-SD-Standard sowohl mit einem normalen DNS als auch mit mDNS möglich, wird bei Bonjour allerdings nur in Kombination mit mDNS eingesetzt.

[Che11, 5:27 - 10:46]

```
pi@raspberrypi:~ $ dig -p 5353 -t ptr _services._dns-sd._udp.local @224.0.0.251

[...]

;; QUESTION SECTION:
;_services._dns-sd._udp.local.  IN      PTR

;; ANSWER SECTION:
_services._dns-sd._udp.local. 10 IN     PTR     _nvstream_dbd._tcp.local.
```

Auflistung 2.1: Auflistung aller Dienste in einem Netzwerk über DNS-SD und mDNS (gekürzt, selbst durchgeführt) [CK13a; CK13b].

Die Kombination dieser Techniken erlaubt etwa ein automatisches Einrichten von Druckern, Kameras, Streaming-Boxen und Computern. Beispielsweise kann dem Benutzer eine Liste von Druckern präsentiert werden, die mithilfe von DNS-SD über mDNS abgerufen wird, ohne dass zu einem Zeitpunkt IP-Adressen oder Hostnamen eingegeben werden müssen. [Che11, 10:48 - 12:21]

2.7 Containerisierung

Containerisierung (Containerization) ist eine leichtgewichtigere Alternative zur Virtualisierung von Servern. Dabei wird eine Anwendung zusammen mit allen ihren Abhängigkeiten, die zur Ausführung benötigt werden (wie Programmbibliotheken oder Konfigurationsdateien) in einem Container gebündelt ausgeliefert. Dadurch können Fehler, die etwa durch verschiedene Laufzeitumgebungen beim Entwickler und beim Kunden entstehen, verhindert werden, da der Entwickler den Großteil der Umgebung selbst kontrolliert. [SUS19]

Während bei der Virtualisierung mehrere virtuelle Maschinen mit kompletten Betriebssystemen auf einem Host-PC virtualisiert werden, teilen sich Container den Betriebssystemkern mit dem Host-Betriebssystem. Im Container werden lediglich die Prozesse der Anwendung isoliert vom restlichen System ausgeführt. Da für den Host und alle Container nur eine Instanz des Betriebssystemkerns benötigt wird, ist dieses Vorgehen ressourcenschonender als Virtualisierung. Allerdings muss der Container dasselbe Betriebssystem wie der Host benutzen, beispielsweise sind Windows-Container auf Linux dadurch nicht möglich. [SUS19]

3 Verwandte Arbeiten

Dieser Abschnitt bietet eine Übersicht über eine Auswahl an freier Software und kommerzieller Produkte, die mit dem in dieser Arbeit entworfenen System vergleichbar sind. Die Aussagen stützen sich zum Großteil auf Angaben des Entwicklers beziehungsweise des Herstellers und sind somit mit Vorsicht zu genießen.

3.1 Freie Software

Nmap[1] ist laut Angaben des Projektinitiators der weltweit beliebteste Netzwerkscanner. Das Programm ist als freie und quelloffene Software erhältlich. Es unterstützt das Scannen von Netzwerkbereichen auf verbundene Hosts sowie das Scannen von Hosts auf verschiedene Parameter, wie offene Ports, die Betriebssystemversion und Anwendungsversionen. Eine Automatisierung und Erweiterung des Tools wird durch die Nmap Scripting Engine (NSE) ermöglicht. Mithilfe der NSE können Hosts etwa auf Sicherheitslücken (wie etwa EternalBlue[2]) oder auf Malware getestet werden. [Lyo10, S. xxi f., 12 f.]

Bei dem Programm handelt es sich um eine Kommandozeilenanwendung mit optionaler grafischer Oberfläche (siehe Abbildung A.2), welche entsprechendes Fachwissen zur Konfiguration und Auswertung der Ergebnisse voraussetzt. Außerdem müssen Scans manuell angestoßen werden, weswegen es sich nicht ohne Weiteres zur kontinuierlichen Überwachung eines Netzwerks eignet. [Lyo10, S. xxi ff.]

Der **Angry IP Scanner**[3] ist ebenfalls ein freier und quelloffener Netzwerkscanner. Er baut auf Java auf und bietet eine grafische Oberfläche. Das Programm kann IP-Bereiche auf verbundene Geräte scannen und offene Ports auf den Geräten auflisten. Ferner können weitere Informationen zu den Geräten, wie beispielsweise der Hostname, die MAC-Adresse oder Anwendungsversionen, ermittelt werden. Ähnlich wie Nmap eignet es sich nicht zur kontinuierlichen Überwachung, da Scans manuell angestoßen werden müssen. Allerdings ist das Interface weniger umfangreich (siehe Abbildung A.3) und somit für unerfahrene Benutzer etwas leichter zu verstehen. [Kek]

3.2 Kommerzielle Produkte

Fing[4] ist ein für iOS und Android verfügbares Netzwerk-Toolkit. Es ermöglicht dem Benutzer, die mit dem Netzwerk verbundenen Geräte aufzulisten und zu identifizieren. Es können IP- und MAC-

[1] `https://nmap.org/`
[2] Server Message Block (SMB)-Exploit, welcher zeitweise alle verbreiteten Versionen von Microsoft Windows betraf [Cal]
[3] `https://angryip.org/`
[4] `https://www.fing.com/products/fing-app`

Adresse, Hostname, Hersteller sowie Modell des Gerätes ermittelt werden. Geschwindigkeit und Latenz einer Internetverbindung können ebenfalls gemessen werden. Außerdem können automatische Sicherheits- und Gerätebenachrichtigungen an ein Smartphone oder eine E-Mail-Adresse versendet werden. Als Werkzeuge zur Fehlersuche stehen Portscans, Pings, Traceroutes und DNS-Abfragen zur Verfügung. [Fin]

Vom selben Unternehmen existiert mit der **Fingbox**[5] seit 2018 eine hardwarebasierte Ergänzung zur Smartphone-App. Dabei handelt es sich um ein Gerät, welches per Ethernet an den Router angeschlossen wird. Es registriert, welche Geräte mit dem Netzwerk verbunden sind und zeigt diese in einer Übersicht an. Zusätzlich können auch WLAN-Geräte, welche sich in Funkreichweite befinden, aber nicht im WLAN eingewählt sind (wie beispielsweise der Laptop eines Nachbarn), getrackt werden. Auf Wunsch kann der Benutzer benachrichtigt werden, sobald sich ein bestimmtes Gerät im Netzwerk oder in Funkreichweite befindet. Außerdem kann Geräten mithilfe von Address Resolution Protocol (ARP)-Spoofing[6] der Internetzugriff entzogen werden. Ähnlich wie bei der Fing-App kann die Geschwindigkeit der WLAN- und Internetverbindung gemessen werden. Die gesammelten Informationen werden auf Servern des Herstellers gespeichert und sind über eine App abrufbar. [Eik18, S. 52]

Domotz[7] bietet mit Domotz Pro eine weitere Lösung an, dessen Funktionsumfang sich an die Fingbox anlehnt. Sie kann das Netzwerk auf verbundene Geräte scannen und diese identifizieren. Weitere Features sind die Überwachung von TCP- und Simple Network Management Protocol (SNMP)-Diensten sowie die Überwachung von Reaktionszeiten und der Geschwindigkeit der Internetverbindung. Beim Eintreten von konfigurierbaren Veränderungen von Parametern oder dem Beitritt neuer Geräte können E-Mail- oder Push-Benachrichtigungen versendet werden. Die Überwachung des Netzwerks ist auch über mehrere Subnetze oder VLANs hinweg möglich. Aufgrund der Ausrichtung auf ein professionelleres Umfeld werden mehrere Benutzerkonten mit einem Rollensystem und einer API zur Erweiterung des Systems unterstützt. Eine Smartphone-App zum mobilen Zugriff steht zur Verfügung. Ähnlich wie bei Fing wird das System als Domotz Box[8] auch als Hardwarelösung angeboten. [Domb; Doma]

Die **Bitdefender BOX**[9] von **Bitdefender** ist eine hardwarebasierte Lösung, die auf Privatkunden ausgerichtet ist. Zusätzlich zur Überwachung der Menge der mit dem Netzwerk verbundenen Geräte überwacht sie laut Herstellerangaben auch den Traffic der Benutzer auf schädliche Webseiten und unterbindet die unverschlüsselte Übertragung sensibler Benutzerdaten, prüft die verbundenen Geräte auf Schwachstellen oder atypisches Verhalten und blockiert die Ausnutzung von Schwachstellen. Hierfür werden Machine Learning und die Antivirus-Technologie des Unternehmens eingesetzt. Im Gegensatz zur Fingbox ermöglicht die Bitdefender BOX eine Überwachung des übertragenen Traffics, da das Gerät entweder als WLAN-Hotspot eingesetzt oder zwischen Modem und WLAN-Router dazwischengeschaltet werden soll. [Hüb18; Bit19, S. 3 f.]

Nagios XI[10] ist ein Netzwerk- und Servermonitoringsystem für Unternehmen. Es ermöglicht das Monitoring von Geräten, Diensten und ihrer Leistung. Ein Dashboard bietet eine Übersicht über

[5] `https://www.fing.com/products/fingbox`
[6] Dabei werden Daten, die an das Gateway adressiert sind, mithilfe von gefälschten ARP-Paketen an eine falsche MAC-Adresse umgeleitet [Bun07, S. 10]
[7] `https://www.domotz.com/`
[8] `https://www.domotz.com/domotz-box.php`
[9] `https://www.bitdefender.de/box/`
[10] `https://www.nagios.com/products/nagios-xi/`

den aktuellen Zustand der Komponenten. Assistenten unterstützen den Administrator bei der Konfiguration der zu überwachenden Komponenten, unter anderem auch durch einen automatischen Scan des Netzwerks nach verbundenen Geräten. Das System kann auf die Bedürfnisse der Benutzer angepasst und mithilfe von Add-ons kann zusätzliche Funktionalität nachgerüstet werden. Es ist außerdem Mehrbenutzerfähig. Mit **Nagios Core**[11] existiert auch eine quelloffene Variante des Systems mit eingeschränkter Funktionalität und weniger Benutzerfreundlichkeit. [Nag19b; Nag19a]

3.3 Abgrenzung zu vorhandenen Lösungen

Das in dieser Arbeit vorgestellte System kann sich in mehreren Punkten von den vorhandenen Lösungen abgrenzen.

Es kann ohne Abhängigkeit zu externen Systemen funktionieren und alle Daten lokal auf einem Datenträger im Netzwerk des Benutzers speichern. Dadurch wird die Privatsphäre des Benutzers gewahrt, da keine Daten zur internen Struktur des Netzwerks auf fremden Servern gespeichert werden. Dies ist ein Vorteil, da aus den Daten zur Geräteaktivität auch persönliche Verhaltensmuster ersichtlich werden können. So könnte man etwa aus der Information, wann sich ein Smartphone mit dem Netzwerk verbunden hat herauslesen, wann eine bestimmte Person zuhause war oder ob sie ihr Telefon zu ungewöhnlichen Zeiten benutzt hat. Außerdem könnte ein Angreifer, der an die Datenbank mit den Informationen gelangt, bestimmen, welche Haushalte verwundbare Geräte besitzen und diese gezielt angreifen.

Das System muss außerdem nicht mit dem Zwang zum Kauf einer bestimmten Hardware einhergehen. Da möglichst auf Standardsoft- und Hardware gesetzt werden soll, steht es dem Benutzer frei, die Anwendung auf der Hardware seiner Wahl zu betreiben (Vorausgesetzt, sie ist bezüglich Leistung, Architektur, Betriebssystem, usw. kompatibel). Der Vorteil, der dadurch entsteht ist, dass man für Sicherheitsupdates des zugrundeliegenden Betriebssystems nicht auf die Kooperation des Systemherstellers angewiesen ist, sondern lediglich auf die des Herstellers der Standardsoft- und hardware.

[11] `https://www.nagios.org/projects/nagios-core/`

4 Analyse der Bedrohungslage

In diesem Abschnitt wird die aktuelle Bedrohungslage in der Bundesrepublik Deutschland beziehungsweise der Europäischen Union analysiert, um eine fundierte Grundlage für die Anforderungen an das zu entwerfende System zu bilden.

4.1 Kategorien von Bedrohungen

Im Jahr 2019 hat das BSI in ihrem Bericht zur Lage der IT-Sicherheit in Deutschland folgende Bedrohungskategorien aufgeführt und erläutert [Bun19, S. 7]:

- Beim **Identitätsdiebstahl** werden Daten, die die Echtheit einer Person bilden (wie beispielsweise Benutzername und Passwort) entwendet. Möglichkeiten hierfür sind unter anderem Phishing[1], der Einsatz von Schadsoftware oder der Diebstahl von Daten bei Diensteanbietern. [Bun19, S. 8; Nat19]
- Der Begriff **Schadprogramme** umfasst sämtliche Programme (unter anderem Ransomware und Botprogramme), die gegen den Willen des Benutzers auf seinem Gerät installiert werden und diesem in irgendeiner Weise schaden. [Bun19, S. 11]
- **Ransomware** ist Software, welche dem Benutzer den Zugriff auf sein Gerät und die darauf befindlichen Daten verwehrt. Damit wird meist versucht, eine Zahlung vom Benutzer zu erwirken, indem man ihm verspricht, er bekomme danach wieder Zugriff auf sein Eigentum. Bei Ransomware handelt es sich um eine spezielle Art der Schadprogramme. [Bun19, S. 15 f.]
- **Botnets** sind Netzwerke aus Geräten, auf denen ohne Kenntnis der Nutzer ein Schadprogramm installiert wurde. Diese können beispielsweise für DDoS-Angriffe oder zum Schürfen von Kryptowährungen eingesetzt werden. [Bun19, S. 21]
- Bei **DDoS**-Angriffen wird ein Gerät über ein Netzwerk mit einer großen Menge an Anfragen überfordert, sodass es abstürzt oder nicht mehr erreichbar ist. Dadurch wird Nutzern der Zugriff auf die Dienste dieses Gerätes verwehrt. Im Kontext eines Heimnetzwerks haben DDoS-Angriffe keine große Relevanz. Dies rührt daher, dass Botnets im Vergleich eher selten für diesen Zweck benutzt werden und die Ziele dieser Angriffe meist öffentlich zugängliche Dienste sind. [Nat18; Bun19, S. 18, 21]
- **Spam** beschreibt E-Mails, welche vom Empfänger nicht gewünscht sind und meist Werbung, Schadprogramme oder Phishing-Versuche enthalten [Bun19, S. 26].

[1] Abgreifen von persönlichen Daten über gefälschte E-Mails, Kurznachrichten oder Webseiten [Bun19, S. 78]

- In die Kategorie **Kryptographie** fallen Angriffe, die Schwächen in kryptographischen Mechanismen oder Protokollen ausnutzen [Bun19, S. 29 f.].
- Eine Menge von **Schwachstellen in modernen Prozessorarchitekturen** wurde beginnend 2018 bekannt. Diese nutzen zumeist Seitenkanäle, um Informationen aus geschützten Speicherbereichen auszulesen oder in diese Bereiche zu schreiben [Bun19, S. 32].

4.2 Mögliche Angriffsvektoren

Die Bedrohungskategorien aus Abschnitt 4.1 treffen nicht zwingend eine Aussage über den Einfallsvektor, über den der Angriff erfolgen kann. Manchmal kann ein Angriffsvektor ein Einfallstor für eine Bandbreite an verschiedenen Bedrohungen sein. Die European Union Agency for Cybersecurity (ENISA) listet folgende Angriffsvektoren auf, die im Zeitraum von ca. Dezember 2017 bis Dezember 2018 beobachtet wurden [Eur19, S. 10]:

- **„Attacking the human element“**
  Social Engineering[2], Phishing und andere Formen des Betrugs
- **„Web and browser based attack vectors“**
  Angriffe auf Browser (beispielsweise Drive-by-Downloads[3] oder bösartige Skripte) oder Angriffe auf Server (beispielsweise SQL-Injections)
- **„Internet exposed assets“**
  Güter, die aus dem Internet erreichbar sind, aber nicht durch eine Firewall oder starke Zugangsdaten geschützt werden
- **„Exploitation of vulnerabilities/misconfigurations and cryptographic/network/security protocol flaws“**
  Ausnutzen von falsch konfigurierter Software oder von Schwachstellen in Software oder in Protokollen
- **„Supply-chain attacks“**
  Manipulation der Soft- oder Hardware entlang der Lieferkette, also bevor diese dem Opfer geliefert wird
- **„Network propagation/lateral movement“**
  Ausbreitung des Angreifers von einem infizierten Gerät auf andere Geräte im selben Netzwerk
- **„Active network attacks“**
  beispielsweise DNS-Poisoning[4]
- **„Privilege or user credentials misuse/escalation“**
  Missbrauch oder Manipulation von Zugängen

[2] Ausnutzen menschlicher Schwächen, damit das Opfer Informationen preisgibt [Bun19, S. 78]
[3] Automatische Installation von Schadsoftware durch Ausnutzung von Sicherheitslücken, die durch das bloße Betrachten einer Webseite angestoßen wird [Bun19, S. 77]
[4] Einfügen von gefälschten Einträgen in den Cache eines DNS-Servers [Clo]

- **„Fileless or memory-based attacks“**
  Angriffe, bei denen sich die Schadsoftware im Arbeitsspeicher befindet und keine Dateien auf persistente Medien geschrieben werden
- **„Misinformation/Disinformation“**
  Verbreitung von Fake News, Trolling, Einsatz von Social Bots oder Missbrauch von sozialen Medien und Suchmaschinen

[Eur19, S. 125 f.]

Von den aufgelisteten Vektoren können folgende eventuell durch Netzwerkmonitoring abgemildert werden:

- **„Internet exposed assets“** sind in Deutschland kein signifikantes Problem, da Geräte in der Regel durch einen Router abgeschirmt und nicht von außen erreichbar sind [Bun19, S. 7]. Dennoch können automatische Portscans nützlich sein. Der Benutzer könnte Ports von seinem Router auf Geräte im Netzwerk weitergeleitet haben. Außerdem könnten offene Ports von einem Angreifer, welcher sich bereits anderweitig Zugriff auf das Netzwerk verschafft hat, genutzt werden, um weitere Geräte im Netzwerk zu infizieren.
- Der **„Exploitation of vulnerabilities/misconfigurations and cryptographic/network/security protocol flaws“** kann durch präventive Erkennung vorgebeugt werden. Dies ist zum Teil automatisiert möglich, etwa wenn auf einem Gerät ein veralteter Anwendungsserver läuft oder kritische Funktionen nicht durch eine wirksame Authentifikation geschützt sind.
- **„Network propagation/lateral movement“** wird durch Schwachstellen und Fehlkonfigurationen begünstigt. Ein automatischer Scan könnte hier durch eine Aufklärung des Benutzers über unsichere Geräte in seinem Netzwerk Abhilfe schaffen.
- **„Privilege or user credentials misuse/escalation“** ist relevant, da Geräte ohne Zugriffskontrolle oder mit den standardmäßig voreingestellten Anmeldeinformationen betrieben werden können. Dies kann ebenfalls durch automatische Scans erkannt werden.

Andere Angriffsvektoren, wie etwa **„Active network attacks“** oder **„Fileless or memory-based attacks“** stehen in keinem direkten Zusammenhang mit dieser Arbeit. Sie können allerdings auch als Einfallstor für weitere Angriffsarten genutzt werden.

4.3 Besonderheiten durch das Internet of Things

Das BSI stellte in ihrem Bericht 2019 fest, dass Angreifer zum Berichtszeitpunkt einen besonderen Fokus auf mobile Endgeräte und IoT-Geräte legen. Letzteres liegt daran, dass die Sicherheit von IoT-Geräten aus wirtschaftlichen Gründen oft vernachlässigt wird und eine Unterstützung durch Sicherheitsupdates des Herstellers somit ausbleibt. Das macht sie zu einem attraktiven Angriffsziel für Angreifer, welche die Konnektivität und Ressourcen dieser Geräte ausnutzen wollen. [Bun19, S. 21]

Die Infektionszahlen bei IoT-Geräten sind in Deutschland aber bisweilen noch gering, da sie sich in deutschen Netzwerken in der Regel hinter einem Router befinden und somit nicht direkt aus

dem Internet erreichbar sind [Bun19, S. 24]. Allerdings beutetet das nicht, dass von diesen Geräten keine Bedrohung ausgeht. Wie Sivaraman et al. in einem Konferenzbericht gezeigt haben, ist es vorstellbar, dass ein Angreifer Schadsoftware auf einem mit dem Heimnetzwerk verbundenen Smartphone einschleusen könnte, um dieses als Sprungbrett in das Netzwerk zu benutzen [Siv+16, S. 195, 199].

Eine weitere Gefahr könnte auch entstehen, wenn Angreifer das Konzept von Ransomware auf das IoT-Umfeld übertragen. Während Ransomware bislang die Dateien auf einem Computer als Geisel nimmt, um eine Zahlung zu erwirken, ist es auch denkbar, dass die Funktionalität von IoT-Geräten als Geisel genommen werden könnte. Im Kontext eines Smart Homes könnte dies beispielsweise bedeuten, dass den Bewohnern der Zugriff auf ihre Beleuchtung, ihre Heizung oder ihre Türschlösser verwehrt wird, bis diese ein Lösegeld bezahlen. [Bun19, S. 15 f. Sch15, S. 117 f.]

Im September 2016 wurde ein neuer Weltrekord für den größten DDoS-Angriff aufgestellt. Für diesen Angriff wurde das Mirai-Botnet benutzt, welches laut Angaben des angeblichen Autors aus über 380.000 infizierten IoT-Geräten bestand. Diese Geräte wurden mit standardmäßig voreingestellten Benutzernamen und Passwörtern betrieben, wodurch es dem Mirai-Bot ermöglicht wurde sie zu infizieren. Durch Vorfälle wie diesen wird deutlich, dass die Sicherheit von IoT-Geräten nicht vernachlässigt werden sollte. [Nat17]

4.4 Bedrohungs- und Risikoanalyse eines Heimnetzwerks

In diesem Abschnitt werden die Bedrohungen und Risiken in einem Heimnetzwerk systematisch analysiert. Die dafür genutzte Vorgehensweise lehnt sich an die von Paulus in *Basiswissen Sichere Software* beschriebene an [Pau11, S. 89 ff.].

Überblick über die Vorgehensweise

Die Vorgehensweise ist wertezentriert. Das bedeutet, dass die zu schützenden Werte im Mittelpunkt der Betrachtung liegen. Deswegen werden als erster Schritt die Werte und Akteure identifiziert. Bei den Werten kann es sich sowohl um Daten als auch um Konzepte wie das Image des Kunden handeln. Es sollten auch die Interaktionen zwischen den Werten und den Akteuren betrachtet werden. [Pau11, S. 90 f.]

Im zweiten Schritt soll eine Übersicht der geplanten Architektur erstellt werden. Dies ist ein iterativer Prozess, welcher die Architektur und die Sicherheitsvorgaben aufeinander abstimmt. Den Kern dieser Übersicht bilden die Datenhaltungen und die Datenflüsse im System. Aus diesen lassen sich die Vertrauensgrenzen ableiten. Sie definieren die Stellen, an denen Daten nicht ungehindert übertragen können werden sollen und sind somit essenziell für die Sicherheit eines Systems. [Pau11, S. 92]

Im dritten Schritt wird das System entlang der zuvor definierten Vertrauensgrenzen in seine Einzelteile zerlegt. Auf den Einzelteilen wird der zweite Schritt dann erneut durchgeführt, um zu prüfen, ob die bisherige Aufteilung ausreichend ist oder ob weiter aufgeteilt werden muss. Im

vierten Schritt werden möglichst viele Bedrohungen anhand der bisherigen Ergebnisse identifiziert. Dafür können Techniken wie Brainstorming, Angriffsbäume oder Checklisten verwendet werden. [Pau11, S. 93 - 103]

Im fünften Schritt werden Bedrohungen bewertet, um zu ermitteln, welche davon eine realistische Gefahr darstellen, die verhindert werden sollte. Hierfür kann ein Bedrohungsmodell wie das Open Web Application Security Project (OWASP) Risk Rating verwendet werden. Zuletzt werden auf Basis der Bewertungen die Gegenmaßnahmen geplant. Hier werden die Gründe für die Schwachstellen ausfindig gemacht und Vorgehensweisen zur Abwehr dieser Angriffe entwickelt. [Pau11, S. 104 - 111]

Identifikation der Werte und Akteure

Zur Identifikation der **Werte**, die in einem Heimnetzwerk schützenswert sind, können die in Abschnitt 4.1 beschriebenen Bedrohungskategorien genutzt werden. Bei den Werten handelt es sich demnach um folgende:

- Die **Identität** des Benutzers (beispielsweise Zugangsdaten),
- die **Daten** des Benutzers (beispielsweise persönliche Daten wie Bilder, Dokumente, Nachrichten, etc.),
- und die **Rechenkapazität** sowie die **Internetverbindung** der Netzwerkteilnehmer (die durch Schadsoftware missbraucht werden kann)

Als **Akteure** sind folgende Parteien in einem Heimnetzwerk beteiligt:

- Die **rechtmäßigen Benutzer** des Netzwerks,
- die **Administratoren** des Netzwerks (eine Teilmenge der Benutzer),
- **gutartige Dritte** (beispielsweise die Anbieter von Diensten im Internet),
- und **bösartige Angreifer** (die die Werte der Benutzer erbeuten wollen).

Übersicht über die Architektur

Die **Datenhaltung** erfolgt zum Großteil auf den Endgeräten des Benutzers. Dafür werden die internen Speicher von PCs, Laptops, Smartphones und Tablets benutzt. Bei diesen Daten handelt es sich sowohl um Zugangsdaten zu externen Diensten, als auch um andere Daten wie Dokumente oder Fotos. Auch auf Network Attached Storage (NAS)-Systemen können Daten für den Zugriff aus dem Heimnetzwerk und aus dem Internet gehalten werden.

Die **Datenflüsse** sind grundsätzlich zwischen allen Netzwerkteilnehmern möglich. Die Teilnehmer sind in diesem Kontext sowohl die Geräte im Heimnetzwerk als auch in anderen verbundenen Netzwerken (wie dem Internet). Ungehinderte Datenflüsse sind zwischen den Teilnehmern des Netzwerks möglich, die Verbindung in andere Netze kann durch Firewalls oder Bandbreitenbeschränkungen eingeschränkt sein.

Zerteilen des Systems

Die Kommunikation zwischen Geräten in einem Heimnetzwerk ist ungehindert möglich, wie auch die Kommunikation zwischen Geräten im Internet. Eine Vertrauensgrenze liegt also zwischen dem Heimnetzwerk und dem Internet vor. Diese wird durch den Router (auch Gateway genannt) gebildet. Dieser stellt eine Firewall bereit und kontrolliert die Datenflüsse zwischen dem Heimnetzwerk und dem Internet.

Identifikation und Bewertung der Bedrohungen

Die Bedrohungen für das Heimnetzwerk basieren auf dem Bericht des BSI, welcher in Abschnitt 4.1 bearbeitet wurde. Aufgrund der Menge an Angriffen, die analysiert werden könnten, werden in diesem Kapitel nur beispielhaft drei Bewertungen mithilfe des OWASP Risk Rating-Systems[5] durchgeführt. Dabei werden vier Faktoren mit je vier Punkten analysiert, um durch eine Mittelwertberechnung ein Gesamtrisiko zu ermitteln. [OWA19]

Beispiel 1: Missbrauch von offenen Ports oder Diensten

Die Gefahr des Missbrauchs von offenen Ports oder Diensten (wie beispielsweise einem Webinterface) wird in Tabelle 4.1 bewertet. Die Werte ergeben eine mittlere Likelihood (5,5), einen hohen Impact (6,5) und somit eine **hohe** Overall Risk Severity.

Beispiel 2: Unberechtigter Zugriff aufgrund von schwachen Zugangsdaten

Die Gefahr eines unberechtigten Zugriffs auf Geräte aufgrund von schwachen Zugangsdaten (beispielsweise dem Zugriff auf Überwachungskameras) wird in Tabelle 4.2 bewertet. Die Werte ergeben eine mittlere Likelihood (5,375), einen hohen Impact (6,5) und somit eine **hohe** Overall Risk Severity.

Beispiel 3: Missbrauch von Schwachstellen in veralteter Software

Die Gefahr eines Missbrauchs von Schwachstellen in Software, welche nicht rechtzeitig aktualisiert wurde, wird in Tabelle A.1 bewertet. Die Werte ergeben eine mittlere Likelihood (5,625), einen hohen Impact (6,5) und somit eine **hohe** Overall Risk Severity.

[5] `https://www.owasp.org/index.php?title=OWASP_Risk_Rating_Methodology&oldid=252633`

Faktor	Bewertung	Begründung
Threat Agent Factors		
Skill level	5 / 9	Fortgeschrittene IT-Kenntnisse notwendig
Motive	3 / 9	Hängt von Stellung des Opfers und Art des Dienstes ab
Opportunity	6 / 9	Eventuell Zugriff aufs Heimnetzwerk notwendig
Size	7 / 9	Entweder Teilnehmer im Heimnetzwerk oder gesamtes Internet
Vulnerability Factors		
Ease of discovery	7 / 9	Portscans können automatisch durchgeführt werden
Ease of exploit	3 / 9	Zusätzliche Schwachstellen müssen verbunden werden
Awareness	5 / 9	Sicherheitslücken können öffentlich dokumentiert sein
Intrusion detection	8 / 9	Logs werden vermutlich nicht gesichtet
Technical Impact Factors		
Loss of confidentiality	7 / 9	Geräte können sensible Informationen enthalten
Loss of integrity	7 / 9	Sensible Daten können beschädigt werden
Loss of availability	7 / 9	Zugriff auf Geräte kann verwehrt werden
Loss of accountability	8 / 9	IP-Adressen vielleicht ermittelbar
Business Impact Factors		
Financial damage	7 / 9	Missbrauch sensibler Daten möglich
Reputation damage	7 / 9	Rufschädigung mithilfe sensibler Daten möglich
Non-compliance	2 / 9	Im privaten Kontext von geringer Relevanz
Privacy violation	7 / 9	Angriffe im großen Maßstab denkbar

Abbildung 4.1: OWASP Risk Rating eines Missbrauch von offenen Ports oder Diensten.

Faktor	Bewertung	Begründung
Threat Agent Factors		
Skill level	3 / 9	Nur grundlegende Kenntnisse erforderlich
Motive	4 / 9	Hängt von Stellung des Opfers und Art des Dienstes ab
Opportunity	6 / 9	Eventuell Zugriff aufs Heimnetzwerk notwendig
Size	7 / 9	Entweder Teilnehmer im Heimnetzwerk oder gesamtes Internet
Vulnerability Factors		
Ease of discovery	5 / 9	Sicherheitslücken können öffentlich dokumentiert sein
Ease of exploit	4 / 9	Vorbedingungen müssen erfüllt sein
Awareness	6 / 9	Sicherheitslücken können öffentlich dokumentiert sein
Intrusion detection	8 / 9	Logs werden vermutlich nicht gesichtet
Technical Impact Factors		
Loss of confidentiality	7 / 9	Geräte können sensible Informationen enthalten
Loss of integrity	7 / 9	Sensible Daten können beschädigt werden
Loss of availability	7 / 9	Zugriff auf Geräte kann verwehrt werden
Loss of accountability	8 / 9	IP-Adressen vielleicht ermittelbar
Business Impact Factors		
Financial damage	7 / 9	Missbrauch sensibler Daten möglich
Reputation damage	7 / 9	Rufschädigung mithilfe sensibler Daten möglich
Non-compliance	2 / 9	Im privaten Kontext von geringer Relevanz
Privacy violation	7 / 9	Angriffe im großen Maßstab denkbar

Abbildung 4.2: OWASP Risk Rating eines unberechtigten Zugriffs aufgrund schwacher Zugangsdaten.

5 Analyse der Anforderungen

Die Anforderungsanalyse baut auf der Definition eines Heimnetzwerks aus Abschnitt 2.1 auf. Als Protokoll auf der Vermittlungsschicht wird nur IPv4 betrachtet. Das liegt daran, dass zur Enumeration von Geräten in IPv4- und IPv6-Netzen verschiedene Ansätze verwendet werden müssen. Einige Ansätze basieren darauf, sämtliche IP-Adressen im Bereich des Heimnetzwerks darauf zu testen, ob sie einem Gerät zugewiesen sind. Aufgrund des stark vergrößterten Addressraums von IPv6 im Vergleich zu IPv4 (2^{128} statt 2^{32} Adressen) ist dies bei IPv6 nicht mehr praktikabel.

5.1 Installation und Einrichtung

Das System verbessert die Sicherheit eines Netzwerks, in dem es dessen Benutzer über die Netzwerkstruktur und über Missstände in diesem aufklärt. Das System muss von den Benutzern daher akzeptiert werden, um effektiv zu sein. Dabei werden die meisten Heimnetzwerke von Benutzern mit wenig Fachwissen administriert [Was10, S. 2]. Aus diesem Grund soll das System in der Installation und Einrichtung möglichst unkompliziert sein (Anforderung R1). Es soll einem Benutzer mit durchschnittlichen Computerkenntnissen möglich sein, sich die nötige Hard- und Software zu beschaffen und alle Komponenten zu installieren und zu konfigurieren.

Auf der Hardwareseite kann diese Anforderung erfüllt werden, in dem nur Produkte verwendet werden, die leicht im stationären oder im Onlinehandel zu erwerben sind. Auf der Softwareseite soll der Benutzer möglichst wenige Schritte durchführen müssen und wenn notwendig durch Anleitungen und Assistenten durch den Prozess geführt werden. Da, wie zuvor gezeigt wurde, Fehlkonfigurationen eine Quelle von Sicherheitsproblemen darstellen, soll das System nach dem Prinzip *Security by Default* ohne weitere Eingriffe des Benutzers sicher sein (R2). Das bedeutet beispielsweise, dass Sicherheitsupdates automatisch installiert werden.

5.2 Überwachung der Netzwerkkomponenten

Eine Beeinträchtigung der Sicherheit eines Netzwerks erfordert meist die Teilnahme eines unsicheren Gerätes an diesem Netzwerk. Deswegen muss dem Benutzer ein Überblick darüber verschafft werden, welche Geräte mit seinem Netzwerk verbunden sind (R3). Dadurch soll erkennbar sein, wo sich potenzielle Gefahrenquellen befinden. Dafür muss das System automatisch und in regelmäßigen Abständen einen Scan des gesamten Netzwerkbereichs durchführen (R8). Die Ergebnisse (darunter MAC-Adresse, IP-Adresse und Hostname) müssen in einer Datenbank gespeichert werden.

Auf den gefundenen Geräten muss ebenfalls automatisch und in regelmäßigen Abständen eine Menge von Tests ausgeführt werden. Dabei sollen das verwendete Gerät sowie das Betriebssystem und dessen Versionsnummer bestimmt werden (R4), um eine Identifikation der Geräte zu erleichtern. Außerdem müssen angebotene Dienste, wie offene Ports, UPnP- oder Bonjour-Dienste, erkannt werden, um die Sicherheit dieser bewerten zu können (R5). Es soll geprüft werden, ob veraltete und damit möglicherweise unsichere Betriebssystem- oder Anwendungsversionen eingesetzt werden (R6). Außerdem soll festgestellt werden, ob Dienste ohne Zugangsdaten oder mit standardmäßig voreingestellten Zugangsdaten betrieben werden (R7).

Eine Analyse des übertragenen Traffics könnte ebenfalls wünschenswert sein. Dabei müsste der Traffic allerdings auf dem Switch beziehungsweise dem Gateway abgefangen werden. Dafür müsste diese Hardware bei der Installation des Systems durch ein kompatibles Gerät ersetzt werden. Da die Einrichtung des Systems möglichst simpel sein soll, wird auf diese Funktion verzichtet.

5.3 Auswertung und Darstellung der Daten

Um eine hohe Akzeptanz des Systems bei den Benutzern zu erreichen, soll die Benutzeroberfläche auch für Laien leicht verständlich sein. Hierfür sollen die gefundenen Geräte in einer intuitiven visuellen Darstellung angeordnet werden, aus welcher der Aufbau des Netzwerks erkennbar ist (R9). Es soll erkennbar sein, welches Gerät in der Oberfläche welchem physischen Gerät entspricht.

Wenn Umstände vorliegen, die die Aufmerksamkeit des Benutzers erfordern, soll diese darauf gelenkt werden (R10). Um erfahrenen Benutzern dennoch die Analyse der Situation zu ermöglichen, soll die Möglichkeit geboten werden, erweiterte Informationen einzusehen. Knapp die Hälfte der Internetnutzung findet heutzutage auf Smartphones statt [Sta19b]. Aus diesem Grund soll die Benutzeroberfläche sowohl auf Computern als auch auf Smartphones benutzbar sein (R11).

5.4 Zusammenfassung

R1 Das System soll einfach zu beschaffen und einzurichten sein.

R2 Es soll standardmäßig so konfiguriert sein, dass es ohne Eingreifen des Benutzers sicher ist (Security by Default).

R3 Es muss eine Übersicht über alle mit dem Netzwerk verbundenen Geräte angezeigt werden.

R4 Die gefundenen Geräte sollen identifiziert werden (Geräteart, Hersteller, Betriebssystem, Betriebssystemversion).

R5 Es muss eine Übersicht über die von den Geräten angebotenen Dienste angezeigt werden (offene Ports, UPnP, Bonjour).

R6 Es soll geprüft werden, ob veraltete Betriebssystem- der Anwendungsversionen eingesetzt werden.

R7 Es soll geprüft werden, ob Dienste keine oder schwache Zugangsdaten verwenden.

R8 Die Suche nach neuen Geräten und ihren Merkmalen muss in konfigurierbaren Intervallen automatisch angestoßen werden.

R9 Die ermittelten Informationen sollen in einer intuitiven Darstellung angeordnet werden.

R10 Die Aufmerksamkeit des Benutzers soll auf Umstände, die diese erfordern, gelenkt werden.

R11 Die Benutzeroberfläche soll sowohl auf Computern als auch auf Smartphones benutzbar sein.

6 Entwurf

In diesem Kapitel werden die zuvor entwickelten Anforderungen an das System zu einer konkreten technischen Vorgehensweise umgesetzt.

6.1 Auswahl der Hard- und Softwarebasis

Um den Code simpel und wartbar zu gestalten, soll auf eine Softwareplattform gesetzt werden, die möglichst verbreitet ist. Dadurch ist ein besseres Ökosystem an Hilfestellungen und Programmbibliotheken vorhanden. Python zeichnet sich mit einer hohen Beliebtheit aus. Laut einem Bericht von Stack Overflow befand sich die Programmiersprache gemessen am Anteil der Entwickler, die die Sprache einsetzen, mit 42% auf dem zweiten Platz hinter JavaScript [Sta19a]. Die Sprache bietet eine umfangreiche Auswahl an Hilfsbibliotheken, beispielsweise Scapy[1] zur Paketmanipulation oder Flask[2] für den Webserver.

Zur Datenpersistenz wird eine relationale SQL-Datenbank gewählt. Das relationale Datenbankmodell ist geeignet, um die Zusammenhänge zwischen den einzelnen Ergebnissen abzubilden. Durch SQL wird eine Analyse der Datensätze bereits auf Datenbankebene ermöglicht. Da in einem deutschen Haushalt 2018 im Durchschnitt nur 1,99 Personen lebten, ist nicht davon auszugehen, dass das System mehr als eine kleine einstellige Anzahl an gleichzeitigen Benutzern haben wird [Sta19c, S. 48]. Aus diesem Grund ist eine SQLite-Datenbank[3] ausreichend. Dabei handelt es sich um eine gemeinfreie Datenbankbibliothek und laut Angaben der Entwickler um die am meisten eingesetzte Datenbank der Welt. Sie ist serverlos, daher greift jeder Prozess, der mit ihr interagiert, direkt auf die Datenbankdatei zu. Aufgrund von Dateisystemsperren ist dennoch eine gleichzeitige Nutzung der Datenbank aus mehreren Prozessen möglich, wodurch ein ausreichendes Maß an Nebenläufigkeit sichergestellt wird. [SQLa; SQLb]

Da die Benutzeroberfläche auf einen webbasierten Zugriff setzt, muss hierfür eine andere Programmiersprache genutzt werden. Die von gängigen Browsern nativ unterstützte Sprache ist JavaScript. Diese wird durch ein Framework zur Erstellung von Benutzeroberflächen ergänzt, um eine bessere Struktur und Wartbarkeit des Frontend-Codes zu gewährleisten. Hierfür wird das von Facebook entwickelte React[4] gewählt.

Als Hardware ist grundsätzlich alles geeignet, was von den genutzten Programmiersprachen und Frameworks unterstützt wird. Mit 30 Millionen verkauften Einheiten (Stand Dezember 2019) erfreut sich die Raspberry Pi-Serie[5] von Einplatinencomputern über immense Beliebtheit [Upt19].

[1] https://scapy.net/
[2] https://palletsprojects.com/p/flask/
[3] https://www.sqlite.org/index.html
[4] https://reactjs.org/
[5] https://www.raspberrypi.org/

Die aus der Verbreitung resultierende Anzahl an Händlern, die das Produkt anbieten, trägt dazu bei, die Anforderung R1 (Einfache Beschaffung) zu erfüllen. Der Raspberry Pi 3B+ bietet mit vier ARMv8-basierten Kernen mit je 1,4 GHz und 1 GB an Arbeitsspeicher ausreichend Leistung zum Ausführen von einfacheren Diensten und Webanwendungen. Die Anbindung an das Netzwerk kann dabei entweder über Gigabit-Ethernet oder über 2,4 GHz und 5 GHz 801.11b/g/n/ac WLAN erfolgen. [Hat19]

6.2 Strategien zur Auflistung der Netzwerkteilnehmer

Da in (deutschen) Haushalten eine Vielzahl von verschiedenen Routern im Einsatz ist, gibt es keine einheitliche Schnittstelle zur Auflistung aller Netzwerkteilnehmer. Deswegen müssen mehrere Varianten, an diese Informationen zu gelangen, evaluiert werden. Dies dient der Erfüllung von Anforderung R3 (Übersicht über alle Geräte).

Die ersten drei Ansätze, die in diesem Kapitel betrachtet werden, basieren auf den von McClure et al. in *Hacking Exposed 7: Network Security Secrets & Solutions* vorgeschlagenen Strategien [MSK12, S. 48 - 59].

Auflistung mithilfe des Internet Control Message Protocols (ICMP)

Das Internet Control Message Protocol (ICMP) ermöglicht es Geräten, Kontrollnachrichten auf der Vermittlungsschicht auszutauschen. Genutzt wird dies unter anderem, um dem Absender eines Paketes mitzuteilen, wenn keine Route zum adressierten Zielnetz gefunden werden konnte. Ein weiteres Beispiel für den Einsatz des ICMP ist das `ping`-Werkzeug, welches Bestandteil der meisten Betriebssysteme ist. Dieses sendet eine *ICMP Echo*-Anfrage und wartet, ob der Empfänger mit einem *ICMP Echo Reply*-Paket antwortet. Dadurch kann geprüft werden, ob ein Host im Netzwerk erreichbar ist oder nicht. [KR17, S. 447]

Dieses Verhalten kann genutzt werden, um ein Netzwerk nach verbundenen Teilnehmern zu scannen. Dafür sendet man *Echo*-Pakete an jede IP-Adresse in einem Netzwerkbereich. Erhält man ein *Echo Reply*-Paket als Antwort, verbirgt sich hinter der IP-Adresse ein Netzwerkteilnehmer. [MSK12, S. 51 f.]

Der Nachteil dieser Strategie ist, dass *Echo*-Anfragen aus Sicherheitsgründen von Firewalls blockiert werden können [Lyo10, S. 60]. Dadurch soll die Sicherheit erhöht werden, da das Gerät nicht so einfach entdeckt werden kann, was hier zu einem Problem führt. Sollte die Firewall nur *Echo*-Anfragen blockieren, können stattdessen auch *Timestamp*- oder *Address Mask*-Pakete verwendet werden. Allerdings können diese ebenfalls von einer Firewall gefiltert werden. [MSK12, S. 52]

Auflistung mithilfe von TCP oder UDP

TCP ist ein Protokoll für verbindungsorientierte Datenübertragung. Da es auf einem paketvermittelten Netzwerk aufbaut, wird beim Aufbau der Verbindung ein beidseitiger Handschlag durchge-

führt. Dieses Verhalten lässt sich verwenden, um Teilnehmer in einem Netzwerk dazu zu bringen, sich preiszugeben. [KR17, S. 261; MSK12, S. 55 ff.]

Dafür wird beispielsweise ein *TCP SYN*-Paket an jede IP-Adresse in einem Netzwerkbereich gesendet. Entspricht das Verhalten des Zielrechners der Spezifikation und werden solche Pakete nicht durch eine Firewall gefiltert, muss er entweder mit einem *TCP SYN ACK* (falls der angesprochene Port offen ist) oder mit einem *TCP RST* (falls der angesprochene Port geschlossen ist) antworten. Dadurch würde das Gerät seine Existenz offenlegen. [Lyo10, S. 61]

Ähnlich kann auch UDP genutzt werden. Der Unterschied hier ist jedoch, dass UDP verbindungslos ist und somit kein Verbindungsaufbau simuliert werden kann. Stattdessen kann ein beliebiges UDP-Paket an einen Port des Zielrechners gesendet werden. Ist der Zielrechner erreichbar und ist der Port geschlossen, sollte er mit einem *ICMP Port Unreachable*-Paket antworten. Ist der Port geöffnet, könnte die dort lauschende Anwendung mit einem Paket antworten. [Lyo10, S. 63 f.]

Ähnlich wie beim ICMP-Ansatz können sowohl TCP- als auch UDP-Pakete durch eine Firewall gefiltert werden, was die Auflistung des Teilnehmers unmöglich machen würde. Dem kann entgegengewirkt werden, indem die Pakete an einen offenen Port gesendet werden. [Lyo10, S. 61 ff.]

Auflistung mithilfe des Address Resolution Protocols (ARP)

Auf der Sicherungsschicht, welche meist durch Ethernet implementiert wird, ist die Kommunikation zwischen Geräten nur durch Adressierung von physischen Adressen (in diesem Falle MAC-Adressen) möglich. Die meisten Anwendungen bauen jedoch auf dem IP-Protokoll auf, welches eine weitere Adressierung auf der Vermittlungsschicht vorsieht. Aus diesem Grund müssen IP-Adressen zu MAC-Adressen aufgelöst werden, ehe die Datagramme an das richtige physische Gerät adressiert werden können. Hierfür dient das ARP. [KR17, S. 496 - 499]

Das ARP lässt sich zweckentfremden, um alle verbundenen Hosts in einem Netzwerk samt ihrer physischen Adressen zu ermitteln. Hierfür sendet man eine ARP-Anfrage zur Anforderung der zugrundeliegenden physischen Adresse an jede IP-Adresse in einem Netzwerkbereich. Empfängt man darauf eine Antwort, befindet sich hinter der IP-Adresse ein verbundenes Gerät. [MSK12, S. 49]

Stark vereinfachter beispielhafter Anfrage- und Antwortverlauf:
→ „Wem gehört 192.168.1.1?“
← „192.168.1.1 gehört aa:bb:cc:dd:ee:ff“ *(Gerät gefunden)*
→ „Wem gehört 192.168.1.2?“
← Keine Antwort *(Kein Gerät gefunden)*
→ ...

Diese Strategie hat den Vorteil, dass das ARP zwingend von allen Netzwerkteilnehmern implementiert werden muss. Das liegt daran, dass eine Erreichbarkeit über das IP-Protokoll immer mit einer Ermittlung der physischen Adresse beginnt. Auch bei einem anderen Ansatz zum Scan eines IP-Bereichs auf Netzwerkteilnehmer muss jedem Verbindungsaufbau zu einem neuen Host erst eine ARP-Anfrage zuvorkommen. Das bedeutet, dass dieser Ansatz mit allen im Netzwerk erreichbaren Geräten funktionieren sollte. [MSK12, S. 49]

Der Nachteil dieses Ansatzes ist, dass das ARP nur innerhalb eines lokalen Netzwerks eingesetzt werden kann. Sobald mehrere Subnetze zu einem Netzwerk zusammengeschlossen werden, erfolgt zwischen diesen ein Routing auf IP-Ebene. Dabei werden keine Ethernet-Pakete (und somit auch keine ARP-Pakete) zwischen den Subnetzen übertragen. Möchte man Hosts hinter diesen Netzwerkgrenzen enumerieren, muss deswegen auf andere Strategien zurückgegriffen werden.

Auflistung mithilfe von TR-064

Im technischen Bericht TR-064 (*LAN-Side DSL CPE Configuration*) hat das Broadband Forum im Jahr 2004 eine Schnittstelle zur Konfiguration von Customer Premises Equipment (CPE) aus dem lokalen Netzwerk ausgearbeitet. Unter CPE versteht man Hardware am physischen Standort des Kunden, in diesem Fall den Router. Die Schnittstelle ermöglicht die Konfiguration dieser Geräte aus dem lokalen Netzwerk. [The15, S. 3 of 7; AVM; Rou16]

Das Protokoll baut auf XML und SOAP auf und ist mit der UDA 1.0-Spezifikation kompatibel. Gängige UPnP-Bibliotheken können zum Umgang mit der Schnittstelle benutzt werden (beispielsweise GUPnP[6], siehe Abbildung A.4). Beim Start und auf Anforderung der Managementanwendung kündigt der CPE mithilfe des SSDP seine verfügbaren TR-064-Dienste im Netzwerk an. Diese Nachricht enthält Informationen zu der angebotenen Funktionalität, sowie zu den Endpunkten, unter denen diese Funktionalität zu erreichen ist. [The15, S. 1; Bol19]

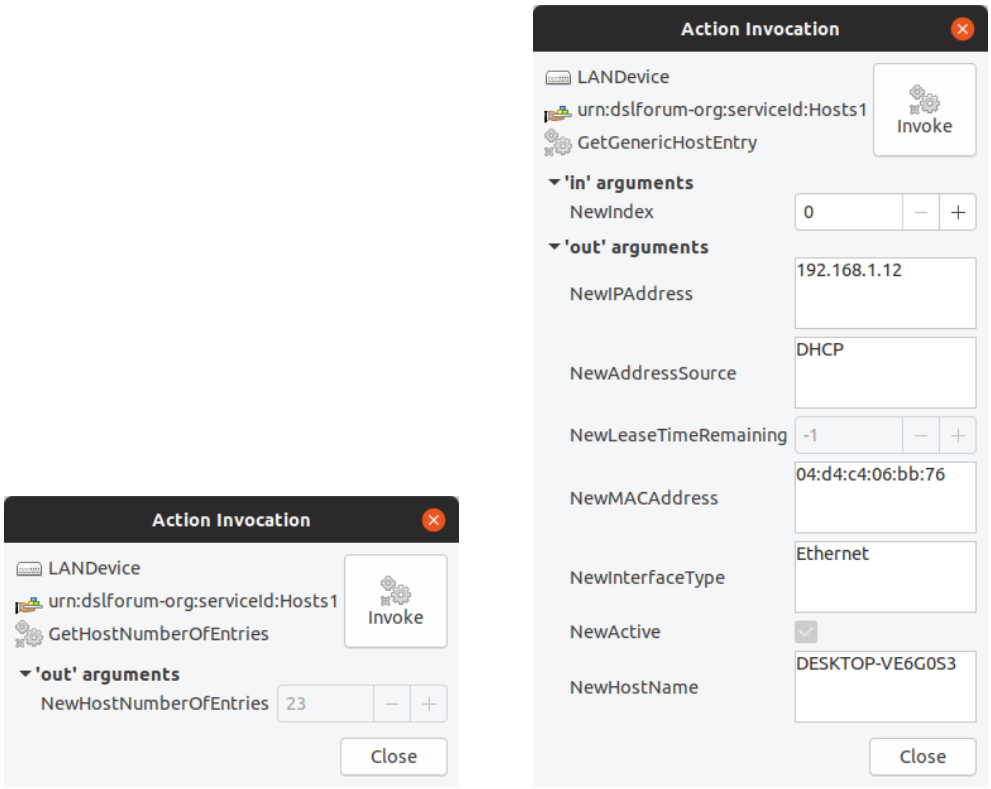
Das Protokoll ermöglicht unter anderem das Setzen und Auslesen bestimmter Parameter in CPEs, wie beispielsweise den Internetzugangsdaten, der WLAN- und Firewall-Konfiguration oder der verbundenen Clients. Im Kapitel *Hosts* des TR-064 wird ein Dienst beschrieben, welcher den Abruf der mit dem CPE verbundenen Geräte mitsamt ihrer IP-Adresse ermöglicht (siehe Abbildung 6.1). Dies kann mithilfe der `GetHostNumberOfEntries`- und `GetGenericHostEntry`-Methoden implementiert werden. [AVM16, S. 1 f. The15, S. 47 f.]

Das TR-064-Protokoll ist auf allen aktuellen Geräten aus der FRITZ!Box-Reihe von AVM standardmäßig aktiviert. Da das Unternehmen auf dem deutschen Markt derzeit Marktführer ist, dürfte diese Strategie in einer beträchtlichen Anzahl von deutschen Haushalten gute Ergebnisse erzielen. Dennoch kann sie nicht als alleinige Lösung eingesetzt werden, da das Protokoll nicht von allen Routern implementiert werden muss und es vom Benutzer deaktiviert worden sein könnte. [AVM16, S. 2; Rüg19]

Auswahl der Auflistungsstrategien

Um einer zu hohen Komplexität entgegenzuwirken, werden in dieser Arbeit nur ausgewählte Strategien implementiert. Die Enumeration über TR-064 ist am saubersten, da hier klar definierte Schnittstellen genutzt und keine Protokolle zweckentfremdet werden. Allerdings müssen nicht alle Heimnetze Unterstützung für dieses Protokoll bieten. Aus diesem Grund wird als zweite Strategie das ARP-Scanning implementiert. Dieses ist aus den verbleibenden Strategien am besten geeignet, da das zugrundeliegende Verhalten essenziell für die Funktionalität eines Netzwerks ist und somit in den meisten Fällen funktionieren sollte.

[6] `https://wiki.gnome.org/Projects/GUPnP`



(a) `GetHostNumberOfEntries()` (b) `GetGenericHostEntry(0)`

Abbildung 6.1: Methodenaufrufe auf einer o2 HomeBox 2 6741 mithilfe von GUPnP Universal Control Point (Screenshot, selbst angefertigt).

6.3 Strategien zur Identifikation der Netzwerkteilnehmer

Ähnlich wie in Abschnitt 6.2 existiert für die Identifikation der Netzwerkteilnehmer keine einheitliche Lösung. Aus diesem Grund muss wieder auf Behelfslösungen zurückgegriffen werden, um die benötigten Informationen zu ermitteln. Durch diese Vorgehensweisen wird Anforderung R4 (Identifikation der Geräte) erfüllt.

Analyse der MAC-Adressen

Die Adressierung auf der Sicherungsschicht erfolgt bei Ethernet mithilfe von MAC-Adressen (auch physische Adressen genannt). Diese sind sechs Byte lang und werden in Hexadezimalschreibweise notiert (beispielsweise `aa:bb:cc:dd:ee:ff`). Jedes Netzwerkinterface besitzt der Idee nach eine weltweit eindeutige MAC-Adresse, die keine zwei Mal vorkommen darf. [KR17, S. 496 f.]

Um Kollisionen zwischen den Adressen der Interfaces verschiedener Hersteller zu vermeiden, müssen die Hersteller einen eigenen Adressbereich beim IEEE kaufen. Dieser Adressbereich besteht aus allen Adressen, die mit einer bestimmten Bytefolge beginnen (beispielsweise würden dann alle Adressen von Hersteller A mit `aa:bb` beginnen). Die Zuweisungen der Präfixe zu den Herstellern werden online veröffentlicht[7]. Durch einen Vergleich der MAC-Adresse eines Gerätes

[7] `https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries`

mit den herausgegebenen Listen kann also ermittelt werden, wer der Hersteller eines Gerätes oder eines Netzwerkinterfaces ist. [KR17, S. 496 f.]

Analyse der offenen Ports

Manche Geräte bieten in der Standardkonfiguration verschiedene Dienste im lokalen Netzwerk an. Anhand dieser geöffneten Ports ist es teilweise möglich, das Betriebssystem zu erkennen [MSK12, S. 73]. Beispielsweise ist auf iPhones und iPads von Apple mindestens seit iOS 4.0.3 der TCP-Port 62078 immer geöffnet. Dort wird der *lockdownd*-Dienst angeboten, der unter anderem für die Synchronisation mit iTunes genutzt wird. [Zdz14, S. 5; Rüt10]

Auf den meisten Windows-Rechnern sind die TCP-Ports 135 (Microsoft RPC services), 139 (NetBIOS Session Service) und 445 (SMB) geöffnet [MSK12, S. 73]. Der Scan eines netzwerkfähigen Druckers (HP ENVY 5030, siehe Auflistung A.8) zeigt, dass mehrere Ports in Verbindung mit der Druckfunktionalität stehen, darunter Port 631 (Internet Printing Protocol) und 9100 (Printer PDL Data Stream). An diesen Ports ließe sich ein Gerät demnach eventuell als Drucker identifizieren. [Ins19]

Diese Methode ist allerdings für Fehlidentifikationen oder Täuschungen anfällig. So könnten Ports auch von anderen Betriebssystemen verwendet werden oder ein Gerät könnte seine Identität böswillig verschleiern, etwa in dem es den Port 62078 öffnet, um sich als iOS-Gerät auszugeben.

Analyse via Banner Grabbing

Beim Banner Grabbing wird eine Verbindung mit einem Dienst hergestellt, um das Betriebssystem, die Anwendungssoftware oder eine Versionsnummer aus den empfangenen Daten auszulesen. Ein Beispiel hierfür ist eine HTTP-Verbindung. Hier geben viele Server beim Verbindungsaufbau im `Server`-Header den verwendeten Webserver und das verwendete Betriebssystem an. In der Auflistung 6.2 ist zu erkennen, wie der Webserver sich als Apache 2.4.18 auf dem Betriebssystem Ubuntu identifiziert. [MSK12, S. 90]

Der Nachteil am Banner Grabbing ist, dass die Implementierung für jedes Protokoll getrennt erfolgen muss. Die Form, in der die Informationen geliefert werden, kann in jedem Protokoll anders spezifiziert sein. Manche Protokolle, darunter SMB, können komplexere Paketabfolgen erfordern, bevor die Versionsnummer des Servers preisgegeben wird (siehe Abbildung 7.3).

Analyse der Hostnamen

Die Hostnamen, die bei Geräten vor- oder fest eingestellt sind, folgen oft einem bestimmten Muster. Daraus können manchmal Rückschlüsse auf das verwendete Gerät oder das verwendete Betriebssystem geschlossen werden.

Beispielsweise verwendet das Betriebssystem Android in der Version 7.0.0 einen festen Hostnamen nach dem Muster „android-“ gefolgt von einer Geräte-ID. Zum Schutz der Privatsphäre wurde

```
pi@raspberrypi:~ $ telnet moodle.thi.de 80
Trying 194.94.240.193...
Connected to moodle.thi.de.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 25 Dec 2019 16:06:51 GMT
Server: Apache/2.4.18 (Ubuntu)
[...]
```

Auflistung 6.2: HTTP-Verbindungsaufbau mit `moodle.thi.de` (mithilfe von `telnet`, selbst angefertigt).

dieses Verhalten ab Android 8 allerdings entfernt, da die Geräte-ID eine Identifikation des Benutzers in öffentlichen Netzwerken erlaubte. Folglich kann anhand des „android-“ Präfixes ermittelt werden, dass ein Gerät Android 7 oder älter benutzt. [The16; Hog17]

Bei iPhones und iPads von Apple ist in der Betriebssystemversion 13.2.3 standardmäßig „iPhone“ bzw. „iPad“ (teils mit einem Suffix) als Hostname voreingestellt (siehe Abbildungen A.5 und A.6). Anhand dieser Hostnamen ließen sich sowohl Betriebssystem als auch Geräteart ermitteln. Microsoft Windows 10 verwendet für den voreingestellten Hostnamen das Muster „DESKTOP-“, gefolgt von einer zufällig wirkenden Zeichenkette (siehe Abbildung A.7).

Hostnamen sind in vielen Betriebssystemen frei konfigurierbar. Das hat zur Folge, dass eine Identifikation anhand des Hostnamens oft fehlschlagen oder falsche Ergebnisse liefern kann. Deswegen sollte der Hostname nicht als alleiniges Merkmal für die Identifikation eines Gerätes dienen.

Analyse des Netzwerkstacks

Verschiedene Betriebssysteme weisen kleine Unterschiede in ihrer Implementierung von Netzwerkstandards wie IP, TCP, UDP oder ICMP auf. Grund hierfür ist meist eine verschiedene Interpretation der Spezifikationen. Anhand dieser Details lässt sich ermitteln, welchen Netzwerkstack ein bestimmtes Gerät benutzt. Es kann zwischen passiver und aktiver Analyse unterschieden werden. [MSK12, S. 74, 77]

Bei der **passiven Analyse** wird der Traffic aus einer Beobachterperspektive auf bestimmte Parameter und Verhaltensweisen analysiert, um Rückschlüsse daraus zu ziehen, ohne aktiv in die Kommunikation einzugreifen. Beispielsweise verwenden Windows Vista und Windows 7 eine IP Time to Live (TTL) von 128, während Linux eine TTL von 64 benutzt. Daran lässt sich erkennen, ob auf einem Gerät Microsoft Windows als Betriebssystem eingesetzt wird. [MSK12, S. 77 f. Hje11]

Bei der **aktiven Analyse** werden speziell angefertigte Pakete als eine Art Messsonden an den Zielrechner versendet, um seine Reaktion darauf zu analysieren. Ein Beispiel für eine aktive Analyse wäre das Versenden eines *TCP FIN*-Pakets an einen offenen Port. Laut Spezifikation soll auf dieses Paket nicht geantwortet werden. Die Netzwerkstacks von Windows Vista und Windows 7 antworten allerdings mit einem *TCP FIN ACK*. Daran lässt sich das Betriebssystem festmachen. [MSK12, S. 74]

Analyse des Dynamic Host Configuration Protocols (DHCP)

Das Dynamic Host Configuration Protocol (DHCP) ermöglicht es Clients, automatisch Konfigurationsdaten aus dem Netzwerk zu beziehen. Dazu gehören unter anderem die IP-Adresse, die Subnetzmaske, der Standardgateway und die Adresse des DNS-Servers. Hierbei sendet der Client eine *DHCP Discover*-Nachricht an die Broadcast-Adresse `255.255.255.255`, um die verfügbaren DHCP-Server zu ermitteln. Darauf antworten diese mit einem *DHCP Offer*, welcher die vorgeschlagenen Konfigurationsdaten (unter anderem die IP-Adresse) enthält. Mit einem *DHCP Request* fordert der Client dann die Zuweisung dieser Konfigurationsdaten an und der Server bestätigt sie mit einem *DHCP ACK*. [KR17, S. 370 ff.]

Unterschiede in der Implementierung des DHCP können zur Ermittlung eines Fingerabdrucks genutzt werden. Anhand dieses kann ermittelt werden, um welches Betriebssystem oder Gerät es sich bei dem DHCP-Client handelt. Jede DHCP-Anfrage enthält eine Liste von Optionen, die vom Absender angegeben werden. Die Option 55 (*Parameter Request List*, siehe Abbildung 6.3) ist hierbei von besonderem Interesse. Diese enthält die Konfigurationsparameter, die der Client vom DHCP-Server anfordert. Welche Parameter und in welcher Reihenfolge diese Parameter angefordert werden, unterscheidet sich zwischen verschiedenen Clients signifikant. Deswegen kann die geordnete Liste der angeforderten Parameter als Fingerabdruck des Betriebssystems verwendet werden. [Bil13]

Die kanadische Firma *Inverse, Inc.* bietet mit *fingerbank.org* eine öffentlich zugängliche Datenbank von DHCP-Fingerabdrücken an. Diese kann nur online über eine API abgerufen werden. Als Alternative finden sich im Internet weitere Datenbanken, welche sich auch Offline benutzen lassen[8]. Dabei handelt es sich meist um Auszüge aus der Fingerbank-Datenbank. [Inva; Invb]

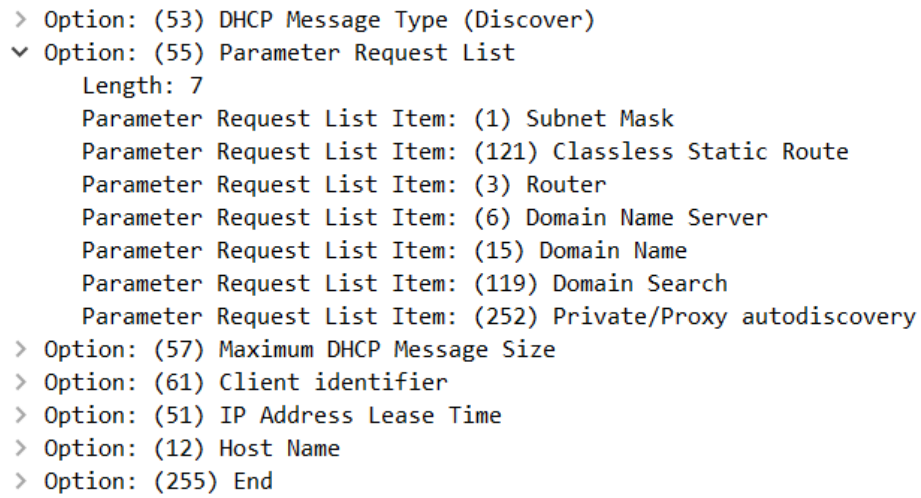


Abbildung 6.3: In der *DHCP Discover*-Nachricht eines Apple iPhone 8 (iOS 13.2.3) enthaltene Optionen (Screenshot eines Netzwerktraffic-Mitschnitts mithilfe von Wireshark, selbst angefertigt).

[8] `https://gfiber.googlesource.com/vendor/google/platform/+/master/taxonomy/dhcp.py` oder `https://github.com/inverse-inc/packetfence/blob/devel/conf/dhcp_fingerprints.conf`

6.4 Strategien zur Auflistung von angebotenen Diensten

Je nach Diensttyp sind verschiedene Strategien zur Auflistung erforderlich. Hiermit wird die Anforderung R5 (Übersicht der angebotenen Dienste) erfüllt.

Suche nach offenen Ports

Anwendungen, die Dienste in einem Netzwerk bereitstellen möchten, erstellen einen Socket auf einem TCP- oder UDP-Port. Unter der Kombination aus der IP-Adresse des Netzwerkinterfaces und der Portnummer lässt sich dieser Dienst dann aus dem Netzwerk erreichen. Solange dieser Socket offen ist und neue Verbindungen annimmt, wird der Port als offen bezeichnet.

Die Internet Assigned Numbers Authority (IANA) führt eine zentrale Liste[9], welche die Zuweisung von TCP- und UDP-Portnummern zu bestimmten Dienstnamen dokumentiert. Die IANA ist eine zentrale Organisation, welche mit der Verwaltung von bestimmten Elementen des Internets beauftragt ist. Darunter befinden sich die zentralen DNS-Server, die Vergabe von IP-Adressen und die Zuweisung von Protokollnummern (dazu gehören auch TCP-/UDP-Ports). Dabei wird der gesamte Portbereich (0 bis 65535) in drei Bereiche unterteilt: Systemports (0 bis 1023), Benutzerports (1024 bis 49151) und dynamische / private Ports (49152 bis 65535). Anhand dieser Zuweisung lässt sich erkennen, welcher Dienst normalerweise unter einem Port angeboten wird. Die Zuweisungen sind jedoch nicht bindend und müssen von Entwicklern auf freiwilliger Basis eingehalten werden. Aus diesem Grund kann es hier zu Fehlidentifikationen kommen. [Int19; Int]

Da beim Scan des gesamten Portbereichs 65535 Pakete pro Gerät versendet werden müssten, nimmt dieser zu viel Zeit in Anspruch. Deswegen werden dabei nur öfter verwendete Ports beachtet. Dazu gehören die Systemports von 0 bis 1023 sowie ergänzend der Port 62078, der auf jedem iOS-Gerät geöffnet ist. [Zdz14]

Suche nach Universal Plug and Play (UPnP)-Diensten

Zur Ankündigung und zur Suche von UPnP-Diensten im Netzwerk existiert das SSDP. Die Dienste setzen sich aus einem Search Target (ST), welcher die Art des Dienstes (beispielsweise ein Bildschirm zur Darstellung von Medien) beschreibt und einem Unique Service Name (USN), welcher die Instanz des Dienstes auf einem spezifischen Gerät identifiziert, zusammen. [Bol19]

Dienste können über SSDP bei initialer Verfügbarkeit im gesamten lokalen Netzwerk angekündigt oder bei Bedarf abgefragt werden. Beides wird über HTTP-ähnliche Anfragen an die UDP-Multicast-Adresse `239.255.255.250:1900` im Netzwerk verteilt. Bei der Suche nach Diensten im Netzwerk kann als gesuchter Typ `ssdp:all` angegeben werden, um Dienste aller Arten in Erfahrung zu bringen. [Bol19]

[9] `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`

Suche nach Bonjour-Diensten

Die DNS-SD-Spezifikation definiert eine Meta-Abfrage, welche die Auflistung aller im Netzwerk angebotenen Diensttypen erlaubt. Dies kann auch für die Verwendung bei Bonjour in Kombination mit mDNS verwendet werden. Hierfür wird nach dem Diensttyp `_services._dns-sd._udp` gesucht (siehe Abbildung 2.1). Dieser fordert die Geräte auf, alle ihre verfügbaren Dienste preiszugeben. [CK13a]

6.5 Strategien zur Erkennung von Sicherheitsproblemen

Es gibt verschiedenste Arten von Sicherheitsproblemen, die bei Geräten auftreten können. An dieser Stelle werden einige davon, welche mit dem Netzwerk im Zusammenhang stehen, betrachtet, um die Anforderungen R6 und R7 zu erfüllen.

Überprüfung auf veraltete Anwendungsversionen

Schwachstellen in Software werden vom Hersteller in der Regel mithilfe von Sicherheitsupdates behoben. Diese müssen gegebenenfalls manuell vom Besitzer eines Gerätes eingespielt werden. Deswegen kann es aufgrund von veralteten Anwendungsversionen zu einer Beeinträchtigung der Sicherheit eines Netzwerks kommen.

Die Prüfung auf veraltete Anwendungsversionen kann mithilfe des in Abschnitt 6.3 beschriebenen Banner Grabbings erfolgen. Dabei wird eine Verbindung mit einem von der Anwendung angebotenen Dienst hergestellt, um aus den übertragenen Daten die Anwendungsversion auszulesen.

Dabei ist zu beachten, dass nicht alle Anwendungen offen im Netzwerk erreichbare Dienste anbieten. Anwendungen, die rein als Client fungieren, können aus der Perspektive eines anderen Netzwerkteilnehmers nicht identifiziert werden. Dadurch wird die Nützlichkeit dieses Vorgehens stark beschränkt. Es ist deswegen darauf zu achten, dass es keine negativen Effekte auf die Sicherheit eines Netzwerks hat, wenn der Benutzer etwa fälschlicherweise in den Glauben versetzt wird, dass alle seine Software auf dem aktuellsten Stand sei.

Überprüfung auf schwache Zugangsdaten

Wie in Abschnitt 4.3 beschrieben wurde, basierte das Mirai-Botnet angeblich aus 380.000 IoT-Geräten. Diese konnten gekapert werden, da sie mit standardmäßig vorkonfigurierten Zugangsdaten, die in einer Liste gefunden werden konnten, betrieben wurden. Daran wird deutlich, dass Sicherheitslücken in Netzwerkgeräten nicht zwingend komplexer Natur sind. In diesem Fall kann auch ein Gerät, welches keine weiteren Sicherheitslücken aufweist, missbraucht werden, wenn der Hersteller davon ausgeht, dass der Kunde das Passwort ändert, dies aber nicht erfolgt. [Nat17]

Im Internet finden sich Listen mit Geräten und ihren standardmäßig voreingestellten Zugangsdaten[10]. Diese lassen sich verwenden, um einen automatischen Login beispielsweise über Telnet oder über Secure Shell (SSH) zu versuchen. Führt eines der Paare von Benutzername und Passwort zu einem erfolgreichen Login, bedeutet dies, dass der Benutzer des Gerätes die Zugangsdaten entweder nie geändert hat, oder dass das Gerät eine Hintertür in Form von fest kodierten Zugangsdaten hat.

Dabei ist zu beachten, dass manche Geräte nach wenigen erfolglosen Anmeldeversuchen weitere Versuche für eine bestimmte Dauer unterbinden. Aus diesem Grund sollten pro Zeitintervall nur eine kleine Anzahl an Zugangsdaten getestet werden. Außerdem sollte die Liste von Zugangsdaten jedes Mal durcheinander gemischt werden, um zu verhindern, dass jedes Mal nur die ersten Einträge in der Liste getestet werden, wenn das Gerät nach ein paar Versuchen blockiert.

6.6 Entwurf einer Komponentenarchitektur

Das Anwendungssystem besteht aus einem webbasierten Frontend und einem Backend. Das Backend setzt sich aus einer Datenbank, einem Webserver und einer Scanner-Komponente als Kernstück des gesamten Systems zusammen. Der Webserver bildet mit einer REST-API das Verbindungsstück zwischen Frontend und Datenbank, während die Datenbank das Verbindungsstück zwischen dem Webserver und dem Scanner ist. Eine grafische Abbildung der Komponentenarchitektur lässt sich Abbildung 6.4 entnehmen.

Um die Erweiterung und die individuelle Konfiguration des Systems zu erleichtern, ist die Scanner-Komponente in Einzelkomponenten aufgegliedert. Diese sind entweder vom Typ **Network Scanner** (Auflisten von Geräten im Netzwerk), **Service Scanner** (Auflisten von Diensten auf einem Gerät), **Vulnerability Scanner** (Erkennen von unsicheren Zugangsdaten) oder **Fingerprinter** (Identifizieren eines Gerätes).

6.7 Entwurf eines Datenmodells

Das Datenmodell soll die anfallenden Daten möglichst simpel, aber ausführlich festhalten, um später Analysen auf diesen Daten ermöglichen. Das **Gerät** (Device) bildet den Mittelpunkt des Modells und speichert die Metadaten des Gerätes (MAC-Adresse, IP-Adresse, Hostname und eine vom Benutzer vergebene Bezeichnung). Obwohl eine MAC-Adresse ein Gerät normalerweise eindeutig identifizieren kann, wird als Primärschlüssel eine generierte ID benutzt, da manche Betriebssysteme ein manuelles Ändern dieser Adresse erlauben [Hof17]. Der **Dienst** (Service) ist aufgrund seiner Kardinalität (0 bis n Dienste pro Gerät) in einer eigenen Tabelle modelliert. Genauso verhält es sich mit dem **Fingerabdruck** (Identität) und dem **Aktivitätsindikator** (Heartbeat), welcher bei jeder erfassten Aktivität des Gerätes angelegt wird. Eine grafische Abbildung des Datenmodells lässt sich Abbildung 6.5 entnehmen.

[10] https://github.com/danielmiessler/SecLists/blob/master/Passwords/Default-Credentials/default-passwords.csv

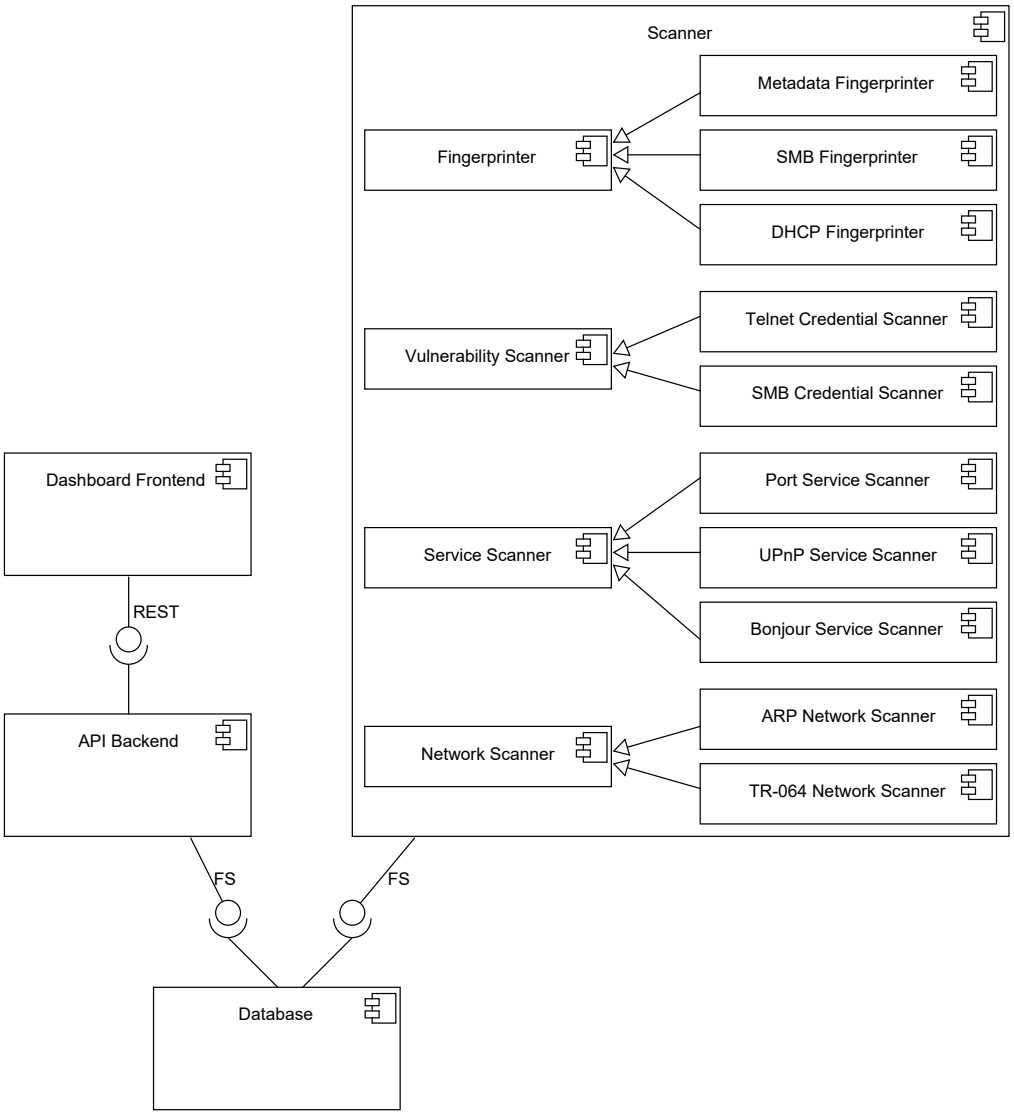


Abbildung 6.4: Komponentendiagramm.

6.8 Automatische Updates

Wie in Abschnitt 4.3 beschrieben, wird die Sicherheit von Heimnetzwerken durch Geräte, die keine regelmäßigen Sicherheitsupdates erhalten, beeinträchtigt. Um Anforderung R2 (Security by Default) zu erfüllen, muss daher eine automatische Versorgung mit Sicherheitsupdates sichergestellt werden.

Das auf dem Raspberry Pi eingesetzte Raspbian bietet mit *unattended-upgrades*[11] ein Paket zur unbeaufsichtigten (auf Englisch *unattended*) Installation von Updates. Dieses durchsucht die Paketquellen des Betriebssystems in regelmäßigen Abständen nach neuen (Sicherheits-)Updates und installiert diese automatisch. Dadurch wird sichergestellt, dass sich alle Betriebssystemkomponenten immer auf dem neuesten Stand befinden, ohne dass der Benutzer sich manuell um die Installation der Updates bemühen muss. [Cal+19]

[11] `https://packages.debian.org/buster/unattended-upgrades`

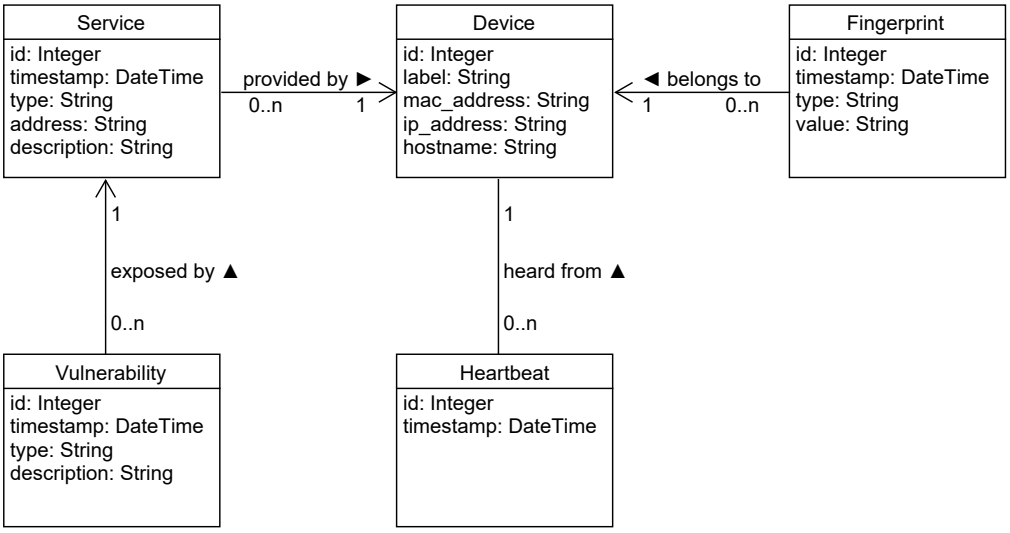


Abbildung 6.5: Datenmodell.

Durch das *unattended-upgrades*-Paket werden allerdings nur Betriebssystemkomponenten sowie Docker selbst aktualisiert. Da das System in dieser Arbeit als Container ausgeliefert wird und nicht in den Paketquellen des Betriebssystems vorhanden ist, muss dieses separat aktualisiert werden. Hierfür eignet sich das *Watchtower*[12]-Projekt. Dieses sucht regelmäßig nach aktualisierten Versionen der auf dem System laufenden Docker-Container. Wird eine neue Version eines Container-Images gefunden, wird sie heruntergeladen und der alte Container wird heruntergefahren, um dann die neue Version zu starten. [DeH+19]

6.9 Auslieferung beim Benutzer

Um Anforderung R1 (Einfache Installation) zu erfüllen, muss eine Art der Auslieferung gewählt werden, die dem Endnutzer möglichst wenig Aufwand bereitet. Das bedeutet, dass die Anwendung mit ihren Abhängigkeiten, wie beispielsweise Programmbibliotheken, der Python-Runtime oder dem Webserver, gebündelt ausgeliefert wird. Hierfür eignet sich das Konzept der Containerisierung, die in Abschnitt 2.7 erläutert wurde. Dabei wird die Anwendung in Verbindung mit einem Großteil des darunterliegenden Betriebssystems als Container ausgeliefert. Dadurch wird verhindert, dass Probleme durch Inkompatibilitäten der Anwendung mit ihrer Laufzeitumgebung auftreten. Die Basis für den Container bildet ein Debian Buster-Image, da als Entwicklungsumgebung Raspbian Buster (ein Debian Buster-Derivat für den Raspberry Pi) genutzt wurde.

Zu den Komponenten, die im Container gebündelt werden, gehören sämtliche Scanner (Network, Service, Fingerprint, Vulnerability), das API-Backend, ein *nginx*-Web-Server für das Routing der API-Zugriffe und dem Bereitstellen des Frontend-Codes und ein Prozesskontrollsystem (*supervisor*[13]) zum Start und zur Koordination der Prozesse innerhalb des Containers.

Eine Installation von Docker und das Einrichten eines Containers benötigt gewisses Fachwissen, da hier mit einem Linux-Terminal gearbeitet werden muss. Deswegen wird zusätzlich noch ein

[12] `https://github.com/containrrr/watchtower`
[13] `http://supervisord.org/`

vorkonfiguriertes Dateisystemimage bereitgestellt. Der Benutzer muss dieses Image lediglich auf eine SD-Karte schreiben und die Karte in einen Raspberry Pi 3B+ einlegen. Hierfür kann die offizielle Dokumentation[14] der Raspberry Pi Foundation befolgt werden. [Rasb]

[14] https://www.raspberrypi.org/documentation/installation/installing-images/README.md

7 Implementierung

In diesem Abschnitt werden die Besonderheiten bei der Implementierung eines Prototyps basierend auf dem zuvor erarbeiteten Entwurf dokumentiert.

7.1 Paketmanipulation

Das Serialisieren und Deserialisieren von Ethernet-, IP-, TCP-, UDP-, ARP- und DHCP-Paketen wird im Regelfall auf einer für die meisten Anwendungen transparente Art und Weise automatisch vom Betriebssystem durchgeführt. Für die Implementierung dieses Systems ist aber eine Kontrolle über die einzelnen Felder in den versendeten Paketen notwendig.

Aus diesem Grund ist es erforderlich, die Pakete für die eingesetzten Protokolle innerhalb der Anwendung manuell zu serialisieren und an einen *Raw Socket* zu übergeben. Bei *Raw Sockets* handelt es sich um Sockets, bei welchen die Behandlung der Protokolle der höheren Netzwerkschichten durch das Betriebssystem deaktiviert ist. Das bedeutet beispielsweise, dass ein *Raw Socket* für das Ethernet-Protokoll das Empfangen und Versenden von Ethernet-Paketen erlaubt, ohne dass die darüber liegenden Protokolle wie IP oder TCP durch das Betriebssystem abgewickelt werden. [Kle17]

Existierende Bibliotheken

Mit Scapy[1] existiert ein Projekt, welches den zuvor beschriebenen Anwendungsfall zum Großteil abdeckt. Dabei handelt es sich um eine Python-Bibliothek zur Paketmanipulation. Diese Bibliothek ermöglicht es, Netzwerkpakete zu senden, zu empfangen, zu untersuchen und zu fälschen. [BS19]

Bei der Implementierung mancher Features auf Basis von Scapy zeigen sich allerdings einige Probleme. Beispielsweise versucht die Bibliothek beim Versenden von 1000 Paketen an eine nicht zugewiesene IP-Adresse insgesamt 1000 Mal, die IP-Adresse mittels einer ARP-Abfrage zu einer MAC-Adresse aufzulösen. Im Gegensatz zur nativen Implementierung des Betriebssystems findet hier kein Caching von fehlgeschlagenen Anfragen statt. Wenn ein Portscan also auf einem Gerät durchgeführt wird, welches derzeit nicht mit dem Netzwerk verbunden ist, führt das zu einer signifikanten Verlangsamung des Programmflusses.

Außerdem gehen bei dem Versuch, einen Portscan mithilfe der Bibliothek durchzuführen, einige Antwortpakete ohne erkennbaren Grund verloren. Obwohl eine Analyse des Netzwerkverkehrs mithilfe des Netzwerkanalysetools Wireshark zeigt, dass das gescannte Gerät mit den richtigen

[1] `https://scapy.net/`

Paketen antwortet, werden manche *TCP SYN ACK*-Pakete nicht von Scapy erfasst. Somit gehen Teile der Ergebnisse eines Portscans verloren.

Versuche, die Fehler im Quellcode von Scapy zu beseitigen und zur Entwicklung der Bibliothek beizutragen, wären aufgrund der immensen Komplexität des Codes allerdings mit zu hohem Zeitaufwand verbunden. Aus diesem Grund wird auf den Einsatz dieser Bibliothek verzichtet und stattdessen eine rudimentäre Implementierung der benötigten Protokolle auf Basis von vorhandenen Betriebssystem- und Python-APIs erstellt.

Raw Sockets

In der Auflistung 7.1 ist zu sehen, wie ein *Raw Socket* zur Kommunikation durch das Ethernet-Protokoll erstellt wird. Die Konstante `ETH_P_ARP` signalisiert dabei, dass nur Ethernet-Pakete, welches das ARP-Protokoll kapseln, empfangen werden sollen. Dies beugt einer Überlastung des Sockets durch Pakete anderer Anwendungen vor.

Aus Sicherheitsgründen benötigt der direkte Zugriff auf Ethernet-Sockets unter Linux die *CAP_NET_RAW* Capability [Kle17]. Deswegen muss die Anwendung mit Administratorrechten ausgeführt werden. Alternativ kann der Programmdatei, welche die Sockets öffnet, auch mithilfe des `setcap`-Kommandos die entsprechende Berechtigung erteilt werden, diese ohne Administratorrechte nutzen zu dürfen [nos17].

```
import socket
ETH_P_ARP = 0x0806
socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(ETH_P_ARP))
```

Auflistung 7.1: Erstellung eines *Raw Sockets* (angelehnt an `scanner/network/arp.py`) [Kle17; Kem+19].

Serialisierung und Deserialisierung von Paketen

Da einige Netzwerkprotokolle auf Basis von *Raw Sockets* manuell implementiert werden, müssen die Strukturen dieser Pakete vor der Übergabe an den Socket manuell serialisiert werden. Hierfür eignet sich das `bitstruct`-Package[2], welches sich an das in Python 3 mitgelieferte `struct`-Package anlehnt. In Abbildung 7.2 ist sichtbar, wie das Format der Serialisierung definiert wird. Dieses besteht aus einer Menge von Buchstaben, welche beschreiben, wie eine Liste von Variablen in die binäre Darstellung überführt werden soll. `r48` steht dabei beispielsweise für einen 48-bit langen Binärstring und `u16` für einen 16-bit Integer. [Rev18]

7.2 Stau- und Überlaststeuerung

Die Überlaststeuerung (Flow Control) sorgt bei einer Netzwerkverbindung normalerweise dafür, dass Pakete nur so schnell versendet werden, wie das Ziel sie empfangen kann. Die Stausteue-

[2] `https://pypi.org/project/bitstruct/`

```python
import bitstruct

ETHERNET_HEADER_STRUCT = ">r48r48u16"
ETHERNET_HEADER_NAMES = ["dst_addr", "src_addr", "type"]

class EthernetPacket:
    def pack(self):
        return bitstruct.pack_dict(ETHERNET_HEADER_STRUCT, ETHERNET_HEADER_NAMES,
        ↪  self.__dict__) + self.data
```

Auflistung 7.2: Serialisierung eines Ethernet-Pakets (angelehnt an `scanner/protocols/ethernet.py`).

rung (Congestion Control) sorgt dagegen dafür, dass Pakete nur so schnell versendet werden, wie sie vom Netzwerk übertragen werden können. [KR17, S. 280, 297]

Eine Implementierung von ARP- und Port-Scannern mithilfe von *Raw Sockets* hat zur Folge, dass diese Mechanismen außer Betrieb gesetzt werden. Werden Pakete nun zu schnell versendet, können sie verloren gehen. Dadurch werden die Ergebnisse der Scans verfälscht, da ein Port beispielsweise als gefiltert statt offen angezeigt wird.

Aus diesem Grund wird die Empfangsroutine so gestaltet, dass die eingehenden Pakete möglichst schnell abgerufen und verarbeitet werden. Dadurch soll dem Verlust von Paketen durch einen überfüllten Empfangspuffer vorgebeugt werden. Das Versenden von Paketen soll dagegen zeitlich gestaffelt durchgeführt werden. Da eine richtige Stau- und Überlaststeuerung eine hohe Komplexität verursachen würde, wird hier auf eine einfache zeitliche Pause zwischen zwei Paketen gesetzt.

7.3 Banner Grabbing über Server Message Block (SMB)

Das SMB-Protokoll ermöglicht es unter Microsoft Windows (und anderen Betriebssystemen), Ressourcen im Netzwerk bereitzustellen. Es wird beispielsweise verwendet, wenn der Benutzer auf Dateien oder Drucker, die sich bei einem anderen Rechner befinden, zugreift. [Rou15]

Das Protokoll kann außerdem verwendet werden, um die Betriebssystem-Version hinter einer Netzwerkfreigabe zu bestimmen. Bei Verwendung von SMBv1 ist dies unter anderem mit *nmap* möglich, diese Version des Protokolls ist unter Microsoft Windows 10 allerdings seit dem Fall Creators Update aus Sicherheitsgründen deaktiviert. Bei SMBv2 und neueren Versionen sind andere Werkzeuge erforderlich, um diese Information abzurufen. [Mic19c; Gra18]

Wie in Abbildung 7.3 zu erkennen ist, wird beim SMB-Verbindungsaufbau im Rahmen der NT LAN Manager (NTLM)-Authentifizierung ein Feld gesendet, welches die Version des Betriebssystems enthält. NTLM ist ein Protokoll zur Authentifizierung von entfernten Benutzern unter Microsoft Windows. Aus der Dokumentation des Protokolls wird ersichtlich, dass es sich bei dem gesuchten Attribut um das `Version`-Feld in der `CHALLENGE_MESSAGE` handelt. Um an dieses zu gelangen, muss der SMB-Verbindungsaufbau also bis zu dem Punkt durchgeführt werden, an dem diese Challenge-Nachricht versendet wird. [Mic19a; Mic19b]

Mit *smbprotocol*[3] von Jordan Borean existiert eine quelloffene Implementierung der SMBv2 und SMBv3-Protokolle in Python. Sie besitzt allerdings keine eingebaute Funktionalität zum Abrufen der benötigten Versionsinformationen. Auf Basis der enthaltenen Beispiele kann aber zumindest ein simples Skript erstellt werden, welches eine Verbindung mit einer SMB-Freigabe herstellt. Durch eine Analyse der Verbindungsroutinen im Code mithilfe eines Python-Debuggers (beispielsweise Visual Studio Code[4]) bei gleichzeitiger Analyse der versendeten und empfangenen Pakete in Wireshark kann der interne Ablauf der Bibliothek nachvollzogen werden. Dabei fällt auf, dass alle gesendeten und empfangenen Pakete in einer Variable namens `preauth_integrity_hash_value` gespeichert werden. Diese wird ab SMB 3.1.1 dafür verwendet, um einen Hash-Wert über alle Pakete zu berechnen, der am Ende den Sitzungsschlüssel für die Verbindung bildet [Olo15].

Aus der Abbildung 7.3 wird ersichtlich, dass die gesuchte `NTLMSSP_CHALLENGE` in einem Paket vom Typ `STATUS_MORE_PROCESSING_REQUIRED` gekapselt ist. Folglich wird in den gespeicherten Paketen nach dieser Struktur gesucht. Der Inhalt dieser kann dann als `SMB2SessionSetupResponse` analysiert und das enthaltene Feld mit der Challenge extrahiert werden. Um sicherzustellen, dass es sich dabei tatsächlich um eine NTLM-Challenge handelt, kann überprüft werden, ob sie, wie von der Dokumentation vorgegeben, mit dem String „NTLMSSP\x00“ beginnt. [Mic19b]

Danach kann die Challenge mit dem *ntlm_auth*-Paket analysiert werden, um das Versionsfeld zu extrahieren. Dieses liegt fälschlicherweise nur als Integer vor, welcher keine sinnvolle Verwendung erlaubt. Deswegen muss es zunächst zurück in einen Binärstring umgewandelt werden, um dann mit dem `struct`-Paket gemäß dem dokumentierten Format zerlegt zu werden. [Mic19b]

7.4 Fingerprinting mittels des Dynamic Host Configuration Protocols (DHCP)

Um DHCP-Pakete zu empfangen, ist es ausreichend, einen *Raw Socket* für das IP-Protokoll mit dem Protokolltyp UDP zu erstellen. Mittels der zuvor beschriebenen Strategien können die so empfangenen IP-Pakete zerlegt und der Typ und die Optionen der DHCP-Anfragen ausgelesen werden.

In Abschnitt 6.3 wurden mehrere Möglichkeiten aufgeführt, DHCP-Fingerabdrücke in lesbare Namen aufzulösen. Die präziseste ist die *fingerbank.org*-Datenbank, die nur über eine REST-API abgerufen werden kann. Da dabei allerdings die Fingerabdrücke der sich neu verbindenden Geräte an einen externen Server gesendet werden müssen, geht damit ein Verlust der Privatsphäre einher. Aus diesem Grund wird neben dem Nachschlagen in einer Online-Datenbank eine zweite, rein offline funktionierende Strategie implementiert. [Invb]

Hierfür wird ein Auszug aus einer DHCP-Fingerabdruck-Datenbank von Google verwendet.[5] Da diese nur verhältnismäßig wenige Einträge enthält, können damit weniger Geräte zuverlässig erkannt werden. Um dem entgegenzuwirken, wird eine unscharfe Suche implementiert. Das bedeutet, dass nicht nach exakten Übereinstimmungen des Fingerabdrucks gesucht wird, sondern nach dem Fingerabdruck, der dem gesuchten am ähnlichsten ist. Hierfür wird der `SequenceMatcher`

[3] `https://github.com/jborean93/smbprotocol`
[4] `https://code.visualstudio.com/`
[5] `https://gfiber.googlesource.com/vendor/google/platform/+/master/taxonomy/dhcp.py`

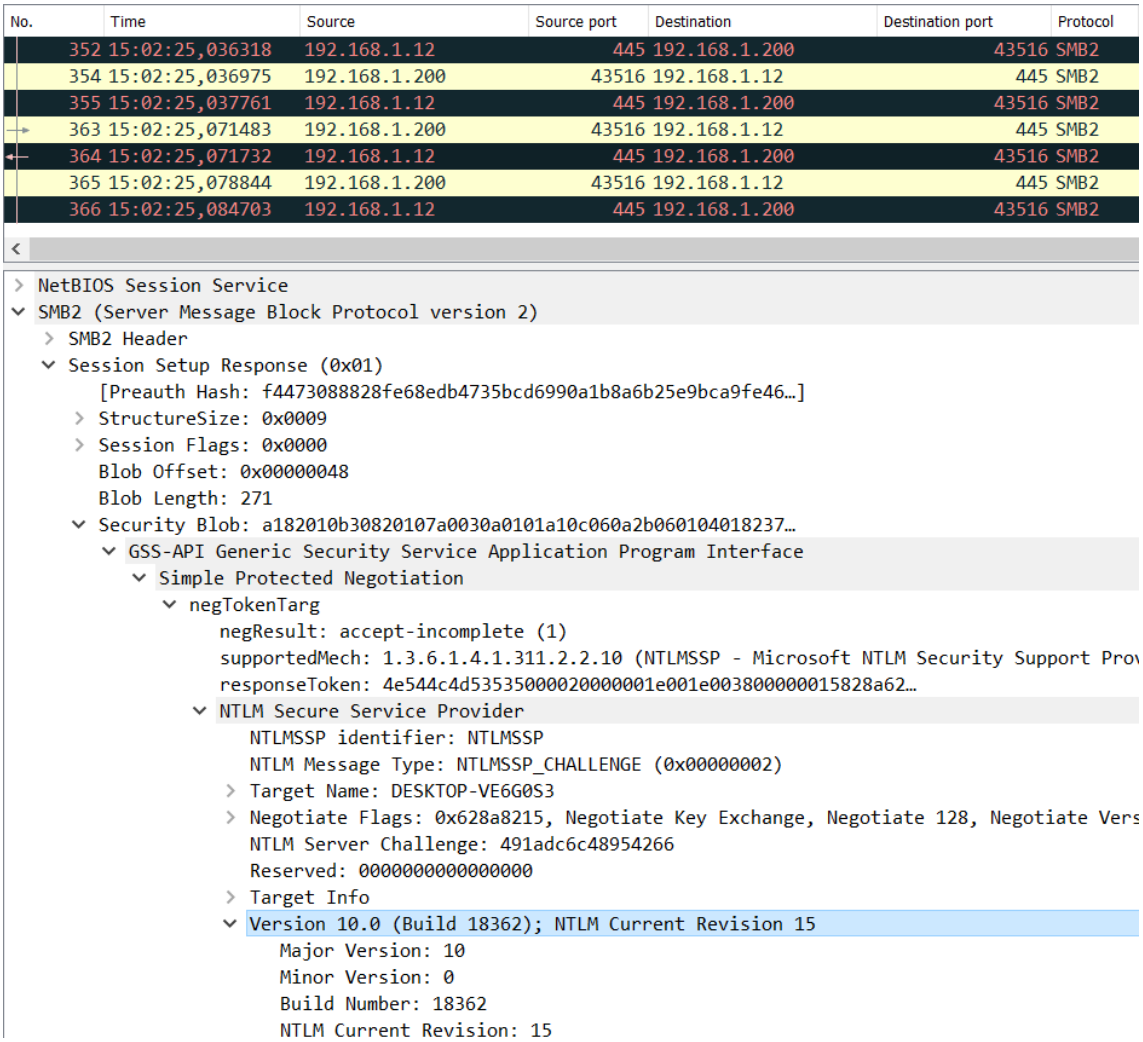


Abbildung 7.3: Paketabfolge bei einem SMB2-Verbindungsaufbau (Screenshot eines selbst angefertigten Netzwerktraffic-Mitschnitts mithilfe von Wireshark).

aus der in Python 3 enthaltenen `difflib`-Bibliothek verwendet. Dieser kann übereinstimmende Blöcke in zwei übergebenen Listen von Werten erkennen. Die Ähnlichkeit zweier Fingerabdrücke wird damit aus den Summen der Längen aller identischer Blöcke geteilt durch die Länge des längeren Fingerabdrucks bestimmt (siehe Auflistung 7.4). [Pyt20]

7.5 Umgebung für automatisierte Tests

Um fehlerfreien Code zu fördern und die Entwicklung zu erleichtern, sollte ein möglichst hoher Anteil des Codes durch automatisierte Unit- oder Integrationstests abgedeckt sein. Als Netzwerkmonitoringsystem führen einige Komponenten bei der Ausführung aber umfangreiche Interaktionen mit dem Netzwerk durch. Um diese Module unabhängig von einer bestimmten Konfiguration des Netzwerks testen zu können, bedarf es deswegen einer komplexen Testumgebung.

Hierfür müsste entweder die gesamte Netzwerkinteraktion mit sogenannten Mocks simuliert werden, oder es müsste eine reproduzierbare Netzwerkumgebung entwickelt werden, die bei jedem

```python
def compare_fingerprints(a, b):
    '''Returns a number between 0..1 that indicates the similarity between the
    ↪  two fingerprints'''

    # [...]

    matcher = SequenceMatcher(None, a, b)
    total_size = 0
    for block in matcher.get_matching_blocks():
        total_size = total_size + block.size
    return total_size / max(len(a), len(b))
```

Auflistung 7.4: Vergleich zweier DHCP-Fingerabdrucke (aus `scanner/fingerprint/dhcp.py`).

Test hochgefahren und in einen vorhersehbaren Zustand versetzt wird. Beides würde mit Komplexität einhergehen, die den Rahmen dieser Arbeit überschreitet. Deshalb werden bei der Entwicklung der Tests vorerst Parameter eines spezifischen Heimnetzwerks eingesetzt. Diese müssen bei jedem Entwickler an das jeweilige Netzwerk, in dem er sich befindet, angepasst werden.

7.6 Auslieferung beim Benutzer

Die Raspberry Pi Foundation bietet auf ihrer Webseite fertige Betriebssystemimages an, welche zur Verwendung mit ihren Einplatinencomputern auf eine SD-Karte geschrieben werden können. Diese werden mit dem *pi-gen*-Tool[6] generiert. Dabei handelt es sich um eine Sammlung von Skripten, die alle notwendigen Pakete herunterladen und zu einem startbaren Image bündeln. [Rasa; RPi19]

Das Dateisystemimage wird vom Tool in sechs Phasen (auf Englisch *stages*) gebaut:

- **Stage 0** generiert dabei ein minimales Dateisystem,
- **Stage 1** generiert ein minimales startbares System,
- **Stage 2** generiert ein minimales System mit grundlegenden Anwendungen,
- **Stage 3** generiert ein grafisches System mit einer Desktopumgebung,
- **Stage 4** generiert ein fast vollständiges Image von maximal 4 GB Größe und
- **Stage 5** generiert ein vollständiges Image mit einer großen Auswahl an Anwendungen.

[RPi19]

Das Tool ist quelloffen und kann den eigenen Bedürfnissen entsprechend angepasst werden. **Stages 3** bis **5** sind für ein Netzwerkmonitoring-System, welches lediglich über eine Weboberfläche bedient wird, nicht notwendig und können entfernt werden. Anstelle dessen wird eine neue **Stage 3** angelegt. Diese enthält *unattended-upgrades*, *Docker* und alle notwendigen Docker-Container. Am Ende des Bauvorgangs wird aus dem generierten System eine Image-Datei im `deploy`-Ordner erstellt, welche auf eine SD-Karte geschrieben werden kann. Das angepasste Projekt liegt dieser Arbeit als `watchpoint-pi-gen.zip` bei.

[6] `https://github.com/RPi-Distro/pi-gen`

7.7 Bau des Containers und des Dateisystemimages

Um die Anwendung beim Endbenutzer auszuliefern, muss aus dem Quellcode ein Container beziehungsweise ein Dateisystemimage gebaut werden. Dieser Prozess wird direkt auf einem Raspberry Pi ausgeführt. Der Ausgangspunkt sind zwei Ordner mit den benötigten Quellcodes (`watchpoint` und `watchpoint-pi-gen`). Die Kommandos werden in einem Terminal, beispielsweise über eine SSH-Verbindung, ausgeführt.

Im `watchpoint`-Ordner wird der Docker-Container wie folgt gebaut:

```
cd ~/workspace/watchpoint
sudo docker build --tag alexhorn/watchpoint:latest .
```

Auflistung 7.5: Bau des Docker-Containers.

Dadurch wird der Container unter dem `alexhorn/watchpoint:latest`-Tag abgelegt. Für den Bau des Images müssen zunächst einige Abhängigkeiten installiert werden:

```
sudo apt-get install coreutils quilt parted qemu-user-static debootstrap zerofree
↪  zip dosfstools bsdtar libcap2-bin grep rsync xz-utils file git curl
```

Auflistung 7.6: Installation der Abhängigkeiten von *pi-gen* (leicht angepasst) [RPi19].

Anschließend kann aus dem gebauten Container ein Dateisystemimage zur Installation auf einem Raspberry Pi generiert werden:

```
cd ~/workspace/pi-gen
sudo ./build.sh
```

Auflistung 7.7: Bau des Dateisystemimages.

Durch dieses Kommando wird eine `image_2019-12-26-Watchpoint-full.zip` im `pi-gen/deploy`-Ordner erzeugt, welche dem Endbenutzer ausgeliefert werden kann.

7.8 Klassendiagramm

In dieser Arbeit wird in Abbildung 7.8 lediglich ein vereinfachtes Klassendiagramm der *scanner*-Komponente dargestellt. Vollständige Klassendiagramme aller Komponenten, darunter Diagramme von *common*, *frontend* und *backend*, werden aus Platzgründen ausgelassen.

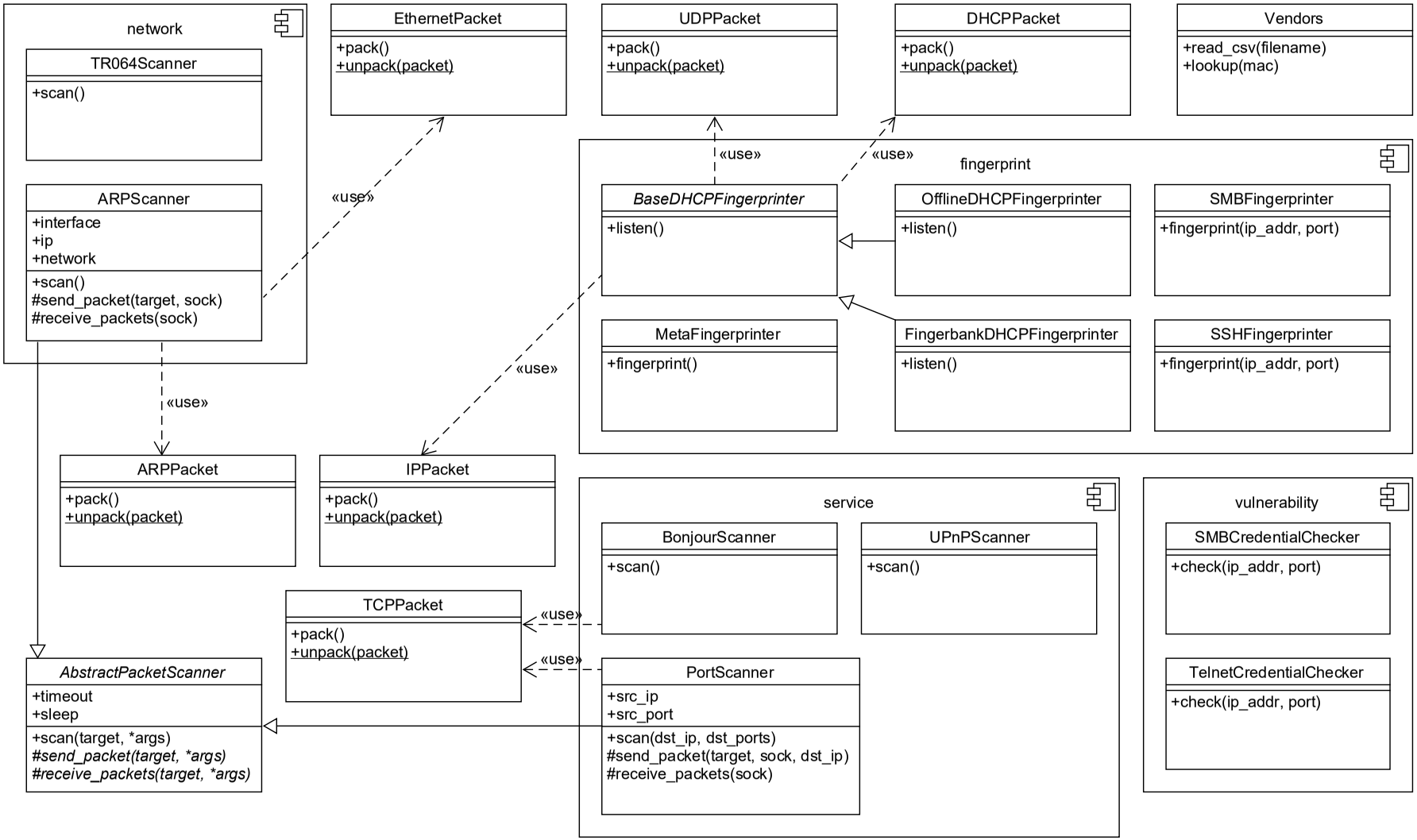


Abbildung 7.8: Vereinfachtes Klassendiagramm der *scanner*-Komponente.

7.9 Struktur des Quellcodes

Der Quellcode des entwickelten Netzwerkmonitoringsystems wird dieser Arbeit als `watchpoint.zip` angehängt. Diese ist wie folgt strukturiert:

`scanner/` enthält die Kernfunktionalität des Monitoring-Systems (siehe Abschnitte 6.2, 6.3, 6.4 und 6.5).

`backend/` enthält den Server zur Bereitstellung der Daten als REST-API.

`frontend/` enthält das Frontend zur Visualisierung der Daten. Dieses Projekt basiert auf der *Create React App*-Projektvorlage[7].

`common/` enthält den von den Modulen gemeinsam genutzten Code.

`docker/` und `docker-compose.yml` enthalten eine Docker-Konfiguration zum Bau und zum lokalen Deployment des Systems.

`README.md` enthält eine kurze Dokumentation des Projekts.

`CREDITS.md` enthält ein Quellenverzeichnis für verwendete Inhalte und Code.

Das modifizierte *pi-gen*-Projekt wird als `watchpoint-pi-gen.zip` angehängt. Diese ist wie folgt strukturiert:

`stage3/` enthält die neue Stage zur Generierung des Dateisystemimages (siehe Abschnitt 6.9).

`CREDITS.md` enthält das Quellenverzeichnis.

[7] `https://github.com/facebook/create-react-app`

8 Evaluation der Ergebnisse

Eine Evaluation des implementierten Systems zeigt, dass die meisten Anforderungen mindestens teilweise erfüllt wurden. Für einen im Rahmen einer wissenschaftlichen Arbeit angefertigten Prototypen ist dies als Erfolg zu werten. Das Aussehen der Benutzeroberfläche wird in den Abbildungen 8.2 und 8.3 gezeigt.

8.1 Erfüllung der Anforderungen

Anforderung R1 (Einfache Einrichtung) wurde, so weit es praktikabel ist, erfüllt. Es muss ein Raspberry Pi 3B+ (inklusive SD-Karte, Netzteil, Netzwerkkabel und gegebenenfalls einem Gehäuse) erworben werden. Zur anschließenden Installation des Systems muss ein Dateisystemimage mithilfe eines grafischen Tools auf eine SD-Karte geschrieben und die Karte in einen Raspberry Pi eingelegt werden.

Anforderung R2 (Security by Default) wurde erfüllt. Der Container und das Dateisystemimage sind beide vorkonfiguriert, um ohne Eingreifen des Benutzers in einem sicheren Zustand zu sein. Updates werden installiert und abgesehen von einem Webinterface sind keine Dienste aus dem Netzwerk erreichbar. Um administrative Tätigkeiten auf dem Gerät auszuführen, ist ein physischer Zugriff zum Anschluss eines Monitors und einer Tastatur notwendig.

Anforderung R3 (Enumeration der Geräte) wurde erfüllt. Es werden alle im selben Netzwerk vorhandenen Geräte angezeigt.

Anforderung R4 (Fingerprinting) wurde zum Teil erfüllt. Die Geräteart und das Betriebssystem können identifiziert werden, soweit passende Signaturen vorhanden sind. Die Betriebssystemversion kann nur bei Microsoft Windows ermittelt werden. Der Hersteller des Gerätes kann bei allen Geräten ermittelt werden.

Anforderung R5 (Enumeration der Dienste) wurde zum Teil erfüllt. Offene Ports und UPnP- sowie Bonjour-Dienste werden angezeigt.

Anforderung R6 (Alte Softwareversionen) wurde zu einem kleinen Teil erfüllt. Die Betriebssystemversionen mancher Windows- und Linux-Geräte können erkannt werden.

Anforderung R7 (Schwache Zugangsdaten) wurde zu einem Teil erfüllt. Schwache Telnet- und SMB-Zugangsdaten können erkannt werden, allerdings ist die Liste mit den Benutzernamen und Passwörtern ausbaufähig.

Anforderung R8 (Intervalle) wurde erfüllt. Die Suche wird in Abständen von fünf Minuten durchgeführt. Das Intervall kann in einer Konfigurationsdatei angepasst werden.

Anforderung R9 (Intuitive Darstellung) wurde zum Großteil erfüllt. Anhand der Fingerprinting-Informationen und der Gerätemetadaten sollte in den meisten Fällen erkennbar sein, um welches Gerät es sich handelt.

Anforderung R10 (Visuelle Hinweise) wurde erfüllt. Wenn die Aufmerksamkeit des Benutzers aufgrund einer erkannten Schwachstelle benötigt wird, wird ihm dies in der Netzwerkübersicht mit einem Warnsymbol signalisiert (siehe Abbildung 8.2).

Anforderung R11 (Geräteübergreifendes Design) wurde erfüllt. Die Webseite ist sowohl auf Smartphones als auch auf PCs benutzbar.

8.2 Performance

Wie Tabelle 8.1 zu entnehmen ist, ergibt der Median von fünf Durchläufen eine Gesamtlaufzeit von 5 Minuten und 24 Sekunden für ein Netzwerk, in dem fünf Geräte gefunden wurden. Dies sollte für die meisten Netzwerke eine akzeptable Geschwindigkeit sein. Dabei ist anzumerken, dass jeweils nur die derzeit aktiven Geräte erkannt werden können. Im realen Betrieb werden Daten über einen längeren Zeitraum gesammelt, was in den meisten Fällen dazu führen sollte, dass schlussendlich alle im Netzwerk vorhandenen Geräte erfasst werden.

Ein Abgleich der im Haus vorhandenen Geräte mit der erstellten Netzwerkübersicht offenbar keine Geräte, die vom Scan nicht erfasst wurden. Auch wurden die angebotenen Dienste und Betriebssystemversionen aller Windows-Rechner im Haus gesammelt.

Durchlauf	Startzeit	Endzeit	Dauer	Erkannte Geräte
1.	17:04:16	17:09:36	5:20	6
2.	17:14:36	17:20:04	5:28	6
3.	17:25:04	17:30:28	5:24	5
4.	17:35:28	17:40:52	5:24	5
5.	17:45:52	17:51:08	5:16	4

Abbildung 8.1: Performanceevaluation in einem exemplarischen Heimnetzwerk.

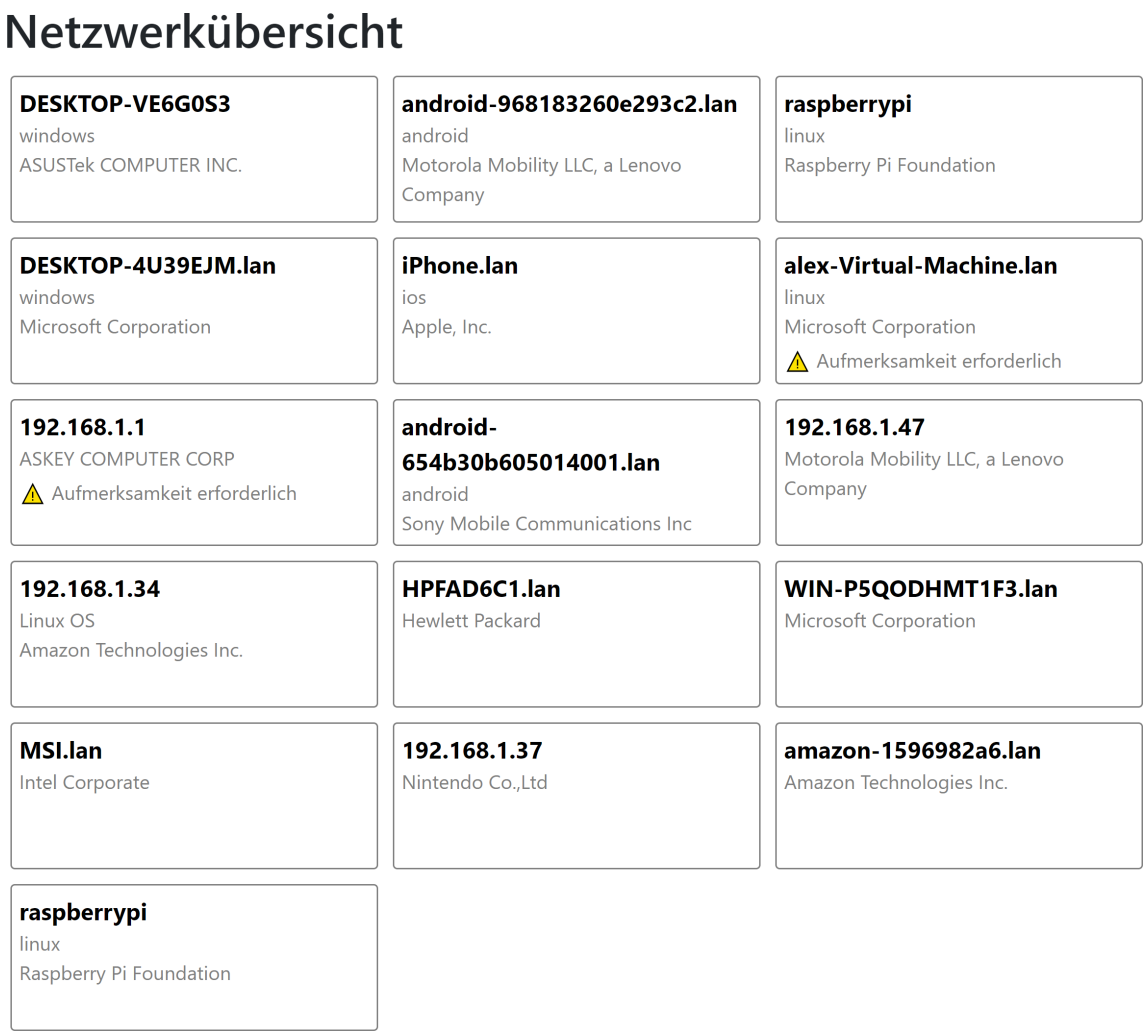


Abbildung 8.2: Netzwerkübersicht auf der Startseite der Weboberfläche.

DESKTOP-VE6G0S3 Umbenennen

Details

Letzte Aktivität	Fri, 10 Jan 2020 15:39:21 GMT
IP-Adresse	192.168.1.12
MAC-Adresse	04:d4:c4:06:bb:76
Hersteller	ASUSTek COMPUTER INC.
Hostname	DESKTOP-VE6G0S3
class	pc
operating_system	windows
operating_system_version	10.0.18362

Schwachstellen

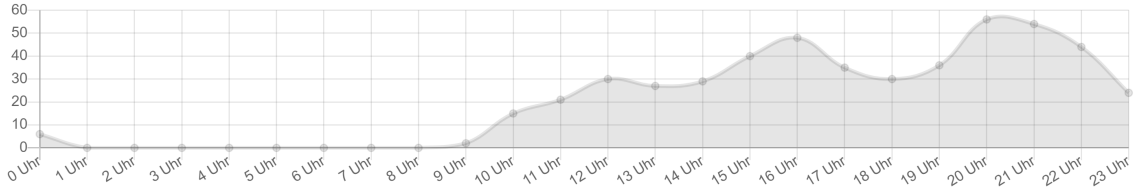
Keine Schwachstellen gefunden

Angebotene Dienste

Typ	Adresse
bonjour	_nvstream_dbd._tcp.local.
port	135/tcp
port	139/tcp
port	445/tcp

Aktivität

Nach Stunden



Nach Wochentag

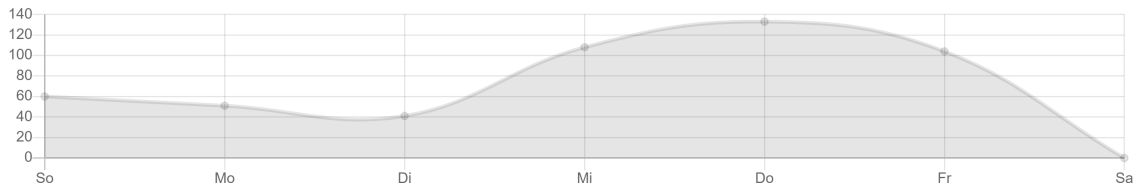


Abbildung 8.3: Detaillierte Anzeige beim Klick auf ein Gerät.

9 Zusammenfassung und Ausblick

Die Struktur der Heimnetzwerke verändert sich im Laufe der Jahre fortwährend. Durch die Verbreitung des Internets wurde das Heimnetzwerk vor Jahren zum Standard in den meisten Haushalten. Später vermehrte sich die Anzahl der im Netzwerk eingebundenen Geräte und es kamen neue Arten wie Smartphones, Tablets oder Smart Home-Geräte dazu. Bei der Betrachtung aus einer Sicherheitsperspektive offenbart sich diese Entwicklung, insbesondere in Bezug auf IoT-Geräte, als nicht unbedenklich.

Maßnahmen, die diese Probleme am Ursprung beheben, wären sicherlich am effektivsten. Doch um dies zu erreichen müsste ein immenser Druck auf die Hersteller ausgeübt werden. Deswegen wurde in dieser Arbeit ein System vorgestellt, welches die Symptome unsicherer Geräte kontrollierbarer macht. Ähnlich wird in Unternehmensumfeldern bereits seit längerer Zeit Netzwerkmonitoring eingesetzt, um die Gefahren im Netzwerk zu überblicken.

In seiner derzeitigen Form kann das entwickelte System dem Benutzer alle im Netzwerk vorhandenen Geräte in einer Übersicht darstellen. Zu jedem Gerät ermittelt es die IP- und MAC-Adresse, den Hersteller und den Hostnamen. Nach Möglichkeit werden verschiedene Parameter analysiert, um die Geräteart, das Betriebssystem und die Betriebssystemversion zu ermitteln. Die von dem Gerät angebotenen offenen Ports, UPnP- und Bonjour-Dienste werden ebenfalls ermittelt. Eine kleine Auswahl an Schwachstellen, konkret schwache Zugangsdaten für Telnet- und SMB-Zugänge, werden ebenfalls erkannt. Um das Verhalten des Gerätes zu verdeutlichen werden auch Aktivitätsdaten gesammelt und nach Stunden und Wochentagen aufgeschlüsselt visualisiert (siehe Abbildung 8.3).

Der implementierte Prototyp bietet in seiner aktuellen Form bereits einen Mehrwert für den Benutzer. Allerdings ist er an einigen Stellen noch ausbaufähig. So ist die Erkennung von Softwareversionen noch unzureichend, da bislang nur die Betriebssystemversionsnummern von manchen Windows- und Linux-Geräten erkannt werden. Die Erkennung von Sicherheitslücken ist ebenfalls ausbaufähig, da eine Vielzahl an weiteren Schwachstellen existiert, welche aus dem Netzwerk geprüft werden könnten. Hier könnten noch die Zugangsdaten für weitere Diensttypen, wie SSH oder webbasierte Oberflächen, geprüft werden. Des Weiteren ist es vorstellbar, automatisch auf ausnutzbare Schwachstellen, wie etwa EternalBlue, zu prüfen.

Im Bereich der Benutzererfahrung können die Installationsroutinen freundlicher gestaltet werden, etwa in dem der Benutzer von einem interaktiven Assistenten durch die Konfiguration geleitet wird. Außerdem könnte ein Passwortschutz für die Weboberfläche die Sicherheit verbessern. Um die internationale Verbreitung des Systems zu fördern, wäre zuletzt eine Übersetzung der Oberfläche in andere Sprachen sinnvoll.

Literatur

[App19] Apple Inc. *Bonjour*. 2019. URL: https://developer.apple.com/bonjour/ (besucht am 16. 12. 2019).

[AVM] AVM Computersysteme Vertriebs GmbH. *Entwicklungssupport*. URL: https://avm.de/service/schnittstellen/ (besucht am 09. 11. 2019).

[AVM16] AVM GmbH. *AVM Technical Note. Automatische Konfiguration der FRITZ!Box*. Jan. 2016. URL: https://avm.de/fileadmin/user_upload/Global/Service/Schnittstellen/AVM_Technical_Note_-_Konfiguration_ueber_TR-064.pdf (besucht am 09. 11. 2019).

[Ben19] Prof. Dr. Oliver Bendel. *Smart Home*. In: *Gabler Wirtschaftslexikon*. Springer Fachmedien Wiesbaden GmbH, Jan. 2019. URL: https://wirtschaftslexikon.gabler.de/definition/smart-home-54137/version-368820 (besucht am 13. 01. 2020).

[BI17] E. Bertino und N. Islam. „Botnets and Internet of Things Security“. In: *Computer* 50.2 (Feb. 2017), S. 76–79. ISSN: 1558-0814. DOI: 10.1109/MC.2017.62.

[Bil13] Olivier Bilodeau. *DEF CON 19 - Olivier Bilodeau - Fingerbank - Open DHCP Fingerprints Database*. Nov. 2013. URL: https://www.youtube.com/watch?v=YR61Ydd0amM (besucht am 22. 11. 2019).

[Bit19] Bitdefender. *Bitdefender BOX. Benutzerhandbuch*. Jan. 2019. URL: https://download.bitdefender.com/resources/media/materials/box/v2/user_guide/2020/BOX_UserGuide_v2_de.pdf (besucht am 30. 10. 2019).

[Bol19] William Boles. *Discovering what's out there*. Apr. 2019. URL: https://williamboles.me/discovering-whats-out-there-with-ssdp/ (besucht am 13. 11. 2019).

[BS19] Philippe Biondi und the Scapy community. *Introduction*. 2019. URL: https://scapy.readthedocs.io/en/latest/introduction.html (besucht am 21. 12. 2019).

[Bun] Bundesamt für Sicherheit in der Informationstechnik. *SYS.4.4 Allgemeines IoT-Gerät*. URL: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKompendium/bausteine/SYS/SYS_4_4_Allgemeines_IoT-Ger%C3%A4t.html (besucht am 13. 12. 2019).

[Bun07] Bundesamt für Sicherheit in der Informationstechnik. *Sichere Anbindung von lokalen Netzen an das Internet (ISi-LANA). BSI-Leitlinie zur Internet-Sicherheit (ISi-L)*. 2007. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Internetsicherheit/isi_lana_leitlinie_pdf.pdf?__blob=publicationFile&v=1 (besucht am 11. 01. 2020).

[Bun19] Bundesamt für Sicherheit in der Informationstechnik (BSI). *Die Lage der IT-Sicherheit in Deutschland 2019*. Okt. 2019. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2019.pdf?__blob=publicationFile&v=6 (besucht am 25. 10. 2019).

[Cal] Paulino Calderon. *File smb-vuln-ms17-010*. URL: https://nmap.org/nsedoc/scripts/smb-vuln-ms17-010.html (besucht am 01. 11. 2019).

[Cal+19] Luke Call u. a. *UnattendedUpgrades*. Aug. 2019. URL: https://wiki.debian.org/UnattendedUpgrades (besucht am 28. 12. 2019).

[Che11] Dr. Stuart Cheshire. *Bonjour Network Discovery and Connectivity*. 2011. URL: https://developer.apple.com/videos/play/wwdc2011/211/ (besucht am 16. 12. 2019).

[CK13a] S. Cheshire und M. Krochmal. *DNS-Based Service Discovery*. RFC 6763. Feb. 2013. URL: https://tools.ietf.org/html/rfc6763.

[CK13b] S. Cheshire und M. Krochmal. *Multicast DNS*. RFC 6762. Feb. 2013. URL: https://tools.ietf.org/html/rfc6762.

[Clo] Cloudflare, Inc. *What is DNS cache poisoning? | DNS spoofing*. URL: https://www.cloudflare.com/learning/dns/dns-cache-poisoning/ (besucht am 10. 01. 2020).

[DeH+19] Brian DeHamer u. a. *Watchtower*. Dez. 2019. URL: https://github.com/containrrr/watchtower (besucht am 09. 01. 2019).

[Doma] Domotz, Inc. *Features. The Tools You Need to Monitor and Manage Remote Networks*. URL: https://www.domotz.com/features.php (besucht am 02. 11. 2019).

[Domb] Domotz, Inc. *The Domotz Box*. URL: https://www.domotz.com/domotz-box.php (besucht am 02. 11. 2019).

[Eik18] Ronald Eikenberg. „Netzwerkpolizei. Netzwerküberwachung mit der Fingbox“. In: *c't* 1 (2018), S. 52.

[Eur19] European Union Agency for Cybersecurity. *ENISA Threat Landscape Report 2018*. Jan. 2019. DOI: 10.2824/622757. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018/at_download/fullReport (besucht am 28. 10. 2019).

[Fin] Fing. *Download the Fing App network toolkit*. URL: https://www.fing.com/products/fing-app (besucht am 01. 11. 2019).

[Gra18] Robert Graham. *SMB version detection in masscan*. Juni 2018. URL: https://blog.erratasec.com/2018/06/smb-version-detection-in-masscan.html (besucht am 23. 11. 2019).

[Hat19] Lucy Hattersley. *Raspberry Pi 3B+ Specs and Benchmarks*. März 2019. URL: https://magpi.raspberrypi.org/articles/raspberry-pi-3bplus-specs-benchmarks (besucht am 27. 11. 2019).

[Hje11] Erik Hjelmvik. *Passive OS Fingerprinting*. Nov. 2011. URL: https://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting (besucht am 25. 12. 2019).

[Hof17] Chris Hoffman. *How (and Why) to Change Your MAC Address on Windows, Linux, and Mac*. Juli 2017. URL: https://www.howtogeek.com/192173/how-and-why-to-change-your-mac-address-on-windows-linux-and-mac/ (besucht am 20. 12. 2019).

[Hog17] Giles Hogben. *Changes to Device Identifiers in Android O*. Apr. 2017. URL: https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html (besucht am 01. 12. 2019).

[Hüb18] Frank Hüber. *Bitdefender Box im Test: Netzwerk-Wachhund schützt vor Botnetzen und Phishing*. März 2018. URL: https://www.computerbase.de/2018-03/bitdefender-box-test/ (besucht am 30. 10. 2019).

[IEE15] IEEE Internet Initiative. *Towards a definition of the Internet of Things (IoT)*. Mai 2015. URL: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf (besucht am 13. 12. 2019).

[Ins19] Insecure.Com LLC. *nmap-services*. 2019. URL: https://svn.nmap.org/nmap/nmap-services (besucht am 23. 11. 2019).

[Int] Internet Assigned Numbers Authority. *About us*. URL: https://www.iana.org/about (besucht am 01. 12. 2019).

[Int19] Internet Assigned Numbers Authority. *Service Name and Transport Protocol Port Number Registry*. Nov. 2019. URL: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (besucht am 01. 12. 2019).

[Inva] Inverse inc. *About*. URL: https://fingerbank.org/about/ (besucht am 22. 11. 2019).

[Invb] Inverse inc. *License*. URL: https://fingerbank.org/license/ (besucht am 22. 11. 2019).

[Kek] Anton Keks. *Angry IP Scanner. About*. URL: https://angryip.org/about/ (besucht am 02. 11. 2019).

[Kem+19] Fred N. van Kempen u. a. *if_ether.h*. Juni 2019. URL: https://github.com/torvalds/linux/blob/master/include/uapi/linux/if_ether.h (besucht am 11. 01. 2020).

[Kle17] Andi Kleen. *raw - Linux IPv4 raw sockets*. Sep. 2017. URL: http://man7.org/linux/man-pages/man7/raw.7.html (besucht am 20. 12. 2019).

[KR17] James F. Kurose und Keith W. Ross. *Computer networking. A top-down approach*. seventh edition, global edition. Literaturverzeichnis Seite 769-810. Boston: Pearson, 2017. ISBN: 978-1-292-15359-9.

[LD18] Stefan Luber und Andreas Donner. *Was ist Netzwerk-Monitoring?* Aug. 2018. URL: https://www.ip-insider.de/was-ist-netzwerk-monitoring-a-642648/ (besucht am 12. 12. 2019).

[Lex] Lexico.com. *home network*. URL: https://www.lexico.com/definition/home_network (besucht am 12. 12. 2019).

[Lyo10] Gordon Lyon. *Nmap network scanning. Official Nmap project guide to network discovery and security scanning*. [Nachdr. Zero-Day release May 2008]. Sunnyvale, CA: Insecure.Com, 2010. ISBN: 978-0-9799587-1-7.

[Mic19a] Microsoft Corporation. *[MS-NLMP]: NT LAN Manager (NTLM) Authentication Protocol*. Nov. 2019. URL: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/b38c36ed-2804-4868-a9ff-8dd3182128e4 (besucht am 24. 11. 2019).

[Mic19b] Microsoft Corporation. *2.2.1.2 CHALLENGE_MESSAGE*. Feb. 2019. URL: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/801a4681-8809-4be9-ab0d-61dcfe762786 (besucht am 23. 11. 2019).

[Mic19c] Microsoft Corporation. *SMBv1 is not installed by default in Windows 10 Fall Creators Update and Windows Server, version 1709 and later versions*. Juni 2019. URL: https://support.microsoft.com/en-us/help/4034314/smbv1-is-not-installed-by-default-in-windows (besucht am 23. 11. 2019).

[MSK12] Stuart McClure, Joel Scambray und George Kurtz. *Hacking Exposed 7. Network Security Secrets and Solutions*. 7. ed. New York [u.a.]: McGraw Hill, 2012. ISBN: 978-0-07178028-5.

[Nag19a] Nagios Enterprises, LLC. *Nagios Core vs. Nagios XI: 4 Key Differences*. Okt. 2019. URL: https://www.nagios.com/news/2019/10/nagios-core-vs-nagios-xi/ (besucht am 22. 12. 2019).

[Nag19b] Nagios Enterprises, LLC. *Nagios XI. Enterprise Server and Network Monitoring Software*. 2019. URL: https://www.nagios.com/products/nagios-xi/ (besucht am 22. 12. 2019).

[Nat17] National Cybersecurity and Communications Integration Center. *Alert (TA16-288A). Heightened DDoS Threat Posed by Mirai and Other Botnets*. Department of Homeland Security, Okt. 2017. URL: https://www.us-cert.gov/ncas/alerts/TA16-288A (besucht am 23. 12. 2019).

[Nat18] National Cybersecurity and Communications Integration Center. *Security Tip (ST04-015). Understanding Denial-of-Service Attacks*. Department of Homeland Security, Juni 2018. URL: https://www.us-cert.gov/ncas/tips/ST04-015 (besucht am 25. 10. 2019).

[Nat19] National Cybersecurity and Communications Integration Center. *Security Tip (ST04-014). Avoiding Social Engineering and Phishing Attacks*. Department of Homeland Security, Aug. 2019. URL: https://www.us-cert.gov/ncas/tips/ST04-014 (besucht am 25. 10. 2019).

[NB09] Kim S. Nash und Alyson Behr. *Network Monitoring Definition and Solutions*. Juni 2009. URL: https://www.cio.com/article/2438133/network-monitoring-definition-and-solutions.html (besucht am 13. 12. 2019).

[nos17] nos. *Linux Raw Socket Permissions Issue*. Sep. 2017. URL: https://stackoverflow.com/a/46485527 (besucht am 31. 10. 2019).

[Olo15] Edgar A Olougouna. *SMB 3.1.1 Pre-authentication integrity in Windows 10*. Aug. 2015. URL: https://blogs.msdn.microsoft.com/openspecification/2015/08/11/smb-3-1-1-pre-authentication-integrity-in-windows-10/ (besucht am 24. 11. 2019).

[Olt16] Jon Oltsik. *Network Security Monitoring Trends*. Aug. 2016. URL: https://www.cisco.com/c/dam/en/us/products/collateral/security/stealthwatch/esg-research-insight.pdf (besucht am 20. 12. 2019).

[OWA19] OWASP Foundation. *OWASP Risk Rating Methodology*. Juni 2019. URL: https://www.owasp.org/index.php?title=OWASP_Risk_Rating_Methodology&oldid=252633 (besucht am 11. 12. 2019).

[Pau11] Sachar Paulus. *Basiswissen sichere Software. Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering*. 1. Aufl. Literaturverz. S. 273 - 279. Heidelberg: dpunkt-Verl., 2011. ISBN: 978-3-89864-726-7.

[Pyt20] Python Software Foundation. *difflib — Helpers for computing deltas*. Jan. 2020. URL: https://docs.python.org/3/library/difflib.html (besucht am 11. 01. 2020).

[Rasa] Raspberry Pi Foundation. *Downloads*. URL: https://www.raspberrypi.org/downloads/ (besucht am 27. 11. 2019).

[Rasb] Raspberry Pi Foundation. *Installing operating system images*. URL: https://www.raspberrypi.org/documentation/installation/installing-images/README.md (besucht am 27. 12. 2019).

[Rev18] Erik Moqvist Revision. *bitstruct - Interpret strings as packed binary data*. 2018. URL: https://bitstruct.readthedocs.io/en/latest/ (besucht am 11. 01. 2020).

[Rou15] Margaret Rouse. *Server Message Block (SMB) -Protokoll*. Apr. 2015. URL: https://www.computerweekly.com/de/definition/Server-Message-Block-SMB-Protokoll (besucht am 26. 12. 2019).

[Rou16] Margaret Rouse. *CPE (Customer Premises Equipment)*. Aug. 2016. URL: https://www.computerweekly.com/de/definition/CPE-Customer-Premises-Equipment (besucht am 13. 11. 2019).

[RPi19] RPi-Distro. *pi-gen*. Dez. 2019. URL: https://github.com/RPi-Distro/pi-gen (besucht am 26. 12. 2019).

[Rüg19] Hannes Rügheimer. *Ist die Fritzbox noch die Schnellste?* Mai 2019. URL: https://www.pressreader.com/germany/pc-magazin/20190503/281603831878172 (besucht am 13. 11. 2019).

[Rüt10] Christiane Rütten. „Ungesicherte Einsichten. Sicherheit von Apps für Android und iPhone“. In: (Okt. 2010). URL: https://www.heise.de/ct/artikel/Ungesicherte-Einsichten-1901539.html?seite=all (besucht am 18. 11. 2019).

[Sch15] M. Schiefer. „Smart Home Definition and Security Threats“. In: *2015 Ninth International Conference on IT Security Incident Management IT Forensics*. Mai 2015, S. 114–118. DOI: 10.1109/IMF.2015.17.

[Siv+16] Vijay Sivaraman u. a. „Smart-Phones Attacking Smart-Homes“. In: *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '16. Darmstadt, Germany: ACM, 2016, S. 195–200. ISBN: 978-1-4503-4270-4. DOI: 10.1145/2939918.2939925. URL: http://doi.acm.org/10.1145/2939918.2939925.

[SQLa] SQLite project. *About SQLite*. URL: https://www.sqlite.org/about.html (besucht am 21. 12. 2019).

[SQLb] SQLite project. *Frequently Asked Questions*. URL: https://www.sqlite.org/faq.html (besucht am 21. 12. 2019).

[Sta] Statista GmbH. *Smart Home*. URL: https://www.statista.com/outlook/279/137/smart-home/germany (besucht am 11. 01. 2019).

[Sta19a] Stack Exchange Inc. *Developer Survey Results. 2019*. 2019. URL: https://insights.stackoverflow.com/survey/2019 (besucht am 25. 11. 2019).

[Sta19b] StatCounter. *Desktop vs Mobile vs Tablet Market Share Worldwide. Jan - Dec 2018*. 2019. URL: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/2018 (besucht am 25. 12. 2019).

[Sta19c] Statistisches Bundesamt (Destatis). *Bevölkerung und Erwerbstätigkeit. Haushalte und Familien Ergebnisse des Mikrozensus*. Juli 2019. URL: https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Bevoelkerung/Haushalte-Familien/Publikationen/Downloads-Haushalte/haushalte-familien-2010300187004.pdf?__blob=publicationFile (besucht am 21. 12. 2019).

[Sta19d] Statistisches Bundesamt (Destatis). *Private Haushalte in der Informationsgesellschaft – Nutzung von Informations- und Kommunikationstechnologien*. Juli 2019. URL: https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Einkommen-Konsum-Lebensbedingungen/IT-Nutzung/Publikationen/Downloads-IT-Nutzung/private-haushalte-ikt-2150400187004.pdf?__blob=publicationFile (besucht am 19. 12. 2019).

[Sto+15] Keith Stouffer u. a. *Guide to Industrial Control Systems (ICS) Security. Supervisory Control and Data Acquisition (SCADA) Systems, Distributed Control Systems (DCS), and Other Control System Configurations such as Programmable Logic Controllers (PLC)*. Mai 2015. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf (besucht am 20. 12. 2019).

[SUS19] SUSE Software Solutions Germany GmbH. *Containerization*. 2019. URL: https://susedefines.suse.com/definition/containerization/ (besucht am 22. 12. 2019).

[The15] The Broadband Forum. *TR-064. LAN-Side DSL CPE Configuration*. Techn. Ber. Aug. 2015. URL: https://www.broadband-forum.org/technical/download/TR-064_Corrigendum-1.pdf (besucht am 09. 11. 2019).

[The16] The Android Open Source Project. *ConnectivityService.java*. Okt. 2016. URL: https://android.googlesource.com/platform/frameworks/base/+/android-7.0.0_r14/services/core/java/com/android/server/ConnectivityService.java#661 (besucht am 01. 12. 2019).

[The19] The Computer Language Co Inc. *Definition of: home network*. 2019. URL: https://www.pcmag.com/encyclopedia/term/44319/home-network (besucht am 12. 12. 2019).

[UPn15] UPnP Forum. *UPnP Device Architecture 2.0*. Techn. Ber. Feb. 2015. URL: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v2.0.pdf (besucht am 16. 12. 2019).

[Upt19] Eben Upton. *Raspberry Pi numbers get stale fast. We sold our thirty-millionth unit some time last week (we think Tuesday).* Dez. 2019. URL: https://twitter.com/EbenUpton/status/1205646606504275968 (besucht am 09. 01. 2019).

[Was10] Rick Wash. „Folk Models of Home Computer Security“. In: *Proceedings of the Sixth Symposium on Usable Privacy and Security*. SOUPS '10. Redmond, Washington, USA: ACM, 2010, S. 1–16. ISBN: 978-1-4503-0264-7. DOI: 10.1145/1837110.1837125. URL: http://doi.acm.org/10.1145/1837110.1837125.

[Zdz14] Jonathan Zdziarski. „Identifying back doors, attack points, and surveillance mechanisms in iOS devices“. In: *Digital Investigation* 11.1 (2014), S. 3–19. ISSN: 1742-2876. DOI: 10.1016/j.diin.2014.01.001. URL: http://www.sciencedirect.com/science/article/pii/S1742287614000036.

Anhang

Faktor	Bewertung	Begründung
Threat Agent Factors		
Skill level	8 / 9	Fortgeschrittene IT-Sicherheitskenntnisse notwendig
Motive	4 / 9	Hängt von Stellung des Opfers ab
Opportunity	6 / 9	Eventuell Zugriff aufs Heimnetzwerk notwendig
Size	7 / 9	Entweder Teilnehmer im Heimnetzwerk oder gesamtes Internet
Vulnerability Factors		
Ease of discovery	3 / 9	Verwundbare Ziele eventuell schwer zu finden
Ease of exploit	3 / 9	Vorbedingungen müssen erfüllt sein
Awareness	6 / 9	Sicherheitslücken können öffentlich dokumentiert sein[1]
Intrusion detection	8 / 9	Logs werden vermutlich nicht gesichtet
Technical Impact Factors		
Loss of confidentiality	7 / 9	Geräte können sensible Informationen enthalten
Loss of integrity	7 / 9	Sensible Daten können beschädigt werden
Loss of availability	7 / 9	Zugriff auf Geräte kann verwehrt werden
Loss of accountability	8 / 9	IP-Adressen vielleicht ermittelbar
Business Impact Factors		
Financial damage	7 / 9	Missbrauch sensibler Daten möglich
Reputation damage	7 / 9	Rufschädigung mithilfe sensibler Daten möglich
Non-compliance	2 / 9	Im privaten Kontext von geringer Relevanz
Privacy violation	7 / 9	Angriffe im großen Maßstab denkbar

[1] Im Internet finden sich Verzeichnisse von öffentlich Bekannten Schwachstellen, wie z.B. `https://cve.mitre.org/`, `https://www.exploit-db.com/` oder `https://www.rapid7.com/de/db/`

Abbildung A.1: OWASP Risk Rating eines Missbrauch von Schwachstellen in veralteter Software.

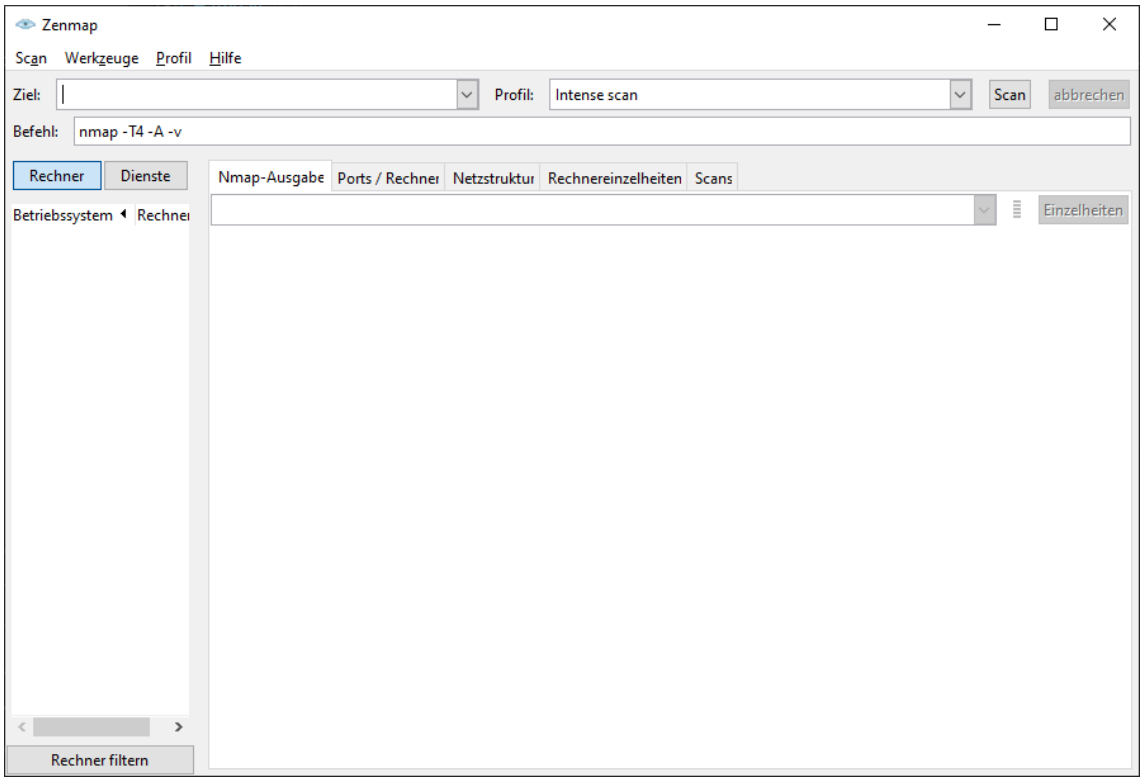


Abbildung A.2: Benutzeroberfläche von Zenmap (Screenshot, selbst angefertigt).

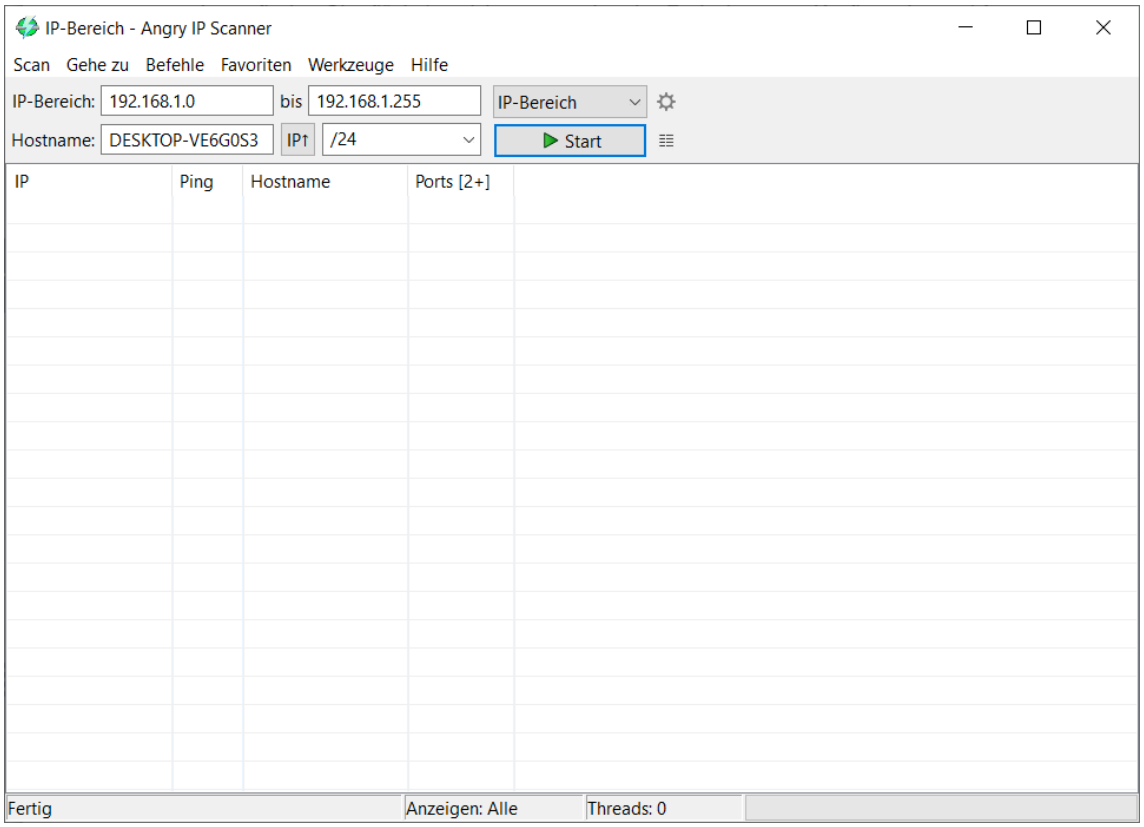


Abbildung A.3: Benutzeroberfläche vom Angry IP Scanner (Screenshot, selbst angefertigt).

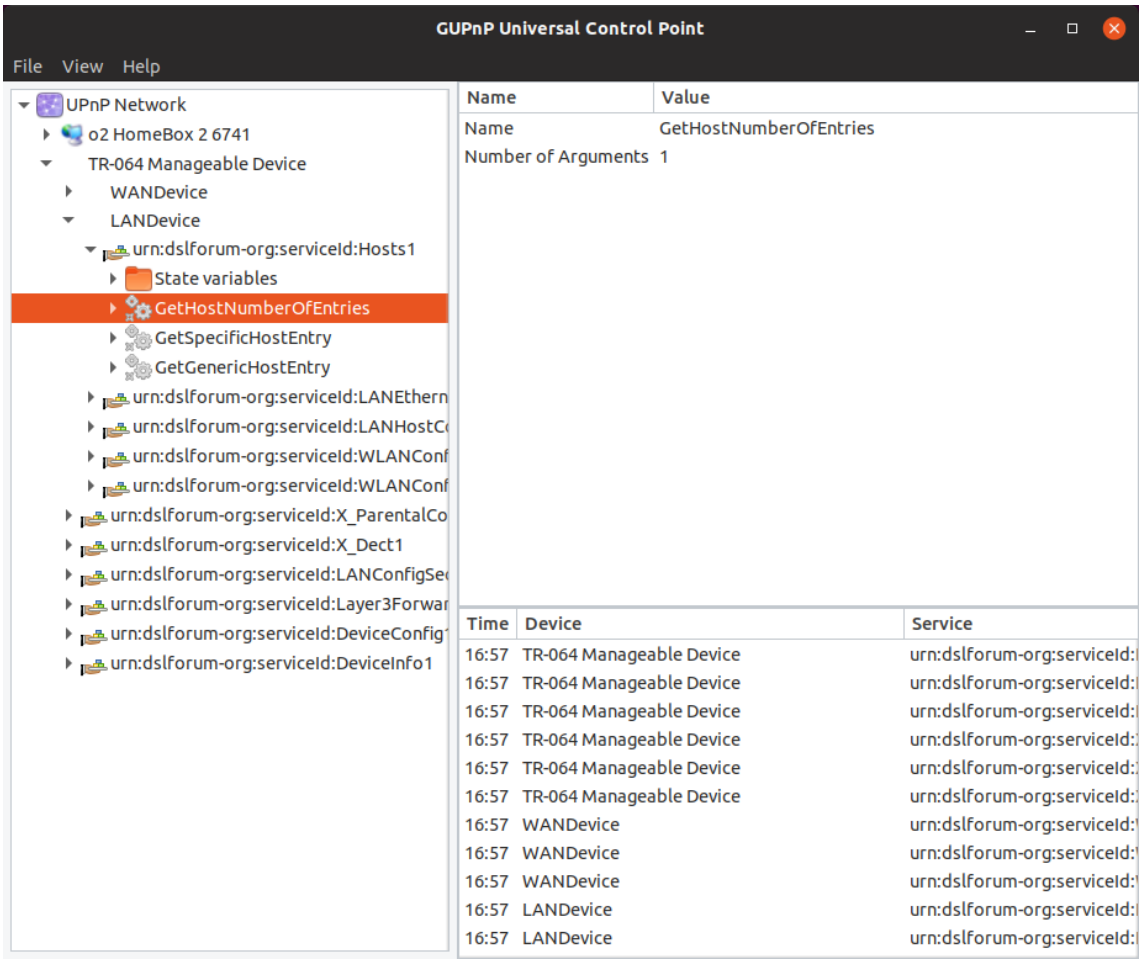


Abbildung A.4: Auflistung der TR-064-Funktionalität über GUPnP Universal Control Point (Screenshot, selbst angefertigt).

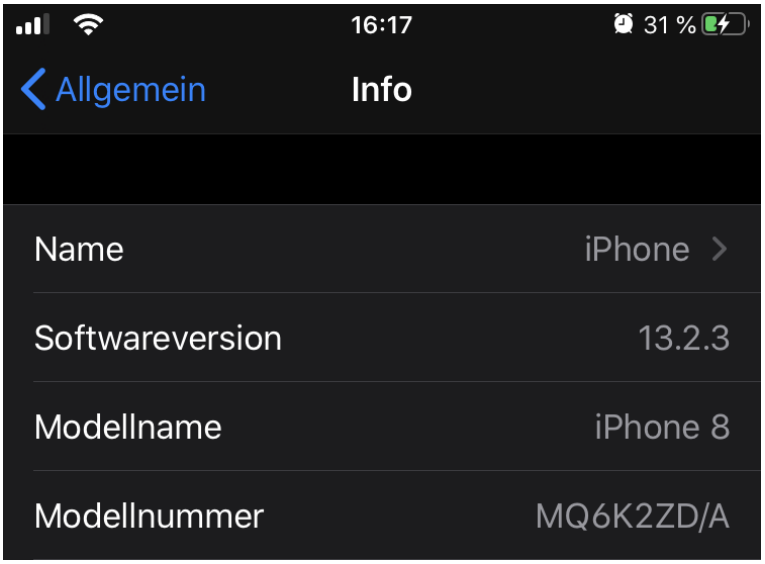


Abbildung A.5: Voreingestellter Hostname auf einem iPhone 8 (Screenshot, selbst angefertigt, zugeschnitten).

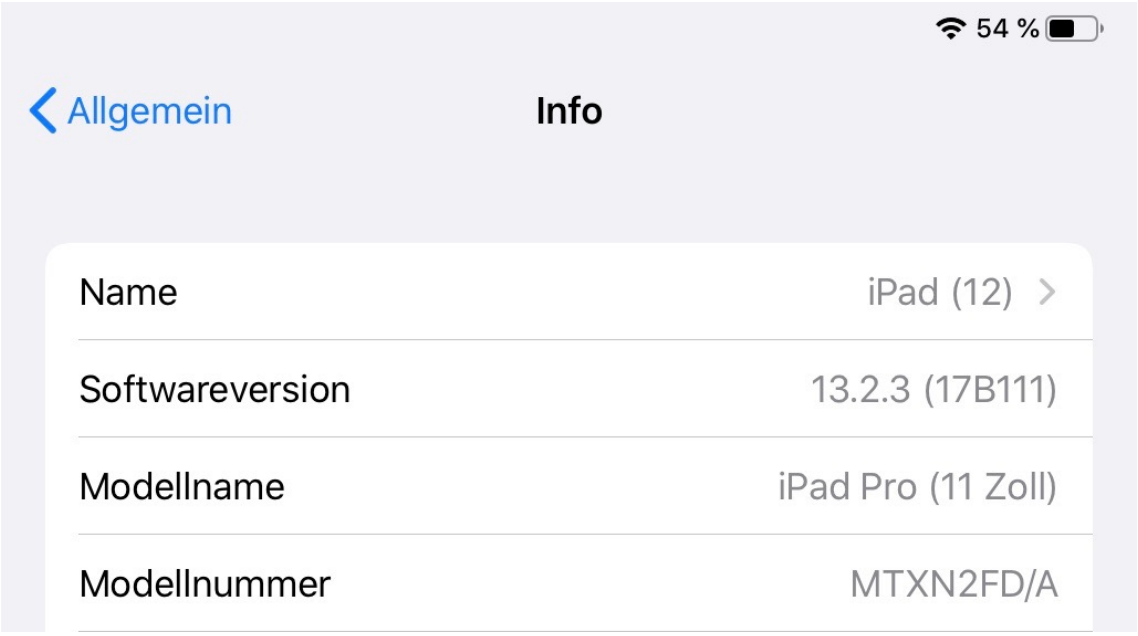


Abbildung A.6: Voreingestellter Hostname auf einem iPad Pro (Screenshot, angefertigt von Thomas Heinrichs, selbst zugeschnitten).

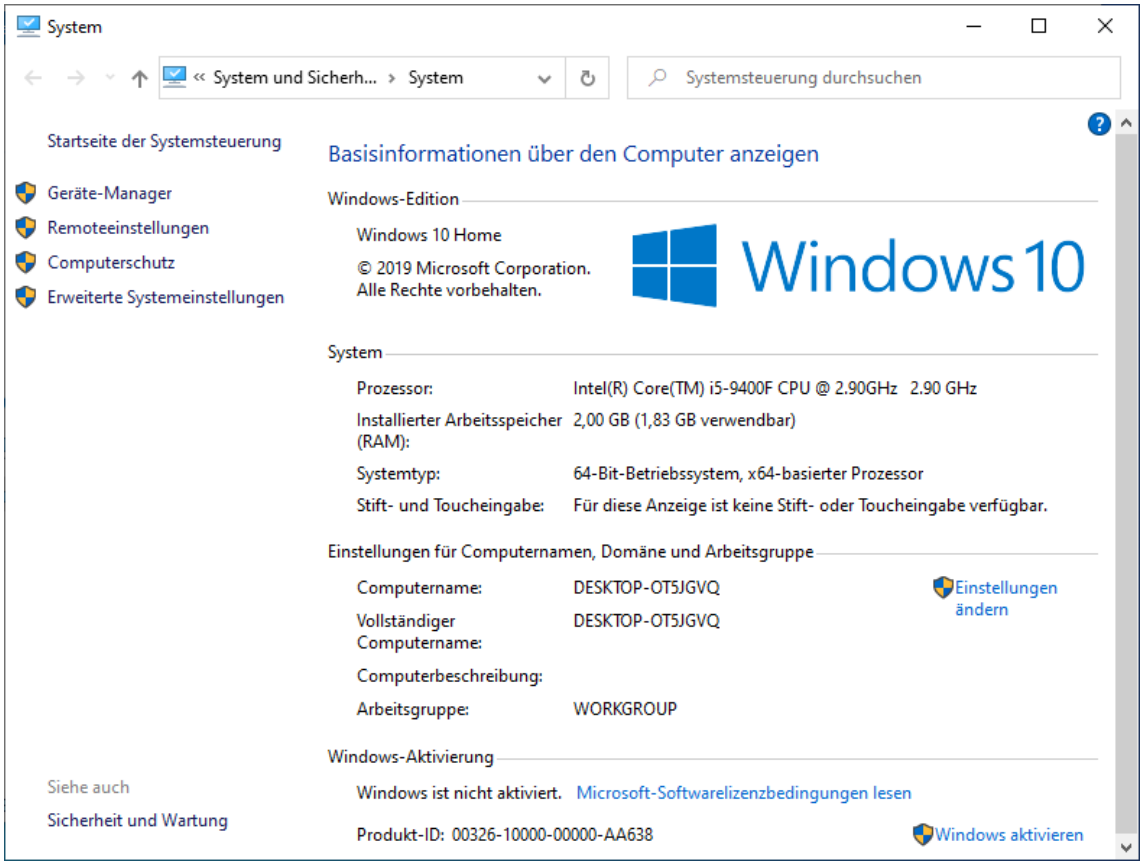


Abbildung A.7: Voreingestellter Hostname in Windows 10 (Screenshot, selbst angefertigt, zugeschnitten).

```
pi@raspberrypi:~ $ nmap 192.168.1.192
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-23 13:49 GMT
Nmap scan report for unknownace2d3fad6c1 (192.168.1.192)
Host is up (0.0068s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
631/tcp  open  ipp
8080/tcp open  http-proxy
9100/tcp open  jetdirect

Nmap done: 1 IP address (1 host up) scanned in 0.80 seconds
```

Auflistung A.8: nmap-Scan eines netzwerkfähigen Druckers (Modell: HP ENVY 5030, selbst durchgeführt).

Erklärung nach § 18 Abs. 4 Nr. 7 APO THI

Ich erkläre hiermit, dass ich die Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ingolstadt, 16. Januar 2020

Alexander Horn