Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Computer Science Department

Bundesamt
für Sicherheit in der
Informationstechnik

Abschlussarbeit

im Bachelorstudiengang Informatik

Untersuchung der Sicherheit von OpenWrt anhand der BSI TR-03148 mittels eines OpenWrt betriebenen Heim-Routers

von Henry Weckermann

Erstbetreuer: **Prof. Dr.-Ing Markus Ullmann**
Zweitbetreuer: **Prof. Dr. Norbert Jung**

Eingereicht am: **TODO**

Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt bzw. nicht veröffentlicht.

Bonn, den 8. Januar 2021

Henry Weckermann

Abstract

In an increasingly digitalized world, the network router often forms the boundary between the home network and the internet. An increasing number of attacks on these very home networks for the spread of ransomware and botnets is observed worldwide [1, p. 11-15]. This makes ways and means for manufacturers and customers to evaluate the security of their products all the more important. The goal of this thesis was to evaluate the open-source router operating system OpenWrt using the “Technical Guideline 03148 - Secure Broadband Routers“ of the German Federal Office for Information Security. Furthermore, solutions for possible shortcommings of OpenWrt should have been worked out in case it cannot meet the requirements of the guideline. Moreover, the source code of various open source router operating systems were statically analyzed to address another possibility for security verification. Finally, the results of this analysis was compared with the results of the “Home Router Security Report 2020“.

To support the premise that OpenWrt is compliant with the Technical Guideline, all applicable test cases were tested against the published test criteria. Furthermore, the source code of seven contemporary open source router firmware images was statically analyzed using the “Firmware Analysis and Comparison Tool“ so that the results were comparable to the “Home Router Security Report 2020“. The results showed that OpenWrt fails to comply in 22% of the test cases, but the predicted effort to correct these weaknesses is low. The static code analysis showed that the open source firmware, while not without shortcomings, is superior to the proprietary firmware which was analysed in the “Home Router Security Report“.

The results show that, with some adjustments, OpenWrt can present a safer alternative to proprietary pre-installed router firmware and that open source firmware performs comparatively well when using static code analysis. Conducting all of the guidelines’s test cases would have allowed a more accurate conclusion. Ultimately, a repetition of the static code analysis with a larger set of open-source firmware is desirable in order to establish a higher degree of comparability.

Zusammenfassung

In einer zunehmend digitalisierten Welt bildet der Netzwerk-Router häufig die Grenze zwischen dem Heimnetzwerk und dem Internet. Weltweit wird eine zunehmende Anzahl an Angriffen auf eben diese Heimnetze für die Verbreitung von Ransomware und Bot-Netzen beobachtet [1, p. 11-15]. Umso wichtiger sind Mittel und Wege für Hersteller und Kunden die Sicherheit ihrer Produkte zu evaluieren. Das Ziel dieser Arbeit war es das quelloffene Router-Betriebssystem OpenWrt mittels der „Technischen Richtlinie 03148 - Sichere Breitband Router“ des deutschen Bundesamtes für Sicherheit in der Informationstechnik (BSI) zu prüfen. Ferner sollten Lösungen für eventuelle Defizite erarbeitet werden, falls OpenWrt den Anforderungen der TR nicht entsprechen kann. Anschließend wurde der Quellcode verschiedener quelloffener Router-Betriebssysteme statisch analysiert, um eine weitere Möglichkeit der Sicherheitsüberprüfung zu thematisieren. Die Ergebnisse dieser Analyse wurden abschließend mit den Ergebnissen des „Home Router Security Reports 2020“ verglichen.

Um die Annahme zu stützen, dass OpenWrt TR-konform ist, wurden alle anwendbaren Testfälle anhand der veröffentlichten Prüfkriterien getestet. Weiterhin wurde der Quellcode von sieben quelloffenen Router Firmware-Abbildern statisch mit dem „Firmware Analysis and Comparison Tool“ analysiert, sodass die Ergebnisse mit dem „Home Router Security Report 2020“ vergleichbar waren. Die Ergebnisse zeigten, dass OpenWrt 22% der Technischen Richtlinie nicht besteht, der Aufwand aber gering ist, diese Defizite auszubessern. Die statische Code-Analyse zeigte, dass die quelloffene Firmware zwar nicht ohne Mängel ist, jedoch im Vergleich der proprietären Firmware überlegen ist.

Die Ergebnisse zeigen, dass OpenWrt mit einigen Anpassungen ggf. eine sichere Alternative zu proprietärer vorinstallierter Firmware bilden kann und die quelloffene Firmware vergleichsweise gut abschneidet. Die vollständige Durchführung aller Testfälle der TR hätte eine genauere Aussage erlaubt. Letztlich ist eine Wiederholung der statischen Code-Analyse mit einem größeren Korpus erstrebenswert, um eine höhere Vergleichbarkeit herzustellen.

Abkürzungsverzeichnis

BSI	Bundesamt für Sicherheit in der Informationstechnik
CCC	Chaos Computer Club
CPE	Common Platform Enumeration
CSRF	Cross-Site-Request-Forgery
CSS	Cascading Style Sheets
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DUT	Device under Test.
EOL	End Of Life
FACT	Firmware Analysis and Comparison Tool
FKIE	Fraunhofer-Institut für Kommunikation, Informationsverarbeitung und Ergonomie
LuCI	OpenWrt web user interface
GOT	Global Offset Table
GPL	GNU General Public License
ICS	Implementation Conformance Statement
IoT	Internet of Things
ISP	Internet Service Provider
LAN	Local Area Network
NAT	Network Address Translation
NX-Bit	No eXecute Bit
OpenWrt	Open Wireless Router
PIE	Position-Independent Executables
RELRO	RELocation Read-Only
SCP	Secure Copy
SDK	Software Developer Kit
SE	Secure Element
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SOHO	Small Office, Home Office
TEE	Trusted Execution Environment
TFTP	Trivial File Transfer Protocol
VoIP	Voice over IP
VPN	Virtual Private Network
VPS	Virtual Private Server
WAN	Wide Area Network
WPA	Wi-Fi Protected Access
WPS	Wi-Fi Protected Setup

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Ausgangspunkt

Das Internet wird ein zunehmend wichtigerer Teil des menschlichen Lebens. Öffentliche Hotspots, internetfähige Alltags-Geräte (IoT-Geräte) und mobiles Arbeiten von Zuhause sind nur einige Beispiele für technologische Neuerungen, welche ohne das Internet nicht möglich wären. Die rund 38 Mio. Netzanbindung an DSL-, Kabel-, oder Glasfaser-Anschlüsse in Deutschland werden in Heimnetzen und Kleinunternehmen überwiegend durch Netzwerk-Router realisiert [2]. In vielen Fällen bildet der Router die direkte Schnittstelle zwischen dem Internet und dem privaten Netzwerk. So stellt dieser durch Paketfilter und eine Firewall meist auch die einzige zentrale Sicherheitskomponente zum Schutz des Netzwerkes bereit. Ein erfolgreicher Angriff auf den Router bietet einem Angreifer unzählige Möglichkeiten, in das Netz einzugreifen und so immensen Schaden anzurichten. Neben bekannten Zielen wie private Daten und Passwörtern kann der Router auch als Teil eines Bot-Netzwerks für Distributed Denial-of-Service (DDoS) verwendet werden oder als Einfallstor auf weitere Geräte des Netzwerkes [3]. In Korrelation mit den stark steigenden Fällen von Cyberkriminalität im privaten und wirtschaftlichen Umfeld zeigt dies wie wichtig ein von Werk aus geschützter Router mit einer sicheren Konfiguration ist [1].

Handelsübliche Router, wie sie in Privathaushalten und Small Office, Home Office (SoHo) Umgebungen eingesetzt werden sind bereits mit einem proprietären Betriebssystem bespielt. Die Sicherheit dieser meist proprietären Distribution kann nur mit großem Aufwand von Endnutzern verifiziert werden und Sicherheitsupdates nur vom Hersteller veröffentlicht werden. Hersteller können in der zunehmend kürzer werdenden Zeit zwischen neuen Iterationen von Malware meist nicht in angemessener Zeit reagieren, um Sicherheitsupdates zur Verfügung zu stellen. Quelloffene Router Firmware wie OpenWrt, DD-Wrt, Tomato oder LibreCMC bieten eine Alternative zu den vorinstallierten, proprietären Betriebssystemen der Router [4, 5, 6, 7]. Diese Projekte können vollständig eingesehen, modifiziert und kompiliert werden, sodass die Sicherheit des Produktes einfacher evaluiert werden kann. Ebenfalls können aufgrund der hohen Anzahl an Mitwirkenden Sicherheits- und Funktionsupdates schnel-

ler entwickelt und veröffentlicht werden. Umfangreiche Überprüfungen dieser Projekte, wie z.B. anhand der BSI TR-03148: Sichere Broadband Router, werden allerdings aufgrund des hohen Zeit- bzw. Kostenaufwands selten durchgeführt, sodass diese auch eine Zertifizierung nicht erlangen können [8]. Eine solche Zertifizierung könnte ungeschulten Endnutzern auch diese quelloffenen Router-Betriebssysteme als Alternativen näherbringen und somit zu einem höheren Sicherheitsniveau in privater und SOHO Netzwerkinfrastruktur führen.

1.2 Was ist OpenWrt?

OpenWrt (Open Wireless RouTer) ist ein quelloffenes Netzwerk-Betriebssystem für Router, welches auf GNU/Linux basiert und durch eine GNU General Public License (GPL) lizensiert ist[4]. Die Installation umfasst einen Bootloader, ein Linux-Kernel, ein eigenes Dateisystem und ausgewählte Anwendungen. Es kann auf Routern, Switches und Wireless Access Points eingesetzt werden, um die vorinstallierte Firmware vollständig zu ersetzen [9]. Es bietet neben standardmäßiger Router Funktionalität einen eigenen Paketmanager, über welchen ca. 3800 (Stand 01.11.20) weitere Pakete installiert werden können [10]. Dies bietet viele weitere Einsatzmöglichkeiten und Funktionen, welche vom Hersteller nicht oder unzureichend unterstützt werden. Ebenfalls beinhaltet die Installation von OpenWrt BusyBox, einen SSH-Dienst, und LuCI, einem Web-Interface, sodass dem Nutzer über den root-Benutzeraccount vollständiger Zugriff auf das Gerät gewährt wird. Nach derzeitigem Stand werden über 1700 Geräte von ca. 270 Herstellern von OpenWrt unterstützt [11]. Diese Anzahl Geräte kann unter anderem deshalb unterstützt werden, da OpenWrt nur minimale Ressourcen auf dem Endgerät benötigt. Nach eigenen Angaben kann die derzeitige Version auf Geräten installiert werden, welche 4MB Flash Speicher und 32MB RAM besitzen. Ab der nächsten „Major Release“ Version (20.XX) werden 8MB Flash und 64MB RAM vorausgesetzt [11]. Diese Voraussetzungen sind jedoch bei den meisten Geräten der letzten Jahre gegeben. OpenWrt zeichnet sich ebenfalls dadurch aus, dass es sich nicht nur um eine statische Firmware handelt, sondern ebenfalls um ein komplettes Framework zur Entwicklung und Erstellung von angepassten Firmware Versionen. Ebenfalls zeichnet sich OpenWrt dadurch aus, dass Geräte solange unterstützt werden, wie sie die minimalen Systemanforderungen erfüllen. Dies steht im Gegensatz zu den meisten proprietären Betriebssystemen, welche nur einige Jahre lang Funktions- und Sicherheitsupdates erhalten und nach ihrem sog. „End of Life“ (EOL) nicht mehr sicher betrieben werden können und ausgetauscht werden müssen. Auch wenn in der Entwicklungsgeschichte von OpenWrt viel für die Benutzerfreundlichkeit des Betriebssystems getan wurde, ist es nicht unbedingt für Laien geeignet. Trotz des Managements über die Weboberfläche, erweist sich die Einrichtung ohne Grundkenntnisse als anspruchsvoll.

Die Entwicklung von OpenWrt begann 2004, nachdem der amerikanische Hersteller Linksys zuvor einen Router auf den Markt brachte, dessen Firmware zu Teilen ebenfalls unter der GPL Lizenz stand und somit öffentlich verfügbar sein musste. Die erste Veröffentlichung von OpenWrt erfolgte im Januar 2006 mit Version 0.9 (White Russian). Seitdem wurde das Projekt stetig weiterentwickelt (siehe Abbildung 1.1). 2016 spaltete sich eine Gruppe Mitwirkender aufgrund interner Diskrepanzen ab und gründete das LEDE Projekt. Jedoch wurde LEDE bereits 2018 wieder in OpenWrt integriert, sodass beide Projekte nun wieder zusammen unter einem Namen entwickelt werden. Die derzeit aktuelle Version ist 19.07.5, welche am 09.12.2020 veröffentlicht wurde (vergleiche Anhang C.1) [12].

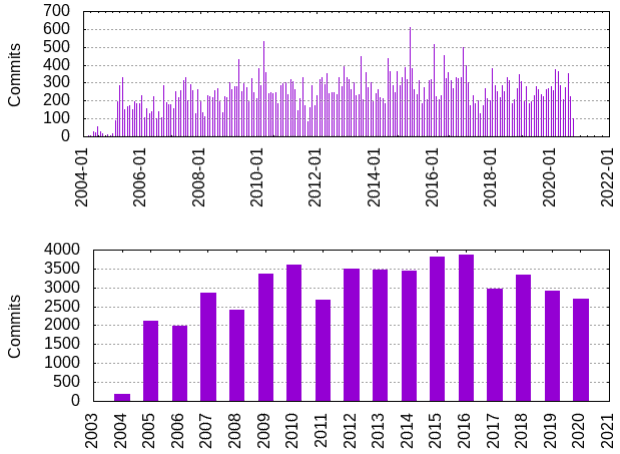


Abbildung 1.1: Zeigt die Git Commits pro Monat und pro Jahr des OpenWrt Git Repositories. Der dargestellte Zeitraum ist 2004 bis 2020. Seit 2005 werden jedes Jahr mindestens 2000 Commits gemacht. Die Grafik wurde mit dem Programm GitStats erstellt (http://gitstats.sourceforge.net).

1.3 Relevanz und Verwendung von OpenWrt

Die Webseite des OpenWrt Projektes verzeichnete im Jahre 2020 bis einschließlich November 1.261.500 Besucher, sowie 52,4 Millionen Seitenaufrufe. Insgesamt wurden bis November 2020 bereits 16,44TB Daten abgerufen [13]. Die aktuelle Version von OpenWrt (19.07.4) wurde dabei allein im November 1981 Mal heruntergeladen. Ebenfalls wurde Version 18.06.8 noch 935 Mal angefragt. Zusammen wurden ca. 10000 Firmware-Abbilder im November 2020 heruntergeladen [13]. Wie die Daten zeigen ist OpenWrt keinesfalls ein kleines Projekt mit nur wenigen Interessierten, sondern eine nachgefragte Alternative für Heimrouter, Unternehmen und Entwickler. Es lässt sich nur schwer abschätzen wie die

Verteilung zwischen dem privaten und wirtschaftlichen Einsatz der Firmware genau ist, jedoch ist eine mehrheitliche Nutzung im privaten Umfeld zu vermuten, da die Downloadzahlen eine Tendenz zu Alltagsroutern, anstelle von professionellen Geräten, zeigen (siehe Abbildung 1.3). OpenWrt ist dennoch nicht nur für Heimrouter relevant, sondern zeichnet sich auch in seinem Nutzen für Unternehmen und Entwickler aus. Es bietet Unternehmen die Möglichkeit ein Netz zu betreiben, welches sie vollständig mit quelloffener Software realisieren und steuern können. Ebenfalls bietet es Dienstleistungsunternehmen einen Weg hochgradig maßgeschneiderte Netzstrukturen für ihre Kunden zu entwerfen, welche quelloffen und leicht anpassbar sind. So können neue oder geänderten Funktionen über ein Paket bereitgestellt und verteilt werden.

Month	Unique visitors	Number of visits	Pages	Hits	Bandwidth
Jan 2020	132,873	197,956	1,127,395	4,665,469	1.67 TB
Feb 2020	110,371	163,646	1,002,759	4,104,171	1.61 TB
Mar 2020	122,633	183,308	1,002,048	5,175,434	1.66 TB
Apr 2020	129,317	189,456	988,607	4,377,669	1.68 TB
May 2020	118,101	173,179	925,788	5,681,240	1.84 TB
Jun 2020	134,365	184,142	790,987	3,189,593	1.17 TB
Jul 2020	97,788	140,153	712,274	7,458,245	1.59 TB
Aug 2020	100,393	143,850	661,420	3,240,342	1.39 TB
Sep 2020	101,636	145,296	767,185	3,216,887	1.21 TB
Oct 2020	124,108	177,852	790,254	3,692,070	1.35 TB
Nov 2020	89,915	127,187	655,436	7,628,492	1.28 TB
Dec 2020	0	0	0	0	0
Total	1,261,500	1,826,025	9,424,153	52,429,612	16.44 TB

Abbildung 1.2: Hier sind verschiedene Statistiken der OpenWrt Webseite aufgeführt. Es werden die Besucher, die Anzahl der Besuche, die Anzahl der Seitenaufrufe, die Nummer der Treffer und die verbrauchte Bandbreite pro Monat und für das gesamte Jahr dargestellt. (Erstellt: 30.11.2020)

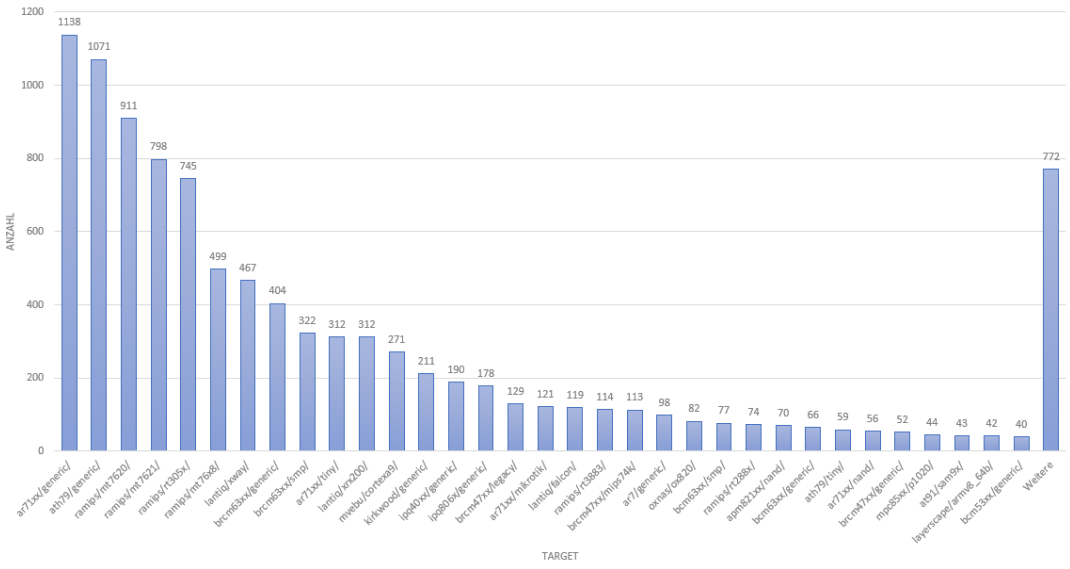


Abbildung 1.3: Anzahl der heruntergeladenen Firmware pro Target im November 2020. Unter weitere sind alle anderen Targets, welche in der Statistik auftauchten, zusammengefasst.

1.4 Beschreibung der BSI TR-03148

Bei der Technischen Richtlinie „Sichere Breitband Router“ (BSI TR-03148) des Bundesamtes für Sicherheit in der Informationstechnik handelt es sich um eine Sammlung von grundlegenden Sicherheitsanforderungen für Breitband Router [8, p. 3]. Der Schwerpunkt der Richtlinie liegt hierbei vor allem auf Heimroutern, sowie auf Geräten, welche im SOHO Umfeld eingesetzt werden. Das Dokument wird durch die Dokumente „BSI TR-03148 Implementation Conformance Statement (ICS)“ sowie „BSI-TR-03148-P ICS and Test Documentation” ergänzt. In diesen Dokumenten sind Testfälle und Dokumentation zur Durchführung einer Prüfung festgehalten. Die Technische Richtlinie definiert 101 „Test Requirement's", welche insgesamt 164 „Test Procedures" beschreiben. Ein „Test Requirement" wird als fehlgeschlagen gewertet, wenn ein zugehöriges „Test Procedure" nicht bestanden wird. Nach Angaben des Bundesamtes für Sicherheit in der Informationstechnik richtet sich die Technische Richtlinie vor Allem an Hersteller von Routern, sie kann jedoch auch für Endnutzer relevant sein, wenn diese einen neuen Router anschaffen möchten und sich daher über den Stand der Technik informieren wollen [14]. Es werden Anforderungen für ein Mindestmaß an verpflichtenden und einigen optionalen IT-Sicherheitsmaßnahmen definiert, um ein grundlegendes Niveau für die Sicherheit dieser Geräte zu schaffen [8, p. 11]. Die angestrebte Zertifizierung von Geräten würde ebenso die Sicherheit der Geräte für den Verbraucher transparenter machen.

Das Dokument entstand aus einer Zusammenarbeit des BSIs mit verschiedenen Herstellern von Routern, Telekommunikationsanbietern, Behörden, dem Innen- und Wirtschaftsministerium, sowie unter Anderem mit Vertretern des OpenWrt Projektes und des Chaos Computer Clubs (CCC) [15]. Diese trugen ihre Ideen und Vorstellungen zur Sicherheit von Routern zusammen und suchten Lösungen für Interessenkonflikte. Nach Veröffentlichung der Richtlinie im Jahre 2018 wurde diese allerdings unter Anderem von Vertretern des OpenWrt Projektes, sowie vom Chaos Computer Club, kritisiert. Nach Meinung dieser Interessengruppe sind die definierten Maßnahmen in der Technischen Richtlinie nicht ausreichend, um tatsächliche Angriffe auf Router zu verhindern [16].

1.5 Bisherige Forschung

Während der Einsatz von OpenWrt im privaten und professionellen Umfeld beliebt zu sein scheint (vgl. 1.3) und auch einige Forschungsarbeiten mit OpenWrt verwirklicht wurden, sind derzeit keine aktuellen Arbeiten zur Sicherheit von OpenWrt verfügbar. Ortega et al. (2009) veröffentlichte eine Arbeit über eine quelloffene Methode zum Verhindern von sogenannten ARP Poisoning Attacken. Sie nutzten in diesem Kontext OpenWrt lediglich als viel-

seitig unterstützte Testplattform [17]. Palazzi et al. (2010) nutzten den Funktionsumfang und die Anpassbarkeit der Firmware, um einen verbesserten Datendurchsatz in Heimnetzen mit verschiedenen WLAN-Geräten zu erreichen [18]. Keine der derzeitigen Veröffentlichungen beschäftigt sich mit der Sicherheit von OpenWrt als Betriebssystem. Einzig Andrew McDonnell (2014) veröffentlichte in seinem Blog zwei Einträge über eine Sicherheitsanalyse von OpenWrt mittels des Tools checksec.sh (`https://github.com/slimm609/checksec.sh`) und entwarf eine verbesserte Version, in welcher bedeutend mehr Maßnahmen zur Verhinderung von Exploits aktiviert waren [19]. Die Ergebnisse der Veröffentlichung basierten jedoch auf Version 14.07 (Barrier Breaker) von OpenWrt, welche stark veraltet ist [12].

Die Forschung an Komponenten, die OpenWrt ausmachen, ist jedoch keinesfalls so eingeschränkt wie zuvor aufgezeigt. Der Linux Kernel, welcher einen grundlegenden Teil des OpenWrt Betriebssystems ausmacht, ist seit seiner Veröffentlichung 1991 ein andauerndes Gebiet der Forschung und Entwicklung, so auch in der IT-Sicherheit [20, 21, 22, 23]. Ebenso definiert sich OpenWrt über seine ca. 3800 zusätzlichen quelloffenen Pakete. Viele dieser Software-Erweiterungen existieren schon seit Jahrzehnten und ihre Integrität und Vertraulichkeit sind von den unzähligen Nutzern auf verschiedensten Plattformen anerkannt [24, 25, 26]. Abschließend kann man festhalten, dass es zwar durchaus Forschung an Komponenten, welche auch in OpenWrt genutzt werden, gibt, jedoch OpenWrt selbst noch nicht im Mittelpunkt der Forschung stand und die Sicherheitslage des Projektes weitestgehend ungeklärt bleibt.

1.6 Zielsetzung

Ziel dieser Arbeit war es, die Technische Richtlinie 03148 des BSI an Version 19.7.04 von OpenWrt durchzuführen und das Router-Betriebssystem auf Konformität zu prüfen. Hierbei wurde ein TP-Link Archer C7 Router genutzt. Es wurden die grundsätzlichen Sicherheitsmerkmale von OpenWrt mittels der technischen Richtlinie evaluiert. Dabei wurde nur die Funktionalität des Betriebssystems geprüft, welche nach der Installation auf dem Gerät bereitgestellt wurde. Funktionen, welche vom Nutzer zusätzlich installiert und eingerichtet werden mussten, wurden nicht betrachtet. Ebenso wurde die Anwendbarkeit der technischen Richtlinie auf quelloffene Netzwerk-Betriebssysteme ermessen. Darüber hinaus wurden statische Software Tests einiger quelloffener Router-Betriebssysteme als weitere Metrik genutzt, um einen differenzierteren Einblick in die Sicherheitslage solcher Projekte zu gewähren. Abschließend wurde sich kritisch mit den Ergebnissen, sowie der technischen Richtlinie, auseinandergesetzt. Die Ergebnisse der Arbeit können sowohl der Entwicklung von OpenWrt als auch unerfahrenen Endnutzern weitere Einblicke in die Sicherheit des Projektes liefern und somit langfristig die Resilienz der Heim- und SoHo Netzinfrastruktur stärken.

Kapitel 2

Grundlagen

2.1 Was könnt hier hin?

Kapitel 3

Methodik

3.1 Übersicht und Begründung der verwendeten Methodik

Die Methodik der Arbeit ist in großen Teilen durch die Technische Richtlinie vorgegeben. Die Testfälle wurden aufgrund ihrer Gruppierung in thematische Module in chronologischer Reihenfolge erarbeitet. Einzig solche Testfälle, welche spezifizierten, dass sie erst zum Ende der Testphase durchgeführt werden sollten, wurden nach hinten gestellt. Da es in erster Linie um die Technische Richtlinie 03148 gehen sollte, wurden weitere Tests, wie ein statischer Test mit dem „Firmware Analysis and Comparison Tool“ (FACT), erst nach Vollendung der Richtlinie begonnen [27].

Die Testfälle der Technischen Richtlinie wurden, soweit möglich, mit den Programmen durchgeführt, welche in der TR selbst spezifiziert wurden. Die Ergebnisse einer Literaturrecherche zeigten, dass die aufgeführte Software für die Überprüfung der Testanforderungen geeignet ist und die Ergebnisse dieser Programme seit vielen Jahren weitestgehend als korrekt akzeptiert sind. Hierzu zählt vor Allem das Programm nmap, welches aufgrund von verschiedenen Testrechnern in den Versionen 7.80, 7.90 und 7.91 verwendet wurde [28]. Die Änderungshistorie von nmap gibt allerdings keinen Anlass zur Annahme, dass dies die Ergebnisse invalidiert [29]. Ebenso wurde airmon-ng / airodump-ng in der Version 1.6 zum Prüfen verwendet. Dieses Softwarepaket ist ebenfalls seit vielen Jahren angesehen und findet in wissenschaftlichen Arbeiten Verwendung [30, 31]. Zur Aufzeichnung von Netzwerkpaketen wurde Wireshark 3.4.2 verwendet, welches neben der Kommandozeilenanwendung tcpdump häufig Verwendung findet [32, 33]. Im Rahmen der Tests wurde des Weiteren auf einige zweckspezifische Skripte in den Programmiersprachen Python und Bash zurückgegriffen. Bei der Entwicklung wurde Wert auf einfache Ausführbarkeit, sowie eine geringe Zahl an externen Abhängigkeiten, gelegt, um eine wiederholbare Ausführbarkeit auch in der Zukunft zu gewährleisten.

Als Router wurde ein TP-Link Archer C7 (AC1750) Dualband-Gigabit-WLAN-Router (v5) verwendet. Dieses Modell wurde aufgrund der Beliebtheit im OpenWrt-Umfeld, der Verfügbarkeit, der aktuellen Ausstattung und dem Preis-Leistungs-Verhältnis gewählt.

Die Statistiken der OpenWrt-Webseite haben gezeigt, dass die zu dem Gerät gehörige Version die dritt häufigst heruntergeladene OpenWrt-Firmware im November des Jahres 2020 ist. Bei dem Xiaomi Mi R3P und dem D-Team Newifi D2 Router, welche öfter abgefragt wurden, handelt es sich zwar um günstigere Geräte, allerdings haben diese nur eine eingeschränkte Verfügbarkeit in Deutschland. Die vom gewählten TP-Link unterstützten Funktionen sind darüber hinaus vergleichbar mit vielen Endgeräten, welche im privaten und SOHO Umfeld eingesetzt werden.

3.2 Aufbau und Beschreibung der Testumgebung

Der genutzte Testaufbau soll einen reibungslosen Ablauf der Testfälle erlauben sowie einfach reproduzierbar sein. Der Internetanschluss wurde durch den Internet Service Provider (ISP) bn:t Blatzheim Networks Telecom GmbH zur Verfügung gestellt. Der Glasfaseranschluss des ISP terminiert in einer FRITZ!Box 5530 Fiber, welche das Subnetz 192.168.178.0/24 bereitstellt. Der WAN Port des mit OpenWrt 19.7.04 bespielten TP-Link Archer C7 v.5 Dualband-Gigabit-WLAN-Router, wurde mit dieser FRITZ!Box verbunden, sodass der OpenWrt-Router das Subnetz 192.168.1.0/24 zur Verfügung stellen konnte. Die Erstinstallation von OpenWrt auf dem TP-Link Router erfolgte über die zur Verfügung stehende Anleitung [34]. Zunächst wurde das Firmware-Abbild heruntergeladen, daraufhin wurden die Hash-Werte mit den veröffentlichten und signierten Hash-Werten abgeglichen. Nachdem sichergestellt wurde, dass diese übereinstimmten, konnte die Datei über das Web-Interface des TP-Link Routers aufgespielt werden. Die OpenWrt Installationsdatei wurde hierzu über die Firmware-Update Funktion hochgeladen und automatisch vom Gerät installiert. Das Gerät startet daraufhin persistent mit OpenWrt anstelle des Betriebssystems von TP-Link. Alternativ besteht die Möglichkeit das Firmware-Abbild von OpenWrt über die „Trivial File Transfer Protocol“ (TFTP) Funktionalität des Routers aufzuspielen.

Ein Testcomputer wurde über das Local Area Network (LAN) angeschlossen, ein weiterer Laptop per WLAN und LAN verbunden (siehe Abbildung 3.1).
Der Testcomputer wurde wahlweise mit Windows 10 Version 20H2 (Build 19042.685) oder Ubuntu 20.04 LTS betrieben. Auf dem Laptop kam Kali Linux zum Einsatz. Dieser Aufbau gibt dem Tester eine flexible Arbeitsumgebung, in welcher die Tests ungestört durchgeführt werden können. Durch die Abtrennung des Netzes in das 192.168.1.0/24 Subnetz durch den OpenWrt Router sind Geräte des allgemeinen Heimnetzes von Portscans und Netzwerkpaketmitschnitten ausgeschlossen. Dadurch können Tests performanter durchgeführt werden, während andere Teilnehmer des Netzes ungestört weiterarbeiten können. Ebenso bietet der beschriebene Aufbau einfach die Möglichkeit weitere Geräte, welche für Tests benötigt werden, hinzuzufügen. Die verwendeten Linux-Distributionen, Ubuntu 20.4 LTS und

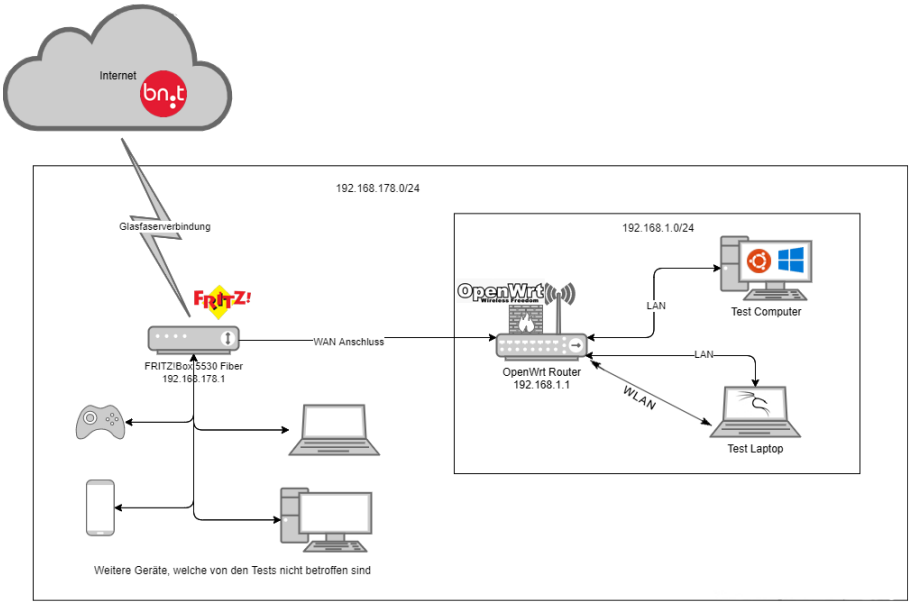


Abbildung 3.1: Aufbau der eingesetzten Testumgebung. Es handelt sich hierbei um einen sog. „double NAT“ Aufbau, d.h. der OpenWrt-Router hat keinen nativen Internetanschluss sondern bezieht die Internetverbindung über einen vorgelagerten Router. Die Tests wurden im Subnetz 192.168.1.0/24 durchgeführt.

Kali Linux, bieten die Möglichkeit die notwendigen Programme reibungslos zu betreiben. Dieser sogenannte „double NAT“ (Network Address Translation) Aufbau (vergleiche Abbildung 3.2) stellt praktisch keinen Nachteil dar [35]. Obwohl der direkte Anschluss des OpenWrt-fähigen Routers präferiert eingesetzt werden sollte, können alle Tests ohne Integritätsverlust durchgeführt werden. Die Tests bezüglich des WAN Anschlusses können über die IP-Adresse des OpenWrt Routers durchgeführt werden, welche durch die FRITZ!Box vergeben wurde. Weiterhin wurde der „Domain Name System Resolver“ (DNS Resolver) der FRITZ!Box auf die IP-Adresse des OpenWrt Routers geändert, um die Tests bzgl. des DNS-Protokolls und der Implementierung nicht zu verfälschen. Alle verfügbaren Firewall- und Filter-Einstellungen der FRITZ!Box wurden ebenfalls während der Tests deaktiviert.

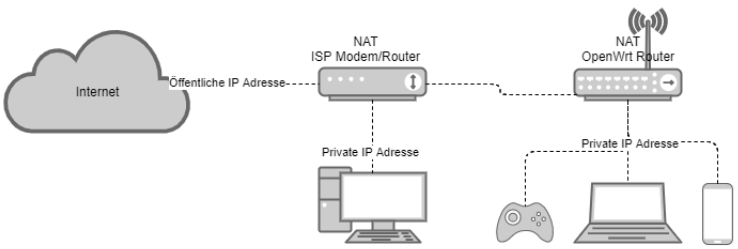


Abbildung 3.2: Beispiel einer allgemeinen „double NAT“ Umgebung. Die Darstellung basiert auf: https://kb.netgear.com/30186/What-is-Double-NAT (abgerufen: 13.12.2020)

3.3 Durchführung der Testfälle

3.3.1 Conformance Statement

Bevor die eigentlichen Tests, welche in der „Technischen Richtlinie 03148: Sichere Broadband Router“ beschrieben sind, durchgeführt werden können, muss zunächst ein sogenanntes „Implementation Conformance Statement“ ausgefüllt werden. In diesem werden maßgebende Informationen über das zu testende Gerät festgehalten. Bei einer Durchführung der Technischen Richtlinie im Kontext einer Zertifizierung würde dieses Conformance Statement zunächst vom Hersteller bzw. Auftraggeber ausgefüllt und eingereicht. Die angegebenen Informationen unterstützen den Tester, sind aber auch selbst Teil der Testprozedur. Zu diesen Informationen gehören neben dem Namen und der betrachteten Software-Version auch eine Übersicht über die zur Verfügung stehende Dokumentation des Gerätes. Hierzu wird auch technische Dokumentation gezählt, welche normalerweise nicht für Endnutzer und Verbraucher zur Verfügung steht. Des Weiteren werden relevante Informationen zu allen Modulen zusammengetragen, welche bei der Durchführung der „Test Procedures“ von Relevanz sind. So werden zum Beispiel für Modul A – Privates Netzwerk alle Dienste gesammelt, welche im privaten Netz zur Verfügung stehen, sowie die dazugehörigen Interfaces und Ports. Im Falle dieser Arbeit wurde das Conformance Statement als Teil der Richtlinie betrachtet und mit den in der Online-Dokumentation von OpenWrt beschriebenen Informationen ausgefüllt. Darüber hinaus konnte der Quellcode Aufschluss über in der Dokumentation ungeklärte Fragestellungen geben. OpenWrt bietet eine vergleichsweise geringe Anzahl an Diensten im Ausgangs- sowie initialisierten Zustand an. Lediglich der Web-Server uHTTPd auf Port 80, der SSH Server auf Port 22 und der von dnsmasq zur Verfügung gestellte DNS-Dienst auf Port 53 sind aktiv. Funktionen wie das Session Initiation Protocol (SIP) für Voice-over-IP-Telefonie oder Protokolle zur externen, automatischen Konfiguration des Geräts, welche oft bei handelsüblichen Routern verwendet werden, fehlen vollends. Ebenso gibt die Dokumentation an, dass das veraltete und als unsicher geltende Wi-Fi Protected Setup (WPS) Verfahren ohne die Installation von zusätzlicher Software nicht verwendet werden kann [36]. Dies ist auf vielen aktuellen Geräten in den Standardeinstellungen aktiviert. Aus dieser eingeschränkten Menge an Diensten wird ersichtlich, dass das Gerät nur über die Netz-Schnittstelle oder per ssh eingerichtet und bedient werden kann. Jedoch steht dem Nutzer standardmäßig der sogenannte „root“ Benutzer zur Verfügung, sodass uneingeschränkter Zugriff auf alle Funktionen und Einstellungen des Gerätes gewährleistet ist. Eine weitere Besonderheit zeigt sich in auch in der Vorkonfiguration des WLAN-Netzes von OpenWrt. Dies ist zunächst deaktiviert und wird standardmäßig ohne Passwort initialisiert. Begründen kann man dies damit, dass OpenWrt nicht mit gerätespezifischer Dokumentation ausgeliefert werden kann wie sonst üblich. Ein Schriftstück mit einzigartigem Passwort für das Gerät, sowie

das voreingestellte WLAN, kann nicht erstellt werden. So muss jedes Passwort, welches für ein OpenWrt Gerät verwendet wird, vom Benutzer selbst erstellt werden. Dies kann sowohl positive als auch negative Implikationen für die Sicherheit des Gerätes haben. Zum einen verhindert dieses Vorgehen, dass ein Hersteller ein unsicheres Passwort festlegt, welches ein unerfahrener Nutzer nicht ändert. Ebenso könnte der Hersteller ein bestimmtes Muster oder Master-Passwort verwenden. Wenn dieses veröffentlicht wird, so sind alle Geräte auf denen das Passwort nicht geändert wurde in Gefahr. Zum anderen könnte der Nutzer ein unsicheres Passwort vergeben, wenn er damit Konfrontiert wird. Wenn vom System keine festen Ansprüche an dieses Passwort gesetzt werden, so würde ein schwaches Passwort die Sicherheit des Gerätes ebenso kompromittieren.

Schon im zweiten Abschnitt des Conformance Statements, welcher sich auf das öffentliche Netz bezieht, wird erkenntlich, dass auch auf Seiten des Internets nur eine minimale Anzahl an Diensten verwendet wird. Die Dokumentation von OpenWrt spezifiziert keinen Dienst, welcher auf dem WAN-Interface angeboten wird. Ein vergleichbarer Trend kann auch bei den angebotenen Funktionen des Geräts beobachtet werden. Lediglich sehr grundlegende Funktionen wie das Dynamic Host Configuration Protocol (DHCP), ssh, secure copy (scp), IPv6 Unterstützung und eine Firewall werden angeboten. Die eigens für OpenWrt entwickelte, quelloffene Packet-Management Software „opkg“, über welche zusätzliche Funktionalität installiert werden kann, bildet jedoch eine Ausnahme. Der geringe Umfang an Funktionen lässt sich in zweierlei Hinsicht begründen. Durch den Packet Manager opkg kann gewünschte Funktionalität leicht vom Benutzer selbst installiert und eingerichtet werden, ohne schon im Vorhinein Speicherplatz für Funktionen zu nutzen, welche unter Umständen nicht verwendet werden. Darüber hinaus kann OpenWrt so auch auf Geräten mit limitiertem persistenten Speicher oder Arbeitsspeicher installiert werden. So kann zum Beispiel das Web-Interface von der Installation ausgeschlossen sein, wenn ein Gerät nicht über genügend Speicher verfügt. Dadurch ist eine minimale Installation auf Geräten mit 4MB Flashspeicher und 32MB RAM möglich, jedoch lediglich bis einschließlich Version 19.07.

Ein Defizit von OpenWrt lässt sich jedoch bereits im Conformance Statement finden. Es besteht keine Möglichkeit sicherheitsrelevante Updates automatisch einzuspielen. Über den Paket Manager bereitgestellte Funktionen könnten zwar mittels sog. CronJobs aktualisiert werden, dies würde jedoch nur periodisch nach Einstellung des Nutzers geschehen. Dies bietet keine Sicherheit, wenn die Periode zu groß gewählt wurde. Sicherheitslücken im Linux Kernel können jedoch nur über vollständige Firmware-Upgrades behoben werden und erfordern das aktive Eingreifen des Nutzers. Dies setzt das Engagement und fachliche Verständnis des Nutzers voraus, über den aktuellen Stand informiert zu bleiben und eine Aktualisierung zeitnah durchzuführen. Jedoch gibt die Dokumenation an, dass die Überprüfung

des Firmware-Upgrades von OpenWrt auf Integrität und Authentizität nicht automatisiert ist. Für einige Abbilder stehen digitale Signaturen zur Verfügung, welche vom integrierten Tool fwtool beim Aufspielen des Updates geprüft werden. Falls dieser Option allerdings nicht zur Verfügung steht, stehen dem Nutzer zur Unterstützung beim Upgrade-Prozess dann die eingebetteten Metadaten bereit, welche ausschließlich sicherstellen, dass es sich überhaupt um ein unterstütztes Gerät handelt. Gleichermaßen sind die berechneten Hash-Werte verfügbar, welche durch den Benutzer mit den signierten Werten des Download-Servers abgeglichen werden können.

Die folgenden Module des Conformance Statements zeigen gleichwohl eine weitere Besonderheit von OpenWrt. Die für Firewall, DNS und DHCP verwendeten Implementierungen sind vollständig quelloffen und werden schon seit vielen Jahren entwickelt. Die Firewall wird durch ein für OpenWrt gestaltetes Programm firewall3 bereitgestellt. Es handelt sich hier um eine einfache Möglichkeit netfilter/iptable Regeln zu gestalten. Iptables ist Bestandteil des Kernels und wird schon seit Version 2.4 mitgeliefert [37]. Der DHCP und DNS-Dienst wird von dnsmasq ermöglicht. Dies ist ebenfalls ein weitverbreitetes Programm, welches bereits 2001 veröffentlicht wurde und seitdem kontinuierlich weiterentwickelt wurde [38]. Da OpenWrt keine Fernwartungs-, VoIP- oder Virtual Private Network (VPN) Funktionalität bereitstellt, ohne die entsprechenden Pakete über den Paketmanager zu installieren, werden diese im weiteren Verlauf nicht betrachtet und dieses Ergebnis im Conformance Statement vermerkt.

3.3.2 Test Documentation

Die Testdokumentation wurde in Form der bereitgestellten Tabellenkalkulationsdatei ausgefüllt. Alternativ können die Ergebnisse auch in einer Textdatei festgehalten werden. Die Anforderungen mit Kriterien zum Bestehen des Testes finden sich im veröffentlichten „Test Specification“. Die Testdokumentation definiert die folgenden Kategorien: Eine durchlaufende Nummerierung und eine Angabe, ob es ein „muss“ oder „soll“ Kriterium ist, eine Beschreibung des Testfalls und die Angabe des Testers, ob der Testfall anwendbar ist oder nicht. Ebenso steht „N/A“ (not applicable) als Option zur Verfügung. Darauf folgen Felder für die jeweiligen Ergebnisse der Tests einer jeden Testreihe, gefolgt von der Möglichkeit für Notizen, Referenzen, benutze Tools, Zugriffsmethoden und einer Referenz für weitere Daten wie Bilder.

Die in der Richtlinie spezifizierten Zustände des DUT wurden vor Beginn der Test wie folgt festgelegt: Das Gerät ist im Auslieferungszustand (factory state), wenn es initial in Betrieb genommen wurde und nach jedem vollständigen Zurücksetzen. Der erste Start nach einem solchen Zurücksetzen des Geräts versetzt dieses in den Auslieferungszustand.

Der initialisierte Zustand (initialized state) ist erreicht, wenn das Gerät im Auslieferungszustand gestartet und ein Passwort für das Root-Konto vergeben wurde. Dies ist vom Nutzer selbst vorzunehmen und nicht verpflichtend. Für alle Testfälle, die den initialisierten Zustand oder den kundenspezifischen (customized state) Zustand vorrausetzen, wurde diese Aktion durchgeführt. Das Gerät befindet sich im kundenspezifischen Zustand, wenn zusätzliche Einstellungen vom Nutzer aktiviert oder angepasst wurden (vgl. Abbildung 3.3).

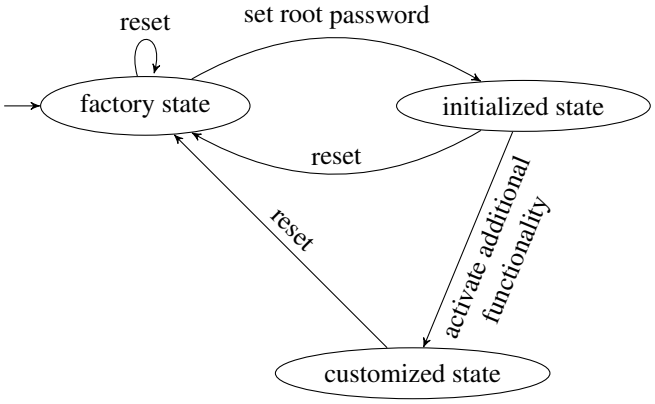


Abbildung 3.3: Darstellung der Zustände des Geräts sowie Übergangskriterien.

Modul A – Private Network

Wie in TP.A.1 nachgewiesen, unterstützt die betrachtete Version von OpenWrt zwei Arten, das Gerät in Betrieb zu nehmen. Zum einen stellt das Gerät einen SSH Zugang zur Verfügung, zum anderen den Web-Server, welches das Web-Interface „LuCI“ bereitstellt. Zur Prüfung des verlangten vollständigen Internetzugangs im initialisierten Zustand (TR.A.1) wurde die DNS-Funktionalität des bei Windows 10 standardmäßig installierte Kommandozeilenprogramm nslookup verwendet. Der FTP-Funktionsumfang wurde ebenfalls mittels des Windows-Kommandozeilenprogramms "ftp"getestet. Hierzu wurde der FTP-Downloadserver von DD-WRT genutzt (ftp.dd-wrt.com), da dieser ohne Benutzerkonto genutzt werden kann. HTTP, sowie HTTPS-Unterstützung können mittels des Programms „curl“ nachgewiesen werden. Hierbei handelt es sich um ein quelloffenes Programm, welches neben HTTP und HTTPS viele verschiedene Protokolle unterstützt und zur Übertragung von Daten über diese Protokolle gedacht ist [SOURCE?]. Das „Simple Mail Transfer Protocol“ (SMTP) kann ebenfalls mit Hilfe von curl getestet werden. Die geforderte IPv4 und IPv6 Konnektivität kann mit den Kommandozeilenapplikationen ping bzw. ping6 geprüft werden. Zur Sicherstellung der SSH-Verbindung kann zum Beispiel der kostenlose öffentliche Server von „SDF Public Access UNIX System, Inc“ genutzt werden (ssh.sdf.org). Ein eigens bereitgestellter SSH-Server kommt ebenfalls in Frage. Die

Unterstützung für das Telnet Protokoll muss unter Windows zunächst aktiviert werden, es ist jedoch auf vielen Linux Distributionen sofort verfügbar. Ein Test kann über die URL „towel.blinkenlights.nl“ durchgeführt werden. Die verwendeten Programme stehen unter den meisten aktuellen Betriebssystemen standardmäßig zur Verfügung und die spezifizierten Server sind weltweit kostenlos zu erreichen. Ebenfalls kann angenommen werden, dass die angegebenen URLs längerfristig zu erreichen sind, da sie schon seit vielen Jahren ihre Dienste anbieten.

Ein wichtiger Aspekt der Technischen Richtlinie wird ebenfalls durch TR.A.2 bis TR.A.5 spezifiziert. Diese „Test Requirement's’"behandeln die durch das Gerät zur Verfügung gestellten Dienste. Es wird vorausgesetzt, dass die angebotenen Dienste durch den Hersteller dokumentiert und auf eine wohldefinierte, minimale Menge beschränkt sind. Die Überprüfung kann mit Hilfe des Tools nmap durchgeführt werden. Nmap ist ein quelloffener Portscanner, welches ursprünglich von Gordon Lyon entwickelt wurde [28]. Es wird genutzt, um offene Ports und die darauf lauschenden Dienste zu identifizieren. Die TCP Ports des DUT wurden mit dem Kommando

```
$ nmap -sS -sC -sV -p- -Pn -oN <Dateiname.txt> 192.168.1.1
... oder ...
$ nmap -sSCV -p- -Pn -oN <Dateiname.txt> 192.168.1.1
```

Listing 3.1: Verwendetes nmap-Kommando zum Scannen aller TCP-Ports des OpenWrt Routers.

überprüft. Ebenfalls kann der Option „-T4“ hinzugefügt werden, um ggf. die Geschwindigkeit durch eine engere Taktung der Anfragen zu erhöhen. UDP Dienste wurden wie folgt getestet:

```
$ nmap -n -sUV --version-intensity 0 -p- --max-retries 1 -v -oN <Dateiname.txt>
    192.168.1.1
```

Listing 3.2: Verwendetes nmap-Kommando zum Scannen aller UDP-Ports des OpenWrt Routers.

Die optionale Erweiterung „-v“ erhöht die Verbosität und liefert bei den zeitintensiven UDP-Scans Informationen über den Fortschrittsgrad. Eine genaue Übersicht über die Funktion der gewählten Kommandos liefert Abbildung 3.4. Die beiden verwendeten Kommandos bzw. leichte Abwandlungen von diesen wurden vor allem aufgrund ihrer detaillierten Ausgabe sowie Performanz gewählt.

```
HOST DISCOVERY:
    -Pn: Treat all hosts as online -- skip host discovery
    -n: Never do DNS resolution [default: sometimes]

SCAN TECHNIQUES:
    -sS: TCP SYN
    -sU: UDP Scan

PORT SPECIFICATION AND SCAN ORDER:
    -p <port ranges>: Only scan specified ports
        Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
        Use -p- to scan all ports

SERVICE/VERSION DETECTION:
    -sV: Probe open ports to determine service/version info
    --version-intensity <level>: Set from 0 (light) to 9 (try all probes)

SCRIPT SCAN:
    -sC: equivalent to --script=default

TIMING AND PERFORMANCE:
    Options which take <time> are in seconds, or append 'ms' (milliseconds),
    's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
    -T<0-5>: Set timing template (higher is faster)
    --max-retries <tries>: Caps number of port scan probe retransmissions.

OUTPUT:
    -oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIddi3,
        and Grepable format, respectively, to the given filename.
    -v: Increase verbosity level (use -vv or more for greater effect)
```

Abbildung 3.4: Auszug aus der Dokumentation des Programms nmap. Die hier dargestellten Optionen des Programmes beziehen sich auf die im Verlauf der Arbeit eingesetzten Optionen. Quelle: https://svn.nmap.org/nmap/docs/nmap.usage.txt (Abgerufen am 02.01.2020)

Zur Prüfung der WLAN-Schnittstelle wurde auf die Programmsuite aircrack-ng zurückgegriffen. Es handelt sich hierbei um eine frei verfügbare Sammlung von Programmen zur Analyse der Sicherheit von Wi-Fi Netzwerken [39]. Zunächst wird das Programm airmon-ng eingesetzt, um die WLAN-Karte in den sogenannten Monitor-Modus zu versetzen:

```
$ airmon-ng start wlan0
```

Listing 3.3: Kommando um die WLAN-Karte in den Monitormodus zu versetzen. Der Name der verwendeten Karte ist wlan0.

Daraufhin kann airodump-ng verwendet werden, um Informationen zu allen verfügbaren WLAN-Netzen bereitzustellen:

```
$ airodump-ng wlan0mon
```

Listing 3.4: Kommando um airodump-ng mit der soeben in den Monitormodus versetzen WLAN-Karte zu starten.

Vor allem die Spalte „ENC“, welche für „encryption“ steht, ist von Bedeutung. Sie zeigt an, dass das Gerät durch Wi-Fi Protected Access 2 (WPA2) geschützt ist. Dies unterstützt die Annahme, dass das Gerät WPA2 nach dem IEEE802.11i Standard bereitstellt.

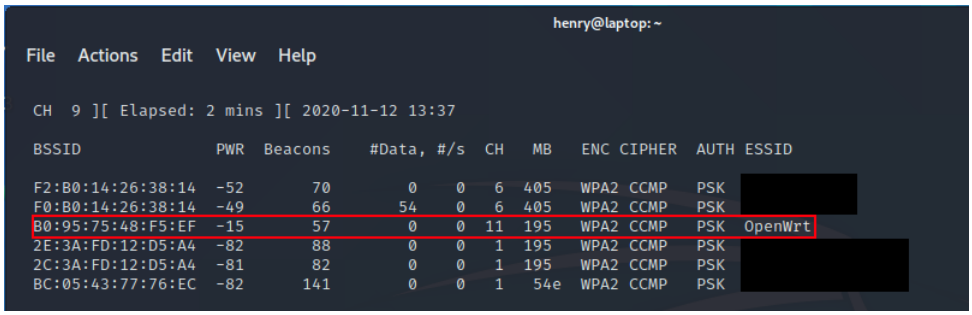


Abbildung 3.5: Die Ausgabe des Programms airodump-ng. Man kann sehen, dass OpenWrt mit WPA2 CCMP verschlüsselt ist und ein Password (Spalte PSK) benötigt.

Modul B - Public Network

Die Teststrategie, welche für Modul B – Public Network eingesetzt wurde, ist nahe an der Vorgehensweise von Modul A – Private Network orientiert. Jedoch wird nun die IP des OpenWrt Geräts im Kontext des Subnetzes 192.168.178.0/24 verwendet (siehe Abbildung 3.1). So wird nicht die LAN-Schnittstelle des Gerätes angesprochen, sondern die Wide Area Network (WAN) Schnittstelle, also die öffentliche IP-Adresse des Gerätes ist.

```
$ # Fuer TCP:
$ nmap -sSCV -p- -Pn -oN <Dateiname.txt> 192.168.178.115
$ # Fuer UDP:
$ nmap -n -sUV --version-intensity 0 -p- --max-retries 1 -v -oN <Dateiname.txt>
    192.168.178.115
```

Listing 3.5: nmap Kommandos zum Scannen der WAN-Schnittstelle von OpenWrt. Das erste Kommando scannt TCP, das zweite UDP. Die angegebene IP-Adresse ist die des OpenWrt Routers.

Auch die VoIP Funktionalität kann effektiv mit nmap getestet werden. Zusätzlich zu einem vollständigen Scan des Geräts können auch die standardmäßig für VoIP verwendeten Ports 5060 und 5061 separat gescannt werden.

```
$ # TCP Port 5060, 5061 testen
$ nmap -sSCV -p 5060,5061 -Pn -oN <Dateiname.txt> 192.168.178.115
$ # UDP Port 5060, 5061 testen
$ nmap -n -sUV --version-intensity 0 -p 5060,5061 --max-retries 1 -v -oN
    <Dateiname.txt> 192.168.178.115
```

Listing 3.6: Kommandos zum gezielten Scan der Ports 5060, 5061 mit nmap. Diese Ports werden standardmäßig für VoIP genutzt.

Jedoch ist eine vollständige Prüfung aller Ports zu bevorzugen, da diese Ports nicht zwingend genutzt werden müssen.

Modul C - Functionalities

Das „Test Requirement" TR.C.2 beschreibt die Anforderung, dass dem Endnutzer keinerlei Funktionalität verheimlicht werden darf. Dies ist eine durchaus schwierig zu prüfende Anforderung, welche erst zum Ende des Tests durchgeführt werden sollte. Im Falle von OpenWrt und dem somit vollständig verfügbaren Quellcode, sowie dem vollumfänglichen root Zugriff auf das Gerät per ssh ist dies vereinfacht, jedoch aufgrund des Funktionsumfangs immer noch eine Herausforderung. Es muss sich hier auf die Eindrücke und Erfahrungen des Testers zum Ende der Testphase verlassen werden.

Modul D – Configuration and Information

Für die meisten modernen Heimrouter ist die Konfiguration durch ein Web-Interface die benutzerfreundlichste Methode, so auch für OpenWrt. Die Sicherung der Datenintegrität und Vertraulichkeit auf dem Transportweg wird heutzutage durch HTTPS erreicht. Diese Transportwegverschlüsselung verhindert, dass eine böswillige dritte Partei die übertragenen Daten auslesen oder verändern kann. Es ist also naheliegend, die Anforderung an eine durch HTTPS gesicherte Verbindung zum Webserver in der Technischen Richtlinie zu finden. Zur Überprüfung des „Test Requirement" TR.D.3 bietet sich ein Skript wie testssl.sh an, welches von Dr. Wetter IT-Consulting frei zur Verfügung gestellt wird [40]. Dieses Skript zeigt detaillierte Informationen zu allen vom Webserver unterstützten Protokollversionen sowie Verschlüsselungsmethoden an. Des Weiteren kann auch ein Paketaufzeichnungs- und Analyse-Software wie Wireshark eingesetzt werden, um die unverschlüsselten Pakete

zu betrachten. Wenn HTTPS aktiv ist, so sollten keine menschenlesbaren Daten in den Pakten gefunden werden. Es ist ebenfalls möglich, Informationen zu HTTPS und dem dazugehörigen Zertifikat in den meisten modernen Browsern in der Nähe der URL-Leiste zu finden.

Nichtsdestoweniger müssen auch andere Angriffsvektoren auf Heimrouter betrachtet bzw. getestet werden. So muss das Einloggen auf dem Gerät gegen Bruteforce Angriffe geschützt sein. Bei einem solchen Angriff wird meist eine Passwortliste verwendet, um automatisch viele Kombinationen von Benutzernamen und Passwörtern zum einloggen auf dem Gerät zu testen. Hierbei wird keine weitere Logik angewendet, sondern möglichst alle Möglichkeiten ausgeschöpft. Eine Schutzmaßnahme kann ein Fehlerzähler sein, welcher die fehlgeschlagenen Versuche protokolliert und das Aufschalten auf das Gerät nach einer gewissen Anzahl an Versuchen unterbindet oder entschleunigt. Ebenso könnte die Eingabe auf Muster geprüft werden, um automatische Login-Versuche zu erkennen. Die Prüfung dieses „Test Requirement's" wurde durch ein Skript in der Programmiersprache Python umgesetzt. Durch den Aufruf

```
$ python3 OpenWrt_Bruteforce_Check.py web
```

Listing 3.7: OpenWrt Webserver mit Python Skript auf Resistenz gegen Bruteforce Angriffe testen. (siehe Anhang für vollständiges Python-Programm.)

wird der Web-Server getestet. Alternativ kann durch

```
$ python3 OpenWrt_Bruteforce_Check.py ssh
```

Listing 3.8: OpenWrt SSH-Server mit Python Skript auf Resistenz gegen Bruteforce Angriffe testen. (siehe Anhang für vollständiges Python-Programm

der SSH Server getestet werden. Vor der Nutzung können der korrekte Benutzername, sowie das korrekte Password, die Anzahl der Versuche, die IP des Geräts, sowie der SSH-Port festgelegt werden. Für den Test des SSH-Servers wurden 40 Versuche eingestellt, wobei die Zeit für die Antwort des Servers gemessen wird. Das Python Modul „SSHLibrary“ wird genutzt, um die Verbindungen mit dem SSH-Server zu handhaben. Zunächst wird geprüft, ob der spezifizierte Server erreichbar ist. Daraufhin werden die spezifizierten Login Versuche durchgeführt und die Zeit bis zur Antwort des Servers gemessen. Die Antwort des Servers bei falschen Daten ist der Abbruch der Session durch eine SSHLibrary Exception. Nachdem die Daten gesammelt wurden, wird eine lineare Regression auf den Daten durchgeführt, um einen Trend in den Antwortzeiten kenntlich zu machen. Wenn ein linearer Anstieg zu erkennen ist, dann werden die Versuche verlangsamt, wenn die Regressionslinie jedoch zur X-Achse parallel ist, so werden die Versuche in konstanter Zeit durchgeführt. Neben der

grafischen Darstellung der Antwortzeiten, sowie der Regressionslinie, werden dem Nutzer der Mittelwert, der Median, der Regressionskoeffizient und der Standardfehler angezeigt. Nachdem die Analyse durchgeführt wurde, werden die korrekten Login Daten verwendet, um eine neue Verbindung herzustellen. Wenn das OpenWrt SSH-Banner korrekt angezeigt wird, lässt der SSH-Server trotz der vorherigen fehlgeschlagenen Versuche noch weitere zu, ohne erkennbare Entschleunigung. Der Test des Webservers wurde durch die POST Anfrage

```
http://192.168.1.1/cgibin/luci/admin/status?luci_username={USERNAME}
                                          &luci_password={PASSWORD}
```

realisiert. Wenn ein falscher Benutzername, oder ein falsches Passwort verwendet wird, so antwortet der Webserver mit dem Statuscode 403. Nach der ersten Überprüfung der Verbindung wurden 100 Versuche eingestellt, um genug Daten für ein aussagekräftiges Ergebnis zu sammeln. Der weitere Ablauf der Analyse verläuft wie bereits beschrieben. Nach der Auswertung der Daten werden die korrekten Login-Daten an den Server geschickt. Ein einfacher Regex-Ausdruck überprüft, ob ein erfolgreicher Login möglich war, und es wird dem Benutzer anschließend angeboten, eine eingeloggte Session im Browser zu öffnen (siehe Anhang A.1)

Neben dem Bruteforce Angriff auf den Webserver ist auch Cross-Site-Request-Forgery (CSRF) ein üblicher Angriffsvektor [SOURCE]. Wenn keine adäquaten Schutzmaßnahmen vom Server getroffen werden, kann ein Angreifer über eine präparierte Website oder einen Phishing Link, schädlichen Code auf Webseiten ausführen, auf denen der Nutzer bereits authentifiziert ist. Dieser Code versetzt den Angreifer in die Lage, Befehle auf der Webseite oder dem Webserver auszuführen, auf welchem der Nutzer angemeldet ist. Es könnte zum Beispiel ein neuer Benutzer durch den Angreifer angelegt werden, oder Einstellungen und Sicherheitsparameter an den Angreifer gesendet werden [siehe Abbildung 3.5]. Eine häufig verwendete Sicherheitsmaßnahme gegen CSRF Angriffe ist ein Anti-CSRF Cookie. Dieser wird im http-Header der Website deklariert und besteht aus einer zufälligen Zeichenkette. Dieser Cookie wird für jede http-Methode benötigt, welche nach dem Setzen des Cookies aufgerufen wird und Aktionen in der Session des Nutzers durchführen möchte. Zur Überprüfung der Anforderung TR.D.12 wird zunächst festgestellt, ob es einen Anti-CSRF Cookie gibt. Zunächst kann der Speicher des Webbrowsers angezeigt werden, um zu prüfen, ob überhaupt ein Cookie eingesetzt wird. Daraufhin wird die Web-Proxy Funktionalität von Burp Suite genutzt, um den Ablauf des Logins und der Erstellung einer gültigen Session zu beobachten. Alle nachfolgenden http-Methoden sollten nach Initialisierung des Cookies diesen als Sicherheitsmerkmal mit versenden. Der Quellcode von OpenWrt gibt darüber hinaus weiteren Aufschluss über die Implementierung der Anti-CSRF Tokens. Die Datei „dispatcher.lua“ des LuCI Interfaces, welche die Erstellung und Vali-

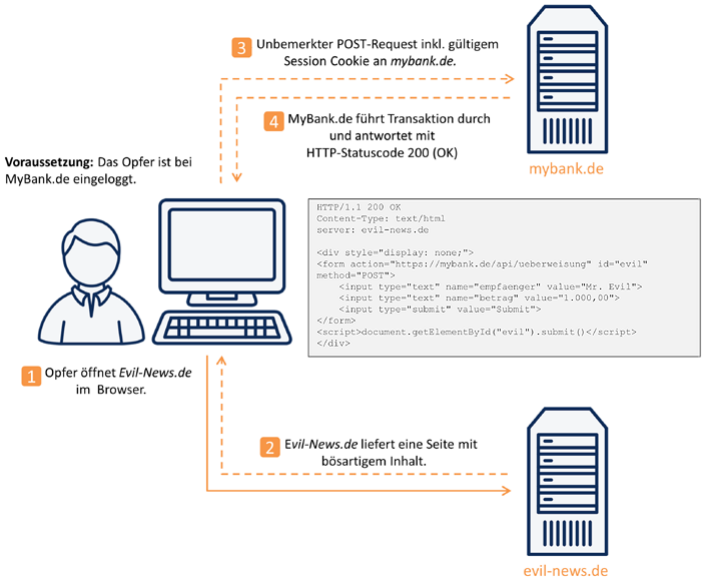


Abbildung 3.6: Eine möglicher Ablauf eines Cross-Site-Request-Forgery Angriffs. Der Nutzer ist bei MyBank.de angemeldet und öffnet Evil-News.de. Diese Seite liefert Schadcode an den Browser des Opfers aus. Der Code tätigt eine Überweisung mittels eines POST Requests und dem gültigen Cookie des Nutzers. Da es keine Abwehrmaßnahmen gibt tätigt der Bankserver die Überweisung.

dierung der Benutzer-Sitzungen handhabt, zeigt in diesem Falle eindeutig, dass es sich um Anti-CSRF Cookies handelt und dass diese durch den als sicher anerkannten Zufallszahlengenerator /dev/urandom generiert werden [41, 42]. Abschließend wurde ein einfaches Python Skript verwendet, welches 100 gültige Sitzungen am Web Server des OpenWrt Routers anmeldet und mittels eines Regulären-Ausdruckes den Wert des Cookies ausließt. Dazu wird das Request Modul von Python verwendet, sowie die POST-Anfrage, welche bereits für das Bruteforce-Skript verwendet wurde. Abschließend wird geprüft, ob die 100 verschiedenen Sitzungen einzigartige SessionIDs und Anti-CSRF Token besitzen.

Modul E – Firmware Updates

Modul E der Technischen Richtlinie prüft die Firmware Update Funktion des Geräts. Hier ist vor allem der Mechanismus der Firmware-Validierung von Interesse. Nach Angaben der Entwickler werden einige Firmware Dateien signiert. OpenWrt liefert standardmäßig ein Kommandozeilenprogramm, mit dem Signaturen und Metadaten aus den Firmwareabbildern extrahiert werden können. Der Aufruf

```
$ fwtool -s - <Dateiname.bin>
```

Listing 3.9: Nutzung des in OpenWrt integrierten Programmes „fwtool“ zur Extraktion von Signaturen aus OpenWrt Firmware-Abbildern

zeigt die Signatur an, wenn diese vorhanden ist. Ebenso muss ermessen werden, wie lange der Hersteller benötigt, um Sicherheitslücken zu beheben. Die sogenannten „Git Hashes“, genaue Identifizierungsmerkmale eines git commits, sind hier förderlich, da sie einen genauen Zeitstempel tragen. Des Weiteren ist der entsprechende git commit, welcher eine Sicherheitslücke behebt, in den Sicherheitsnotizen auf der OpenWrt Website spezifiziert, sodass das Erstellen einer Zeitleiste mit Sicherheitsvorfällen und deren Beheben einfach realisierbar ist.

Modul G – Domain Name System (DNS)

Zur weiteren Einschränkung der Angriffsoberfläche wird in Modul G die Implementierung des DNS-Dienstes des DUT geprüft. Ein möglicher Angriff auf DNS-Dienste ist eine sogenannte DNS Rebinding Attacke. Bei dieser Art von Angriff wird die vom Browser durchgesetzte „Same Origin Policy“ umgangen, um arbiträre Anfragen an das lokale Netzwerk des Opfers zu stellen. Die Herkunft („Origin“) eines Web Dokumentes ist dabei wie folgt definiert:

Abbildung 3.7: Origin (dt. Herkunft) eines Web-Dokumentes [43]

Zwei Dokumente haben also die gleiche Herkunft („same origin“), wenn sie identische „scheme“, „host“ und „port“ Komponenten haben. Die „Same Origin Policy“ setzt also durch, dass Skripte, oder auch Cascading Style Sheets (CSS), nur auf Daten von anderen Webseiten zugreifen können, wenn diese sich dieselbe Herkunft teilen. Wenn diese Richtlinie nicht implementiert wäre, dann wäre eine bösartige Webseite zum Beispiel in der Lage auf ein Bankkonto zuzugreifen, auf dem ein Opfer ebenfalls eingeloggt ist. Dort könnten Daten ausgelesen oder Aktionen ausgeführt werden.

Bei einem DNS Rebinding Angriff ruft das Opfer zunächst eine kompromittierte, oder bösartige, Website auf. Für diesen Aufruf wird ein DNS-Server beauftragt mit der IP-Adresse des angefragten Web-Servers zu antworten. Der vom Angreifer kontrollierte DNS-Server antwortet mit einem DNS A Record, welcher auf die Angreifer-Webseite verweist und den Browser des Opfers anweist, die DNS-Daten nur für eine geringe Zeit im Cache zu behalten. Ein Skript, welches auf der Webseite des Angreifers platziert wurde, wartet nun darauf, dass die DNS-Daten aus dem Cache verfallen, sodass der Browser eine

neue Anfrage stellen muss. Diesmal antwortet der DNS-Server allerdings nicht mit der eigenen IP-Adresse, sondern mit einer IP-Adresse des lokalen Netzwerks des Opfers. Nun kann das Skript Anfragen an den lokalen Dienst stellen, z.B. Daten exfiltrieren oder weitere Angriffe starten [siehe Abbildung 3.8].

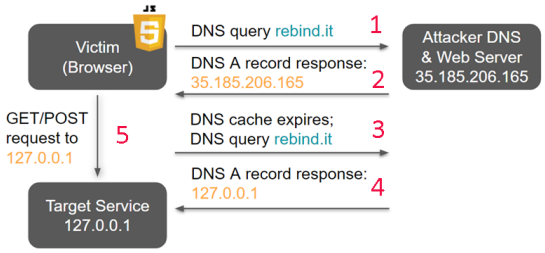


Abbildung 3.8: Ablauf eines DNS Rebinding Angriffs mittels des des Webtools der NCC Group (`http://rebind.it:8080/manager.html`

Die Überprüfung der Anforderung TR.G.2 basiert auf der Untersuchung der verwendeten Methoden zur Mitigation von DNS Rebinding Attacken und einem funktionalen Test dieser Umsetzung. Da OpenWrt DNS-Dienste mittels dnsmasq anbietet, muss geprüft werden, ob die Option „–stop-dns-rebind“ aktiviert ist. Dies ist sowohl über die Kommandozeile als auch über das LuCI Web-Frontend möglich. Ein funktionaler Test dieser Sicherheitsmaßnahme kann mittels des Singularity of Origin Web-Toolkits der NCC Group getestet werden [43]. Als Target Host wird dabei die IP-Adresse des OpenWrt Routers spezifiziert. Desweiterten wurde das Intervall auf zwei reduziert und die Option „Flood DNS Cache“ aktiviert, da der Test mit einem Chromium basierten Browser durchgeführt wurde. Es bietet sich ebenfalls an verschiedene „Attack Payloads“ und Strategien zu testen.

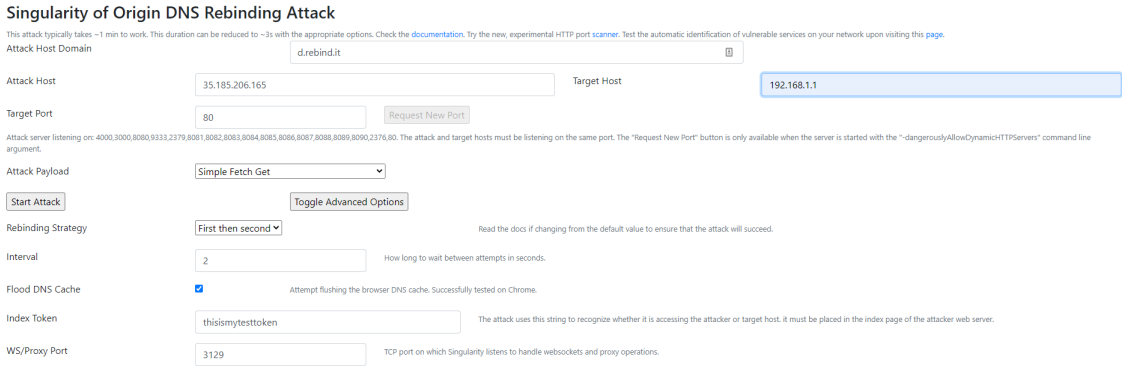


Abbildung 3.9: Das Singularity of Origin Web-Interface. Die Webseite wird von der NCC Group bereitgestellt, um verschiedene Arten von DNS Rebinding Angriffen zu testen. In diesem Kontext wurde es genutzt, um zu prüfen, ob OpenWrt diese Angriffe erfolgreich abwehrt bzw. erkennt.

Eine ebenso relevante Sicherheitsfunktion von DNS-Diensten ist die sogenannte „Source Port Randomization“ und „Transaction ID Randomization“, also die zufällige Wahl eines Quell-Ports, sowie einer Transaktions-ID für eine DNS-Anfrage. Diese Werte, welche vom DNS-Client generiert werden, dienen als Synchronisationsmethode zwischen dem DNS-Server und Client. Wenn der Quell-Port und die Transaktionsidentifikationsnummer von einem Angreifer berechnet oder geraten werden können, dann kann ein Angreifer diese nutzen, um dem Opfer manipulierte DNS-Antworten zu senden. Der DNS-Client würde diese aber als korrekt akzeptieren und eine potenziell schädliche Verbindung zu einem dritten Server aufbauen [44]. Für einen funktionalen Test werden zunächst mithilfe des Python Skriptes send_dns_requests.py eine große Anzahl verschiedene DNS-Anfragen generiert. Dazu wird eine Liste mit 1000 häufig besuchten Webseiten genutzt [45]. Dies bietet sich an, da so sichergestellt wird, dass diese Webseiten verfügbar sind und in einer geringen Zeit antworten. Während die DNS-Anfragen gestellt werden wird ein Mitschnitt aller Netzwerkpakete durch das Programm Wireshark gemacht. Die so erstellte Datei wird in einem weiteren Schritt analysiert. Dazu liest das Python Skript „analyze_pcap.py“ diese ein und selektiert im ersten Schritt alle DNS-Pakete, welche vom OpenWrt Router gesendet wurden. Daraufhin werden der DNS-Quell-Port sowie die Transaktions-ID aus diesen Paketen ausgelesen. Im letzten Schritt werden die Anzahl der DNS Anfragen, die Anzahl der einzigartigen Ports und Transaktions-IDs, die jeweiligen minimalen und maximalen Werte, die Standardabweichung und die häufigsten Werte angezeigt. Des Weiteren wird ein Kolmogorow-Smirnow-Test durchgeführt, um zu prüfen, ob die Verteilung der Daten mit einer Gleichverteilung übereinstimmt. Schlussendlich werden noch jeweils zwei Grafiken generiert, welche die Daten in einem Säulendiagramm und einen Streudiagramm darstellen. Auf diese Art kann visuell prüfen, ob Muster in den Darstellungen zu erkennen sind.

```
Test results for DNS port randomization:

  Number of samples: 1086
  Number of unique ports: 1086
  Range: 61 - 65508
  Standard Deviation: 18668.148912615263
  5 most common ports: [(59336, 1), (27475, 1), (14318, 1), (57429, 1), (31375,
      1)]



Test results for transaction ID randomization:

  Number of samples: 1086
  Number of unique ports: 1037
```

```
  Range: 45 - 65415
  Standard Deviation: 19128.480563438716
  5 most common ports: [(22681, 2), (16653, 2), (25739, 2), (20337, 2), (57986,
      2)]

KstestResult(statistic=0.03222836095764273, pvalue=0.6257210434465147)
KstestResult(statistic=0.027624309392265192, pvalue=0.8019298430829213)
```

Listing 3.10: Ergebnisse der DNS Port und DNS Transaktions-Identifikationsnummern Analyse. Der Kolmogorow-Smirnow Test zeigt, dass die Daten annähernd gleichverteilt sind.

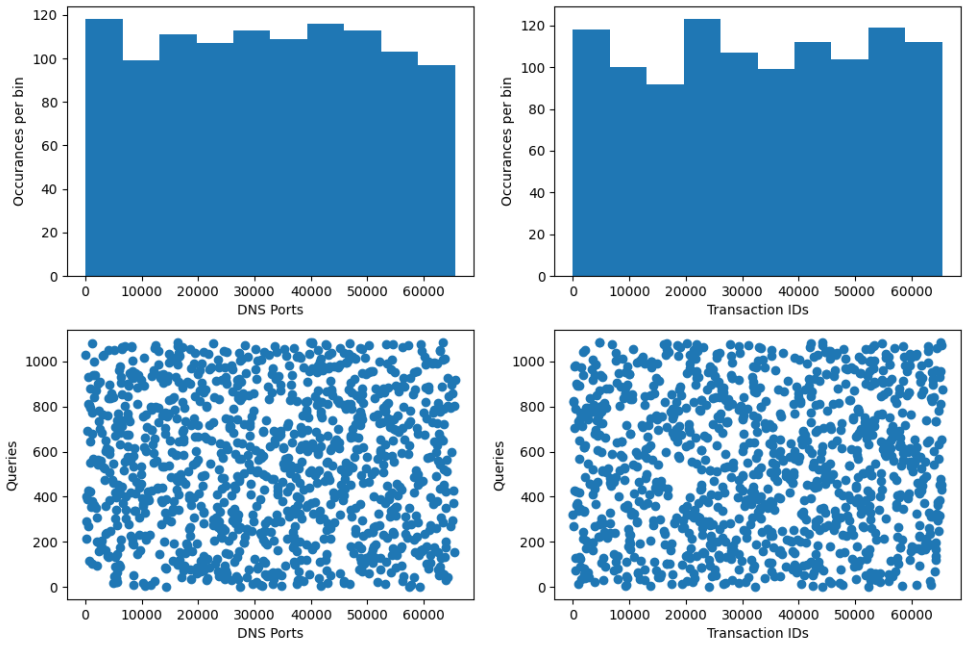


Abbildung 3.10: Darstellung der Ergebnisse als Säulendiagramm und als Streudiagramm. Die X-Achse beschreibt die Source-Ports bzw. die Transaktions-ID. Aus der Y-Achse der Säulendiagramme sind ist die Häufigkeit pro Sammelbehälter dargestellt. Die Y-Achse der Streudiagramme zeigt die Anzahl der Anfragen. Man kann sehen, dass die volle Spannbreite von 0 bis 65535 ausgenutzt wird. Da kein Muster in den Streudiagrammen erkennbar ist und die Säulendiagramme eine Gleichverteilung andeuten, kann angenommen werden, dass zufällige Source-Ports und Transaktions-IDs gewählt wurden.

Modul I – Factory Reset

Das Testen der Zurücksetzfunktion des OpenWrt Routers fällt aufgrund des uneingeschränkten Systemzugriffs einfach. Es können verschiedene Methoden eingesetzt werden. Zunächst sollte eine Leitlinie (Baseline) erstellt werden. Dazu dient ein Konfigurationsbackup, wel-

ches direkt nach dem ersten Einschalten des Geräts erstellt wurde. Dieses wird anschließend mittels des Kommandozeilenprogramms „diff“ mit einem Backup verglichen, welches nach der Nutzung des Routers und einem anschließenden Zurücksetzen des Geräts erstellt wurde. Alternativ kann das ebenfalls auf OpenWrt zur Verfügung stehende Kommandozeilenprogramm „md5sum“ verwendet werden, um die Hash-Werte aller lesbarer Dateien auf dem System zu generieren und diese zu exportieren. Diese sollten nach dem Zurücksetzen des Geräts wieder übereinstimmen. Die Erstellung von CRC-Prüfsummen mit dem „cksum“ Befehl kann unter Umständen schneller sein, als MD5 Hash-Werte. Es bietet sich also an diese Prüfsummen auf leistungsschwachen Geräten zu verwenden.

3.3.3 Nicht anwendbare Test Prozeduren

[Absatz streichen oder neu schreiben!] Ebenso wie die Natur des OpenWrt Projektes ein einfaches Testen vieler „Test Requirement
textquotesingle s" ermöglicht, so werden einige Aspekte der Firmware anders gehandhabt als bei handelsüblichen Heimroutern. So sucht man vergeblich nach einem initial verfügbaren WLAN-Netz, nachdem der Router gestartet und eingerichtet wurde. Ebenso sind viele Funktionen, die ein Nutzer vielleicht von anderen Geräten gewöhnt ist, nur als zusätzliches Software-Paket verfügbar, oder durch aufwendige Konfiguration. Beispiele sind Wi-Fi Protected Setup, ein Community WLAN, Fernwartung, automatische Firmware-Updates oder Meldungen zu neuen Firmware-Updates, Voice over IP und Virtual Private Network Funktionen.

3.4 Statische Code-Analyse einiger quelloffenen Router Firmware Alternativen mittels FACT

Neben der Methodik der Technischen Richtlinie des BSI gibt es noch viele weitere Arten, um Aspekte einer Software zu evaluieren. Die Sicherheit einer betrachteten Software, in diesem Fall OpenWrt, lässt sich unter anderem durch sogenannte statische Tests oder durch dynamische Tests abschätzen. Diese Verfahren gehören zu den analytischen Softwaretests und unterscheiden sich darin, dass bei einem dynamischen Test die Software während der Laufzeit (execution based) getestet wird, während sie bei einem statischen Test nicht ausgeführt wird (non-execution based) [46]. Es wird sich für die Durchführung einer statischen Code-Analyse von Router-Firmware an der Methodik des „Home Router Security Reports 2020“ des Fraunhofer-Institut für Kommunikation, Informationsverarbeitung und Ergonomie (FKIE) orientiert [47]. In dieser Veröffentlichung des FKIE wurden 127 verschiedene,

aktuelle Firmware-Abbilder von sieben Herstellern automatisch durch das ebenfalls vom FKIE entwickelte Firmware Analysis and Comparison Tool analysiert und ausgewertet.

3.4.1 Installation und Testumgebung

FACT, welches vom FKIE kostenfrei zur Verfügung gestellt wird, wurde lokal auf einem Desktop Computer installiert. Es handelt sich hierbei um ein System mit 12 Prozessoren, welche jeweils auf einer Taktfrequenz von 4.2GHz betrieben werden, sowie 16GB RAM. Ebenfalls stehen dem System 256GB persistenter Speicher zur Verfügung. Da die Installation auf Ubuntu 16.04, 18.04, 20.04 (stable) empfohlen wird, wurde Ubuntu 20.04 als aktuellster Vertreter des Ubuntu-Betriebssystems ausgewählt [48]. Die zum Zeitpunkt der Arbeit aktuelle Version von FACT, FACT_core v3.1.1, wurde mittels der bereitgestellten Anleitung installiert (siehe 3.11) [27].

```
$ sudo apt update && sudo apt upgrade && sudo apt install git
$ git clone https://github.com/fkie-cad/FACT_core.git ~/FACT_core
$ ~/FACT_core/src/install/pre_install.sh && sudo mkdir /media/data && sudo chown
    -R USER /media/data
$ sudo reboot
$ ~/FACT_core/src/install.py
$ ~/FACT_core/start_all_installed_fact_components
```

Listing 3.11: Installationsschritte für das Firmware Analysis and Comparison Tool des FKIE. Diese Schritte Updaten die installierten Pakete, downloadet das Git Repository und führt alle nötigen Schritte zur Installation aus. Zuletzt wird das Programm gestartet.

Da das System den minimalen Software Anforderungen von FACT entspricht ist die Installation und Nutzung des Programms prinzipiell möglich, jedoch empfiehlt sich ein System mit mehr RAM, da dies die Performanz der Analyse erhöht. Ebenfalls kam es bei dem eingesetzten System vermehrt dazu, dass kein RAM mehr zur Verfügung stand und der Rechner während der Analyse aufgrund der Auslastung nicht anderweitig genutzt werden konnte. Der Einsatz eines separaten Test Computers oder eines Virtuellen Privaten Servers (VPS) ist zu empfehlen.

3.4.2 Erstellung des Firmware-Corpus

Der zu testende Firmware-Corpus besteht aus sieben verschiedenen, quelloffenen Router-Firmwares. Neben dem für die Technische Richtlinie verwendeten Abbild von OpenWrt

Minimal	Recommended	Software
4 Cores	16 Cores	git
8GB RAM	64GB RAM	python 3.5 - 3.8
10 GB disk space	10* GB disk space	OS see below

Abbildung 3.11: Minimale und empfohlene Systemvorraumsetzungen für FACT. Die Grafik stellt auch die benötigte Software dar. [47]

Version 19.7.04, wurden noch sechs weitere Alternativen gewählt, von denen fünf spezifisch für das gewählte TP-Link Model Archer C7 v5 kompiliert sind. Zu den betrachteten Firmwares gehören DD-WRT, Gargoyle Router Management, Gluon, LibreCMC, AdvancedTomato, sowie Version 19.7.05 von OpenWrt. Einzig AdvancedTomato bietet keine Version für den getesteten Router an, weshalb auf eine Version für einen NETGEAR WNDR3700v3 Dual-Gigabit-WLAN-Router zurückgegriffen wurde, da dieser Router ebenfalls eine MIPS Architektur nutzt und im Leistungsumfang vergleichbar ist.

Die aufgelistete Firmware wurde gewählt, da sie in Funktion und Umfang OpenWrt ähnlich sind und die Projekte, denen sie entstammt, ebenfalls mehrere Heimrouter mit einer Codebasis unterstützen. Es wurden keine Firmware-Alternativen gewählt, die auf Desktop Computern oder Servern installiert werden, da diese aufgrund der zur Verfügung stehenden Rechenkapazitäten im Leistungsumfang nicht vergleichbar sind. Das analysierte Korpus wurde am 21.12.2020 erstellt. Es wurde für jede analysierte Firmware die aktuellste Version für den TP-Link AC1750-Dualband-Gigabit-WLAN-Router genutzt, mit Ausnahme des Abbildes von OpenWrt Version 19.07.4 und der Tomato Firmware. Version 19.07.4 wurde getestet, da es sich um die mittels der Technischen Richtlinie geprüfte Version handelt.

Projekt	**Geeignetes Produkt**	**Firmware Version**
AdvancedTomato	NETGEAR WNDR3700v3	3.4-138
DD-WRT	TP-Link Archer C7 v5	12-18-2020-r45036
Freifunk Gluon	TP-Link Archer C7 v5	V2-v2020.2.1
Gargoyle Router Management	TP-Link Archer C7 v5	1.12.0 (stable)
LibreCMC	TP-Link Archer C7 v2	v1.5.3:2020-10-02
OpenWrt	TP-Link Archer C7 v5	19.07.4
OpenWrt	TP-Link Archer C7 v5	19.07.5

Tabelle 3.1: Übersicht über die für die Analyse ausgewählte Firmware. Dargestellt ist jeweils der Name des Projektes, das Produkt für das die Firmware geeignet ist und die ausgewählt Version der Firmware (vollständige Tabelle: Anhang 3.1)

3.4.3 Durchgeführte Tests und Metriken

Um einen Vergleich mit den Ergebnissen des Home Router Security Reports 2020 des FKIE zu ermöglichen, wurden die gleichen Aspekte auch bei den quelloffenen Firmwares analysiert. Es wurden die folgenden sicherheitsrelevanten Aspekte betrachtet:

- Wann wurde das letzte Update für das Gerät veröffentlicht?
- Welches Betriebssystem wird verwendet und wie viele kritische Schwachstellen sind für dieses bekannt?
- Welche vorbeugenden Maßnahmen gegen Exploits werden eingesetzt und wie häufig sind diese aktiviert.
- Ist privates kryptografisches Schlüsselmaterial enthalten?
- Können hartkodierte Login-Daten und bekannte Passwörter in dem Firmware-Abbild gefunden werden?

Die einzelnen Komponenten des „Firmware Analysis and Comparison Tools“ werden mittels des Befehls

```
$ ~/FACT_core/start_all_installed_fact_components
```

Listing 3.12: Befehl zum starten alle Komponenten von FACT

gestartet. Nachdem der lokale Server gestartet ist, werden die Firmware-Abbilder einzeln über die Upload-Funktion hochgeladen. Die folgenden Analyse-Methoden wurden gewählt:

- CPU Architecture
- Crypto Material
- CVE Lookup
- CWE Checker
- Exploit Mitigations
- Known Vulnerabilities
- Software Components
- Source Code Analysis
- Users and Passwords

Die Ergebnisse der automatischen Analyse werden anschließend durch die REST API von FACT ausgelesen und als Grafiken dargestellt, sodass eine direkte Gegenüberstellung der Ergebnisse des FKIE mit den erhobenen Daten möglich ist.

Kapitel 4

Ergebnisse

4.1 Ergebnisse der Technischen Richtlinie

Von 101 „Test Requirement's" konnte der TP-Link Router in 69 getestet werden. Bei den 32 nicht getesteten Fällen handelt es sich in den meisten Instanzen um Funktionalität, welche von dem Gerät ohne weitere Software-Pakete nicht unterstützt wird. So wurden die “Module K – Remote Configuration”, “Modul L – Voice over IP” und “Modul M – Virtual Private Network” vollkommen vom Testvorgang ausgeschlossen. Ebenso wurden „Test Requirement's" nicht geprüft, welche mit den bereits genannten Modulen Gemeinsamkeiten haben. Darüber hinaus fielen Testfälle bezüglich der standardmäßig gesetzten Passwörter und Login-Daten ebenso weg wie solche, die Community-Funktionen testen. Die 69 getesteten Requirements umfassten 109 Test Prozeduren, von denen wiederum 9 als ergebnislos gewertet wurden. Grafik 4.1 zeigt, dass 72% (78 Test Prozeduren) als bestanden gelten, während 22% (24 Test Prozeduren) als durchgefallen gewertet wurden. Nachfolgend werden zunächst die bestandenen Testfälle betrachtet und anschließend Änderungsvorschläge für nicht bestandene Testfälle beschrieben (siehe Unterkapitel 4.2).

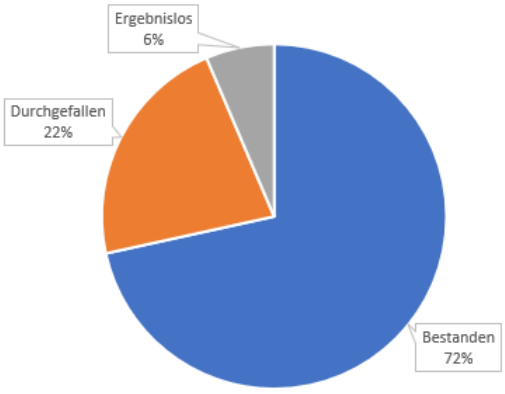


Abbildung 4.1: Darstellung der bestandenen, durchgefallenen und nicht anwendbaren Testfälle der Technischen Richtlinie bei der Untersuchung von OpenWrt. Es wurden insgesamt 69 „Test Procedures“ durchgeführt.

Die Durchführung der Technischen Richtlinie an dem mit OpenWrt betriebenen Gerät zeigte, dass OpenWrt die eigenen Ansprüche an Speicherverbrauch und Funktionalität einhalten kann. Das Gerät liefert Kernfunktionen eines Routers und legt dabei besonderes Augenmerk auf die Reduzierung der angebotenen Dienste auf ein Minimum. Die wiederholten Port-Scans mit nmap zeigten, dass lediglich der Webserver, SSH und der DNS/DHCP Dienst über TCP und UDP betrieben werden (siehe 4.1, 4.2)

```
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times
    will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-17 11:44 CET as nmap -sSCV
    -T4 -p- -Pn 192.168.1.1
Nmap scan report for OpenWrt.lan (192.168.1.1)
Host is up (0.00038s latency).
Not shown: 65532 closed ports
PORT  STATE SERVICE VERSION
22/tcp open ssh Dropbear sshd (protocol 2.0)
53/tcp open domain Cloudflare public DNS
80/tcp open http
MAC Address: B0:95:75:48:F5:EF (Tp-link Technologies)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
    https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2773.20 seconds
```

Listing 4.1: Ergebnisse des OpenWrt nmap Scans für alle **TCP**-Ports des **LAN**-Interfaces.

```
# Nmap 7.91 scan initiated Sat Nov 21 15:20:54 2020 as: nmap -n -sUV
    --version-intensity 0 -p- --max-retries 1 -v -oN UDP_SCAN_RESULT.txt
    192.168.1.1
Warning: 192.168.1.1 giving up on port because retransmission cap hit (1).
Nmap scan report for 192.168.1.1
Host is up (0.00047s latency).
Not shown: 65023 open|filtered ports, 511 closed ports
PORT  STATE SERVICE VERSION
53/udp open domain Cloudflare public DNS
MAC Address: B0:95:75:48:F8:02 (Tp-link Technologies)

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
    https://nmap.org/submit/ .
# Nmap done at Sat Nov 21 15:33:06 2020 -- 1 IP address (1 host up) scanned in
    732.88 seconds
```

Listing 4.2: Ergebnisse des OpenWrt nmap Scans für alle **UDP**-Ports des **LAN**-Interfaces.

Des Weiteren wurden an keiner Stelle Defizite bezüglich der Testkriterien in der Dokumentation der Software gefunden. Alle in der Technischen Richtlinie geforderten Informationen der Dokumentation konnten ermittelt werden. Neben diesen Ergebnissen sind der vollständig quelloffene Code des Betriebssystems und der vollständige root Zugriff auf das Gerät deutliche Indikatoren dafür, dass dem Nutzer keine Funktionen vorenthalten werden (TR.C.2). Durch die Reduzierung auf die wesentlichen Funktionen eines Routers verringert OpenWrt deutlich die Angriffsoberfläche und spielt somit den Zielen der TR-03148 zu. Die Ergebnisse in „Module B – Private Networks“ unterstützen diese Aussage. Der OpenWrt Router stellt keinen Dienst auf der WAN-Schnittstelle zur Verfügung.

```
# Nmap 7.91 scan initiated Thu Nov 26 18:13:44 2020 as: nmap -sSCV -T4 -Pn -p-
    -oN nmap_wan.txt 192.168.178.115
Nmap scan report for OpenWrt.fritz.box (192.168.178.115)
Host is up (0.012s latency).
All 65535 scanned ports on OpenWrt.fritz.box (192.168.178.115) are closed
    (65494) or filtered (41)
MAC Address: B0:95:75:48:F5:F0 (Tp-link Technologies)

Service detection performed. Please report any incorrect results at
    https://nmap.org/submit/ .
# Nmap done at Thu Nov 26 19:06:31 2020 -- 1 IP address (1 host up) scanned in
    3167.21 seconds
```

Listing 4.3: Ergebnisse des OpenWrt nmap Scans für alle **TCP**-Ports des **WAN**-Interfaces.

```
# Nmap 7.91 scan initiated Thu Nov 26 19:21:13 2020 as: nmap -sUV
    --version-intensity 0 -p- --max-retries 2 -v -oN nmap_wan_udp.txt
    192.168.178.115
Warning: 192.168.178.115 giving up on port because retransmission cap hit (2).
Increasing send delay for 192.168.178.115 from 0 to 50 due to 11 out of 20
    dropped probes since last increase.
Nmap scan report for OpenWrt.fritz.box (192.168.178.115)
Host is up (0.0027s latency).
All 65535 scanned ports on OpenWrt.fritz.box (192.168.178.115) are open|filtered
    (56258) or closed (9277)
MAC Address: B0:95:75:48:F5:F0 (Tp-link Technologies)

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
    https://nmap.org/submit/ .
# Nmap done at Thu Nov 26 21:55:05 2020 -- 1 IP address (1 host up) scanned in
    9232.30 seconds
```

Listing 4.4: Ergebnisse des OpenWrt nmap Scans für alle **UDP**-Ports des **WAN**-Interfaces.

So wurde auch das Protokoll Voice over IP (VoIP) nicht in der Standardinstallation mitgeliefert, da dieses für die Funktionalität als Router nicht relevant ist. Ebenfalls konnten die Tests nachweisen, dass OpenWrt international angesehene Standards wie IEEE802.11i erfolgreich inkorporiert.

OpenWrt besteht auch einige weitere Testfälle aufgrund der umfänglichen Informationen und Logs, welche das System für den Nutzer bereitstellt. Das Gerät führt umfassende System- und Kernel-Log Dateien ebenso wie Informationen über verbundene Geräte, aktive und bereitstehende Dienste, Firewall-Funktionen und das System selbst. Auch werden relevante Informationen wie End-of-Support und Mitteilungen zu Sicherheitslücken klar strukturiert auf der Webseite der Entwickler veröffentlicht. Die Veröffentlichung von Updates, welche Sicherheitslücken beheben, erfolgt dabei stets in wenigen Tagen oder Wochen. Vor allem Module F bis I, welche sich mit Firewall, Domain Name System, Dynamic Host Configuration Protocol und dem Zurücksetzen des Gerätes beschäftigen, wurden von OpenWrt vollständig bestanden. Dies ist auf die ebenso quelloffenen Komponenten zurückzuführen, welche schon seit vielen Jahren weiterentwickelt werden und in einigen Bereichen weit verbreitet sind. So nutzt OpenWrt iptables als Firewall und dnsmasq für DNS und DHCP Funktionalität.

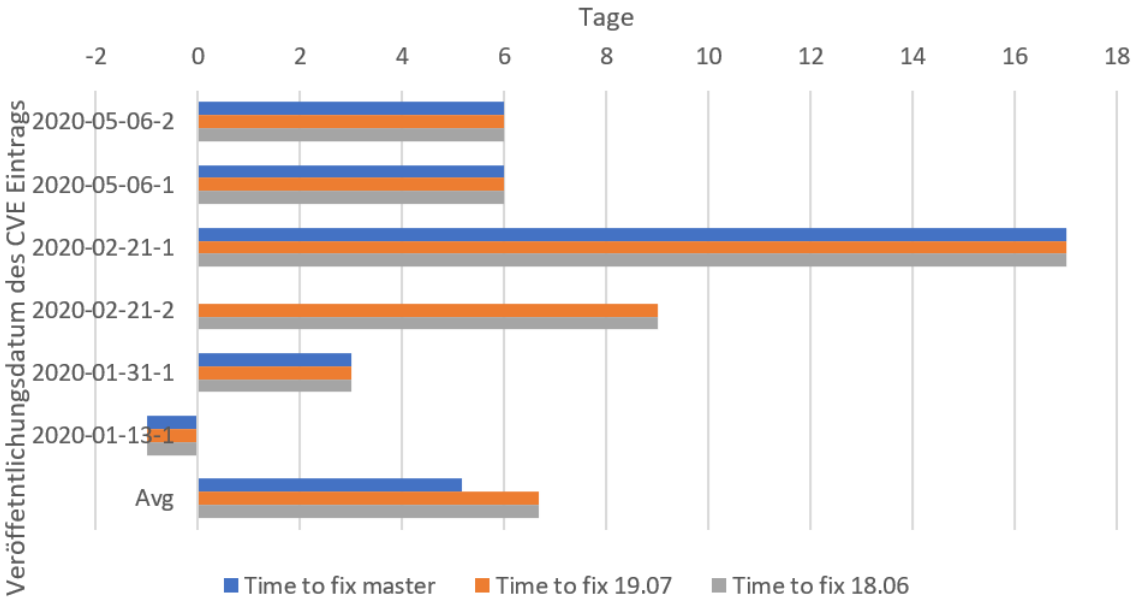


Abbildung 4.2: Benötigte Zeit zur Implementierung und Veröffentlichung von Sicherheitsupdates für OpenWrt im Jahr 2020. Basiert auf dem Veröffentlichungsdatum des CVE Eintrags und dem Datum des git commits, welcher im Sicherheitshinweis spezifiziert ist.

Einige Defizite bzw. gemischte Ergebnisse liefern die Testfälle des WLAN-Gästenetzes. Da OpenWrt keine klassische Funktionalität liefert, welche automatisch ein WLAN-Netz für Gäste bereitstellt, wurde hier auf die Anleitung in der Dokumentation zurückgegriffen, die ein ähnliches Ergebnis erzielen soll, jedoch dem Nutzer alle Freiheiten lässt, Änderungen zu machen. So ist es kein Garant, dass das Gäste-Netz Nutzer tatsächlich separiert oder, dass ein Nutzer nicht von dort aus auf die Konfiguration des Gerätes zugreifen kann.

4.2 Notwendige Änderungen zum Bestehen der TR

Wie bereits im vorherigen Kapitel festgehalten, benötigt OpenWrt nur einige Änderungen, um die Technische Richtlinie 03148 vollumfänglich zu bestehen. Es wird vor allem Wert auf die mit „MUST“ gekennzeichneten Testfälle gelegt. Für Testfälle, die mit „SHOULD“ gekennzeichnet sind wird nachfolgend eine Änderung vorgeschlagen, wenn es sich hierbei um eine simple Anpassung handelt. Darüber hinaus wurde die Funktionalität des Gäste-WLAN nicht weiter betrachtet, da dieses vom Nutzer vollständig konfiguriert werden muss. OpenWrt bietet keine Möglichkeit ein Gäste-Netzwerk mit einer einzigen Option zu aktivieren. Die vollständige Implementierung einer solchen Funktionalität müsste in Betracht gezogen werden.

Es sind nur einige Änderungen von Nöten, um Modul A vollständig zu bestehen. Der 4. Test des Testfalls TR.A.9 (TP.A.9.4) schlägt fehl, da die Verschlüsselung von WLAN-Netzwerken standardmäßig ausgeschaltet ist. Im initialen Zustand ist die gesamte WLAN-Funktionalität von OpenWrt abgeschaltet. Sie muss zunächst vom Benutzer selbst in den Einstellungen, entweder über das Web-Interface oder SSH, eingeschaltet werden. In der Konfigurationsübersicht des jeweiligen WLAN-Netzes ist das Passwort jedoch erst in einem zweiten Reiter untergebracht. Dort ist standardmäßig „No Encryption (open network)“ angewählt. Diese verzweigte Aufteilung kann dazu führen, dass unerfahrene Nutzer lediglich die ESSID anpassen und daraufhin den Speichern-Button betätigen. Auf diese Weise würde das Netzwerk ohne Passwort initialisiert. Das Verschieben des Passwortfeldes, sowie der Auswahl der Verschlüsselung in den ersten (initialen) Reiter der Übersicht, könnte diesem Problem entgegenwirken. Ebenso könnte die initiale Konfiguration der WLAN-Netzwerkes statt „No Encryption“ stattdessen „WPA2-PSK“ ausgewählt haben. So könnte der Nutzer die Konfiguration nicht speichern ohne ein Passwort einzufüllen. Wenn WPA2 ausgewählt ist, erscheint bei einem unzureichenden Passwort eine Fehlermeldung und die Konfiguration wird nicht gespeichert. Auf diese Weise kann dennoch gezielt ein Netzwerk ohne Passwort erstellt werden, wenn der Nutzer bewusst auf diese Option umgeschaltet hat. Der Nutzer könnte ebenfalls von einem Mechanismus unterstützt werden, welcher die Stärke des

WLAN-Passwortes darstellt. Ein ähnlicher Mechanismus wird bereits bei der Prüfung des Geräte-Passwortes eingesetzt und könnte auch im Umfeld des PSK dem Nutzer zusätzliche Hilfestellung bei der Wahl eines Passwortes geben. Dabei sollte der bestehende Mechanismus zur Evaluation des Passwortes allerdings angepasst werden, sodass die Vorgaben der Technischen Richtlinie eingehalten werden.

Ein weiterer Test, welcher während der Durchführung der Technischen Richtlinie scheiterte, ist TR.D.2. Dieser beschreibt, dass der Zugang zur Konfiguration des Gerätes mindestens durch ein Passwort geschützt sein muss, wenn das Gerät sich im initialen oder kundenspezifischen Zustand befindet. Aufgrund der Natur vom OpenWrt als Alternatives Router-Betriebssystem, welches erst nach Erhalt des Gerätes vom Nutzer aufgespielt wird, ist ein Passwort im „factory“-Zustand nicht sinnvoll. Da kein einzigartiges Passwort vergeben werden kann, bevor OpenWrt vom Nutzer eingesetzt wird, würde das Gerät keinen höheren Sicherheitsansprüchen genügen, wenn ein Benutzeraccount mit Passwort voreingestellt wäre. Aufgrund der anhaltenden Nutzung des root Benutzers auf OpenWrt Systemen ist es dem Benutzer allerdings vollkommen freigestellt, diesen Account ohne Passwort zu betreiben. Lediglich ein kleiner Informationstext im Web-Interface erinnert an das Setzen eines Passwortes. Ebenfalls kann über den SSH-Zugang ein bereits gesetztes Passwort gelöscht werden, sodass der Account dann wieder ohne Passwort eingesetzt werden kann. Dies stellt ein hohes Sicherheitsrisiko dar. Jedoch könnte dieses Problem umgangen werden, wenn der Nutzer entweder gezwungen würde, ein Passwort für den root Nutzer zu verwenden, um das Gerät zu initialisieren, oder wenn der Nutzer dazu gezwungen werden würde, einen neuen Nutzeraccount anzulegen und sowohl für den root-Benutzer als auch für den eigenen Nutzeraccount ein Passwort festzulegen. Daraufhin sollte der Nutzer seinen eigenen Account zur Konfiguration des Gerätes nutzen und lediglich auf den root-Benutzer zurückgreifen, wenn höhere Privilegien benötigt werden. Es könnte standardmäßig ein unprivilegierter Nutzeraccount installiert sein und zusätzlich auf ein Programm wie „sudo“ gesetzt werden. Dadurch, dass das „passwd“ Programm durch den root-Nutzer ausgeführt wird, werden alle Überprüfungen des Passwortes übersprungen, bzw. alle Fehlermeldungen ignoriert. Das „passwd“ Dienstprogramm wird verwendet, um Benutzerpasswörter zu ändern oder zu entfernen. Dieses Vorgehen würde ebenfalls dafür sorgen, dass Kriterien wie TR.D.10 und TR.D.15 kein Problem mehr darstellen. So müsste ein Nutzer zunächst das alte Passwort eingeben, um ein Neues zu wählen. Ebenfalls könnte ein Nutzer gehindert werden, ein schwaches Passwort zu wählen.

Auch wenn es sich dabei nur um ein „SHOULD“-Kriterium handelt, ist HTTPS mit Transport Layer Security eine sicherheitskritische Technologie (TR.D.3) [SOURCE]. Außer eine eigene „Certification Authority“ wird durch die Entwickler von OpenWrt etabliert, verbleibt realistisch die Möglichkeit selbst-signierte Zertifikate zu verwenden, auch wenn

ein Nutzer dann in den meisten Fällen eine Sicherheitswarnung des Browsers akzeptieren muss. Ebenfalls besteht die Möglichkeit den gesamten Verkehr mit SSH zu verschlüsseln. Der Einsatz von HTTPS ist außerdem zu empfehlen, da z.B. der Firefox Browser mit dem „https-only-mode“ den Einsatz von HTTPS erzwingen kann, sodass HTTP-Webseiten nicht mehr aufgerufen werden können [SOURCE].

OpenWrt zeigt zusätzlich einige Schwächen bei der Implementierung von Sicherheitsmaßnahmen gegen Bruteforce Angriffe auf den Login-Bereich sowie gegen Session-Hijacking Attacken. Da OpenWrt durchaus die fehlgeschlagenen Login-Versuche registriert, wäre ein Zähler die einfachste Option Bruteforce Angriffe auf die Login-Bereiche zu verhindern. Es könnten z.B. eine begrenzte Anzahl an Versuchen zur Verfügung stehen. Nachdem diese abgelaufen sind, muss eine gewisse Zeit gewartet werden, bis ein neuer Versuch unternommen werden kann. Ebenso könnte ein Zeitlimit zwischen jedem Login-Versuch implementiert werden, sodass ein Angriff verlangsamt wird. Wenn das Passwort ausreichend komplex gewählt ist, so würde dies die Geschwindigkeit und Attraktivität von Bruteforce Angriffen deutlich mindern. Session-Hijacking Angriffe könnten verhindert werden, wenn zusätzlich zu den „anti-cross-site-forgery-request“ (anti-CSFR)–Token der „Session-Timer“ verringert würde. Das kontinuierliche, automatische Updaten der auf der Seite dargestellten Informationen setzt diesen Timer jedoch alle fünf Sekunden zurück. Dieses Verhalten muss unterbunden werden. 300 Sekunden (5 Minuten) für eine Sitzung wären ein geeigneteres Intervall anstelle von den derzeit eingesetzt 1300 Sekunden (60 Minuten).

Abschließend deckte die Durchführung der Technischen Richtlinie noch einen weiteren Schwachpunkt des OpenWrt Betriebssystems auf. Die Testfälle TR.E.5 bis TR.E.8 wurden alle nicht bestanden, da es keinen automatischen Authentifizierungsmechanismus gibt, welcher Firmware-Updates prüft. Dem Nutzer werden lediglich signierte SHA256 Hash-Werte des Firmware-Updates zur Verfügung gestellt, sodass die Authentizität und Integrität der Datei vom Nutzer geprüft werden muss.

> „ [...] while release image files are usually signed by one or more developers with detached GPG signatures to allow users to verify the integrity of installation files. [...] Note that not every file is signed individually but that we’re signing the sha256sums or - for repositories - the Packages files to establish a chain of trust: The SHA256 checksum will verify the integrity of the actual file while the signature will verify the integrity of the file containing the checksums. [49]“

Dieser Ansatz bietet natürlich nur die geforderten Schutzziele, wenn ein Nutzer tatsächlich die Authentizität und Integrität der sha256sums Datei prüft und anschließend den darin enthalten Hash-Wert mit dem der heruntergeladenen Datei vergleicht. Ebenso wie der OPKG Paket Manager könnte auch der Firmware-Update-Mechanismus auf die usign Ed22519 Signaturen zurückgreifen oder eine GPG-Signatur der Entwickler tragen. Eine automatische

Verifizierung der Signatur über das Internet, mit entsprechenden Sicherheitsmaßnahmen, wäre dann möglich. In den Fällen, in denen das Gerät keine Internetverbindung aufweist, könnte dann auf die SHA256 Hash-Werte zurückgefallen werden. Durch das beschriebene Vorgehen könnte das Gerät TR.E.5 bis TR.E.8 bestehen sowie den Nutzern mehr Sicherheit und Nutzerfreundlichkeit bieten.

In einigen Fällen lässt sich keine sinnvolle Lösung finden, die für ein Projekt wie OpenWrt geeignet ist. So auch bei TR.D.24, welches fordert, dass dem Nutzer eine Nachricht auf dem Gerät angezeigt wird, wenn eine neue Firmware verfügbar ist. Für diese Anforderung müsste das Gerät mit dem Internet verbunden sein und zudem müssten von Seiten der OpenWrt Entwickler ein Update-Server zur Verfügung gestellt werden. Dies würde Kosten auf Seiten der freiwilligen Entwickler sowie eine größere Angriffsoberfläche auf Seiten der Nutzer verursachen. Eine Anmeldung beim E-Mail Newsletter der Entwickler wäre die simpelste Möglichkeit um über neue Versionen sowie Sicherheitslücken informiert zu bleiben.

4.3 Ergebnisse der statischen Code-Analyse sowie Gegenüberstellung mit ausgewählten Ergebnissen des Home Router Security Reports 2020

Im Rahmen dieser statischen Code-Analyse durch das „Firmware Analysis and Comparison Tool “ des FKIE wurden sieben verschiedene quelloffene Router-Firmware Alternativen analysiert. Dabei waren fünf Fragen von besonderem Interesse.

- Wann wurde das letzte Update für das Gerät veröffentlicht?
- Welches Betriebssystem wird verwendet und wie viele kritische Schwachstellen sind für dieses bekannt?
- Welche vorbeugenden Maßnahmen gegen Exploits werden eingesetzt und wie häufig sind diese aktiviert.
- Ist privates kryptografisches Schlüsselmaterial enthalten?
- Können hartkodierte Login-Daten und bekannte Passwörter in dem Firmware-Abbild gefunden werden?

FACT konnte während der Analyse erfolgreich 92,73% der Daten aus den Firmware-Abbildern extrahieren. Bei der gesamten betrachteten Firmware wurde durch Analyse von Metadaten eine MIPS 32-Bit Architektur mit „big-endian“ Byte-Reihenfolge festgestellt. Für die Analyse der „Critial Vulnerabilities and Exposures“ (CVE) wurde aufgrund einiger Fehler in FACT nicht das Ergebnis der automatischen Analyse gewählt. Stattdessen wurden

die Ergebnisse durch die Webseite www.cvedetails.com, welche wiederum auf die Daten der „National Vulnerability Database“ der US-Regierung, zugreift, bereitgestellt. Da cvedetial.com ausschließlich CVSS v2 Bewertungen bereitstellt, wurden einzig diese für die Analyse verwendet. Ein CVE-Eintrag hat einen Schweregrad von „Hoch“, wenn der CVSS v2 Wert sieben oder höher beträgt. Um Vergleichbarkeit mit den Ergebnissen des FKIE zu gewährleisten wurden lediglich CVE-Einträge mit einem Schweregrad von „Hoch“ gezählt. [ERKLÄRUNG ZU CVE SYSTEM UND CVSS]

4.3.1 Vergangene Tage seit der letzten Veröffentlichung eines Firmware-Updates

In diesem Abschnitt soll evaluiert werden, wann für die betrachteten Firmware-Abbilder das letzte Mal eine neue Version seit dem 24.12.2020 veröffentlicht wurde. Alle Abbilder des Firmware-Corpus spezifizierten das Veröffentlichungsdatum im Dateinamen selbst oder auf der jeweiligen Webseite. Dieses Kriterium wurde untersucht, da es die Bereitschaft der Entwickler andeutet, ihr Projekt regelmäßig mit Funktions- und Sicherheitsupdates zu unterstützen. Eine neuere Version bedeutet also zumeist, dass weniger sicherheitsrelevante Lücken bekannt sind und das System sicherer ist. Da die Unterstützer der quelloffenen Projekte in vielen Instanzen auf weitere Software zurückgreifen und auch diese Updates erfährt, ist es wahrscheinlich, dass Firmware bekannte Lücken hat, wenn diese längere Zeit nicht erneuert wurde.

Grafik 4.3 zeigt, dass für fünf von sieben untersuchten Firmware-Abbildern in den letzten 365 Tagen eine neue Version veröffentlicht wurde. Aufgrund des Ausreißers in den erhobenen Daten wird im folgenden der Median statt des Mittelwertes betrachtet. Es ergibt sich, dass die Router-Betriebssysteme nach Median-Berechnung alle 83 Tage und im Schnitt alle 309 Tage eine neue Version erhalten. Ebenfalls muss erwähnt werden, dass bei der Veröffentlichung einer neuen Version meist alle von dem jeweiligen Projekt unterstützen Geräte diese neue Version zur Verfügung gestellt bekommen. So werden bei einer neuen Version von OpenWrt alle ca. 1700 Geräte von diesem neuen Update unterstützt und erfahren somit alle Sicherheitsupdates, die bereitgestellt werden. Dies steht im Gegensatz zu etablierten Herstellern von Routern, welche oft pro Gerät eine eigene Version entwickeln. Gargoyle Router Management wurden nicht in den letzten 365 Tagen erneuert und das Tomato Betriebssystem hat in den letzten 1480 Tagen kein Update erfahren. Der Zyklus von 83 Tagen ist höher als der Update-Zyklus von Desktop- oder Server-Betriebssystemen, jedoch noch im Rahmen der 90 Tage, welche normalerweise das Zeitfenster darstellen, in dem Entwickler Zeit haben auf Sicherheitslücken und Probleme zu reagieren (responsible disclosure) [50]. Darüber hinaus muss bedacht werden, dass in manchen Fällen ein Paketmanager zur Verfügung steht, über welchen Updates für Pakete während der Laufzeit installiert werden können.

Somit sind Updates der Firmware nur notwendig, um Kernfunktionalität zu erweitern oder Fehler in dieser zu beheben, sowie um den Kernel zu aktualisieren. Nur eine von acht bisher veröffentlichten Sicherheitslücken im Jahr 2020 konnte ausschließlich durch ein Update auf eine neuere Version von OpenWrt behoben werden, wobei alle weiteren durch ein einfaches Update des betroffenen Paketes nachgebessert werden konnten [SOURCE]. Verglichen mit den Ergebnissen des Home Router Security Reports 2020 zeigt sich, dass für die quelloffenen Betriebssysteme häufiger neue Versionen veröffentlicht werden. Wenn man die analysierten Abbilder als Gruppe betrachtet, dann schneidet diese vergleichsweise gut ab. Einzig Tomato fällt als Ausreißer heraus. Lediglich ASUS, AVM und Netgear, als Hersteller von handelsüblichen Routern, können mithalten.

Ebenso wie im Home Router Security Report 2020 festgestellt, muss zusätzlich beachtet werden, dass alle betrachteten Produkte kleinere Updates auch über die Geräte selbst zu Verfügung stellen könnten, sodass die aktuellste Version nicht im Internet veröffentlicht wird. Darüber hinaus handelt es sich bei den hier festgestellten Daten ausschließlich um eine Momentaufnahme, die keine Aussagekraft darüber hat, ob regelmäßig Updates bereitgestellt werden, oder ob diese Sicherheitslücken überhaupt adressieren [47].

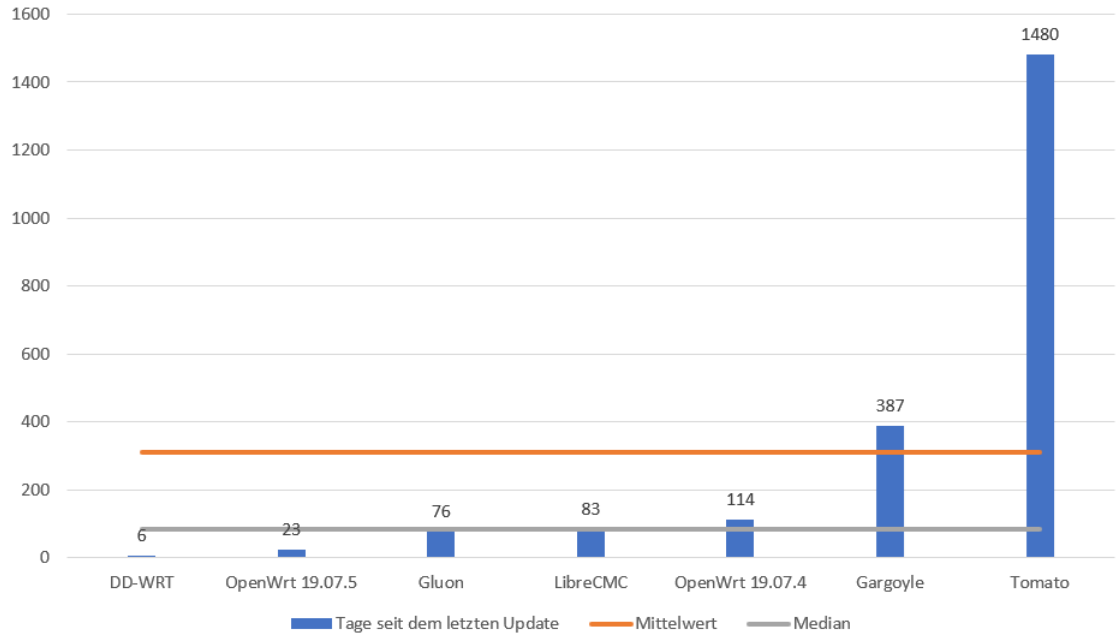


Abbildung 4.3: Vergangene Tage seit der letzten Aktualisierung der betrachteten Firmware. Seit dem letzten Update von DD-WRT ist am geringsten Zeit vergangen. Für Tomato ist seit 1480 Tagen keine neue Version erschienen. (Stand: 30.12.2020)

4.3.2 Betriebssysteme

Da es sich bei allen analysierten Firmware-Abbildern um quelloffene Projekte handelt, ist es nicht verwunderlich, dass der Linux-Kernel dominant vertreten ist. Der Linux-Kernel, welcher 1991 von Linus Tovalds entwickelt wurde und seither stetig weiterentwickelt wird stellt einen der am häufigsten genutzten Betriebssysteme für IoT Geräte dar [51]. Die geringe Größe des Kernels, der große Funktionsumfang und die umfangreiche Dokumentation und Verbreitung sind für eine Community-getriebene Entwicklung auf Speicher- und Rechenleistungslimitierten Geräten wie z. B. Heim-Routern gut geeignet. Grafik 4.4 zeigt, dass alle untersuchten Projekte einen Linux Kernel verwenden. Dieser Trend deckt sich ebenfalls mit den Ergebnissen des Home Router Security Reports des FKIE. In den untersuchten Produkten des Verbrauchermarktes wurde Linux in 91% der Fälle verwendet [47, p. 2].

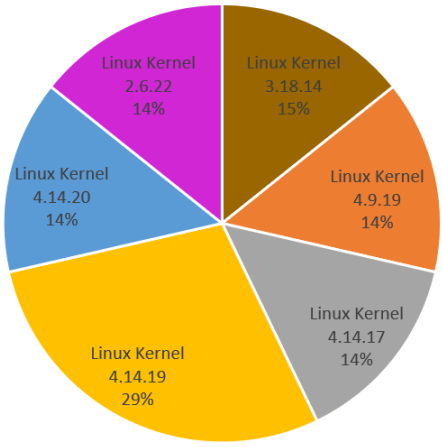


Abbildung 4.4: Verwendete Linux Kernel der betrachteten Firmware.

Aufgrund der unzureichenden Ergebnisse der FACT-Analyse bezüglich der vorhandenen CVE-Einträge für die verwendeten Linux-Kernel wurden die Ergebnisse in diesem Fall direkt über www.cvedetails.com abgerufen. FACT erstellt zunächst eine „Common Platform Enumeration“ (CPE) der Software Version und stellt mit dieser CPE eine Anfrage an die „National Vulnerability Database“. Da die zurückgegebenen Ergebnisse allerdings auch Schwachstellen beinhalten, welche nur für bestimme Geräte mit der jeweiligen Linux-Kernel Version gelten, wurden die jeweiligen Schwachstellen des Kernels über die Website cvedetail.com abgefragt. Bei einer Stichprobe der von FACT gelieferten CVE-Einträge sind verschiedene Einträge aufgefallen, welche z.B. nur für bestimmte IoT-Geräte wie Smartwatches eingetragen sind. Cvedetail nutzt ebenfalls die „National Vulnerability Database“, stellt jedoch noch zusätzliche Informationen und Statistiken bereit. Auf diese Art wurde sicher-

gestellt, dass die betrachteten Schwachstellen spezifisch für den Kernel sind und nicht für ein bestimmtes Gerät, welches diesen Kernel nutzt. Da nicht alle eingetragenen CVEs eine direkte Bedrohung darstellen, wurden die Ergebnisse weiter eingeschränkt. So wurden lediglich solche CVEs betrachtet, welche mit einen CVSS2 (Common Vulnerability Scoring System) Wert von sieben oder höher eingestuft wurden. Dies ist ein Bewertungssystem, mit welchem CVE Einträge kategorisiert werden, sodass der Schweregrad der Sicherheitslücke durch einen Wert definiert werden kann. Da das neuere Format, CVSS3, nicht durch cvedetails.com bereitgestellt wird, wird es in der Analyse vernachlässigt. Wie Grafik 4.6 zeigt, stehen für alle betrachteten Geräte einige CVE Einträge des Linux Kernels zur Verfügung. Ebenfalls kann man sehen, dass der in DD-WRT verwendete Kernel mehr CVE Einträge hat als der von Gargoyle Router Management, obwohl bei DD-WRT die geringste Zeit seitdem letzten Firmware Update vergangen ist. Tomato schneidet erneut am schlechtesten ab. Grafik 4.5 zeigt zusätzlich, dass für zwei der sechs verschiedenen Linux Kernel schon seit einigen Jahren keine Sicherheitsupdates entwickelt werden. Sowohl der von Tomato verwendete Kernel, 2.6.22, als auch Linux Kernel 3.8.14, welcher von DD-WRT verwendet wird, werden nicht mehr mit Updates unterstützt. Dies spiegelt sich auch in der hohen Anzahl CVE Einträge wider (siehe Abbildung 4.6).

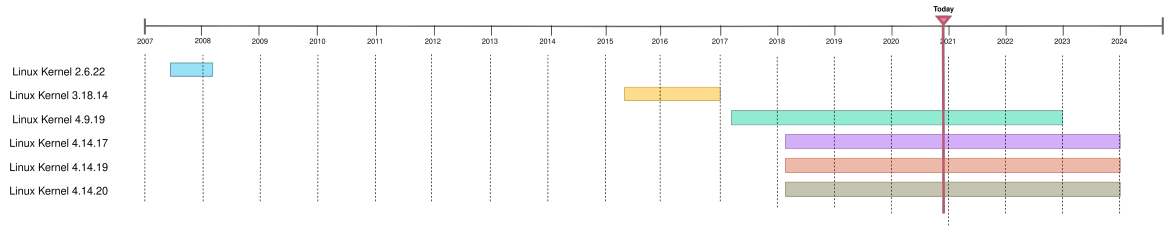


Abbildung 4.5: Stellt jeweils den Zeitraum von Veröffentlichung bis zum Ende der Entwicklerunterstützung für alle gefundenen Linux Kernel dar. Der Kernel 4.14.19 wurde in zwei Firmware-Abbildern genutzt.

Die Ergebnisse sind aufgrund der unterschiedlichen Beschaffung sowie der fehlenden CVSS3 Werte nicht wirklich mit denen des Home Router Security Reports 2020 vergleichbar. Jedoch kann man sagen, dass die quelloffenen Router-Betriebssysteme mehrheitlich modernere Linux-Kernel Versionen nutzen. Lediglich zwei der betrachteten Firmware-Abbilder nutzen einen Kernel, der nicht mehr unterstützt wird. Der „Security Report“ gibt an, dass ein Drittel der betrachteten Geräte einen Kernel vor Version 3 nutzen und lediglich ca. 22% einen aktuellen Kernel der 4. Version [47, p. 8]. Im Gegensatz dazu nutzen ca. 70% der betrachteten quelloffenen Software-Projekte einen Linux-Kernel der Version 4.9.19 oder höher (siehe Abbildung 4.5).

Im Gegensatz zu den Ergebnissen des Security Reports können falsch positive Ergebnisse bei der Erkennung der Kernel Version beinahe ausgeschlossen werden, da diese ebenfalls von den Entwicklern auf der Webseite oder in den Veröffentlichungsdokumenten

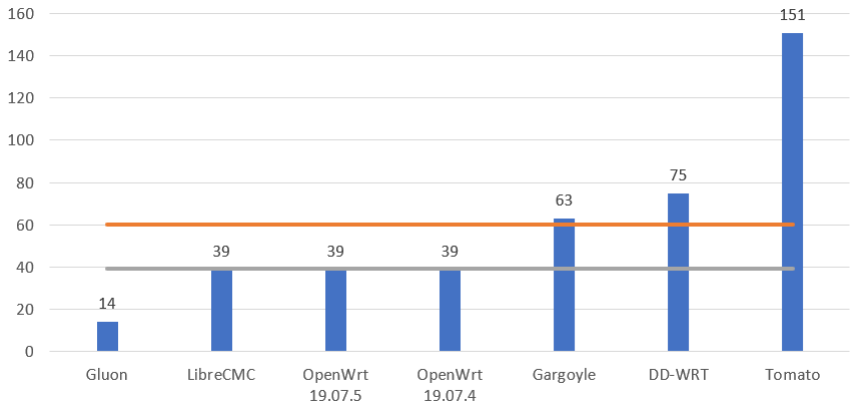


Abbildung 4.6: Anzahl der gefundenen CVE Einträge pro Firmware mit einer CVSS-Einschätzung von sieben oder höher.

der jeweiligen Version veröffentlicht wird. Jedoch besteht die Möglichkeit, dass die Entwickler eigene Korrekturen für Sicherheitslücken des Kernels entwickeln und veröffentlichen. Dies ist bei dieser Art Community-getriebener Entwicklung nicht unwahrscheinlich, da hier keine Entwickler bezahlt werden müssen, welche zusätzlich zu ihren anderen Aufgaben für das Beheben von Sicherheitslücken im Kernel eingesetzt werden. Ebenfalls ist es möglich, dass aufgrund der uneindeutigen CPE-Spezifikation einige CVE-Einträge nicht von cvedetails.com gelistet werden [47, p. 7].

4.3.3 Härtungsmaßnahmen

Die betrachteten Firmware Abbilder wurden auf die folgenden Exploit Mitigationen getestet:

- **Stack Canary**: Es handelt sich hierbei um eine zufällig gewählte Byte-Sequenz, welche vor die „return“-Adresse auf den Stack geschrieben wird, um Overflows zu erkennen. Wenn es zu einem Buffer-Overflow kommt, würde diese Sequenz überschrieben und diese kann somit nicht vor dem Zurückkehren (return) verifiziert werden [52].
- **FORTIFY_SOURCE** ist eine zusätzliche Option der GCC Compiler Collection. Wenn diese Option beim Kompiliervorgang von Dateien ausgewählt wird, werden verschiedene Funktionen zur Manipulation von Zeichenketten und Speicher (memcpy, memset, strcpy, strcat, sprintf, gets, ...) während der Ausführung auf Pufferüberläufe (buffer overflow) geprüft. Dies schützt meistens nicht vor gezieltem Ausnutzen dieser Funktionen aber vor der Korruption des Heaps und Stacks durch Systemfehler [53].
- **Non-Executable Bit (NX)**: Dieses besondere Bit markiert Bereiche des Speichers als reine Datenspeicherbereiche. Dadurch wird sichergestellt, dass in diesen Bereichen, in

denen kein Code ausgeführt werden sollte, auch kein Code ausgeführt werden kann. Diese Separierung findet sich sonst nur in Harvard-Architekturen [47, p. 11].

- **Position-Independent Executable (PIE)** (positionsunabhängiges ausführbares Programm) bezeichnet eine Technik, bei welcher Programm-Code an einer zufälligen Speicheradresse geladen wird. Hierbei wird nicht mit absoluten, sondern relativen Speicheradressen gearbeitet. Dies erschwert zwar Angriffe, da ein Angreifer zunächst die absolute Speicheradresse finden muss, jedoch verlangsamt diese Technik unter Umständen auch die Ausführung des Codes [54].
- **RELocation Read-Only (RELRO)** schützt den "Global Offset Table" (GOT) gegen Manipulationen während der Laufzeit. Der Global Offset Table beinhaltet die Speicheradressen von gemeinsam genutzten Libraries oder globalen Variablen, sodass diese von einem Programm genutzt werden können. Wenn die RELRO Option beim Kompiliervorgang ausgewählt wurde, dann wird nach dem Start des Programms ein reiner Lesezugriff auf den GOT festgelegt [47, p. 12].

Sowohl RELRO als auch das NX-Bit werden vermehrt bei den quelloffenen Router-Betriebssystemen eingesetzt (siehe Abbildung 4.7). Außer Tomato nutzen alle der betrachteten Firmwares zu beinahe 100% das NX-Bit. Mit Ausnahme von Tomato und DD-WRT nutzen im Schnitt ca. 50% aller ausführbarer Dateien der Firmware-Abbilder RELRO. Tomato und DD-WRT setzen hingegen kaum auf RELRO. PIE wird andererseits im Schnitt zu ca. 40% genutzt. Tomato scheint bevorzugt auf PIE zu setzen (siehe Abbildung 4.8). Die Nutzung von Stack Canaries und FORTIFY_SOURCE verhält sich pro Firmware nahezu identisch. Gargoyle Router Management, LibreCMC und OpenWrt nutzen es bei ca. 19% aller Dateien, Gluon bei ca. 8%, während DD-WRT und Tomato beinahe vollständig auf diese Techniken verzichten. Die Verbreitung von PIE ist vergleichbar mit den Ergebnissen der FKIE Veröffentlichung (siehe Abbildung 4.7) [47, p. 15]. Ebenso wie im Security Report berichtet, nutzen auch die quelloffenen Betriebssysteme annähernd alle vollumfänglich NX-Bits. Dies lässt sich leicht durch den vergleichsweise guten Schutz bei infinitesimalen Geschwindigkeitseinbußen erklären. Die Daten des FKIE zeigten, dass RELRO nur selten von allen Herstellern eingesetzt wird mit Ausnahme von AVM. Dem steht eine Nutzung von ca. 50% bei den freien Firmware-Produkten gegenüber. Ebenso wie die betrachtete Firmware der Markthersteller, wird nur selten auf Stack Canaries und FORTIFY_SOURCE gesetzt. Obwohl Stack Canaries keinen merkbaren Einfluss auf die Geschwindigkeit eines Systems hat, scheint diese Technik nur bei wenigen Dateien angewendet worden zu sein. Es könnte sich hierbei um systemkritische Dateien handeln. Dies gilt ebenso für die FORTIFY_SOURCE Option (siehe Abbildung 4.8).

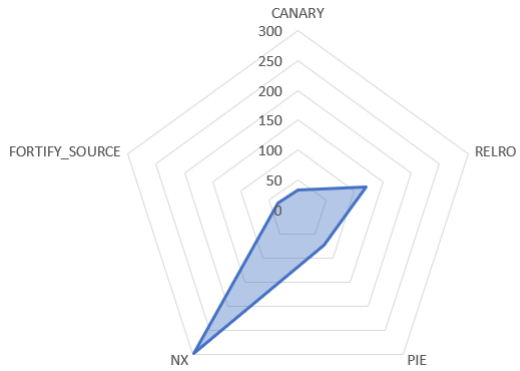


Abbildung 4.7: Netzdiagramm der verwendeten Härtungsmaßnahmen. Es werden alle Firmware-Abbilder gemeinsam dargestellt. NX, RELRO und PIE werden am häufigsten genutzt. Stack Canaries und FORTIFY_SOURCE wird kaum eingesetzt.

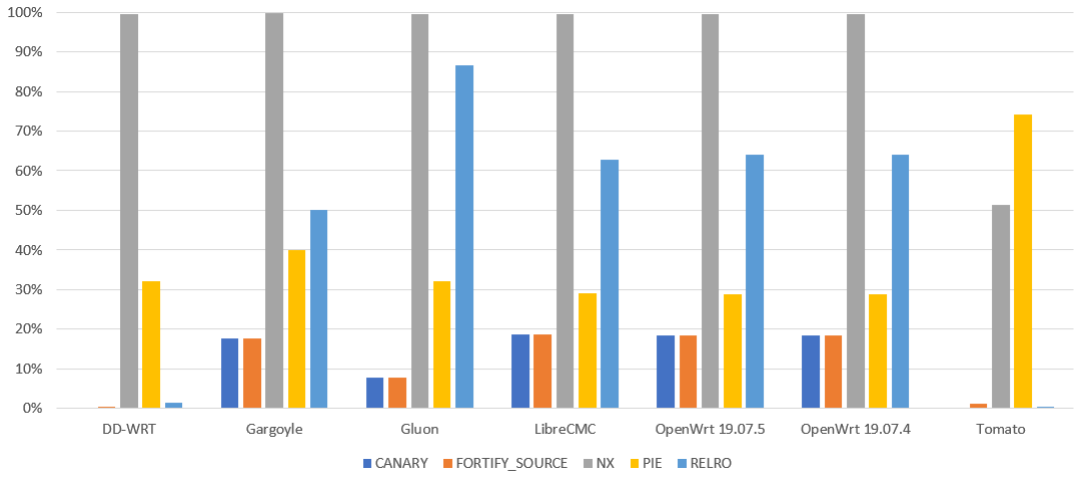


Abbildung 4.8: Säulendiagramm der verwendeten Härtungsmaßnahmen nach Analyse der FACT Daten. Pro Firmware sind Stack Canaries, FORTIFY_SOURCE, NX, PIE und RELRO dargestellt.

Zusammenfassend kann man sagen, dass vor allem auf NX und RELRO für den Großteil der Dateien gesetzt wird. PIE, Stack Canaries und FORTIFY_SOURCE wird nur bei wenigen ausführbaren Dateien genutzt.

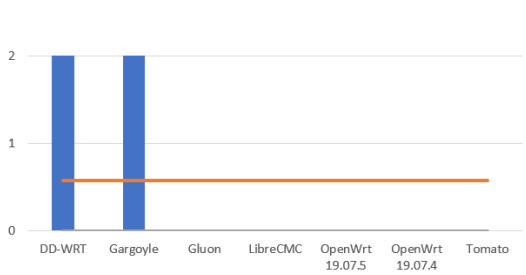
4.3.4 Privates Schlüsselmaterial

Wenn private kryptographische Schlüssel in den Firmware-Abbildern enthalten sind, so haben diese keine Sicherheitsfunktion mehr. Um die korrekte Funktionalität zu gewährleisten, in dem Fall, dass private Schlüssel enthalten sein müssen, so sollten die Vorgaben der OWASP eingehalten werden:

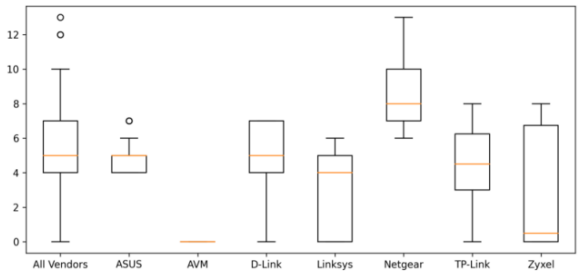
> “Do not hardcode secrets such as passwords, usernames, tokens, private keys or similar variants into firmware release images. This also includes the storage of sensitive data that is written to disk. If hardware security element (SE) or Trusted Execution Environment (TEE) is available, it is recommended to utilize such features for storing sensitive data. Otherwise, use of strong cryptography should be evaluated to protect the data. If possible, all sensitive data in clear-text should be ephemeral by nature and reside in a volatile memory only.” [55]

Die Einhaltung dieser Vorgaben ist jedoch deutlich erschwert, wenn die Firmware nicht spezifisch für ein Gerät geschrieben ist. Ebenso stehen den Entwicklern der quelloffenen Firmware nicht alle Entwicklerwerkzeuge der Hersteller zur Verfügung um z.B. auf ein „Hardware Security Element“ zuzugreifen. Zugleich wird für den Zugriff in einigen Fällen ein physischer Zugang zu dem Gerät benötigt.

Trotz dieser Probleme konnte FACT nur aus DD-WRT und Gargoyle Router Management private Schlüssel extrahieren. Bei beiden Betriebssystemen wurden jeweils ein Pkcs8PrivateKey sowie ein SSLPrivateKey gefunden. Da PKCS#8 ein Container-Format für private kryptographische Schlüssel ist, kann man ohne weitere Nachforschung nicht bestimmen, welchen nutzen diese Schlüssel für die Systeme haben. Die gefundenen SSL-Schlüssel dienen vermutlich dazu den vom Webbrowser an den Webserver gesendeten Session-Key zu entschlüsseln [56]. Es lässt sich also vermuten, dass DD-WRT und Gargoyle Transport Layer Security verwenden, jedoch kann ein Man-in-the-Middle Angriff einfach durchgeführt werden, wenn der private SSL Schlüssel bekannt ist [57]. Die genauen Details der Implementierung und Nutzung der gefundenen Schlüssel ist jedoch vollkommen unbekannt. Es könnte sich ebenso um ungenutztes oder veraltetes Material handeln. Darüber hinaus könnte der SSL Schlüssel auch nur für die initiale Konfiguration des Gerätes genutzt werden, um danach durch einen neuen ersetzt zu werden.



(a) Teilbild 1: Quelloffene Betriebssysteme



(b) Teilbild 2: Ergebnisse der FKIE Untersuchung

Abbildung 4.9:
Teilbild 1: Anzahl der privaten Schlüssel in den untersuchten quelloffenen Betriebssystemen. Lediglich in DD-WRT und Gargoyle wurden private Schlüssel gefunden.
Teilbild 2: Anzahl der privaten Schlüssel in der vom FKIE untersuchten Firmware. Außer bei AVM wurden bei allen Hersteller private Schlüssel gefunden [47, p. 17].

4.3.5 Angelegte Benutzeraccounts

Diese Analyse dient dazu hartkodierte Accountinformationen in der untersuchten Firmware zu finden. FACT extrahiert dazu die Daten aus der „/etc/shadow“ und „/etc/passwd“ Datei. Diesen Dateien speichern Informationen zu allen Nutzeraccounts, welche auf dem System angelegt sind. Dazu gehören unter anderem der Nutzername, das Passwort, die Nutzerrechte und weitere Informationen der Nutzer. Das Passwort, welches in der „/etc/shadow“ Datei gespeichert wird, liegt in Hash-Form vor. FACT nutzt eine Passwortliste mit häufig genutzten Passwörtern und das Programm „john“, um das Passwort im Klartext darzustellen. Problematisch sind bereits angelegte Nutzeraccounts vor allem, wenn diese nicht geändert oder abgeschaltet werden können. Ebenso bergen sie das Risiko, dass ein unerfahrener Nutzer diesen Account benutzt, ohne ein neues Password für den Account festzulegen. Auf diese Art kann ein Angreifer sehr einfach auf die Konfiguration des Gerätes zugreifen.

Die Analyse der quelloffenen Firmware zeigt, dass lediglich Gargoyle Router Management einen bereits angelegten Nutzeraccount mit schwachem Passwort ausweist. Im Test des FKIE wurden auf 50 Geräten (40%) vom Hersteller angelegte Accounts gefunden [47, p. 19]. Da es sich bei dem Befund des Gargoyle Betriebsystems allerdings um den root Account handelt, ist es nicht unwahrscheinlich, dass der Nutzer nach einmaliger Eingabe des Passwortes „password“ ein neues Passwort wählen muss. In diesem Falle bietet Gargoyle Router Management nicht mehr Sicherheit als OpenWrt, bei welchem der root Account ohne Passwort initialisiert ist. In dieser Instanz ist es umso wichtiger, dass der Nutzer auf das Risiko ausreichend hingewiesen wird, bzw. aufgefordert wird, dass Passwort zu ändern.

Kapitel 5

Diskussion

5.1 Zusammenfassung der Ergebnisse

Die Durchführung der Technischen Richtlinie hat gezeigt, dass OpenWrt trotz einiger Schwächen eine gute Ausgangslage zum vollständigen Bestehen der TR hat. 69 von 101 „Test Requirement's" konnten geprüft werden, dies entspricht 109 Test Prozeduren bzw. 68% aller Testfälle der TR. OpenWrt konnte 72% der getesteten Fälle bestehen, während das Ergebnis zu 22% negativ ausfiel. In 6% der Fälle wurde der Testfall als ergebnislos gewertet. Ebenso lieferte die Analyse mittels FACT weitere positive Ergebnisse für OpenWrt und die anderen quelloffenen Betriebssysteme. Die meisten der analysierten Firmware wurde im letzten Jahr mit Updates versorgt, fünf von sieben Nutzen einen noch unterstützen Linux-Kernel, darunter auch die beiden betrachteten Version von OpenWrt. Es zeigte sich auch eine vergleichbare Verteilung bei der Nutzung von Härtungsmaßnahmen und Exploit Mitigationen. FACT fand nur vereinzelt privates Schlüsselmaterial und Nutzeraccounts. Hier handelte es sich erneut nicht um die beiden OpenWrt Versionen.

5.2 Limitationen

Eine mögliche Limitation bei der Durchführung der Technischen Richtlinie ist die Testumgebung. Besonders die double NAT Konfiguration ist nicht optimal zur Durchführung der Technischen Richtlinie. Der beschriebene Aufbau kann dazu führen, dass einige Ergebnisse nicht zuverlässig angegeben werden können, vor Allem wenn der Zugriff auf die Konfiguration des ersten Routers oder des Modems nicht gegeben ist. So könnten einige Pakete nicht zum eigentlichen OpenWrt Router zugstellt werden, wenn diese bereits von der vorgelagerten Firewall abgefangen wurden. Ebenfalls hätte ein zusätzlicher Testrechner und ggf. weitere Router den Testvorgang weiter beschleunigt, indem nmap-Scans über Nacht oder parallel ausgeführt hätten werden können. Ebenso hätte dies die Durchführung einiger zusätzlicher Tests erlaubt, welche nun als „inconclusive“ markiert wurden, da nicht genug

Systeme zur Verfügung standen, um die vorgegebene Testprozedur durchzuführen. Des Weiteren musste in diesem Falle das Conformance Statement der Technischen Richtlinie vom Tester selbst ausgefüllt werden, statt vom Hersteller oder Entwickler. Dies stellt eine sehr einseitige Betrachtung des Gerätes durch den Tester dar. Auch könnte hier eine gewisse Voreingenommenheit unterstellt werden, da der Tester ggf. gewisse Ergebnisse bereits erwartet. Anschließend muss an dieser Stelle ebenso betrachtet werden, dass die Möglichkeit besteht, bereits bei der Anfertigung des Conformance Statements etwas zu übersehen, wodurch Ergebnisse der TR verfälscht werden können.

Eine weitere Limitation zeigt sich im Zusammenspiel von OpenWrt und der Technischen Richtlinie selbst. OpenWrt ist zwar durchaus für Heim-Router und Router aus dem SOHO-Bereich gedacht, jedoch müssen die Nutzer schon für die Installation einige technische Grundkenntnisse vorweisen, sowie überhaupt von der Möglichkeit wissen. So wird OpenWrt durch die TR an einigen Stellen für die Umsetzung von Funktionen bestraft, welche für den durchschnittlichen Nutzer von OpenWrt vielleicht geeignet sind. Darüber hinaus darf die TR nur als ein Mittel von vielen gesehen werden, um die Sicherheit solcher komplexen Systeme zu untersuchen. Es ist vielmehr das Ziel der Technischen Richtlinie ein Grundmaß an Sicherheit auf Heim-Routern zu schaffen, statt in jeder Hinsicht sichere Router. Aus diesem Grunde geht die Richtlinie zu Teilen tiefer in Details hinein als anderswo, wo die Existenz einer Funktion wichtiger ist als die perfekte Implementierung. Auch wirkt die Technische Richtlinie nicht direkt automatisierten Angriffen wie Heartbleed, Sambacry oder BCMUPnP entgegen [58, 58, 59]. Die TR sorgt allerdings für eine verringerte Angriffsoberfläche und viele Maßnahmen, die den Nutzer dabei unterstützen sollen, sein Gerät sicherer zu betreiben. Schon sicherere Login- und WLAN-Passwörter können einige Angriffe verlangsamen und uninteressant machen. Die Technische Richtlinie sollte also lediglich als Teil des Weges zu sichereren Geräten verstanden werden. Weitere Techniken zum Testen von Software sollten dennoch weiter eingesetzt werden, um einen weiteren Blick auf die IT-Sicherheitslage zu bekommen.

Zu den genannten Limitationen kommt zusätzlich eine zeitliche Komponente. Die Durchführung anhand von OpenWrt war im gesetzten zeitlichen Rahmen machbar, jedoch wäre es dennoch interessant gewesen, eine möglichst vollständige Durchführung der TR anzustreben. OpenWrt hätten durch den Paket-Manager sämtliche zusätzlichen Komponenten geboten, um jedes Modul der TR vollständig zu testen. Dies hätte als konzeptioneller Beweis weitere Einblicke in die Möglichkeiten und Limitationen der Technischen Richtlinie und auch OpenWrt gegeben. Ebenso interessant wie die Ergebnisse von OpenWrt bei der TR wäre ein Vergleich mit dem Resultat von anderen handelsüblichen Routern. Der Zertifizierungsprozess hat jedoch erst vor kurzem begonnen und die Daten der bisher durchgeführten Testdurchläufe stehen nicht für die Öffentlichkeit zur Verfügung. So wird es noch eine Weile

dauern, bis Vergleichswerte verfügbar sind.

Neben den Limitationen bezüglich der Technischen Richtlinie, müssen auch einige Einschränkungen bei der Durchführung der statischen Code-Analyse aufgezeigt werden. Die betrachteten Firmware Abbilder hätten schneller und ausführlicher analysiert werden können, wenn mehr Zeit und mehr Rechenkapazitäten zur Verfügung gestanden hätten. Neben den gewählten Metriken liefert FACT noch weitere interessante Plugins. Ebenso kann nur eine eingeschränkte Vergleichbarkeit mit den Ergebnissen des Home Router Security Reports 2020 dargestellt werden. Dies liegt unter Anderem an der geringen Anzahl an untersuchter Firmware. Im Gegensatz zu den Herstellern, welche im FKIE Report betrachtet wurden, wird bei der quelloffenen Firmware in den meisten Fällen eine Codebasis für alle unterstützen Geräte kompiliert, sodass es die Ergebnisse nicht beeinflusst hätte, wenn Firmware für verschiedene Prozessorinstruktionssätze vertreten gewesen wäre. Ebenfalls wurden in dieser Analyse bereits die bekanntesten quelloffenen Alternativen betrachtet, welche gefunden werden konnten. Ein weiterer Punkt, welcher die Vergleichbarkeit mit dem Home Router Security Report 2020 betrifft, war die Analyse der veröffentlichten CVE-Einträge pro verwendeter Linux Kernel. So lassen sich diese Werte zwar nicht direkt vergleichen, jedoch geben die Angaben des FKIE Reports und der in dieser Arbeit aufgeführten Analyse einen Einblick in die Lage der IT-Sicherheit der betrachteten Geräte.

5.3 Implikationen und zukünftige Forschung

Die Ergebnisse zeigen, dass die Technischen Richtlinie durchaus auch für quelloffenen Router-Betriebssysteme geeignet ist. Nur aufgrund der relativ guten Ergebnisse von OpenWrt kann man jedoch nicht sagen, dass es sich hier von einem IT-Sicherheitsstandpunkt aus um ein sicheres System handelt. Lediglich ein Mindestmaß an Sicherheit kann festgestellt werden und für das vollständige Bestehen der TR sind noch einige Änderungen notwendig. Ebenso wurden nur einige Tests mit FACT durchgeführt, sodass auch in dieser Hinsicht nicht von einem vollständig nachgewiesen sicheren System gesprochen werden darf. Es handelt sich hier nur um Indikatoren und Momentaufnahmen. FACT selbst erweist sich jedoch als geeignetes Programm, um mit wenig Aufwand und geringen technischen Fähigkeiten eine statische Code-Analyse an Firmware durchzuführen.

Zukünftig wäre ein Vergleich verschiedener Geräte anhand der Technischen Richtlinie von Interesse. Ebenso wäre die Durchführung an anderen quelloffenen Router-Betriebssystemen sowie eine Gegenüberstellung interessant. Wie bereits erwähnt wäre es zusätzlich eine Möglichkeit die TR vollständig anhand von OpenWrt durchzuführen und darüber hinaus mithilfe des Software Developer Kits (SDK) der OpenWrt-Entwickler eine Version bereitzustellen, welche alle Anforderungen der TR erfüllt. Weiterhin kann eine sinn-

volle Erweiterung der Technischen Richtlinie um mehr Testfälle in Betracht gezogen werden. Auf diese Weise könnte mehr Funktionalität geprüft werden oder bereits geprüfte Funktionen eingehender getestet werden. Wenn die Verbreitung der TR fortgeschritten ist und mehrere Geräte eine Zertifizierung erhalten haben, so wäre eine Marktanalyse interessant. So könnte die Auswirkung der TR auf die Hersteller und auf die Wahrnehmung der Kunden betrachtet werden.

Kapitel 6

Fazit

Literaturverzeichnis

[1] Bundesamt für Sicherheit in der Informationstechnik, "Die Lage der IT-Sicherheit in Deutschland 2020." https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2020.pdf?__blob=publicationFile&v=2, 2020.

[2] Statistisches Bundesamt, "Ausstattung privater Haushalte mit Internetzugang und Breitbandanschluss im Zeitvergleich." https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Einkommen-Konsum-Lebensbedingungen/Ausstattung-Gebrauchsgueter/Tabellen/zeitvergleich-ausstattung-ikt.html, 2020.

[3] S. Triantopoulou, D. Papanikas, and P. Kotzanikolaou, "An Experimental Analysis of Current DDoS attacks Based on a Provider Edge Router Honeynet," in *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pp. 1–5, IEEE, 15.07.2019 - 17.07.2019.

[4] OpenWrt Webseite. https://openwrt.org.

[5] DD-WRT Webseite. https://dd-wrt.com.

[6] AdvancedTomato Webseite. https://advancedtomato.com.

[7] libreCMC Webseite. https://librecmc.org.

[8] Bundesamt für Sicherheit in der Informationstechnik, "BSI TR-03148:Secure Broadband Router: Requirements for secure Broadband Routers." https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03148/TR03148.pdf?__blob=publicationFile&v=3, 2020.

[9] OpenWrt Webseite, "About the OpenWrt/LEDE project." https://openwrt.org/about, 16.10.2020.

[10] OpenWrt Webseite, "Package table." https://openwrt.org/_media/packages_dump_tab_separated.zip, 03.01.2021.

[11] OpenWrt Webseite, "Table of Hardware." https://openwrt.org/toh/start, 18.01.2020.

[12] OpenWrt Webseite, "OpenWrt Version History." https://openwrt.org/about/history, 13.12.2020.

[13] OpenWrt Webseite, "OpenWrt Download Statistik November 2020," 29.11.2020.

[14] Bundesamt für Sicherheit in der Informationstechnik, "BSI TR-03148 Sichere Breitband Router." https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03148/tr03148_node.html.

[15] Ulrich Hottelet, "BSI will Router mit neuer Technischen Richtlinie sicherer machen." https://www.heise.de/security/meldung/BSI-will-Router-mit-neuer-Technischen-Richtlinie-sicherer-machen-4222689.html, 16.11.2018.

[16] Friedhelm Greis, "CCC und OpenWRT kritisieren Router-TR als Farce." https://www.golem.de/news/bsi-richtlinie-ccc-und-openwrt-kritisieren-router-tr-als-farce-1811-137796.html, 19.11.2018.

[17] A. P. Ortega, X. E. Marcos, L. D. Chiang, and C. L. Abad, "Preventing ARP cache poisoning attacks: A proof of concept using OpenWrt," in *2009 Latin American Network Operations and Management Symposium*, pp. 1–9, IEEE, 19.10.2009 - 21.10.2009.

[18] C. E. Palazzi, M. Brunati, and M. Roccetti, "An OpenWRT solution for future wireless homes," in *2010 IEEE International Conference on Multimedia and Expo*, pp. 1701–1706, IEEE, 19.07.2010 - 23.07.2010.

[19] Andrew McDonnell, "Evaluating the security of OpenWRT." https://blog.oldcomputerjunk.net/2014/evaluating-the-security-of-openwrt-part-1/, 2014.

[20] Linus Torvalds, "Linux—a free unix-386 kernel." https://tech-insider.org/linux/research/acrobat/911010.pdf, 1991.

[21] G. K.-H. Jonathan Corbet, "Linux Kernel Development: How Fast It is Going, Who is Doing It, What They Are Doing and Who is Sponsoring the Work." https://www.linuxfoundation.org/wp-content/uploads/linux-kernel-report-2016.pdf, 2016.

[22] H. Chen, Z. Zhang, S. Moon, and Y. Zhou, eds., *Proceedings of the Second Asia-Pacific Workshop on Systems - APSys '11*, (New York, New York, USA), ACM Press, 2011.

[23] M. Jimenez, M. Papadakis, and Y. Le Traon, "An Empirical Analysis of Vulnerabilities in OpenSSL and the Linux Kernel," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, pp. 105–112, IEEE, 06.12.2016 - 09.12.2016.

[24] Lennart Beringer, Adam Petche, Katherine Q. Ye, Andrew W. Appel, "Verified Correctness and Security of OpenSSL HMAC," in *24th USENIX Security Symposium*.

[25] J. Viega, M. Messier, and P. Chandra, *Network Security with OpenSSL: Cryptography for Secure Communications*. Sebastopol: O'Reilly Media Inc, 2009.

[26] V. J. D. Barayuga and W. E. S. Yu, "Packet Level TCP Performance of NAT44, NAT64 and IPv6 Using Iperf in the Context of IPv6 Migration," in *2015 5th International Conference on IT Convergence and Security (ICITCS)*, pp. 1–3, IEEE, 24.08.2015 - 27.08.2015.

[27] Fraunhofer FKIE, "FACT Core." https://github.com/fkie-cad/FACT_core, 2020.

[28] G. Lyon, *nmap network scanning*. Sunnyvale, CA: Insecure.Com LLC, 2008.

[29] Gordon Fyodor Lyon, "Nmap 7.90 Released! First release since August 2019.." https://seclists.org/nmap-announce/2020/1, 2020.

[30] A. Acosta-López, E. Y. Melo-Monroy, and P. A. Linares-Murcia, "Evaluation of the WPA2-PSK wireless network security protocol using the Linset and Aircrack-ng tools," *Revista Facultad de Ingeniería*, vol. 27, no. 47, 2018.

[31] M. Waliullah, A. B. M. Moniruzzaman, and M. S. Rahman, "An Experimental Study Analysis of Security Attacks at IEEE 802.11 Wireless Local Area Network," *International Journal of Future Generation Communication and Networking*, vol. 8, no. 1, pp. 9–18, 2015.

[32] P. Goyal and A. Goyal, "Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark," in *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 77–81, IEEE, 16.09.2017 - 17.09.2017.

[33] C. Sanders, *Practical packet analysis: Using Wireshark to solve real-world network problems*. San Francisco: No Starch Press, 3rd edition ed., 2017.

[34] OpenWrt Webseite, "Factory install: First-time installation on a device." https://openwrt.org/docs/guide-quick-start/factory_installation, 2019.

[35] OpenWrt Webseite, "Router vs switch vs gateway and NAT." https://openwrt.org/docs/guide-user/network/switch_router_gateway_and_nat, 2020.

[36] Stefan Viehböck, "Brute forcing Wi-Fi Protected Setup." https://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf, 2011.

[37] Paul Russell, "Netfilter - Project history." https://www.netfilter.org/about.html#history.

[38] Simon Kelley, "dnsmasq Archiv." http://www.thekelleys.org.uk/dnsmasq/archive/?C=M;O=A, 2001.

[39] Thomas d'Otreppe de Bouvette. https://github.com/aircrack-ng/aircrack-ng, 25.01.2020.

[40] D. Wetter, "testssl.sh," 2020.

[41] OpenWrt Team, "Luci dispatcher.lua." https://github.com/openwrt/luci/blob/master/modules/luci-base/luasrc/dispatcher.lua, 2020.

[42] Patrick Lacharme, Andrea Röck, Vincent Strubel, Marion Videau, "The Linux Pseudo-random Number Generator Revisited." https://eprint.iacr.org/2012/251.pdf, 2012.

[43] R. M. Gérald Doussot, "State of DNS Rebinding: Attack & Prevention Techniques and the Singularity of Origin." https://docs.google.com/presentation/d/1O7MxvbIfRcPSlbyZbFxD-fAR34XlquQSlRAHPb2kR4E/edit#slide=id.p, 2019.

[44] Microsoft, "MS08-020 : How predictable is the DNS transaction ID?." https://msrc-blog.microsoft.com/2008/04/09/ms08-020-how-predictable-is-the-dns-transaction-id/, 2008.

[45] Ben Sooter, "URLchecker." https://github.com/bensooter/URLchecker/blob/master/top-1000-websites.txt, 15.03.2016.

[46] A. Spillner and T. Linz, *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester : Foundation Level nach ISTQB-Standard*. 6., überarbeitete und aktualisierte auflage ed., 2019.

[47] Peter Weidenbach, Johannes vom Dorp, "Home Router Security Report 2020." https://www.fkie.fraunhofer.de/content/dam/fkie/de/documents/HomeRouter/HomeRouterSecurity_2020_Bericht.pdf, 2020.

[48] Fraunhofer FKIE, "FACT Core." https://github.com/fkie-cad/FACT_core/blob/master/README.md, 2020.

[49] OpenWrt Webseite, "Release Signing." https://openwrt.org/docs/guide-user/security/release_signatures, 2019.

[50] A. Lazarov, B. Shishkov, D. Mitrakos, and M. Janssen, eds., *Proceedings of the Eighth International Conference on Telecommunications and Remote Sensing - ICTRS '19*, (New York, New York, USA), ACM Press, 2019.

[51] P. Gaur and M. P. Tahiliani, "Operating Systems for IoT Devices: A Critical Survey," in *2015 IEEE Region 10 Symposium*, pp. 33–36, IEEE, 13.05.2015 - 15.05.2015.

[52] xorl, "Linux GLibC Stack Canary Values." https://xorl.wordpress.com/2010/10/14/linux-glibc-stack-canary-values/, 2010.

[53] Siddharth Sharma, "Enhance application security with FORTIFY_SOURCE." https://access.redhat.com/blogs/766093/posts/1976213, 2014.

[54] M. Payer, "Too much PIE is bad for performance." https://doi.org/10.3929/ethz-a-007316742.

[55] OWASP Embedded Application Security Project, "Securing Sensitive Information." https://scriptingxss.gitbook.io/embedded-appsec-best-practices/4_securing_sensitive_information, 2019.

[56] "SSL Private-Key." https://ssl.de/ssl-glossar/private-key.html.

[57] J. Du, X. Li, and H. Huang, "A Study of Man-in-the-Middle Attack Based on SSL Certificate Interaction," in *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 445–448, IEEE, 21.10.2011 - 23.10.2011.

[58] M. Carvalho, J. DeMott, R. Ford, and D. A. Wheeler, "Heartbleed 101," *IEEE Security & Privacy*, vol. 12, no. 4, pp. 63–67, 2014.

[59] R. K. Konoth, R. van Wegberg, V. Moonsamy, and H. Bos, "Malicious cryptocurrency miners: Status and Outlook."

Abbildungsverzeichnis

Anhang A

Auswahl verwendeter Programme

```
#!/bin/sh

# /etc/config/show_wifi_clients.sh
# Shows MAC, IP address and any hostname info for all connected wifi devices

echo  "# All connected wifi devices, with IP address,"
echo  "# hostname (if available), and MAC address."
echo -e "# IP address\tname\tMAC address"
# list all wireless network interfaces
# (for universal driver; see wiki article for alternative commands)
for interface in `iwinfo | grep ESSID | cut -f 1 -s -d" "`
do
  # for each interface, get mac addresses of connected stations/clients
  maclist=`iwinfo $interface assoclist | grep dBm | cut -f 1 -s -d" "`
  # for each mac address in that list...
  for mac in $maclist
  do
    # If a DHCP lease has been given out by dnsmasq,
    # save it.
    ip="UNKN"
    host=""
    ip=`cat /tmp/dhcp.leases | cut -f 2,3,4 -s -d" " | grep -i $mac | cut -f 2 -s
        -d" "`
    host=`cat /tmp/dhcp.leases | cut -f 2,3,4 -s -d" " | grep -i $mac | cut -f 3
        -s -d" "`
    # ... show the mac address:
    echo -e "$ip\t$host\t$mac"
  done
done
```

```
# Configure network
uci -q delete network.guest
uci set network.guest="interface"
uci set network.guest.type="bridge"
uci set network.guest.proto="static"
uci set network.guest.ipaddr="192.168.3.1"
uci set network.guest.netmask="255.255.255.0"
uci commit network
/etc/init.d/network restart


# Configure wireless
WIFI_DEV="$(uci get wireless.@wifi-iface[0].device)"
uci -q delete wireless.guest
uci set wireless.guest="wifi-iface"
uci set wireless.guest.device="${WIFI_DEV}"
uci set wireless.guest.mode="ap"
uci set wireless.guest.network="guest"
uci set wireless.guest.ssid="guest"
uci set wireless.guest.encryption="none"
uci commit wireless
wifi reload


# Configure DHCP
uci -q delete dhcp.guest
uci set dhcp.guest="dhcp"
uci set dhcp.guest.interface="guest"
uci set dhcp.guest.start="100"
uci set dhcp.guest.limit="150"
uci set dhcp.guest.leasetime="1h"
uci commit dhcp
/etc/init.d/dnsmasq restart


# Configure firewall
uci -q delete firewall.guest
uci set firewall.guest="zone"
uci set firewall.guest.name="guest"
uci set firewall.guest.network="guest"
uci set firewall.guest.input="REJECT"
uci set firewall.guest.output="ACCEPT"
uci set firewall.guest.forward="REJECT"
uci -q delete firewall.guest_wan
uci set firewall.guest_wan="forwarding"
uci set firewall.guest_wan.src="guest"
uci set firewall.guest_wan.dest="wan"
uci -q delete firewall.guest_dns
uci set firewall.guest_dns="rule"
uci set firewall.guest_dns.name="Allow-DNS-Guest"
uci set firewall.guest_dns.src="guest"
uci set firewall.guest_dns.dest_port="53"
uci set firewall.guest_dns.proto="tcp udp"
uci set firewall.guest_dns.target="ACCEPT"
uci -q delete firewall.guest_dhcp
uci set firewall.guest_dhcp="rule"
uci set firewall.guest_dhcp.name="Allow-DHCP-Guest"
uci set firewall.guest_dhcp.src="guest"
uci set firewall.guest_dhcp.dest_port="67"
```

```
uci set firewall.guest_dhcp.family="ipv4"
uci set firewall.guest_dhcp.proto="udp"
uci set firewall.guest_dhcp.target="ACCEPT"
uci commit firewall
/etc/init.d/firewall restart


# Configure wireless encryption
WIFI_PSK="GUEST_WIFI_PASSWORD"
uci set wireless.guest.encryption="psk2"
uci set wireless.guest.key="${WIFI_PSK}"
uci commit wireless
wifi reload


# Configure client isolation
uci set wireless.guest.isolate="1"
uci commit wireless
wifi reload


# Configure firewall - icmp
uci rename firewall.@rule[1]="icmp"
uci rename firewall.@rule[5]="icmp6"
uci set firewall.icmp.src="*"
uci set firewall.icmp6.src="*"
uci commit firewall
/etc/init.d/firewall restart


# Configure firewall - restrictions
uci -q delete firewall.guest_wan
uci -q delete firewall.guest_fwd
uci set firewall.guest_fwd="rule"
uci set firewall.guest_fwd.name="Allow-HTTP/HTTPS-Guest-Forward"
uci set firewall.guest_fwd.src="guest"
uci set firewall.guest_fwd.dest="wan"
uci add_list firewall.guest_fwd.dest_port="80"
uci add_list firewall.guest_fwd.dest_port="443"
uci set firewall.guest_fwd.proto="tcp"
uci set firewall.guest_fwd.target="ACCEPT"
uci commit firewall
/etc/init.d/firewall restart

# Configure firewall
uci rename firewall.@zone[0]="lan"
uci set firewall.lan.masq="1"
uci commit firewall
/etc/init.d/firewall restart
```

```
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 23 14:17:11 2020

@author: Henry Weckermann
"""

import re
import requests
from tqdm import tqdm
from collections import Counter
import matplotlib.pyplot as plt

USERNAME = "root"
PASSWORD = "1234"

no_of_requests = 100

# Change this if you want to test the webserver
URL = f"http://192.168.1.1/cgi-bin/luci/admin/status?luci_username={USERNAME}&
    luci_password={PASSWORD}"

# ---------------------------------------------------------------------------- #

sessionID_list, token_list = [], []

for _ in tqdm(range(no_of_requests)):
   response = requests.post(URL)
   sessionID, token = re.findall("[a-zA-Z0-9]{32}", response.text)

   sessionID_list.append(sessionID)
   token_list.append(token)

cnt = Counter(token_list)

# ---------------------------------------------------------------------------- #

# Define the matplotlib figure
fig = plt.figure(figsize=(12, 8))
ax1 = fig.add_subplot(111)

# Define plots
ax1.bar(cnt.keys(), cnt.values(), align='center', alpha=0.5)

# Define looks
plt.xlabel('Token')
plt.ylabel('Occurrences')
plt.xticks(rotation=90, fontsize=8)
plt.yticks([1,2])
plt.grid(True)

plt.show()

# ---------------------------------------------------------------------------- #

print(f"""
     \n\tNumber of requests: {len(token_list)}\n
     Number of unique tokens: {len(set(token_list))}
     Number of unique sessionIDs: {len(set(sessionID_list))}""")
```

```
from scapy.all import *
from scipy import stats
import matplotlib.pyplot as plt
import random
import numpy as np
from collections import Counter

# Open pcap file
scapy_cap = rdpcap('dns_dump.pcap')

# select only DNS packages from 192.168.1.1 (openwrt router)
important_packages = []
for pkt in scapy_cap:
    if IP in pkt and pkt[IP].src == '192.168.1.1':
        try:
            if pkt.haslayer(DNS):
                important_packages.append(pkt)
        except:
            pass

# extract port and transaction ID
DNS_PORT = []
TRANS_ID = []
for pkt in important_packages:
    DNS_PORT.append(pkt.dport)
    TRANS_ID.append(pkt.getlayer(DNS).id)

dns_count = Counter(DNS_PORT)
trans_count = Counter(TRANS_ID)

# Printing results

print('Test results for DNS port randomization:\n')
print(f'\tNumber of samples: {len(DNS_PORT)}')
print(f'\tNumber of unique ports: {len(set(DNS_PORT))}')
print(f'\tRange: {min(DNS_PORT)} - {max(DNS_PORT)}')
print(f'\tStandard Deviation: {np.std(DNS_PORT)}')
print(f'\t5 most common ports: {dns_count.most_common(5)}\n')

print('#'*100, '\n')
print('Test results for transaction ID randomization:\n')
print(f'\tNumber of samples: {len(TRANS_ID)}')
print(f'\tNumber of unique ports: {len(set(TRANS_ID))}')
print(f'\tRange: {min(TRANS_ID)} - {max(TRANS_ID)}')
print(f'\tStandard Deviation: {np.std(TRANS_ID)}')
print(f'\t5 most common ports: {trans_count.most_common(5)}\n')


print(stats.kstest(DNS_PORT, random.sample(range(0, 65535), len(DNS_PORT))))
print(stats.kstest(TRANS_ID, random.sample(range(0, 65535), len(TRANS_ID))))

fig = plt.figure(figsize=(12, 8))
sub1 = fig.add_subplot(221)
sub1.hist(DNS_PORT)
sub1.set_xlabel('DNS Ports')
sub1.set_ylabel('Occurances per bin')

sub2 = fig.add_subplot(222)
sub2.hist(TRANS_ID)
```

```
sub2.set_xlabel('Transaction IDs')
sub2.set_ylabel('Occurances per bin')

sub3 = fig.add_subplot(223)
sub3.scatter(range(0,len(DNS_PORT)), DNS_PORT)
sub3.set_xlabel('DNS Ports')
sub3.set_ylabel('Queries')

sub4 = fig.add_subplot(224)
sub4.scatter(range(0,len(TRANS_ID)), TRANS_ID)
sub4.set_xlabel('Transaction IDs')
sub4.set_ylabel('Queries')

plt.show()
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 15 14:34:27 2020

@author: Henry Weckermann (henry.weckermann@smail.inf.h-brs.de)

A script to visualize the behavior of OpenWrt when faced with multiple
wrong login attempts to the ssh server or the webserver.
Uses linear regresion to fit a line in the response data of the server.

Used in testing TR.D.11 of BSI TR-03148
"""

import sys
import time
import pprint
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
from scipy import stats

# Global parameters. Change these to your needs
USERNAME = "root"
PASSWORD = "imawrongpassword"
CORRECT_PW = "1234"

no_of_requests = 50

# Change this if you want to test ssh
IP = '192.168.1.1'
PORT = 22

# Change this if you want to test the webserver
URL = f"http://192.168.1.1/cgi-bin/luci/admin/status?luci_username={USERNAME}&
    luci_password={PASSWORD}"


def plot_results(xlabel_text, ylabel_text):
   # Define x values as the number of requests (so 1, 2, ..., x)
   x = range(1, no_of_requests+1)

   # Calculate the linear regression
   slope, intercept, r_value, p_value, std_err = stats.linregress(x, times)

   # Define the matplotlib figure
   fig = plt.figure()
   ax1 = fig.add_subplot(111)

   # Define plots
   ax1.scatter(x, times)
   ax1.plot(x, intercept + slope*x, 'black', label='Regression line')

   # Define looks
   plt.xlabel(xlabel_text)
   plt.ylabel(ylabel_text)
   plt.legend()
   plt.grid(True)

   plt.show()
```

```
    # ------------------------------------------------------------------------ #

    # Define results
    results = {'Mean' : np.round(np.mean(times),3),
            'Median' : np.round(np.median(times),3),
            'Regression coefficient' : str(np.round(r_value,5)) + f" (p = {np.
                round(p_value,3)})",
            'Standard error' : np.round(std_err, 3)
            }

    # Plot results in a nice way. You don't have to understand the print statement
        . It justs looks better
    myStr = pprint.pformat(results)
    print(myStr.translate(myStr.maketrans("'{},", "  ")))

def check_success_web():

    import re

    # Checking if the webserver still responds to a correct login attempt
    id_string = "Invalid username and/or password! Please try again."

    URL_new = URL.replace(PASSWORD, CORRECT_PW)
    r = requests.post(URL_new)
    sessionID, token = re.findall("[a-zA-Z0-9]{32}", r.text)

    if id_string not in r.text:
        print(f"""Successfully logged in with user {USERNAME}. Got Session-ID {
            sessionID} and token {token}.
              The IP does not seem to be blacklisted by the DUT.""")

    # ------------------------------------------------------------------------ #

    browser_open = input("Do you want to open the OpenWrt web-interface in your
        browser (y / n): ")
    if browser_open.lower() == "y":
        import webbrowser
        webbrowser.open_new(URL)

def check_success_ssh():
    # Final login to see if login is still possible
    print("\n\nThis should print the OpenWrt Banner if the ssh server did not
        block the IP\n")
    ssh.open_connection(IP, port=PORT)
    openwrt_banner_new = ssh.login(USERNAME, CORRECT_PW)
    print(openwrt_banner_new)


if sys.argv[1] == "web":

    import requests

    # ------------------------------------------------------------------------ #

    # Check if webserver is alive and connection is possible
    tmp = requests.post(URL.replace(PASSWORD, CORRECT_PW))
    if tmp.status_code == 200 and "Invalid username and/or password!" not in tmp.
        text:
```

```
        print(f"Status code {tmp.status_code} -> Webserver found and is accessable
            via password {CORRECT_PW}")
    else:
        print(f"Error. Please check the password and if the webserver is accessable
             at the specified URL -> {URL[:URL.find('?')]}")
        sys.exit(0)

    # ------------------------------------------------------------------------ #

    # Making the requests to the OpenWrt login page
    times = []
    for _ in tqdm(range(no_of_requests)):
        start = time.time()
        r = requests.post(URL)
        stop = time.time()
        if r.status_code == 403:
            times.append(stop - start)
        elif r.status_code == 200:
            print("This should not happen if you set a wrong password")
        else:
            print(f"Status Code: {r.status_code}. Please seek help.")

    assert len(times) == no_of_requests

    plot_results('Number of POST requests to the OpenWrt webserver', 'Time to
        complete the request (sec)')
    check_success_web()



elif sys.argv[1] == "ssh":

    from SSHLibrary import SSHLibrary

    ssh = SSHLibrary()

    # ------------------------------------------------------------------------ #

    # Sanity checks
    connection_status = ssh.open_connection(IP, port=PORT)

    if connection_status != 1:
        print("The SSH Server is not reachable. Please try something different")
        sys.exit(0)

    try:
        openwrt_banner = ssh.login(USERNAME, CORRECT_PW)
        if "built-in shell (ash)" not in openwrt_banner:
            print("Server is reachable but no login is possible. Is the password
                correct?")
            sys.exit(0)
    except:
        print("Server is reachable but no login is possible. Is the password
            correct?")
        sys.exit(0)


    # ------------------------------------------------------------------------ #
```

```
    # Testing
    times = []
    for _ in tqdm(range(no_of_requests)):
        failed = False
        start = time.time()
        try:
            ssh.open_connection(IP, port=PORT)
            ssh.login(USERNAME, PASSWORD)
        except:
            ssh.close_connection()
            failed = True
        stop = time.time()

        if failed:
            times.append(stop-start)
        else:
            print("This should not happen.")
            sys.exit(0)

    # Checking if all login attempts have been made.
    assert len(times) == no_of_requests

    plot_results('Number of login attempts at the ssh server', 'Time to complete
        the request (sec)')
    check_success_ssh()


else:
    print("Please choose either 'web' or 'ssh'")
```

Listing A.1: Ein Skript, welches die Resilenz des OpenWrt Web-Servers und SSH-Servers gegen Bruteforce-Angriffe testet

Anhang B

Verwendete Firmware für FACT Analyse

PROJEKT	ROUTER	VERSION	VERÖFFENTLICHUNGS-DATUM	LINK
OPENWRT	ARCHER C7 v5	19.07.4	09.2020	https://downloads.openwrt.org/releases/19.07.4/targets/ath79/generic/openwrt-19.07.4-ath79-generic-tplink_archer-a7-v5-squashfs-factory.bin
OPENWRT	ARCHER C7 v5	19.07.5	12.2020	https://downloads.openwrt.org/releases/19.07.5/targets/ath79/generic/openwrt-19.07.5-ath79-generic-tplink_archer-c7-v5-squashfs-factory.bin
LIBRECMC	ARCHER C7 v2	v1.5.3:2020-10-02	02.10.2020	https://librecmc.org/librecmc/downloads/snapshots/v1.5.3/targets/ath79/generic/librecmc-ath79-generic-tplink_archer-c7-v2-squashfs-factory.bin
GLUON	ARCHER C7 v5	V2-v2020.2.1	09.10.2020	https://images.ffkbu.de/BonnV2/stable/Wireguard/factory/gluon-ffkbu-V2-v2020.2.1-Wireguard-tp-link-archer-c7-v5.bin
GARGOYLE	ARCHER C7 v5	1.12.0 (stable)	03.12.2019	https://www.gargoyle-router.com/downloads/images/ar71xx/gargoyle_1.12.0-ar71xx-generic-archer-c7-v5-squashfs-factory.bin
TOMATO	NETGEAR WNDR3700v3	3.4-138	05.12.2016	https://advancedtomato.com/downloads/router/wndr3700v3
DDWRT	ARCHER C7 v5	12-18-2020-r45036	18.12.2020	ftp://ftp.dd-wrt.com/betas/2020/12-18-2020-r45036/tplink_archer-c7-v5/

Anhang C

OpenWrt Veröffentlichungshistorie

Version (Code Name)	Veröffentlichungsdatum	Linux Kernel	libc
White Russian 0.9	2007 January	2.4.30	uClibc
Kamikaze 7.06	2007 June	2.6.19	
Kamikaze 7.07	2007 July	2.6.21	
Kamikaze 7.09	2007 September		
Kamikaze 8.09	2008 September	2.6.26	
Kamikaze 8.09.1	2009 June	2.6.26.8	
Kamikaze 8.09.2	2010 January		
Backfire 10.03	2010 April	2.6.32	
Backfire 10.03.1	2011 December		
Attitude Adjustment 12.09	2013 April	3.3	
Barrier Breaker 14.07	2014 October	3.10.49	
Chaos Calmer 15.05	2015 September	3.18.20	
Chaos Calmer 15.05.1	2016 March	3.18.23	
LEDE 17.01.0	2017 February	4.4.50	musl
LEDE 17.01.1	2017 April	4.4.61	
LEDE 17.01.2	2017 June	4.4.71	
LEDE 17.01.3	2017 August	4.4.89	
LEDE 17.01.4	2017 October	4.4.92	
LEDE 17.01.5	2018 July	4.4.140	
LEDE 17.01.6	2018 September	4.4.153	
OpenWrt 18.06.0	2018 July	4.9.111, 4.14.52	
OpenWrt 18.06.1	2018 August	4.9.120, 4.14.63	
OpenWrt 18.06.2	2019 February	4.9.152, 4.14.95	
OpenWrt 18.06.3	skipped	skipped	
OpenWrt 18.06.4	2019 July	4.9.184, 4.14.131	
OpenWrt 18.06.5	2019 November	4.9.198, 4.14.151	
OpenWrt 18.06.6	2020 January	4.9.208, 4.14.162	
OpenWrt 18.06.7	2020 January	4.9.211, 4.14.167	
OpenWrt 18.06.8	2020 March	4.9.214, 4.14.171	
OpenWrt 19.07.0	2020 January	4.14.162	
OpenWrt 19.07.1	2020 January	4.14.167	
OpenWrt 19.07.2	2020 March	4.14.171	
OpenWrt 19.07.3	2020 May	4.14.180	
OpenWrt 19.07.4	2020 September	4.14.195	
OpenWrt 19.07.5	2020 December	4.14.209	

Tabelle C.1: Veröffentlichungshistorie von OpenWrt, sowie genutzer Linux Kernel und libc Version