Bachelorarbeit im Studiengang Wirtschaftsinformatik

Analyse der Sicherheit von IoT-Geräten und Methoden zur Durchführung von Penetrationstests für IoT-Geräte

Autor:	Lars Ursprung
Matrikel-Nr.:	5172962
E-Mail:	lars.ursprung@mnd.thm.de
1. Prüfer:	Prof. Dr.-Ing. Rahamatullah Khondoker
2. Prüfer:	Prof. Dr.-Ing. Nicolas Stein
Abgabedatum:	31. Januar 2024

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit im Studiengang Wirtschaftsinformatik selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Weiterhin bin ich damit einverstanden, dass meine Arbeit der THM-internen Plagiatsprüfung unterworfen wird.

Cölbe, 31. Januar 2024 Lars Ursprung

Inhaltsverzeichnis

Glossar

Botnetz Ein Bot ist ein Computerprogramm, welches unbemerkt auf mehreren Computern installiert ist und aus der Ferne gesteuert wird. Sobald mehrere tausende solcher Bots per Fernsteuerung zusammengeschlossen werden, spricht man von einem Botnetz. Nicht nur klassische PCs können zu Bots werden, auch andere Geräte, welche einen Internetzugang haben oder Teil eines Netzwerkes sind, sind gefährdet. Beispiele sind hier mobile Geräte wie Smartphones oder Tablets, Wearables oder Teile des IoT wie Webcams oder Router.[1]. 10

Debugging Das schrittweise Ausführen von Quellcode, um Variablen zur Laufzeit auszulesen und Informationen über fehlerhaftes Verhalten und das Verhalten einer Anwendung im Allgemeinen zu gewinnen.[1]. 34, 56, 74, 75

Firmware Ein Computerprogramm, welches für gewöhnlich auf Read-Only Memory (ROM) oder Programmable Read-Only Memory (PROM) Speicherbausteinen abgespeichert ist und während der Laufzeit nicht veränderbar ist.[2]. 3, 8, 10, 17, 19–21, 29, 33, 34, 47, 52–54, 56, 61, 62, 64, 78

Framework Eine Struktur die einen Rahmen zur Entwicklung von Programmen oder Programmtypen vorgibt. Ein Framework kann dabei nicht nur die generelle Struktur, sondern auch Schnittstellen und Programmierwerkzeuge zur Verfügung stellen. Frameworks sind umfangreicher als Protokolle und präziser als Strukturen.[3]. 54, 55, 70, 77

Git Ein quelloffenes System zur Versionskontrolle, welches Softwareprojekte jeglicher Größe effizient verwalten kann.[4]. 65, 68, 72, 78, 79

IEEE 802.15.4 Diese Norm definiert die Spezifikationen für die physikalische Schicht (PHY) und die Teilschicht der Medienzugriffskontrolle (MAC) für drahtlose Verbindungen mit niedriger Datenrate mit festen, tragbaren und beweglichen Geräten, welche keine oder nur eine sehr begrenzte Batterieleistung benötigen. Darüber hinaus sieht die Norm Modi vor, die eine präzise Ortung ermöglichen.[5]. 14

[1] Vgl. 2, S. 135 f.
[2] Vgl. 3.
[3] Vgl. 4.
[4] Vgl. 5.
[5] Vgl. 6.

Repository Ein effizienter Datenspeicher zur Speicherung von Objekten, welcher die Verzeichnisstruktur und Dateien eines Softwareprojektes und deren Versionshistorie verwaltet.[6]. 68, 72, 78, 79

[6]Vgl. 7, S. 4 f.

Akronyme

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks. 14

AMQP Advanced Message Queuing Protocol. 13

API Application Programming Interface. 22, 26, 27, 45, 76, 77

BLE Bluetooth Low Energy. 2, 15, 46

BSI Bundesamt für Sicherheit in der Informationstechnik. 53, 54, 69

CI/CD Continuous Integration/Continuous Delivery. 27

CoAP Constrained Application Protocol. 12

CVSS Common Vulnerability Scoring System. 9, 67

CWE Common Weakness Enumeration. 45

DDoS Distributed Denial of Service. 10

DNS Domain Name System. 48

DoS Denial of Service. 28, 61, 62

DREAD Damage Reproducibility Exploitability Affected Discoverability. 58, 61

ERP Enterprise resource planning. 1

GHz Gigahertz. 14, 15

GND Ground. 32, 33

GSM Global System for Mobile Communications. 18

HTTP Hypertext Transfer Protocol. 12, 40, 48

HTTPS Hypertext Transfer Protocol Secure. 12, 40, 48

IoT Internet of Things. 1–4, 6–8, 10–12, 14–22, 24, 25, 28–30, 33–35, 43, 44, 46, 48–50, 52–59, 62–65, 68–80, VIII

IP Internet Protocol. 14, 30, 31, 47

IPv4 Internet Protocol version 4. 14

IPv6 Internet Protocol version 6. 14

ISM Industrial Scientific Medical. 15

JTAG Joint Test Action Group. 34

KI künstliche Intelligenz. 72

LoRA Long Range. 2

LowPan Low-Power Wireless Personal Area Networks. 2

MAC Media-Access-Code. 16, 29, 75

MASVS Mobile Application Security Verification Standard. 45

MITM Man-in-the-middle. 34

MobSF Mobile Security Framework. 45

MQTT Message Queuing Telemetry Transport. 12

NFC Near Field Communication. 16

OSI Open Systems Interconnection. 11

OSINT Open-Source Intelligence. 41, 56, 62, 63, 66, 78

OWASP Open Worldwide Application Security Project. 3, 4, 22, 25, 26, 28, 52, 74, 75, 77

PDF Portable Document Format. 70

PKI Public-Key-Infrastruktur. 21

PROM Programmable Read-Only Memory. III

RFID Radio-Frequency Identification. 10, 15, 16

ROM Read-Only Memory. III

RPL Routing Protocol for Low-Power and Lossy Networks. 14

RTMP Real-Time Messaging Protocol. 48

RTOS Real-time operating system. 17

RTSP Real-Time Streaming Protocol. 48

RX Receive. 32, 33

S3 Simple Storage Service. 49–51, VIII

SIM Subscriber Identity Module. 16

SoC System-on-a-Chip. 17, 74

SPI Serial Peripheral Interface. 34

SQL Structured query language. 76

SSH Secure Shell. 43

SSL Secure Sockets Layer. 12

SSRF Server-Side Request Forgery. 24

STRIDE Spoofing Identity, Tampering with Data, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege. 57, 58, 61

TCP Transmission Control Protocol. 12–14

TCP/IP Transmission Control Protocol/Internet Protocol. 11, 74

TLS Transport Layer Security. 12, 13

TX Transmit. 32, 33

UART Universal Asynchronous Receiver Transmitter. 18, 20, 30–34, 56, 59, 61, 74, 75

UDP User Datagram Protocol. 12–14

URL Uniform Resource Locator. 43, 45

USB Universal Serial Bus. 18, 31, 33, 75

V_{CC} Voltage at the common collector. 32

WLAN Wireless Local Area Network. 15, 18, 44, 46, 47, 49, 59, 61

XML Extensible Markup Language. 12

XMPP Extensible Messaging and Presence Protocol. 12, 13

Abbildungsverzeichnis

Tabellenverzeichnis

Zusammenfassung

Aufgrund der stetig zunehmenden Digitalisierung werden immer mehr physische Objekte miteinander und dem Internet vernetzt und bilden ein sogenanntes İnternet der Dinge"(IoT). Dies führt sowohl im privaten, als auch im industriellen Bereich zu neuen Möglichkeiten der Automatisierung und Datenerfassung. Diese Bachelorarbeit setzt sich mittels einer umfassenden Literaturrecherche mit dem aktuellen Stand der IoT-Sicherheit und der Durchführung von Penetrationstests an IoT-Geräten auseinander. Um das zum Verständnis notwendige Wissen zu schaffen, werden zunächst die Grundlagen für die Themengebiete IoT und Penetrationstests im einzelnen vermittelt. Angesichts der limitierten Ressourcen und damit einhergehend oft reduzierten Sicherheitsmechanismen von IoT-Geräten, zielt die Arbeit darauf ab, häufige Schwachstellen im Bereich des IoT aufzuzeigen und Methoden und Techniken zur Durchführung von Penetrationstests zu erläutern. In diesem Zuge werden einige der Techniken an realen Objekten wie einer Netzwerkkamera und einer Smart Home Bridge demonstriert. Hierbei wird aus den gewonnenen Erkenntnissen eine Methodik zur Durchführung von Penetrationstests an IoT-Geräten entwickelt, welche sich praktisch von Unternehmen und Penetrationstestern einsetzen und flexibel auf eine Vielzahl unterschiedlicher Geräte anwenden lässt. Dabei wird zudem darauf geachtet, dass sich die Methode effektiv von mehreren Beteiligten gleichzeitig in einem Team anwenden lässt. Anschließend wird die Methodik mit zuvor erläuterten bestehenden Methodiken verglichen. Dabei werden sowohl theoretische als auch praktische Aspekte berücksichtigt, als auch die Anpassbarkeit der Methode an zukünftige Entwicklungen.

1 Einleitung

Die zunehmende Digitalisierung beschränkt sich nicht mehr nur auf Dinge wie Textdokumente, Bilddaten oder Prozesse, sondern breitet sich zunehmend auch auf physische Objekte des alltäglichen Lebens aus. So ist es nicht ungewöhnlich, wenn selbst einfachste Objekte wie Glühbirnen oder Armbanduhren mittlerweile eine permanente Netzwerkverbindung haben, um Daten senden und empfangen zu können. Dies kann dazu genutzt werden, um diese Objekte aus der Ferne mittels mobiler Anwendungen zu kontrollieren, oder intelligent miteinander interagieren zu lassen. So kann etwa eine Glühbirne in Verbindung mit einem Lichtsensor bei Dunkelheit eingeschaltet werden, während gleichzeitig automatisch die Rollläden herunterfahren. Auch die Heizung lässt sich mittels vernetzter Thermostate in Verbindung mit diversen Sensoren automatisch regulieren. So können Sensoren das Öffnen eines Fensters erkennen und die Heizung anschließend automatisiert herunterregeln, bis das Fenster wieder geschlossen wurde. Auch Kraftfahrzeuge sind nicht selten mit einer permanenten Internetverbindung ausgestattet, um dem Besitzer verschiedenste Komfortfunktionen anzubieten. Bei vielen Fahrzeugen kann der Besitzer beispielsweise über eine mobile Anwendung den Füllstand des Tankes, oder den Ladestand des Akkus von überall aus ablesen. Auch weitere Fahrzeugdaten wie der Standort des Fahrzeuges, oder der Kilometerstand können jederzeit abgefragt werden. Zudem können moderne Fahrzeuge Verkehrsdaten austauschen, um die Navigationsdienste zu verbessern. Im Allgemeinen spricht man bei dieser Integration von vernetzten Sensoren und Aktoren in Objekte des täglichen Lebens vom *Internet der Dinge*, im englischen auch *Internet of Things*, kurz IoT genannt.[1]

Während die Heimautomatisierung durch IoT-Geräte als Smart-Home bezeichnet wird, findet die Vernetzung und Automatisierung von Industriebetrieben unter der Bezeichnung *Industrie 4.0* statt. Dabei werden Maschinen und Sensoren miteinander vernetzt, um möglichst viele Daten zu sammeln und den Menschen bei Entscheidungen zu unterstützen, sowie um automatisiert eigene Entscheidungen zu treffen.[2] Dies soll langfristig zu einer sich selbst organisierenden Produktion führen, welche den gesamten Lebenszyklus von Produkten umfasst.[3] Erreicht werden soll dies durch die Vernetzung von Menschen, Sensoren, Maschinen, Robotern, Anlagen, Logistik, Produkten sowie Enterprise resource planning (ERP)-Systemen untereinander, um die gesamte Wertschöpfungskette durch einen möglichst hohen Automatisierungsgrad zu optimieren.[4]

Für die Datenübertragung von und zu IoT-Geräten existiert dabei nicht nur ein einziger einheitlicher Standard. Die Techniken zur Datenübertragung im Bereich des IoT sind so

[1] Vgl. 8, S. 39.
[2] Vgl. 9, S. 1123.
[3] Vgl. 10, S. 15.
[4] Vgl. 10, S. 20 f.

unterschiedlich wie die IoT-Geräte selbst und reichen von weit verbreiteten Technologien wie WLAN oder Bluetooth bis hin zu speziellen Netzwerkprotokollen wie ZigBee, Long Range (LoRA) und Low-Power Wireless Personal Area Networks (LowPan).[5] Diese Protokolle und Technologien sind oft auf das jeweilige Gerät und dessen Einsatzzweck abgestimmt und aufgrund der geringen Größe und Portabilität vieler IoT-Geräte auf das Sparen von Energie ausgelegt. Protokolle wie ZigBee oder Bluetooth Low Energy (BLE) finden sich zum Beispiel häufig im Bereich der Heimautomatisierung wieder und sorgen für die lokale Kommunikation der Geräte untereinander. Gesteuert wird die Kommunikation dabei über einen zentralen Konzentrator, oft in Form eines sogenannten Smart-Home-Hubs, welcher mit dem Internet verbunden ist und sich auch über dieses ansteuern lässt.[6]

1.1 Einführung in das Thema

Den Vorteilen wie der Automatisierung von unterschiedlichsten Prozessen und der schnellen Erfassung von Daten stehen jedoch auch Probleme gegenüber, welche die zunehmende Vernetzung durch IoT-Geräte mit sich bringt. Denn Geräte wie Produktionsanlagen in Industrieunternehmen, die in der Vergangenheit ohne Netzwerkverbindung auskamen und somit nur physisch angreifbar waren, sind mittlerweile häufig mit dem Unternehmensnetzwerk oder dem Internet verbunden und schaffen Angreifern damit neue Möglichkeiten in das Netzwerk eines Unternehmens einzudringen.

Zudem versuchen Hersteller von IoT-Geräten die Kosten für Produktion, Entwicklung, Wartung und Betrieb möglichst gering zu halten. Dies führt dazu, dass die Geräte in ihren Ressourcen hinsichtlich Speicher, Energie und Rechenleistung oft limitiert sind.[7] Aufgrund dieser limitierten Ressourcen sind Sicherheitsmechanismen wie eine starke Kryptographie und Möglichkeiten zur Aktualisierung der Software im Vergleich zu klassischen vernetzten Systemen wie Servern oft reduziert oder fehlen gänzlich.[8] Daher ist es notwendig, die Sicherheit von IoT-Geräten gründlich zu evaluieren, bevor sie im Unternehmen oder privat eingesetzt werden.

Eine effektive und bewährte Methode für die Evaluierung der Sicherheit von IT-Systemen ist ein Penetrationstest. Dabei werden Systeme gezielt angegriffen, um eventuell vorhandene Schwachstellen aufzudecken.[9] Die Vorgehensweisen unterscheiden sich dabei nur geringfügig von denen realer Angreifer. Penetrationstester müssen im Gegensatz zu diesen beispielsweise nicht zwingend unentdeckt agieren und richten für gewöhnlich auch keinen Schaden am zu testenden System an.[10] Gängig sind Penetrationstests beispielsweise im Bereich der Webanwendung, in dem Penetrationstester mit spezialisiertem Wissen über Webtechnologien Anwendungen auf deren Sicherheit überprüfen. Die Technologien im Bereich der Webanwendungen sind allerdings recht homogen und standardisiert. Anders sieht dies

[5] Vgl. 9, S. 1122.
[6] Vgl. 9.
[7] Vgl. 11, S. 12.
[8] Vgl. 8, S. 12 f.
[9] Vgl. 9, S. 23.
[10] Vgl. 9, S. 23.

bei IoT-Geräten aus, da Kommunikationsprotokolle, Betriebssysteme und weitere zu den Ökosystemen gehörende Softwares eher fragmentiert sind. Auch die einzelnen Bestandteile von IoT-Ökosystemen sind umfangreicher als einfache Webanwendungen oder klassische Computernetze. Ein IoT-System kann sich nämlich aus Bestandteilen wie Hardware, welche elektronische Bausteine als auch Gehäuse und Schnittstellen umfassen kann, als auch aus Softwarekomponenten wie einer Firmware, verschiedenen Netzwerkprotokollen, mobilen Anwendungen zur Steuerung aus der Ferne, Cloud-Anbindungen zum zentralen Verwalten von Daten oder auch Webanwendungen zur Administration der Geräte selbst zusammensetzen.[11] Diese umfangreiche Stuktur von IoT-Ökosystemen macht die Durchführung von Penetrationstests umfangreich und erfordert viel Fachwissen aus unterschiedlichen Bereichen.

1.2 Problemstellung und Zielsetzung

Die Durchführung von Penetrationstests im IoT-Umfeld ist also komplex, da neben der Sicherheit der Anwendersoftware auch die Sicherheit von Hardware, sowie die Sicherheit von eventuell zugehörigen Cloud-Diensten, Firmware und mehr gründlich geprüft werden muss. Für eine effektive Durchführung von Penetrationstests bedarf es daher stets aktueller Informationen und fundiertes Fachwissen über gängige Schwachstellen der zu testenden Systeme sowie strukturierte Leitfäden und Methoden zur Identifizierung dieser Schwachstellen. Als angesehene Informationsquelle zu aktuellen Themen und Forschungsergebnissen im Bereich des Penetrationstestens hat sich beispielsweise die Konferenz *blackhat* erwiesen.[12] Die auf dieser Konferenz vorgestellten Informationen lassen sich für Interessierte über deren Archive abrufen. Die letzten Vorträge zum Thema Internet of Things sind allerdings aus dem Jahr 2020 und damit zum jetzigen Zeitpunkt etwa drei Jahre alt.[13] Eine weitere weit verbreitete Informationsquelle für Penetrationstester ist das *Open Worldwide Application Security Project (OWASP)*, eine gemeinnützige Organisation, welche dabei helfen möchte, die Sicherheit von Software aus unterschiedlichen Bereichen zu verbessern.[14] Zu diesem Zweck aggregiert das Open Worldwide Application Security Project (OWASP) Informationen und Leitfäden mit Hilfe von freiwilligen Unterstützern und stellt diese der Allgemeinheit zur Verfügung. So gibt es unter dem Namen *OWASP IoT Security Testing Guide* beispielsweise auch einen Leitfaden zum Durchführen von Penetrationstests von IoT-Geräten.[15] Dieser befindet sich allerdings noch in einem frühen Stadium und ist von geringem Umfang.[16] So enthält dieser Leitfaden aktuell nur kurze Beschreibungen und keinerlei Hilfestellung zur praktischen Anwendung. Zwei wichtige Informationsquellen aus dem Bereich der Penetrationstests enthalten also entweder nur wenige Informationen oder Informationen, die schon mehrere Jahre alt sind.

Aus diesem Grund soll die vorliegende Arbeit einige Forschungsfragen beantworten. Zunächst soll die Frage beantwortet werden, welche die häufigsten Schwachstellen in IoT-

[11] Vgl. 12, S. 13 ff.
[12] Vgl. 13.
[13] Vgl. 14.
[14] Vgl. 15.
[15] Vgl. 16.
[16] Vgl. 17.

Geräten sind und wie sich diese Schwachstellen zwischen den verschiedenen Kategorien von IoT-Geräten unterscheiden. Anschließend soll evaluiert werden, welche Methoden für die Durchführung an Penetrationstests für IoT-Geräte derzeit existieren. Aus den daraus resultierenden Informationen sollen die Fragen beantwortet werden, welche kritischen Elemente bei der Entwicklung einer Penetrationstest-Methodik für IoT-Geräte berücksichtigt werden müssen und wie eine selbst entwickelte Methodik an eine große Menge an unterschiedlichen IoT-Geräten angepasst werden kann, ohne dabei an Effektivität zu verlieren. Zudem soll geklärt werden, wie die entwickelte Methode effektiv von Penetrationstestern und Unternehmen angewendet werden kann und welche Herausforderungen bei der Umsetzung auftreten können und wie diese Herausforderungen gelöst werden können. Zudem soll die entwickelte Methode mit zuvor evaluierten bestehenden Methoden verglichen werden, um zu erwartende Vorteile zu identifizieren. Abschließend wird die Frage, wie die entwickelte Methode angesichts der schnellen Entwicklung von IoT-Technologien an zukünftige Entwicklungen angepasst und relevant bleiben kann.

1.3 Vorgehensweise

Zur Beantwortung der zuvor genannten Forschungsfragen soll zunächst eine ausführliche Literaturrecherche durchgeführt werden, um Informationen über den aktuellen Stand im Bereich Penetrationstests von IoT-Geräten zusammenzutragen. Anschließend wird dem Leser auf Basis dieser Informationen zunächst das notwendige Grundlagenwissen über Internet of Things sowie den allgemeinen Ablauf von Penetrationstests vermittelt. Hierfür werden anfangs die wichtigsten Begriffe aus den Bereichen Internet of Things und Penetrationstests erläutert. In diesem Zuge werden zudem die verschiedenen Netzwerkprotokolle und Dienste von IoT-Systemen erläutert. Sobald die Grundlagen vermittelt, sowie wichtige Fachbegriffe und Definitionen geklärt wurden, sollen übliche Methoden für Penetrationstests bei IoT-Geräten aufgezeigt werden. Hierbei werden auch gängige Softwaretools, welche bei Penetrationstests von IoT-Geräten zum Einsatz kommen, erläutert. Dies soll gängige Schwachstellen von IoT-Geräten aufzeigen und Methoden beschreiben, mit denen ein Angreifer diese Schwachstellen ausnutzen kann. Als Orientierung dient hierfür unter anderem die Liste der zehn häufigsten Schwachstellen von IoT-Geräten, welche vom OWASP gepflegt wird.[17] Sofern möglich sollen diese Methoden an haushaltsüblichen IoT-Geräten wie zum Beispiel Smart Home Geräten oder Netzwerk-Überwachungskameras demonstriert werden. Auf Grundlage der durch dieses Vorgehen gewonnenen Erkenntnisse werden anschließend bestehende Penetrationstest-Methodiken erläutert und bewertet. Daraufhin soll eine eigene Methodik entwickelt werden, welche sich praktisch von Penetrationstestern und Unternehmen einsetzen lässt und sich flexibel auf unterschiedliche Gerätetypen anwenden lässt. Dieses Vorgehen ermöglicht es, theoretisches Wissen aus verfügbarer Literatur, praktisch zu verifizieren und zu demonstrieren und anschließend strukturiert einzusetzen. Als Resultat der Bachelorthesis wird ein strukturierter Leitfaden zum Testen und Bewerten der Sicherheit von IoT-Geräten erwartet, welcher sich auf eine Vielzahl an IoT-Geräten anwenden lassen soll und sowohl

[17] Vgl. 18.

Theorie als auch Praxis miteinander vereint.

2 Grundlagen von Penetrationstests

Wie bereits zuvor erwähnt, bezeichnet ein Penetrationstest das Suchen nach Sicherheitslücken und sensiblen Informationen in IT-Systemen, bei dem sich Angreifer ohne kriminelle Absichten den gleichen Mitteln bedienen, wie kriminelle Akteure.[1] Für eine effektive Durchführung eines Penetrationstests ist es daher notwendig, die generellen Abläufe und Werkzeuge, welche auch kriminelle Akteure verwenden können, genau zu kennen und auf das zu testende System abzustimmen.[2] Dieses Kapitel widmet sich daher zuerst den Grundlagen von Penetrationstests im allgemeinen, bevor zu einem späteren Zeitpunkt eine Methodik zum Penetrationstesten von IoT-Geräten im speziellen entwickelt wird. Im Allgemeinen unterscheidet man bei Penetrationstests zwischen Black-, White- und Grey Box Tests.[3] Diese drei Arten von Penetrationstests unterscheiden sich durch die Informationen, über welche die Tester über das zu testende System verfügen. So sind beispielsweise bei einem Black Box Test für die Tester nur die Informationen verfügbar, über die auch ein Angreifer von außen verfügen könnte. Dadurch ist dieser Typ von Penetrationstest besonders realistisch, aber auch entsprechend aufwändig.[4] Zudem ist die Zeit für Penetrationstests in der Regel begrenzt und besonders komplexe Schwachstellen, die mit mehr Informationen eventuell gefunden werden könnten, bleiben im schlimmsten Fall unentdeckt. Der Gegensatz zum Black Box Test ist der White Box Test. Hier stehen den Testern möglichst viele Informationen zu Verfügung, um schnell und effektiv testen zu können. Dies können Informationen wie die Dokumentation und den Quellcode des Systems umfassen.[5] Eine Kombination beider Techniken wird Grey Box Test genannt. Hierbei werden zum einen mittels White Box Techniken Schwachstellen direkt gesucht, zum anderen aber auch mittels Black Box Techniken geprüft, wie gut ein System mit Angriffen im Allgemeinen umgehen kann.[6]

Der generelle Ablauf unterschiedlicher Penetrationstests unterscheidet sich unabhängig davon, was getestet werden soll nur geringfügig. Dies hat den Vorteil, dass Penetrationstests nach einem konsistenten Schema ablaufen und für alle Beteiligten verständlich sind.[7] Penetrationstests bestehen übergreifend aus den drei Phasen der Planung, der Ausführung und den Abschließenden Tätigkeiten.[8]

[1] Vgl. 9, S. 13 f.
[2] Vgl. 9, S. 14.
[3] Vgl. 19, S. 11.
[4] Vgl. 19, S. 11.
[5] Vgl. 20, S. C–1.
[6] Vgl. 20, S. C–1.
[7] Vgl. 20, S. 2.1.
[8] Vgl. 20, S. 2.1.

2.1 Planungsphase

Eine gründliche Planungsphase ist essenziell und legt den Grundstein für einen erfolgreichen Penetrationstest.[9] In der Planungsphase (Engl. Planning) liegt der Fokus auf dem Festlegen von Umfang (Scope) und Zielen sowie der Vorbereitung des Penetrationstests im Allgemeinen.[10] Das Sammeln von weiteren Informationen, die für den Test notwendig oder hilfreich sind kann unter Umständen auch in der Planungsphase erledigt oder begonnen werden, sofern es für den jeweiligen Penetrationstest sinnvoll erscheint. Aber auch Zugänge zu dem zu testenden System werden, wenn möglich bereits in dieser Phase organisiert und beim Testen von IoT-Geräten wird in dieser Phase auch die dafür notwendige Hardware beschafft.[11] Dies dient dazu, einen möglichst reibungslosen Ablauf während des Penetrationstests sicherzustellen. Sollten fehlende Zugänge oder Hardware erst bei der Ausführung des Penetrationstests auffallen, so kann dies Zeit kosten und damit eine gründliche Durchführung gefährden.

Auch sollte bedacht werden, ob für das zu testende System kurz vor oder während des Penetrationstests Wartungsarbeiten oder Aktualisierungen geplant sind. Diese können die Effektivität des Penetrationstests beeinflussen. Während normalerweise zu empfehlen ist, Penetrationstests anzukündigen und nicht an Produktivsystemen zu testen, um Ausfälle und Störungen zu vermeiden, kann es in manchen Fällen dennoch sinnvoll sein, Penetrationstests unangekündigt durchzuführen, um spezielle Vorbereitungen etwa durch Administratoren zu verhindern.[12] Dies verhindert die Implementation temporärer Sicherheitsmaßnahmen, welche außerhalb des Penetrationstests nicht zum Einsatz kommen würden.

Es empfiehlt sich daher, eine auf das eigene Unternehmen angepasste Richtlinie für Penetrationstests zu erstellen, auf der anschließend alle Tests aufbauen können.[13] In dieser Richtlinie sollten unter anderem Rollen und Verantwortlichkeiten geregelt werden, sowie die Anforderungen an die Dokumentation der Durchführung und der Ergebnisse von durchgeführten Tests.[14] Somit kann ein strukturierter Ablauf gewährleistet werden und die Ergebnisse bleiben untereinander vergleichbar. Zudem müssen Zuständigkeiten und Aufgaben nicht bei jedem Test erneut erarbeitet werden, was die Durchführung beschleunigt. Weiterhin kann das Verfahren im Laufe der Zeit sukzessive verbessert werden, sollten sich mit der Zeit Probleme bemerkbar machen, welche regelmäßig auftreten. Daher empfiehlt es sich am Ende eines jeden Tests, eventuell aufgetretene organisatorische Probleme im Rahmen der später näher erläuterten abschließenden Tätigkeiten noch einmal zu evaluieren.

2.2 Ausführungsphase

Während der Ausführungsphase (Engl. Execution) wird das zu testende System gezielt angegriffen, um mögliche Schwachstellen ausfindig zu machen und diese anschließend zu

[9] Vgl. 20, S. 6–1.
[10] Vgl. 12, S. 20.
[11] Vgl. 20, S. 2.1.
[12] Vgl. 20, S. 7–2.
[13] Vgl. 20, S. 6–1.
[14] Vgl. 20, S. 6–1.

verifizieren.[15] Die Tätigkeiten die in dieser Phase durchgeführt werden, unterscheiden sich je nachdem welche Art von Penetrationstest durchgeführt wird und auch durch die verwendeten Technologien des zu testenden Systems. So können Penetrationstests von IoT-Geräten beispielsweise aufwändiger sein, als beim Testen einer alleinstehenden Webanwendung. Dies ist der Fall, da IoT-Geräte Hardware, eine Firmware und gegebenenfalls eine Webanwendung zur Administration beinhalten. Die verwendeten Mittel zur Durchführung eines Penetrationstests, unterscheiden sich demnach je nach zu testendem System.

Sollten beim Testen Programme wie Schwachstellenscanner genutzt werden, die einfache Tests auf Schwachstellen in großer Menge automatisiert durchführen, so sollten die Ergebnisse anschließend noch einmal manuell validiert werden, da es bei diesen Programmen zu Falschmeldungen kommen kann.[16] Auch ist es wichtig, die einzelnen Schwachstellen nicht isoliert zu betrachten und zu beheben, sondern auch den Grund der Schwachstelle herauszufinden.[17] Dies können beispielsweise Gründe wie fehlerhafte Prozesse, mangelhafte Konfiguration und Härtung von Systemen, oder das gänzliche Fehlen von Konfigurationsrichtlinien sein.[18] Die Erkenntnisse, die während der Ausführungsphase gewonnen werden, sollten für die Erstellungen eines späteren Berichtes bereits während des Testes ausführlich dokumentiert werden. Dies stellt sicher, dass beim späteren Verfassen des Reports, keine Informationen verloren gehen und die Vorgehensweisen auch für Entwickler oder Administratoren nachvollziehbar sind. Besonders kritische Schwachstellen, welche beispielsweise auch Produktivsysteme betreffen, sollten hingegen bereits während der Ausführungsphase gemeldet und behoben werden. Weiterhin sollten Schwachstellen, die bei einem vorherigen Test aufgetreten sind, noch ein weiteres Mal getestet werden. Somit kann sichergestellt werden, dass diese Schwachstellen dauerhaft behoben sind. Des Weiteren kann es vorteilhaft sein, bei einem erneuten Test andere Penetrationstester einzusetzen, da diese unterschiedliche Stärken und Schwächen besitzen, die das Auffinden von möglichst vielen Schwachstellen verhindern können.

Vor dem praktischen Beginn eines Penetrationstests ist es zudem sinnvoll, frei zugängliche Informationen über das Internet über das zu testende System zu recherchieren, um die praktische Durchführung so effektiv wie möglich zu gestalten.[19] Auf diese Weise lässt sich bereits vorab ein Grundverständnis erlangen. Als mögliche Informationsquelle kann hier beispielsweise die Herstellerwebseite dienen, auf der oftmals Dokumentationen zum jeweiligen Produkt zum Herunterladen angeboten werden. Bei Open Source Anwendungen ist zudem der Quellcode leicht zugänglich und kann näher untersucht werden.

2.3 Abschließende Tätigkeiten

Die Phase der abschließenden Tätigkeiten folgt auf die Ausführungsphase, in welcher der eigentliche Penetrationstest stattgefunden hat. Die Abschließenden Tätigkeiten sind essenziell,

[15] Vgl. 20, S. 2.1.
[16] Vgl. 20, S. 7–3.
[17] Vgl. 20, S. 7–3.
[18] Vgl. 20, S. 7–4.
[19] Vgl. 8, S. 197 f.

um aus den Erkenntnissen der Ausführungsphase einen Mehrwert zu generieren und die Sicherheit nachhaltig zu erhöhen. Ein Schwerpunkt in dieser Phase ist beispielsweise das Auswerten der zuvor generierten Daten und das Erstellen eines umfassenden Berichtes über den vorangegangenen Penetrationstest.

Der Bericht enthält für gewöhnlich eine kurze Zusammenfassung über die Ergebnisse, sowie ausführliche Informationen über die eventuell gefundenen Schwachstellen. Die ausführlichen Beschreibungen können Vorgehensweisen, verwendete Tools, gesammelte Logdaten, sowie den Schweregrad der Schwachstellen enthalten. Für die Bewertung des Schweregrades empfiehlt sich das Bewertungssystem Common Vulnerability Scoring System (CVSS), welches sich als Standard für die Bewertungen von Schwachstellen in Unternehmen, Regierungen und Sicherheitsanwendungen wie Schwachstellenscanner durchgesetzt hat.[20] Auch werden die potenziellen Auswirkungen der Schwachstellen betrachtet, sowie die Wahrscheinlichkeit für eine Ausnutzung dieser. Auf Basis der ermittelten Schwachstellen und Risiken, werden zudem Empfehlungen und Vorschläge zur Beseitigung der Schwachstellen gegeben.[21]

Dabei ist es wichtig, die gewonnenen Erkenntnisse effektiv an alle relevanten Partien zu vermitteln und zu besprechen, statt den Bericht nur auszuhändigen. Relevante Parteien können dabei technische Teams, Management und eventuell auch Endbenutzer sein. Die gewonnenen Erkenntnisse können dabei auch in die Gesamtstrategie für IT-Sicherheit einfließen, wenn sie sich auch auf andere Gebiete anwenden lassen und somit die Sicherheit im Allgemeinen verbessern. Besonders wichtig ist es, konkrete Aufgabenstellungen zur Behebung der Schwachstellen und Risiken zu definieren, welche innerhalb eines zuvor definierten Zeitraums erledigt werden sollen. Die Erledigung dieser Aufgaben sollte dabei aktiv nachverfolgt werden. Die Abschließenden Tätigkeiten sind mehr als nur ein formaler Abschluss eines Penetrationstests. Sie sind ein entscheidender Faktor zur Sicherstellung, dass die gewonnenen Erkenntnisse in nachhaltigen Verbesserungen der IT-Sicherheit resultieren.

[20] Vgl. 21, S. 129.
[21] Vgl. 20, S. 2.1.

3 Grundlagen des Internet of Things

Als IoT wird die Vernetzung von physischen Objekten aus dem Alltag bezeichnet. Diese vernetzten Objekte werden mit Sensoren und Aktoren ausgestattet und können Daten erfassen und verarbeiten, sowie mit anderen Objekten interagieren. Beispielhaft wäre hier der Einsatz von Radio-Frequency Identification (RFID)-Chips in der Logistik, um Objekte über alle Logistikprozesse hinweg überwachen zu können.[1] Mittels IoT ist es möglich, ein Ökosystem vernetzter Objekte zu schaffen, welche große Mengen an präzisen Daten sammeln, austauschen und verarbeiten können. Die Verbreitung von IoT-Geräten kann dabei alle denkbaren Bereiche durchdringen. So können IoT-Geräte in privaten Haushalten unter dem Begriff Smart-Home eingesetzt werden, in Industrieunternehmen unter dem Begriff *Industrie 4.0* die Produktionsprozesse digitalisieren, oder in der Stadtentwicklung in Verbindung mit vernetzten Automobilen zur intelligenten Verkehrsoptimierung eingesetzt werden. Da IoT-Geräte oft sehr klein und kostengünstig sein sollen, besitzen sie nicht selten nur eingeschränkte Ressourcen, was Speicher und Rechenleistung angeht.[2] Diese Einschränkungen können sich direkt auf die Sicherheit der Geräte auswirken.

Aufgrund der oft geringen Sicherheit sind IoT-Geräte für Angreifer ein lukratives Ziel. Diese zielen nicht nur auf sensible Daten oder die Manipulation der Geräte ab, sondern nutzen IoT-Geräte auch zur Erstellung von Botnetzen. Botnetze können dabei aus Geräten wie Fernsehern, Überwachungskameras oder Druckern bestehen und werden dazu genutzt, um Distributed Denial of Service (DDoS)-Angriffe auf unterschiedlichste Ziele durchzuführen.[3] So wurden große Mengen an mit der Schadsoftware *Mirai* infizierter IoT-Geräte zu einem Botnetz zusammengeschlossen und dafür verwendet, um DDoS-Angriffe durchzuführen.[4] Zur Infizierung der Geräte, welche hauptsächlich aus Routern und Netzwerkkameras bestanden, hat die Schadsoftware konstant das Internet nach IoT-Geräten durchsucht und eine Liste von 62 gängigen Benutzernamen und Passwörtern dafür verwendet, Zugriff auf die Geräte zu erlangen.[5]

Da IoT-Geräte für Angreifer ein interessantes Ziel darstellen, sollte die Sicherheit dieser Geräte möglichst gründlich überprüft werden. Bei der Überprüfung der Sicherheit von IoT-Geräten sollten alle möglichen Angriffsvektoren betrachtet werden. Diese bestehen unter anderem aus der Hardware selbst, der auf der Hardware installierten Firmware, den Kommunikationsschnittstellen des Gerätes, sowie dem Netzwerk, in dem das Gerät sich befindet, als auch Steuerungssoftware, wie beispielsweise Web-Interfaces, Mobile Apps, oder Cloud-APIs.

[1] Vgl. 8, S. 39.
[2] Vgl. 22, S. 1.
[3] Vgl. 23.
[4] Vgl. 24.
[5] Vgl. 24.

3.1 Netzwerkprotokolle und Dienste

Um reibungslose Kommunikation innerhalb von Computernetzwerken zu gewährleisten, ist es notwendig, standardisierte Protokolle zu verwenden, die alle Beteiligten gleichermaßen verstehen.[6] Das Transmission Control Protocol/Internet Protocol (TCP/IP) Referenzmodell und das Open Systems Interconnection (OSI) Referenzmodell legen einen einheitlichen Ablauf für alle verwendeten Protokolle untereinander fest.[7] Je nach Einsatzzweck und physischen Voraussetzungen nutzen IoT-Geräte unterschiedliche Netzwerkprotokolle und Dienste. Die verschiedenen Dienste und Netzwerkprotokolle können dabei den unterschiedlichen Schichten des TCP/IP Referenzmodells zugeordnet werden.[8] Die Schichten des TCP/IP Referenzmodells entsprechen einer Zusammenfassung der Schichten des OSI Referenzmodells. Im Folgenden werden die wichtigsten Protokolle und Dienste aus dem Bereich IoT kurz erläutert und den zugehörigen Schichten des TCP/IP Referenzmodells zugeordnet.

ZigBee ist jedoch ein IoT-Protokoll, welches auf allen Schichten des TCP/IP Referenzmodells arbeitet.[9] ZigBee erlaubt die Erstellung von energieeffizienten Mesh-Netzwerken, welche aus einem Koordinator (ZC), einem Router (ZR) und den ZigBee Endgeräten (ZED) bestehen.[10] In ZigBee-Netzwerken gibt es dabei nur einen Koordinator, welcher alle Geräte authentifiziert und beim Beitritt zum Netzwerk für die Validierung der Geräte einen einzigartigen Netzwerkschlüssel vergibt, welcher mit einem Verbindungsschlüssel verschlüsselt übertragen wird.[11] Eine Besonderheit von ZigBee ist, dass Endgeräte als Router fungieren können und somit eine Erstellung von großflächigen Netzwerken möglich ist, wobei die einzelnen Geräte bei Inaktivität in einem energiesparenden Ruhemodus verweilen können.[12]

3.1.1 Anwendungsschicht

Die Anwendungsschicht übernimmt die Funktionen, die für das Verwalten der Sitzungen notwendig sind, sowie die Funktionen für visuelle Darstellung der Anwendung und die Interaktion mit dem Anwender.[13] Dabei erfolgt eine Koordination der Kommunikation verschiedener Anwendungen und eine Wiederherstellungsmöglichkeit für unterbrochene Sitzungen wird zur Verfügung gestellt. Mit Hilfe von Sicherungspunkten können so große Datenübertragungen im Falle einer Unterbrechung bei Wiederaufbau der Verbindung fortgesetzt werden.[14] Um die übertragenen Daten für den Anwender visuell darstellen zu können, ist die Anwendungsschicht dazu in der Lage die vorliegenden Daten zu dekomprimieren, zu entschlüsseln und zu interpretieren, um sie dem Anwender in geeigneter Form anzeigen

[6] Vgl. 8, S. 99 f.
[7] Vgl. 8, S. 99.
[8] Vgl. 25.
[9] Vgl. 12, S. 196.
[10] Vgl. 26, S. 23.
[11] Vgl. 26, S. 23.
[12] Vgl. 27, S. 1155 f.
[13] Vgl. 28, S. 38.
[14] Vgl. 8, S. 102.

zu können.[15] Zudem werden in der Anwendungsschicht mehrere Protokolle zur Verfügung gestellt, um dem Benutzer Dienste wie SMTP zum Versenden von E-Mails, oder beispielsweise auch Hypertext Transfer Protocol (HTTP) und Hypertext Transfer Protocol Secure (HTTPS) für den Zugriff auf Webinhalte zur Verfügung gestellt.[16]

HTTP ist ein Protokoll der Anwendungsschicht, welches zur Übertragung von Webinhalten unterschiedlicher Art zwischen einem Webserver und einem Empfänger wie beispielsweise einem Webbrowser verwendet wird.[17] Im HTTP Protokoll sind verschiedene Methoden definiert, die ein Client an einen Server stellen kann, um Daten zu senden oder zu empfangen.[18] Gängige Methoden sind *GET* zum Anfordern einer Ressource vom Webserver, oder *POST*, zum übertragen von Inhalten vom Client an eine Ressource des Webservers.[19] HTTP ist ein Klartextprotokoll und überträgt Daten grundsätzlich unsicher.[20] Daher ist der Einsatz von HTTPS nach Möglichkeit zu bevorzugen. HTTPS erweitert das HTTP-Protokoll um das Transport Layer Security (TLS)-Protokoll und ermöglicht damit eine verschlüsselte Übertragung von Daten durch einen sicheren TLS-Tunnel.[21]

Constrained Application Protocol (CoAP) ist ein Übertragungsprotokoll, welches speziell für IoT-Geräte mit geringen Hardwareressourcen entwickelt wurde und Daten von Punkt-zu-Punkt in 1:1-Relation übertragen kann. CoAP ist für viele verschiedene Einsatzzwecke nutzbar und überträgt Daten per User Datagram Protocol (UDP) und kann mit den Protokollen HTTP und Extensible Messaging and Presence Protocol (XMPP) interagieren.[22] Durch die Verwendung von speziellen CoAP-Uniform Resource Identifiers (URIs), können CoAP-Dienste eigenständig untereinander kommunizieren und sich durch eine Suche innerhalb des Netzwerkes über Port 5683 finden und ansprechen.[23]

Message Queuing Telemetry Transport (MQTT) unterscheidet sich von CoAP vorrangig darin, dass zum einen das Transmission Control Protocol (TCP)-Protokoll verwendet wird und zum anderen eine Kommunikation in 1:n-Relation möglich ist.[24] MQTT arbeitet nach dem Publish-Subscribe-Verfahren, was bedeutet, dass andere MQTT Dienste im Netzwerk nicht nur Daten gezielt abfragen, sondern auch so bereitstellen können, dass andere Dienste die Daten abonnieren können und bei Änderungen automatisch benachrichtigt werden.[25]

XMPP ist ein Protokoll zum dezentralen Austausch von Nachrichten im Extensible Markup Language (XML)-Format, welches vor allem in der Anfangszeit von IoT-Geräten eingesetzt wurde, um Nachrichten auszutauschen.[26] Ein positiver Aspekt beim Einsatz von XMPP ist die Flexibilität die das Protokoll bietet und durch den Einsatz von Secure Sockets Layer (SSL)

[15] Vgl. 8, S. 102.
[16] Vgl. 8, S. 102.
[17] Vgl. 29, S. 234 f.
[18] Vgl. 28, S. 58 f.
[19] Vgl. 28, S. 58 f.
[20] Vgl. 29, S. 236.
[21] Vgl. 29, S. 236.
[22] Vgl. 28, S. 339.
[23] Vgl. 28, S. 339.
[24] Vgl. 28, S. 343 f.
[25] Vgl. 28, S. 344.
[26] Vgl. 9, S. 1144.

und TLS auch die gute Sicherheit.[27] Allerdings ist das XMPP-Protokoll ursprünglich für das Versenden von Direktnachrichten konzipiert worden und daher nicht für den Einsatz auf Kleinstgeräten mit limitierten Ressourcen optimiert.[28]

Advanced Message Queuing Protocol (AMQP) ist ein binäres Client-Server-Protokoll, welches Nachrichten auf einem Server in einer Warteschlange speichern kann, um Nachrichten zu einem späteren Zeitpunkt zustellen zu können, wenn der Empfänger im Moment der Zustellung nicht erreichbar ist.[29]

3.1.2 Transportschicht

Die Transportschicht ist verantwortlich für eine reibungslose Übertragung der Daten. Dies beinhaltet die Verwaltung von Verbindungen, sowie die Verwendung von Portnummern, um die Daten an die richtigen Dienste auf dem Endgerät zu vermitteln.[30] Die Transportschicht nimmt dabei Anfragen von höheren Ebenen entgegen und baut eine Verbindung zwischen den Endsystemen her. Diese Schicht ermöglicht die Kommunikation zwischen verschiedenen Diensten und kann dabei entweder je eine einzelne logische Verbindung für die Übertragung der Daten nutzen, oder mehrere Verbindungen auf einem einzelnen Datenkanal im sogenannten Multiplexbetrieb bündeln. Innerhalb der Transportschicht werden die Daten in Pakete aufgeteilt und in korrekter Reihenfolge an den Empfänger übermittelt.[31]

UDP ist ein verbindungsloses Transportprotokoll, welches eine geringe Komplexität besitzt und weder die Reihenfolge der Datenpakete kontrolliert, noch deren Integrität überprüft. Daher ist ein Einsatz von UDP für kritische Daten nicht empfehlenswert. Aufgrund der geringen Komplexität eignet sich UDP allerdings sehr gut für die Übertragung zeitkritischer Daten, bei denen ein Paketverlust nicht sonderlich relevant ist.[32] Ein Beispiel für eine solche Datenübertragung wäre etwa das Videostreaming.

TCP ist ein Transportprotokoll, welches im Gegensatz zu UDP nicht mit verbindungslosen Einzelnachrichten arbeitet, sondern die Daten innerhalb eines Datenstroms überträgt, bei dem stets die korrekte Reihenfolge der Pakete eingehalten wird und der erfolgreiche Eingang eines jeden Paketes von der Gegenseite bestätigt wird.[33]

3.1.3 Internetschicht

Die Internetschicht ist verantwortlich für eine korrekte Übertragung der Daten zwischen zwei Endgeräten über das Internet.[34] Hierbei können nicht benachbarte Teilnetze miteinander verknüpft werden und eine Ende-zu-Ende-Kommunikation über unterschiedliche Schemata

[27] Vgl. 30, S. 4.
[28] Vgl. 30, S. 4.
[29] Vgl. 9, S. 1144.
[30] Vgl. 28, S. 38.
[31] Vgl. 8, S. 101 f.
[32] Vgl. 28, S. 49 f.
[33] Vgl. 28, S. 50 f.
[34] Vgl. 28, S. 38.

zur Adressierung und heterogene Protokolle hinweg betreiben. Die Internetschicht ist demnach dafür verantwortlich, stets den effizientesten Weg für die Daten durch das Netzwerk ausfindig zu machen und zu nutzen. Diese Aufgabe wird *Routing* genannt.[35]

Das **Internet Protocol (IP)** stellt einen verbindungslosen Dienst der Internetschicht dar, welcher Daten in Form von Paketen überträgt, ohne deren Eingang bei der Gegenseite zu bestätigen. Da weder die vollständige Übertragung aller Pakete noch deren korrekte Reihenfolge sichergestellt werden kann, gilt IP als unzuverlässig. Da die maximal mögliche Größe eines IP-Paketes zwar bis zu 65.535 Bytes betragen darf, aber Ethernet beispielsweise nur eine Paketgröße von 1.518 Bytes unterstützt, können Pakete in kleinere Fragmente aufgeteilt werden. IP-Pakete werden durch die Verwendung von Routern zum Ziel geleitet, wobei nicht alle Fragmente eines Paketes den gleichen Pfad zum Ziel durchlaufen müssen.[36] Damit die Empfänger von IP-Paketen diese richtig verarbeiten können, ist es notwendig das Protokoll welches auf das IP-Protokolls folgt innerhalb des Paketes zu definieren. Nachfolgeprotokolle können beispielsweise die beiden Transportprotokolle TCP oder UDP sein.[37]

IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) ist ein offener Standard, welcher es ermöglicht Internet Protocol version 6 (IPv6) über IEEE 802.15.4 basierende Niedrigenergie-Funknetzwerke zu verwenden. 6LoWPAN wurde ursprünglich für die Verwendung im 2,4-Gigahertz (GHz)-Bereich entwickelt, kann aber unter anderem auch im Frequenzbereich unter 1-GHz-Bereich oder mit Bluetooth verwendet werden. Über diesen Standard ist es möglich 6LoWPAN-Netzwerke unter Verwendung von Edge-Routern in IPv6- und Internet Protocol version 4 (IPv4)-Netzwerke zu integrieren.[38]

Routing Protocol for Low-Power and Lossy Networks (RPL) ist ein speziell für 6LoWPAN entwickeltes Routingprotokoll und gilt als Standardroutingprotokoll des IoT. RPL bildet in Zusammenarbeit mit weiteren Standards die Grundarchitektur des IoT.[39] Zur Wegefindung erzeugt RPL einen gerichteten azyklischen Graphen, welcher sich an einem Hauptknoten, beispielsweise einem Edge-Router orientert.[40] Zum Aufbau des Graphen sendet jeder Knoten Kontrollnachrichten aus, welche zudem Rückschlüsse über die Qualität der Verbindung geben.[41] RPL kann im Speichermodus in dem jeder Knoten eine Tabelle mit Wegen zu allen möglichen Zielen hält betrieben werden, oder aber im Nicht-Speichermodus, bei dem nur der Hauptknoten diese Informationen hält und sie für die Wegfindung als Teil der Pakete mitsendet.[42]

[35] Vgl. 8, S. 101 f.
[36] Vgl. 28, S. 107 f.
[37] Vgl. 29, S. 127 f.
[38] Vgl. 31, S. 2.
[39] Vgl. 32, S. 16.
[40] Vgl. 32, S. 16.
[41] Vgl. 32, S. 16.
[42] Vgl. 32, S. 16.

3.1.4 Netzzugangsschicht

Die Netzzugangsschicht sorgt für die Adressierung der Endgeräte untereinander mittels physikalischer Adressen (MAC-Adressen) und die Absicherung der Daten.[43] Die Netzzugangsschicht hat die Aufgabe, eine physische Verbindung zwischen zwei Endgeräten zu etablieren und die Daten in Paketen zusammenzufassen und unter Verwendung von Prüfsummen und fehlerkorrigierender Codes fehlerfrei zu übertragen.[44]

Bluetooth ist eine Funktechnologie der Netzzugangsschicht, welche für den Einsatz auf kurze Distanzen konzipiert wurde und im Lizenzfreien Industrial Scientific Medical (ISM)-Band auf den Frequenzen 2,4 und 2,485 GHz arbeitet.[45] Bluetooth ist dazu in der Lage inaktive Verbindungen abzuschalten und arbeitet recht stromsparend, da es vor allem für den Einsatz auf mobilen Endgeräten entwickelt wurde.[46] Noch energieeffizienter wurde Bluetooth in der Version 4.0, auch BLE genannt, welche häufig in IoT-Geräten anzutreffen ist.[47] Da viele moderne Smartphones Bluetooth in Version 4.0 unterstützen, ist es möglich IoT-Geräte welche auch mit Bluetooth 4.0 arbeiten damit anzusteuern.

Wireless Local Area Network (WLAN) ist eine kabellose Variante von Ethernet, welche in den Frequenzbereichen um 5 GHz und 2,4 GHz arbeitet.[48] Unter Idealbedingungen kann wlan eine Reichweite von 100 bis 300 Metern erreichen.[49] Besonders der Standard 802.11ax auch als Wi-Fi 6 bekannt, spielt im Bereich IoT eine wichtige Rolle, da er energieeffizient arbeitet, dabei aber gleichzeitig flexibel und skalierbar bleibt.[50] Der Einsatz von WLAN ist vor allem im Bereich der Heimautomatisierung sinnvoll, da heutzutage die meisten Haushalte bereits mit WLAN ausgestattet sind und daher zum Betrieb vernetzter Geräte keine zusätzliche Hardware angeschafft werden muss. Dies kann die Einstiegshürde für die Anschaffung von vernetzten Geräten im Haushalt senken.

RFID dient der Kommunikation zwischen Objekten mittels Radiowellen und ist verbreitet in Chipkarten, Maschinen und Sensoren.[51] Das System funktioniert aus einer Kombination aus Transponder und Empfänger und funktioniert sowohl als passives System, welches aus einer Kombination aus Lesegerät und Tag besteht, bei der nur das Lesegerät eine Konstantstromquelle benötigt.[52] Das Tag bezieht seine Energie in dieser Kombination über ein feldstarkes Abfragesignal, welches vom RFID-Tag über die integrierte Antenne zur Versorgung des integrierten Schaltkreises mit Strom verwendet wird.[53] Dieses System ist kostengünstig und kompakt, bietet allerdings nur eine geringe Reichweite und eine geringe Speicherkapazität für Daten, da permanente Speicher größerer Kapizität mehr Energie benötigen.[54]

[43] Vgl. 28, S. 37 f.
[44] Vgl. 8, S. 100 f.
[45] Vgl. 33, S. 196 f.
[46] Vgl. 33, S. 196 f.
[47] Vgl. 12, S. 167 f.
[48] Vgl. 33, S. 233.
[49] Vgl. 33, S. 240.
[50] Vgl. 11, S. 381 f.
[51] Vgl. 10, S. 7.
[52] Vgl. 10, S. 8 f.
[53] Vgl. 10, S. 9.
[54] Vgl. 10, S. 9.

Aktive RFID-Systeme hingegen bestehen aus einer Kombination aus Tag mit großer Batterie und aktivem Sender, wodurch sich Reichweiten von bis zu 90 Metern und die Möglichkeit zur Verwendung von leistungsstarken Prozessoren ergeben.[55] Nachteilig bei solchen Geräte ist allerdings die geringe Lebensdauer von nur drei bis fünf Jahren, weshalb solche Systeme hauptsächlich in Logistiksystemen eingesetzt werden, bei denen die zu verfolgenden Objekte einen hohen monetären Wert besitzen.[56]

Near Field Communication (NFC) ist eine Funktechnologie, welche mit 13,56 MHz arbeitet, auf RFID aufbaut und für den Einsatz auf besonders kurzen Distanzen von etwa 4cm konzipiert ist.[57] Eine Besonderheit von NFC ist, dass keine interne Stromversorgung beispielsweise durch Batterien benötigt wird, da die Geräte vom Kommunikations-Host kabellos mit Strom versorgt werden.[58] Weit verbreitet sind NFC Chips vor allem in Bankkarten, welche zum kabellosen Bezahlen verwendet werden können.[59] Aufgrund der kleinen und flachen Bauweise, lassen sich NFC-Chips in nahezu alle Objekte integrieren, ohne dabei besonders aufzufallen.

Mobilfunk wird im Bereich von IoT außerhalb von lokalen Netzen eingesetzt und ist großflächig landesweit verfügbar.[60] Für die Einwahl in ein Mobilfunknetz benötigen Endgeräte eine sogenannte Subscriber Identity Module (SIM)-Karte.[61] Für IoT ist die Mobilfunktechnologie 5G besonders wichtig, da sie mehrere gleichzeitige Verbindungen ermöglicht, hohe Datenraten ermöglicht, eine geringe Latenz aufweist, wenig anfällig für Störungen ist und dabei gleichzeitig eine hohe Energieeffizienz aufweist.[62] Hohe Verbreitung findet Mobilfunk neben dem Einsatz in Smartphones unter anderem auch in modernen Kraftfahrzeugen, welche permanent vernetzt sind und somit Daten empfangen und senden können.

Ethernet ist eine kabelgebundene Netzwerktechnologie und bildet einen wichtigen Standard in der Vernetzung von IT-Systemen. Ethernet wird zur Übertragung von Daten in lokalen Netzwerken verwendet und basiert auf der direkten Adressierung der Geräte mittels Media-Access-Code (MAC)-Adressen, welche für die Regelung der Sendereihenfolge und Adressierung verantwortlich sind.[63] Für Ethernetverbindungen existieren unterschiedliche Standards, welche sich vor allem im Übertragungsmedium unterscheiden und Übertragungsraten von 10 Mbit pro Sekunde bis 10 Gbit pro Sekunde bieten.[64]

[55] Vgl. 10, S. 10 f.
[56] Vgl. 10, S. 10 f.
[57] Vgl. 11, S. 390.
[58] Vgl. 9, S. 1157.
[59] Vgl. 9, S. 1158.
[60] Vgl. 29, S. 108 f.
[61] Vgl. 29, S. 110.
[62] Vgl. 34, S. 16.
[63] Vgl. 29, S. 28 f.
[64] Vgl. 35, S. 44 ff.

3.2 Firmwares und Betriebssysteme

Ein elementarer Bestandteil von IoT-Geräten ist die Firmware. Die Firmware ist eine zentral im Gerät gespeicherte Software, welche dazu dient, die Kommunikation zwischen Hardware und Anwendungssoftware zu ermöglichen, wobei die Firmware für Endanwender für gewöhnlich nicht zugänglich ist.[65] Die Firmware eines IoT-Gerätes lässt sich daher nicht ohne weiteres analysieren. Sie muss entweder mittels verschiedener Techniken, welche später erläutert werden, extrahiert werden oder über unterschiedlichste Quellen aus dem Internet heruntergeladen werden.[66]

Hinsichtlich der geringen Ressourcen vieler IoT-Geräte ist es notwendig, spezielle Betriebssysteme zu verwenden, welche auf der vergleichsweise schwachen Hardware ausgeführt werden können und wenig Speicherplatz in Anspruch nehmen. Dennoch muss ein solches Betriebssystem mit unterschiedlichsten Hardwarekonfigurationen und Netzwerkprotokollen umgehen können, im Bedarfsfall Echtzeit-Anforderungen für eine präzise Datenverarbeitung erfüllen und dabei noch energieeffizient und möglichst sicher sein.[67] Daher gibt es je nach Einsatzzweck unterschiedliche Betriebssysteme im Bereich IoT.

So hat sich das Betriebssystem TinyOS beispielsweise im Bereich der Sensornetzwerke etabliert, da es aufgrund seiner besonderen Merkmale wie Unterstützung für verschiedene Geräte, einfacher Programmierung und der speziell entwickelten Programmiersprache NesC, auch für IoT-Anwendungen geeignet ist und sich zusammen mit Betriebssystem Contiki in diesem Bereich durchgesetzt hat.[68] Für unterschiedlichere IoT-Anwendungsfälle existiert das quelloffene Betriebssystem RIOT, welches sich vor allem durch seine Echtzeit-Funktionalität und die Verwendung eines Mikrokernels auszeichnet, sowie das Ausführen von Anwendungen welche in den weit verbreiteten Programmiersprachen C und C++ entwickelt wurden, erlaubt.[69] Betriebssysteme mit Echtzeit-Funktionalität (Real-time operating system (RTOS)) zeichnen sich dadurch aus, dass sie auch bei Verarbeitung vieler Ereignisse besonders kurze Latenzzeiten versprechen und Antworten innerhalb einer bestimmten Zeit garantieren.[70]

3.3 Hardware

Die Hardware von IoT-Geräten unterscheidet sich von anderer IT-Hardware dadurch, dass sie oft eine geringe physische Größe hat und stromsparend arbeiten muss. Dies erreichen Hersteller unter anderem durch eine Reduzierung der Komplexität und Rechenleistung sowie Speicherkapazität. Häufig verwenden Hersteller daher sogenannte System-on-a-Chip (SoC)s. SoCs zeichnen sich dadurch aus, dass mehrere Bestandteile eines Gerätes wie Prozessor, Funkmodule, Speicher und Peripherie auf einem einzigen Chip in geringer Größe zusam-

[65] Vgl. 12, S. 132.
[66] Vgl. 12, S. 135.
[67] Vgl. 36, S. 721 f.
[68] Vgl. 37, S. 2065.
[69] Vgl. 37, S. 2065.
[70] Vgl. 38, S. 96.

mengefasst werden.[71] Funkmodule für die Kommunikation über Bluetooth, WLAN, Global System for Mobile Communications (GSM) oder auch Zigbee können jedoch auch separat auf der Platine verbaut werden, um beispielsweise ein störungsfreies Signal zu Ermöglichen.[72] Da IoT-Geräte jeweils einen individuellen Einsatzzweck mit unterschiedlichen Anforderungen haben, unterscheidet sich der physische Aufbau von Gerät zu Gerät und lässt sich schwer verallgemeinern.

Bauteile von IoT-Geräten besitzen oft physische Schnittstellen und Bussysteme, damit beispielsweise Entwickler oder Techniker direkt mit den Geräten interagieren können. Häufige Verwendung findet unter anderem Universal Asynchronous Receiver Transmitter (UART), welches sich mit der seriellen Schnittstelle von älteren Computern vergleichen lässt.[73] UART verwendet eine Empfangsleitung (RX), eine Sendungsleitung (TX) und Erdung für die Kommunikation und stellt damit eine asynchrone Verbindung her, bei der sich die beteiligten Teilnehmer eigenständig um die Synchronisation kümmern müssen.[74] Ein gängiger Verwendungszweck von UART in IoT-Geräten ist das Debugging, wodurch es möglich ist, mittels Universal Serial Bus (USB) zu UART Adaptern über eine serielle Konsole direkt mit dem Betriebssystem des IoT-Gerätes kommunizieren zu können.[75] Über diese Konsole lässt sich mehr über die verwendete Software von einzelnen IoT-Geräten herausfinden und eventuelle Schwachstellen wie etwa im Klartext abgespeicherte Passwörter lassen sich identifizieren.[76]

[71] Vgl. 12, S. 70.
[72] Vgl. 12, S. 70.
[73] Vgl. 12, S. 76 f.
[74] Vgl. 12, S. 76 f.
[75] Vgl. 12, S. 77.
[76] Vgl. 12, S. 77.

4 Häufige Schwachstellen im Bereich des Internet of Things

Mit der Erläuterung der Grundlagen zu den Themen Penetrationstests und IoT-Geräte in den letzten beiden Kapiteln kann nun auf häufig anzutreffende Schwachstellen in IoT-Geräten eingegangen werden. Hierbei ist das Ziel, ein Verständnis für die unterschiedlichen Arten von Schwachstellen im IoT-Bereich zu entwickeln und die Bedingungen zu beleuchten, unter denen diese vorkommen können. Diese Informationen werden in den späteren Kapiteln als Grundlage für die Entwicklung einer Methode zum Durchführen von Penetrationstests an IoT-Geräten dienen. Als Informationsgrundlage über wichtige Schwachstellen im IoT-Bereich dient hier unter anderem die OWASP Internet of Things Top 10. OWASP ist eine Organisation, bestehend aus IT-Sicherheitsexperten aus der ganzen Welt, welche sich dafür einsetzt, Schwachstellen in Softwareprodukten unterschiedlichster Art zu verhindern und die mittlerweile eine der meistgenutzten Informationsquellen zu diesem Thema geworden ist.[1]

4.1 Unterschiedliche Schwachstellen verschiedener Kategorien von IoT-Geräten

Da IoT-Geräte in den unterschiedlichsten Ausprägungen existieren und eine Vielzahl unterschiedlicher Protokolle, Kommunikationstechnologien und Programmiersprachen verwenden, können sie verschiedenste Schwachstellen und unterschiedliche Kategorien enthalten.[2] Zudem haben die Geräte oft nur geringe Hardwareressourcen, was die Implementierung von komplexen Sicherheitsmechanismen wie umfangreicher Zugriffskontrolle und Rechteverwaltung nicht immer möglich macht.[3] Aus diesem Grund werden zur Administration und Verwaltung häufig externe Dienste wie mobile Anwendungen oder Cloud-Dienste benötigt, was zusätzliche Angriffsvektoren schafft.[4] Durch diese Varianz an Komplexität haben IoT-Geräte oft unterschiedliche Schwachstellen. Ein modernes Fahrzeug mit großer Rechenleistung kann so völlig andere Schwachstellen enthalten als ein kleiner Sensor oder eine vernetzte Überwachungskamera. Im Folgenden werden die Schwachstellen daher in verschiedene Kategorien gruppiert, da der Fokus in einem Penetrationstest beispielsweise nur auf dem Cloud-Backend liegen kann, oder ein Zugriff auf die Firmware des IoT-Gerätes nicht immer möglich ist.

[1] Vgl. 39, S. 74.
[2] Vgl. 39, S. 72 f.
[3] Vgl. 39, S. 72.
[4] Vgl. 39, S. 72.

4.2 Hardware

IoT-Geräte können für verschiedenste Szenarien an unterschiedlichsten Orten eingesetzt werden. Daher kann ein physischer Zugriff durch Dritte nicht ausgeschlossen werden, welche versuchen könnten, Geräte zu kompromittieren oder an geistiges Eigentum wie etwa die Firmware der Geräte zu gelangen.[5] Daher ist es notwendig, nicht nur alle Bereiche der Software von IoT-Geräten abzusichern, sondern auch die physische Sicherheit zu gewährleisten.

Auf Platz 10 der kritischsten Schwachstellen im Bereich IoT ist laut OWASP daher der **Mangel an physischer Härtung** (Lack of physical hardening), beziehungsweise Sicherheitsmaßnahmen.[6] Ein Angreifer könnte etwa versuchen, das Gerät zu öffnen und mit Hilfe von vorhandenen Anschlüssen wie UART, welche eigentlich für Tätigkeiten wie das Debugging beim Hersteller gedacht sind, Daten aus dem internen Speicher zu extrahieren oder das System zu manipulieren.[7]

4.3 Software

Die Software von IoT-Geräten bietet eine Vielzahl verschiedener Angriffsvektoren. Dies resultiert unter anderem aus den unterschiedlichsten Betriebssystemen und Firmwares, die für IoT-Geräte verwendet werden. Auf diesen Betriebssystemen sind zudem weitere oft selbst entwickelte Anwendungen anzutreffen, die für den Betrieb des Gerätes und die Integration in Ökosysteme wie Cloud-Umgebungen und mobile Anwendungen notwendig sind. Es muss also sichergestellt sein, dass Betriebssystem und Firmware stets aktuell und korrekt konfiguriert sind. Auch muss die darauf installierte Zusatzsoftware sicher und deren Abhängigkeiten auf dem aktuellen Stand sein. Zusätzlich müssen alle Schnittstellen, durch die diese Komponenten miteinander interagieren, eine sichere Kommunikation ermöglichen.

4.3.1 Firmware

Als Firmware wird eine Software bezeichnet, die spezifisch für eine bestimmte Hardware entwickelt wurde und dazu dient, Anwendungen Zugriff auf die Hardware eines Gerätes zu ermöglichen.[8] Die wichtigsten Bestandteile einer Firmware sind der Bootloader, Kernel und Dateisystem. Der Bootloader initialisiert verschiedene Systemkomponenten wie Arbeitsspeicher oder Hardware-Treiber und ist dabei mit dem BIOS eines klassischen Computers vergleichbar.[9] Der Bootloader ist auch dafür zuständig den Kernel zu starten und damit das Dateisystem zu initialisieren.[10] Demnach ist die Firmware einer der wichtigsten Bestandteile eines IoT-Gerätes, da ohne die Firmware kein Betrieb der Geräte möglich wäre.

[5] Vgl. 40, S. 421.
[6] Vgl. 39, S. 81.
[7] Vgl. 39, S. 81.
[8] Vgl. 26, S. 10.
[9] Vgl. 26, S. 12.
[10] Vgl. 26, S. 12.

Zum Beheben von Schwachstellen, sind regelmäßige Software-Updates unerlässlich, doch der Mangel an Rechenleistung, welche für Public-Key-Infrastruktur (PKI)-Systeme benötigt wird, sowie eine oft unsichere Schlüsselverwaltung erhöhen das Risiko von Angriffen auf IoT-Geräte.[11] Daher befindet sich das **Fehlen eines sicheren Mechanismus zur Aktualisierung** (Lack of secure update mechanism) an vierter Stelle der kritischsten Schwachstellen nach OWASP. Das Fehlen eines sicheren Mechanismus zur Aktualisierung kann dabei unweigerlich zur fünftkritischsten Schwachstelle führen, der **Verwendung von unsicheren oder veralteten Komponenten** (Use of insecure or outdated components).[12] Diese Schwachstelle kann schon zum Zeitpunkt der Entwicklung auftreten, wenn Softwareentwickler unsichere oder veraltete Bibliotheken und Frameworks für die Entwicklung ihrer Software verwenden.[13]

IoT-Geräte werden nicht nur häufig mit veralteter Software, sondern auch mit unsicheren Standardeinstellungen verkauft, um eine unkomplizierte und schnelle Inbetriebnahme beim Kunden sicherzustellen. Werden diese Einstellungen nach der Inbetriebnahme jedoch nicht verändert, so ist es für Angreifer möglich durch etwa die Verwendung von Standardzugangsdaten die Kontrolle über ein Gerät zu übernehmen.[14] Aus diesem Grund stehen **unsichere Standardeinstellungen (Insecure default settings)** an neunter Stelle der kritischsten Schwachstellen nach OWASP.[15]

Um die vorangegangenen Schwachstellen bei einer größeren Anzahl an Geräten effizient kontrollieren zu können ist es wichtig, dass sich die Geräte im Idealfall zentral überwachen, konfigurieren und aktualisieren lassen. Ist das nicht möglich, kann es schwierig werden, alle Geräte in einem sicheren Zustand zu behalten. Das **Fehlen einer Geräteverwaltung** (Lack of device management) ist daher auch in den kritischsten Schwachstellen aufgelistet und befindet sich an achter Stelle.[16]

Während der Entwicklung von Software verwenden Entwickler häufig einfache oder vom Hersteller vergebene Passwörter in verwendeten Komponenten wie Datenbanken oder Betriebssystemen, um die Entwicklung zu vereinfachen.[17] Diese Passwörter lassen sich von Angreifern mittels Brute Force-Angriffen leicht herausfinden. Bei in großen Mengen produzierten iot-Geräten können sich Angreifer solche Geräte besorgen, in Ruhe analysieren und anschließend die gewonnenen Informationen gegen ein Gerät, welches sich im produktiven Einsatz befindet, verwenden. Der Aufwand diese Schwachstelle auszunutzen ist also sehr gering. **Schwache, erratbare oder fest eingebaute Passwörter** (Weak guessable, or hardcoded passwords) befinden sich daher nicht ohne Grund auf dem ersten Platz der kritischsten Schwachstellen für iot-Geräte laut OWASP.[18] Beispielhaft wäre hier der Hersteller Kankun zu erwähnen, welcher in der Firmware seiner smarten Steckdose einen AES256-Schlüssel fest einprogrammiert hat und es Angreifern somit ermöglicht hat, Netzwerkverkehr zu entschlüsseln

[11] Vgl. 39, S. 79.
[12] Vgl. 39, S. 79.
[13] Vgl. 39, S. 79.
[14] Vgl. 39, S. 81.
[15] Vgl. 39, S. 81.
[16] Vgl. 41.
[17] Vgl. 39, S. 76.
[18] Vgl. 41.

und mitzulesen.[19]

An siebter Stelle befindet sich die **unsichere Übertragung und Speicherung von Daten** (Insecure data transfer and storage).[20] Diese Schwachstelle beschreibt etwa das Übertragen von Daten im Klartext, ohne die Verwendung einer Verschlüsselung, wodurch es anderen Netzwerkteilnehmern beim Mitlesen des Netzwerkverkehrs möglich ist, sensible Daten auszuleiten.[21]

Das Abgreifen sensibler Daten ist auch ein Problem der Schwachstelle an sechster Stelle der kritischsten IoT-Schwachstellen. Diese beschreibt einen **mangelhaften Datenschutz** (Insufficient privacy protection), welcher etwa durch die unnötige Übertragung und Speicherung von Daten mit eventuell fehlender Anonymisierung[22] Diese könnten von einem Angreifer erbeutet werden und gefährden die Privatsphäre von Nutzern, oder vertrauliche Informationen offenbaren.

4.3.2 Webanwendungen

Ein weiterer Bestandteil einiger IoT-Geräte sind Webanwendungen. Diese finden häufig Verwendung, um Geräte über einen Webbrowser konfigurieren zu können. Zum Beispiel enthalten vernetzte Überwachungskameras häufig einen integrierten Webserver, welcher eine Weboberfläche zur Konfiguration und Administration zur Verfügung stellt. Diese Weboberfläche kann unabhängig vom Hersteller im lokalen Netzwerk des Anwenders fungieren, oder aber auch in einem hybriden Modell mit Hersteller-Cloud interagieren und somit eine permanente Internetverbindung benötigen.[23] An dritter Stelle der kritischsten Schwachstellen nach OWASP stehen daher **unsichere Schnittstellen zum Ökosystem** (Insecure ecosystem interfaces), womit etwa unsichere Weboberflächen, Application Programming Interface (API)s, oder auch mobile Anwendungen gemeint sind.[24] Aufgrund der möglichen Komplexität von Webanwendungen, existiert allein für diesen Bestandteil von IoT-Geräten bereits eine Vielzahl von potenziellen Angriffsmöglichkeiten. Im Folgenden werden daher die 10 kritischsten Schwachstellen in Webanwendungen nach OWASP aufgelistet:[25]

- Broken Access Control (Fehler in der Zugriffskontrolle)
  - Eine der häufigsten und kritischsten Schwachstellen in Webapplikationen ist eine fehlerhafte Zugriffskontrolle, bei der ein Angreifer Kontrollmechanismen umgehen kann und somit Zugriff auf sensible Daten erhalten kann.[26]
- Cryptographic Failures (Kryptografische Fehler)

[19] Vgl. 12, S. 147.
[20] Vgl. 41.
[21] Vgl. 39, S. 80.
[22] Vgl. 39, S. 79.
[23] Vgl. 26, S. 14.
[24] Vgl. 41.
[25] Vgl. 42.
[26] Vgl. 42.

  - Fehler im Umgang mit Kryptografie können zum Abgreifen sensibler Daten führen. Es sollte daher stets sichergestellt werden, dass sensible Daten nicht im Klartext übermittelt werden und dass die kryptografischen Algorithmen, welche für die Verschlüsselung genutzt werden, stets aktuell und ausreichend sicher sind.[27]

- Injections
  - Durch fehlende Validierung von Benutzereingaben, können Angreifer versuchen, Befehle oder Schadcode in eine Webanwendung einzuschleusen und damit die Anwendung oder das darunterliegende System zu kompromittieren.[28]

- Insecure Design (Unsicheres Design)
  - Die Sicherheit von Webanwendungen sollte von Anfang an eine wichtige Rolle spielen und in jede Überlegung mit einbezogen werden. Denn Sicherheitslücken können nicht nur bei der Implementierung von Funktionen einer Anwendung entstehen, sondern Anwendungen können von vornerein unsicher gestaltet werden, sodass die Sicherheit auch bei einwandfreier Implementierung nicht gewährleistet werden kann.[29] Beispielhaft kann hier der Prozess zum zurücksetzen von Passwörtern mittels Sicherheitsfragen genannt werden, da Dritte die Antworten auf diese Fragen eventuell erlangen und sich somit Zugriff verschaffen könnten.[30]

- Security Misconfiguration (Sicherheitsrelevante Fehlkonfiguration)
  - Schwachstellen können nicht nur durch Fehler in Code und Design auftreten, sondern auch durch eine fehlerhafte Konfiguration. So besteht zum Beispiel die Gefahr, dass ungenutzte Funktionen einer Anwendung installiert und aktiviert sind, welche die Standardzugangsdaten des Herstellers verwenden.[31] Diese sind für Angreifer leicht herauszufinden und auszunutzen.

- Vulnerable and Outdated Components (Angreifbare und veraltete Komponenten)
  - Ein weiteres Risiko ist die Verwendung von veralteten Softwarekomponenten, welche bekannte Schwachstellen enthalten. Ein Angreifer könnte versuchen herauszufinden, welche Komponenten in einer Webanwendung verwendet werden und anschließend versuchen, bekannte Schwachstellen auszunutzen. Daher ist es notwendig, Komponenten regelmäßig zu aktualisieren und wenn möglich ein Schwachstellenmanagementsystem einzusetzen, welches alle Komponenten auf bekannte Schwachstellen überwacht.

- Identification and Authentication Failures (Identifikations- und Authentifizierungsfehler)

[27] Vgl. 43.
[28] Vgl. 44.
[29] Vgl. 45.
[30] Vgl. 45.
[31] Vgl. 46.

  - Schwachstellen im Bereich Identifikation und Authentifizierung können viele Ausprägungen haben. Dies kann die Verwendung schwacher oder häufig verwendeter Passwörter sein, oder auch das Fehlen eines Schutzmechanismusses gegen Brute Force-Angriffe.[32] Auch ein fehlerhaftes Verwalten von Benutzersitzungen und den zugehörigen Sitzungs-IDs gehört in diese Kategorie, ebenso wie die Speicherung von Passwörtern im Klartext, oder die Nutzung schwacher Hash-Verfahren.[33]

- Software and Data Integrity Failures (Software- und Datenintegritätsmängel)
  - Unter Software- und Datenintegritätsmängeln versteht man Schwachstellen, welche durch nicht-vertrauenswürdige Quellen verursacht werden.[34] Diese können unter anderem bei der Verwendung von Modulen oder Bibliotheken anderer Entwickler entstehen, sowie durch manipulierte Softwareupdates, welche ohne eine Integritätsprüfung eingespielt werden.[35]

- Security Logging and Monitoring Failures (Fehler bei Logging und Monitoring)
  - Ein unzureichendes Logging und Monitoring von Aktivitäten einer Webanwendung kann dazu führen, dass Angriffe weder erkannt noch nachträglich analysiert werden können.[36] Daher sollte neben dem Erstellen von aussagekräftigen Logdaten auch ein Benachrichtigungssystem implementiert werden, welches verdächtiges Verhalten erkennen und bei Bedarf Alarme auslösen kann.[37]

- Server-Side Request Forgery
  - Server-Side Request Forgery (SSRF) beschreibt das Verhalten, wenn eine Webanwendung entfernte Ressourcen auf Basis von Benutzereingaben abruft, ohne diese zu validieren.[38] Mögliche Angreifer können dieses Fehlverhalten ausnutzen, um externen Schadcode zu laden, oder sensible Daten von innerhalb der Webanwendung an ein vom Angreifer kontrolliertes Ziel zu senden.

4.3.3 Mobile Anwendungen

Es werden allerdings nicht ausschließlich Webanwendungen für die Steuerung von IoT-Geräten verwendet. Auch mobile Anwendungen für Smartphones werden von einigen Herstellern eingesetzt, um sich mit den Geräten zu verbinden.[39] Mobile Anwendungen können nativ für ein bestimmtes mobiles Betriebssystem entwickelt sein, oder aber mittels eines sogenannten Cross-Platform Frameworks, welches die Entwicklung mobiler Anwendungen für verschiedene Gerätetypen mittels einer einheitlichen Codebasis ermöglicht.[40]

[32] Vgl. 47.
[33] Vgl. 47.
[34] Vgl. 48.
[35] Vgl. 48.
[36] Vgl. 49.
[37] Vgl. 49.
[38] Vgl. 50.
[39] Vgl. 26, S. 17.
[40] Vgl. 26, S. 17 ff.

Auch für diesen Bereich stellt die Organisation OWASP eine Top 10 Liste zur Verfügung. In dieser Liste finden sich beispielsweise Schwachstellen wie die **unsachgemäße Verwendung von Anmeldedaten**, oder die **unsichere Authentifizierung oder Autorisierung**.[41] Des weiteren enthält die Liste Schwachstellen wie das **unzureichende Absichern von Binärdaten**, bei denen Angreifer sensible Daten aus den Installationsdateien von mobilen Anwendungen extrahieren können.[42] Aber auch andere bereits aus Webanwendungen bekannte Schwachstellen wie eine **sicherheitsrelevante Fehlkonfiguration** sind häufige Schwachstellen in mobilen Anwendungen.[43]

4.4 Netzwerk

Ein zentraler Bestandteil von IoT-Geräten ist die Netzwerkverbindung. Diese wird benötigt um mit dem Internet, untereinander, oder mit Steuerungskomponenten wie Backend oder sonstigen Anwendungen kommunizieren zu können. Je nach Gerätetyp und Anwendungszweck kann sich die Methode, welche zum Aufbau einer Netzwerkverbindung verwendet wird, unterscheiden. Die Verbindung kann kabellos oder kabelgebunden stattfinden.

Auf Platz zwei der kritischsten Schwachstellen für IoT-Geräte befindet sich laut OWASP die **Verwendung von unsicheren Netzwerkdiensten** (Insecure network services), womit die Verwendung von nicht benötigten Diensten auf dem Gerät gemeint ist, welche dazu führen können, dass ein Angreifer entweder Daten abgreifen und manipulieren oder einen Fernzugriff auf das Gerät etablieren könnte.[44]

4.5 Cloud

Im Bereich des IoT wird Cloud-Computing vorwiegend dafür genutzt, um große Mengen an Daten von IoT-Geräten zentral zu sammeln und zu verarbeiten.[45] Dies sorgt dafür, dass IoT-Geräte nur wenig internen Speicherplatz mitbringen müssen und der Speicherplatz bei Bedarf in der Cloud an größere Datenmengen angepasst werden kann, sollte sich dafür ein Bedarf aufzeigen. Darüber hinaus können über Cloud-Dienste nicht nur Daten von IoT-Geräten erfasst werden, sondern selbige können auch Daten aus der Cloud entgegennehmen und auf entfernte Anweisungen reagieren.[46] Dies spart neben lokalem Speicherbedarf auch Rechenleistung und ermöglicht es, Geräte in geringer physischer Größe kostengünstig herzustellen. Allerdings bringt die Verwendung von IoT-Geräten in Verbindung mit Cloud-Diensten auch Risiken mit sich, die es zu beachten gilt. So muss neben einer funktionierenden Identitätsverwaltung oder Zugriffskontrolle auch die Sicherheit der gesamten Cloudinfrastruktur und der

[41] Vgl. 51.
[42] Vgl. 51.
[43] Vgl. 51.
[44] Vgl. 41.
[45] Vgl. 52, S. 14.
[46] Vgl. 52, S. 14.

Kommunikation mit dieser sichergestellt werden.[47] Da die Cloud-Dienste aber nicht immer in Eigenleistung betrieben werden, muss man entweder dem Anbieter vertrauen, oder aber die Sicherheit regelmäßig auditieren.

Um Cloud-Dienste effektiv auditieren zu können, ist es notwendig die kritischsten Schwachstellen im Cloud-Umfeld zu kennen und diese in die Untersuchung mit einzubeziehen. Da das Cloud-Umfeld ein eigenes komplexes Gebiet mit verschiedenen Schwachstellen und Risiken ist, werden im Folgenden die 10 kritischsten Schwachstellen im Cloud-Umfeld laut OWASP erläutert:[48]

- Insecure cloud, container or orchestration configuration (Unsichere Konfiguration von Cloud oder Container-Orchestration)
  - Unsichere Konfigurationen von Cloud-Komponenten können Risiken verursachen. So können öffentlich zugängliche Speicher oder nicht ordnungsgemäß gesetzte Berechtigungen Außenstehenden Zugriff auf interne Daten ermöglichen.[49] Bei Verwendung von Container-Technologien wie Docker, sollten zudem die Berechtigungen der Container geprüft und so restriktiv wie möglich gesetzt werden. Container die mit root-Rechten laufen und sich Ressourcen wie Netzwerk-Schnittstellen mit dem Host-System teilen, können einem Angreifer bei Kompromittierung Zugriff auf das darunterliegende Host-System ermöglichen.[50]
- Injection flaws (Injektionen)
  - Genau wie bei Webanwendungen besteht auch bei Cloud-Anwendungen die Gefahr, dass Angreifer versuchen, Schadcode oder Befehle wie beispielsweise Datenbankabfragen unerwünscht in das System einzuschleusen und so den Weiterbetrieb der Anwendung zu gefährden oder Daten zu entwenden.[51] Es sollte daher stets geprüft werden, ob geeignete Maßnahmen ergriffen wurden, um solche Angriffe zu verhindern.
- Improper authentication & authorization (Unsichere Authentifizierung und Autorisierung)
  - Ein weiteres Risiko in Cloud-Umgebungen ist der unbefugte Zugriff durch Dritte. Dies kann über API-Endpunkte im Backend geschehen, welche keine Authentifizierung erfordern, oder wenn Benutzerrollen mehr Rechte enthalten, als für ihren Einsatzzweck erforderlich sind.[52]
- CI/CD pipeline & software supply chain flaws (Fehler in CI/CD Pipelines und Softwarelieferketten)

[47] Vgl. 52, S. 14.
[48] Vgl. 53.
[49] Vgl. 53.
[50] Vgl. 53.
[51] Vgl. 53.
[52] Vgl. 53.

  - Bei der automatischen Bereitstellung von Software aus dem Code heraus können weitere Risiken entstehen. So kann neben unzureichender Authentifizierung bei Continuous Integration/Continuous Delivery (CI/CD)-Systemen auch die Verwendung veralteter Software und Container-Abbilder ein Risiko darstellen.[53] Werden in einer CI/CD-Pipeline zur Bereitstellung von Software innerhalb von Containern beispielsweise veraltete Abbilder verwendet, so sollten auch die zugrundeliegenden Basisabbilder regelmäßig aktualisiert werden.

- Insecure secrets storage (Unsicheres Speichern von Geheimnissen)
  - Cloud-Dienste und Anwendungen müssen häufig miteinander kommunizieren und sich untereinander authentifizieren, hierfür werden in der Regel API-Keys oder Passwörter verwendet. Sollten diese Geheimnisse nicht ordnungsgemäß verwahrt werden, oder sind diese gar im Klartext im System selbst abgespeichert, so stellt dies eine Schwachstelle dar, die von Angreifern ausgenutzt werden kann.[54] Es sollte daher geprüft werden, ob alle Authentifizierungsdaten sicher und vor unbefugten Zugriffen geschützt aufbewahrt werden.

- Over-permissive or insecure network policies (Unsichere Netzwerkrichtlinien)
  - Eine weitere Gefahr stellen unsichere Netzwerkverbindungen dar. Cloud-Dienste werden häufig als Microservices realisiert, bei denen eine Anwendung in einzelne Funktionen unterteilt werden und jede Funktion einen eigenständig lauffähigen Dienst darstellt. Sollten interne Dienste von außen erreichbar sein oder können Dienste miteinander kommunizieren, obwohl dafür keine Notwendigkeit besteht, so kann ein Angreifer versuchen dies auszunutzen.[55] Daher sollte sichergestellt sein, dass neben einer restriktiven und verschlüsselten Netzwerkkommunikation auch der Netzwerkverkehr auf maliziöse Aktivitäten überwacht wird, um Angriffsversuche frühzeitig zu erkennen.[56]

- Using components with known vulnerabilities (Verwendung von Komponenten mit bekannten Schwachstellen)
  - Ein weitere Gefahr ist die Verwendung von Softwarekomponenten mit veralteter Software, welche bekannte Schwachstellen enthält. Hier sind insbesondere Softwarepakete von Drittanbietern oder Container-Images betroffen, welche nicht regelmäßig aktualisiert werden.[57] Die Verwendung eines Schwachstellenmanagementsystems, welches automatisch alle Komponenten überwacht und über Schwachstellen benachrichtigt, kann hier ebenso hilfreich sein wie regelmäßige Aktualisierungen der verwendeten Komponenten.

- Improper assets management (Unsachgemäßes Asset-Management)

[53] Vgl. 53.
[54] Vgl. 53.
[55] Vgl. 53.
[56] Vgl. 53.
[57] Vgl. 53.

  - Selbst eine unzureichende Dokumentation kann Risiken hervorrufen. Beispielsweise können undokumentierte Microservices, oder API-Endpunkte von Angreifern entdeckt und ausgenutzt werden.[58] Auch kann eine mangelhafte Dokumentation dazu führen, dass nicht mehr genutzt Dienste vergessen werden und mit vorhandenen Schwachstellen weiterlaufen und für potenzielle Angreifer erreichbar sind.[59]

- Inadequate 'compute' resource quota limits (Unzureichende Limitierung von Rechenleistung)
  - Auch eine fehlende oder unzureichende Limitierung von Rechenleistung und Ressourcen stellt ein Risiko dar. Angreifer könnten beispielsweise versuchen durch eine große Anzahl an Anfragen an API-Endpunkte einen Denial of Service (DoS) herbeizurufen, oder einen Container derart stark zu belasten, dass dieser die gesamte Rechenleistung des Host-Systems beansprucht und dieses damit verlangsamt oder unbenutzbar macht.[60]

- Ineffective logging & monitoring (Ineffektive Protokollierung und Überwachung)
  - Das Ausnutzen von fehlender Limitierung von Ressourcen kann durch zielgerichtetes Logging von Container- und Prozessaktivitäten frühzeitig erkannt werden.[61] Fehlendes oder unzureichendes Logging kann demnach ein Risiko darstellen, da Angriffe eventuell weder erkannt noch analysiert werden können.

Da sich IoT-Systeme stetig weiter entwickeln, und aus unterschiedlichen Komponenten wie mobilen Apps, Cloud-Diensten oder Webanwendungen mit einer Vielzahl an eigenen Schwachstellen bestehen, verändern sich auch die möglichen Schwachstellen im Bereich IoT permanent. Projekte wie die Top 10 Listen von OWASP sollten daher stets im Auge behalten werden, um auf dem neusten Stand zu bleiben.

[58] Vgl. 53.
[59] Vgl. 53.
[60] Vgl. 53.
[61] Vgl. 53.

5 Durchführung von Penetrationstests an IoT-Geräten

Das folgende Kapitel setzt das zuvor erläuterte theoretische Wissen in die Praxis um und bildet die Grundlage für die Entwicklung einer Methodik für Penetrationstests, die im nachfolgenden Kapitel weiter ausgeführt wird. Durch die Durchführung von Penetrationstests an alltäglichen IoT-Geräten und Testumgebungen werden die in vorherigen Kapiteln erlernten Konzepte praktisch umgesetzt, wobei der Fokus auf die Identifikation von Schwachstellen in den Bereichen Hardware, Software (einschließlich Firmware, Web- und Mobile-Anwendungen), Netzwerk und Cloud gelegt wird. Die hierbei vermittelten Kenntnisse sind wichtig, um ein umfassendes Verständnis für die spezifischen Sicherheitsrisiken in jedem Bereich zu erlangen und liefern zudem wichtige Erkenntnisse für die spätere Gestaltung einer effektiven Penetrationstestmethodik.

5.1 Hardware

IoT-Geräte bestehen aus einer Kombination aus Hardware und Software. Die Hardware ist dabei nicht immer standardisiert und kann sich von Gerät zu Gerät stark unterscheiden. Zudem befindet sich die Hardware oftmals nicht in vor Zugriffen Dritter geschützten Umgebungen wie Rechenzentren. Aus diesem Grund sollte nach Möglichkeit bei einem Penetrationstest auch die Hardware auf Schwachstellen oder Möglichkeiten zur Manipulation durch Dritte untersucht werden. Im Folgenden wird erläutert, auf was dabei geachtet werden sollte und welche gängigen Methoden zur Untersuchung von IoT-Geräten auf Hardware-Ebene existieren und wie diese durchgeführt werden. Einige der Methoden werden dabei an haushaltsüblichen IoT-Geräten aus dem Bereich Smart Home demonstriert.

Ein erster Indikator für den Grad der Sicherheit eines Gerätes können die Verpackung und das Gehäuse sein. Sollte sich das Gehäuse leicht öffnen und Bauteile gut zugänglich sein, kann dies entweder ein Indikator für gute Reparierbarkeit sein oder dafür, dass der Hersteller wenig Aufwand bei der Absicherung des Gerätes betrieben hat.[1] Des weiteren sollte die Verpackung auf Parameter wie MAC-Adressen untersucht werden, welche sich später bei der Untersuchung des Netzwerkverkehrs wiederfinden könnten und von einem Angreifer verwendet werden können. Auch wäre es denkbar, dass Angreifer sich MAC-Adressen von unverkauften Geräten in Ladengeschäften notieren und diese Geräte zu einem späteren Zeitpunkt mittels spezieller IoT-Suchmaschinen aufzuspüren und anzugreifen. Die nachfolgende Abbildung 5.1 demonstriert die Suche nach einer MAC-Adresse einer

[1] Vgl. 12, S. 84 f.

Überwachungskamera mit der IoT-Suchmaschine *shodan.io*, welche neben der IP-Adresse und dem Hostnamen der Kamera noch weitere Informationen ausgibt.

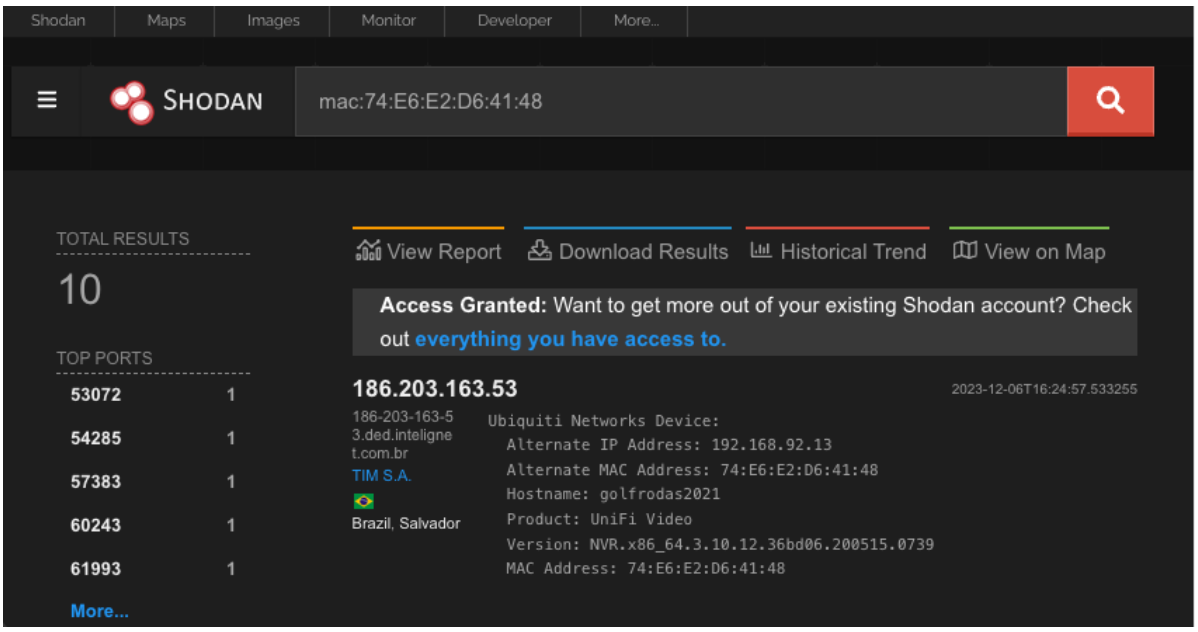

Abbildung 5.1: Suche nach einer MAC-Adresse mittels der IoT-Suchmaschine Shodan.io (Eigene Darstellung)

Indikatoren für eine gute physische Sicherheit des Gehäuses kann die Verwendung von unüblichen Spezialschrauben, oder Schrauben die sich durch abfallende Griffkanten der Schraubköpfe nur in eine Richtung drehen lassen.[2] Weitere Indikatoren können starke Verklebungen oder Verschweißungen sein, die ein Öffnen des Gehäuses erschweren oder verhindern sollen, indem das Gerät beim Öffnen zerstört wird.[3] Bei besonders schutzbedürftiger Hardware können Hersteller auch holographische Siegel auf der Hardware anbringen, die bei Öffnung des Gehäuses zerstört werden und damit eine mögliche Manipulation des Gerätes offenbaren können.[4] Verblendungen sind eine weitere Maßnahme von Herstellern, die eine Analyse der Komponenten erschweren sollen. Hierbei können physische Blenden oder Epoxyd-Harz auf den Bauteilen aufgebracht werden, um den Zugriff zu erschweren, oder die Bauteilebezeichnung wird unkenntlich gemacht, damit eine möglichst aufwändige Recherche notwendig ist, um Informationen über die Bauteile zu erhalten.[5]

Zusätzlich zu einer optischen Untersuchung kann die Hardware von IoT-Geräten daher auch auf technischer Ebene untersucht werden, um Informationen offenzulegen. Eine gängige Methode ist es, eine Verbindung mit Debugging-Schnittstellen wie UART herzustellen. Auf diese Weise können die Informationen, welche das Gerät beim Hochfahren ausgibt, auf

[2] Vgl. 12, S. 86.
[3] Vgl. 12, S. 86 f.
[4] Vgl. 12, S. 87 f.
[5] Vgl. 12, S. 89 f.

einem Konsolenfenster mitgelesen werden. Die Abbildung 5.2 zeigt eine solche Verbindung. Zu erkennen ist die Hauptplatine einer Philips Hue Bridge, welche zur Verwaltung von Smart-Home-Geräten verwendet wird. Die Platine ist an der UART-Schnittstelle mit einem UART zu USB Wandler verbunden. Dieser Wandler ermöglicht es die Informationen aus dem Startprozess der Hue Bridge auf einem Konsolenfenster anzuzeigen.

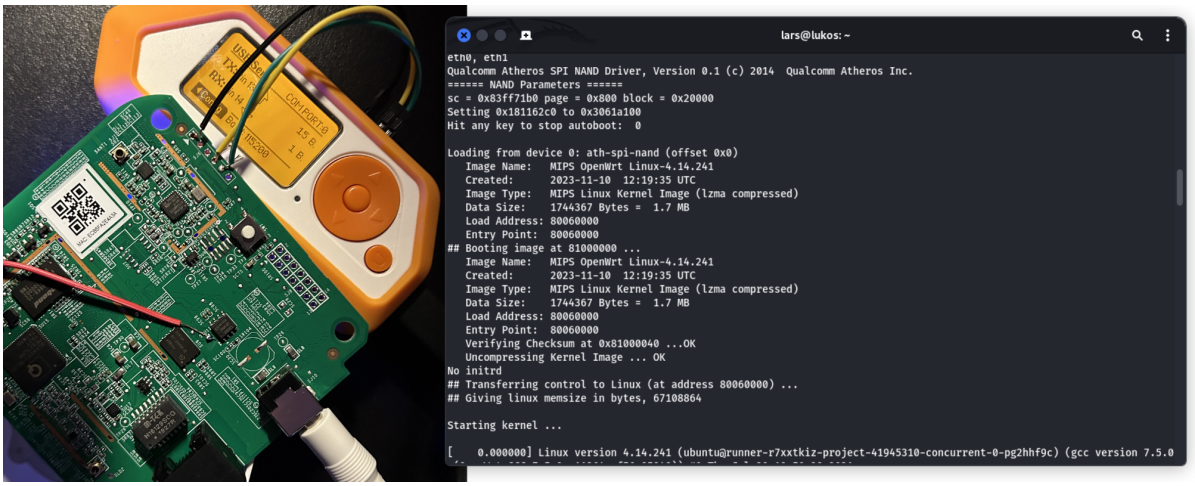

Abbildung 5.2: Auslesen von Systeminformationen einer Philips Hue Bridge mittels USB- zu UART-Verbindung (Eigene Darstellung)

In dem Konsolenfenster sind viele Details des Gerätes zu erkennen, welche sich mit anderen Methoden wie einer äußeren Sichtprüfung nicht herausfinden lassen. So ist dort etwa das vom Gerät verwendete Betriebssystem *MIPS OpenWRT Linux-4.14.241* zu erkennen, welches ein verbreitetes quelloffenes Linux-Betriebssystem ist und überwiegend in Routern zum Einsatz kommt. Des weiteren sind verwendete Treiber und deren Version ersichtlich.

Das gleiche Vorgehen lässt sich auch bei einer IP-Kamera des Herstellers *ieGeek* durchführen. In Abbildung 5.3 ist wieder ein Aufbau zu sehen, bei dem die UART-Schnittstelle der Kamera über einen Wandler mit dem USB-Anschluss eines Computers verbunden ist und die Darstellung von Informationen auf einem Konsolenfenster ermöglicht. Zu erkennen sind wieder verschiedene Informationen zur verbauten Hardware, sowie die verwendete Version des Linux Betriebssystems.

Abbildung 5.3: Auslesen von Systeminformationen einer IP-Kamera mittels USB zu UART Verbindung (Eigene Darstellung)

Schnittstellen wie UART sind allerdings nicht immer ohne weiteres auf den Platinen der zu untersuchenden Geräte erkennbar. Die Verbindungskontakte sind rein visuell nicht immer identifizierbar und falls doch, oftmals nicht eindeutig beschriftet. In diesem Fall ist es notwendig, die einzelnen Kontakte auf der Platine unter Einsatz eines Multimeters zu untersuchen. Im Idealfall kann man einen geeigneten Kandidaten für die UART-Schnittstelle mit bloßem Auge an vier nebeneinander liegenden Kontakten bestimmen.[6] Die vier Kontakte für UART bestehen aus Transmit (TX), welcher Daten vom Gerät weg übertragt, Receive (RX), welcher Daten von einem anderen Gerät entgegennimmt, sowie Ground (GND) und Voltage at the common collector (V_{CC}).[7] Falls diese Kontakte nicht bereits beschriftet sind, können sie unter Einsatz eines Multimeters bestimmt werden.[8]

Um den Erdungsreferenzkontakt (GND) zu finden, wird das Multimeter auf den Modus zur Durchgangsprüfung gestellt und die schwarze Messleitung an einen Erdungspunkt angelegt und mit der roten Messleitung werden alle Kontakte geprüft, bis ein akustisches Signal ertönt.[9] Zum identifizieren des Kontakts für die Versorgungsspannung (V_{CC}) wird das Multimeter auf den Modus zum Messen der Stromspannung eingestellt und wie zuvor alle Kontakte geprüft, bis einer der Kontakte eine Konstante Spannung von 3.3V oder 5V aufweist.[10] Für gewöhnlich gibt es zwei Kontakte die eine solche Spannung aufweisen. Daher werden beide Kontakte direkt nach einem Neustart des Gerätes noch einmal gemessen, wobei einer der Kontakte Spannungsschwankungen aufweisen sollte. Dieser Kontakt ist für das Senden von Daten verantwortlich (TX).[11] Nach dem Ausschlussprinzip ist der verbleibende Kontakt derjenige für das Empfangen von Daten und sollte zudem die geringste Spannung

[6] Vgl. 54, S. 66 f.
[7] Vgl. 54, S. 69.
[8] Vgl. 54, S. 70.
[9] Vgl. 54, S. 71.
[10] Vgl. 54, S. 72.
[11] Vgl. 54, S. 72.

aller Kontakte aufweisen.[12]

Um die UART-Kontakte der Platine mit einem Computer zu verbinden, wird ein USB- zu UART-Adapter benötigt. Dort werden die UART Kontakte RX, TX und GND angeschlossen. Zu beachten ist hierbei, dass der TX-Kontakt der Platine an den RX Kontakt des USB zu UART Adapters und der RX Kontakt der Platine an den TX Kontakt des USB zu UART Adapters angeschlossen wird.[13] Im Anschluss muss noch die Symbolrate (baud rate) des zu testenden Gerätes bestimmt werden.[14] Dies kann etwa durch Ausprobieren gängiger Symbolraten geschehen. Eine weit verbreitete Symbolrate ist 115200.[15] Alternativ können auch Programme oder Skripte zur Hilfe genommen werden, welche in der Lage sind, die Symbolrate automatisch zu identifizieren.[16] Sobald die Symbolrate identifiziert ist, lässt sich mit dem Linux-Programm *Screen* mit dem Befehl *sudo screen /dev/<usb-anschluss> <symbolrate>* oder mit dem Windows-Programm *Putty* eine Kommandozeilenverbindung mit dem Gerät herstellen, um mit dem Gerät zu interagieren und Informationen auszulesen.[17]

5.2 Software

Nicht nur die Hardware eines IoT-Gerätes sollte bei einem Penetrationstest untersucht werden, sondern auch die Software. Aufgrund unterschiedlichster Technologien und Komplexität der verwendeten Software, welche bei IoT-Geräten zum Einsatz kommen, sollte dies möglichst strukturiert geschehen, um alle Bereiche bestmöglich abzudecken. Im Folgenden werden daher die unterschiedlichen Kategorien von Software bei IoT-Geräten beschrieben und worauf bei einem Penetrationstest geachtet werden sollte. Zudem werden einige gängige Methoden zum Untersuchen von Schwachstellen praxisnah an Beispielen erläutert.

5.2.1 Firmware

Firmware-Extraktion beschreibt den Vorgang, die Firmware von einem IoT-Gerät zu extrahieren und einer tieferen Analyse zu unterziehen. Dies soll verwendete Softwares und deren Versionsstand offenlegen sowie ein generelles Verständnis dafür schaffen, wie ein Gerät im Detail arbeitet. Zudem können so eventuell im Klartext abgelegte Kennwörter oder Kommunikationsschlüssel gefunden werden, sowie nützliche Informationen wie API-Endpunkte und Konfigurationsdateien.[18] Aus diesen Gründen ist eine Analyse der Firmware ein wichtiger Bestandteil von IoT-Penetrationstests. Allerdings birgt der Versuch einer Firmware-Extraktion auch die Gefahr das Gerät unbrauchbar zu machen, da einige Geräte eine sogenannte Read-out-Protection besitzen, welche ein Auslesen verhindern sollen und bei Umgehung dafür

[12] Vgl. 54, S. 72.
[13] Vgl. 54, S. 75.
[14] Vgl. 54, S. 75.
[15] Vgl. 12, S. 166.
[16] Vgl. 12, S. 166.
[17] Vgl. 54, S. 75.
[18] Vgl. 26, S. 74.

sorgen, dass alle Speicher des Gerätes gelöscht werden.[19] Um an die Firmware eines Gerätes zu gelangen, ist nicht immer eine Extraktion direkt vom Gerät notwendig. In vielen Fällen ist es möglich, die Firmware direkt von der Webseite des Herstellers zu beziehen und für eine Analyse auf den eigenen Computer herunterzuladen.[20] Sollte ein direkter Bezug über die Webseite des Herstellers nicht möglich sein, so kann man versuchen den Datenverkehr während eines Softwareupdates als sogenannter Man-in-the-middle (MITM) abzufangen, oder herauszufinden von welchem Endpunkt das Gerät Softwareupdates herunterlädt und die Firmware von diesem Endpunkt beziehen.[21] Sollte es nicht möglich sein die Firmware über das Internet oder Abfangen des Datenverkehrs zu beziehen, kann die Firmware auch über Debugging-Schnittstellen wie UART, Serial Peripheral Interface (SPI) oder Joint Test Action Group (JTAG) direkt vom Gerät extrahiert werden.[22]

Nach erfolgreichem Erhalten der Firmware, kann mit der **Firmware-Analyse** begonnen werden. Sollte die Firmware in einem Archivformat wie GZip, Tar, oder ZIP vorliegen so muss diese zunächst entpackt werden.[23] Mit der weit verbreiteten Software *Binwalk*, lassen sich Firmware-Images automatisiert entpacken und anschließend analysieren.[24] Bei der Analyse von Firmware wird zwischen der **statischen Analyse**, bei der man in der entpackten Firmware nach Auffälligkeiten wie im Klartext abgespeicherten Passwörtern oder Schlüsseln, Konfigurationsdateien oder API-Endpunkten sucht[25] und der **dynamischen Analyse** unterschieden, bei der man die Firmware im laufenden Betrieb auf dem Gerät selbst, oder mittels Emulation untersucht.[26] Eine einfache und oberflächliche statische Analyse lässt sich mittels Programmen wie *firmwalker* durchführen, welches innerhalb eines entpackten Dateisystems selbständig nach Zugangsdaten, Sicherheitsschlüsseln, Konfigurationsdaten, Passwörtern und weiteren für eine Sicherheitsanalyse relevanten Daten sucht.[27]

5.2.2 Webanwendungen

Viele IoT-Geräte oder Dienste, die mit IoT-Geräten in Verbindung stehen, nutzen zur Steuerung oder Verwaltung Webanwendungen, auf die mittels Webbrowser zugegriffen werden kann. Webanwendungen basieren auf einem Client-Server-Modell, bei dem das sogenannte Frontend im Browser als Client dient und die Anwendungslogik über das Backend der Webanwendung auf einem entfernten Server läuft.[28] Die Kommunikation zwischen Client und Server erfolgt über Protokolle wie HTTP(S) und kann mittels Scriptsprachen wie JavaScript auch asynchron erfolgen und bei Bedarf Daten vom Server nachladen.[29] Aufgrund der

[19] Vgl. 12, S. 105.
[20] Vgl. 26, S. 76 ff.
[21] Vgl. 26, S. 79 f.
[22] Vgl. 26, S. 84.
[23] Vgl. 12, S. 138.
[24] Vgl. 12, S. 142.
[25] Vgl. 12, S. 145 ff.
[26] Vgl. 12, S. 145 ff.
[27] Vgl. 12, S. 149.
[28] Vgl. 27, S. 731.
[29] Vgl. 27, S. 731.

vielen unterschiedlichen Technologien, aus denen Webanwendungen bestehen können, ist ein Penetrationstest von Webanwendungen komplex und kann viele Schwachstellen zum Vorschein bringen. Auf die gängigsten Schwachstellen und wie man diese findet, wird im Folgenden eingegangen. Da es für einen Penetrationstest gewöhnlich keine Rolle spielt, ob die Anwendung auf einem IoT-Gerät läuft oder nicht und da es schwierig ist alle Schwachstellen über IoT-Geräte abzudecken, werden für Demonstrationszwecke die Anwendungen DVWA und OWASP Juice Shop verwendet, welche absichtlich gängige Schwachstellen erhalten.

Gibt es innerhalb einer Webanwendung **Fehler in der Zugriffskontrolle (Broken Access Control)**, so ist es möglich, dass ein Benutzer auf die Daten eines anderen Nutzers zugreifen kann, ohne dass dies in der Anwendungslogik so vorgesehen ist. Demonstrieren lässt sich diese Schwachstelle in OWASP Juice Shop, der Nachbildung eines Onlineshops mit verschiedensten Schwachstellen. Um Webanwendungen effektiv auf Schwachstellen zu untersuchen, ist es hilfreich, einen sogenannten *Interception Proxy* wie beispielsweise *Burp Suite* zu verwenden. Dieser ist in der Lage dazu den Netzwerkverkehr zwischen Webbrowser und Server abzufangen, zu analysieren und zu manipulieren. Die Abbildung 5.4 zeigt den Warenkorb des derzeit in OWASP Juice Shop angemeldeten Benutzers. Der dort angemeldete Benutzer ist der Administrator und hat die Benutzer-ID 1.

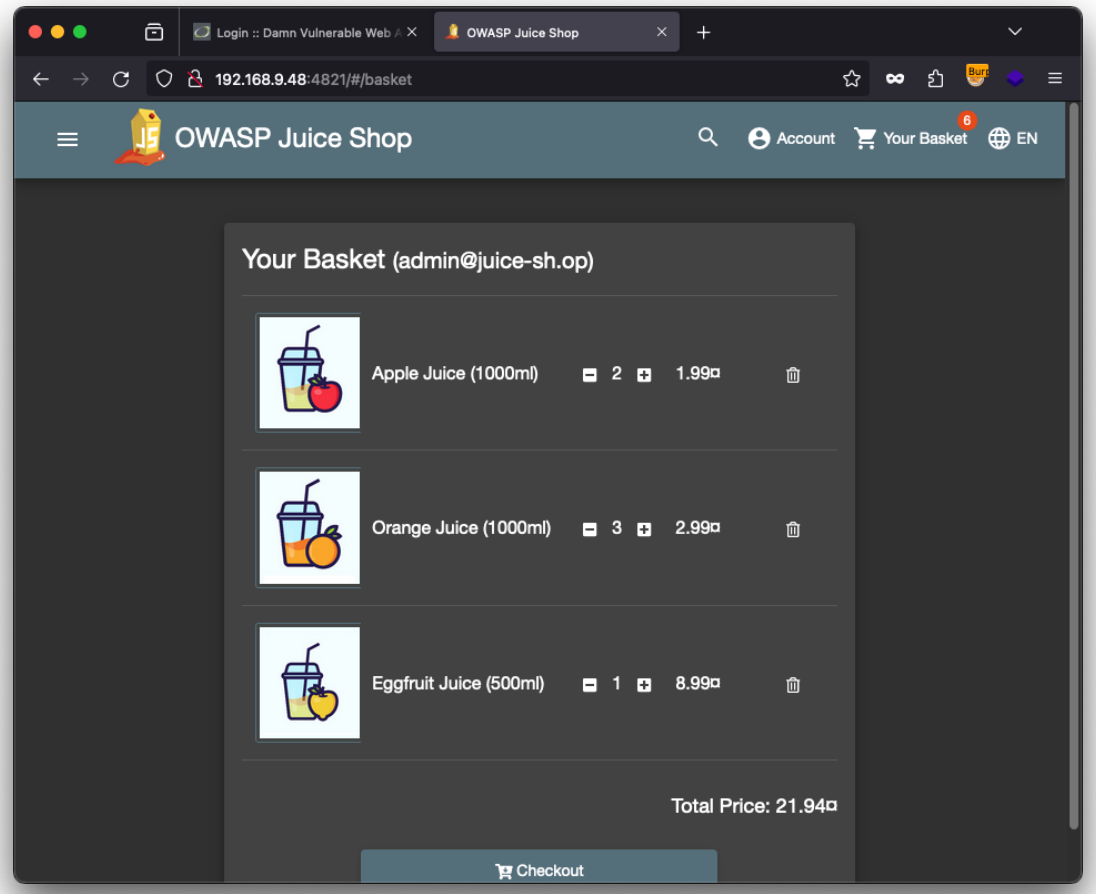

Abbildung 5.4: Warenkorb des Benutzers mit der Benutzer-ID 1 des fiktiven Onlineshops OWASP Juice Shop (Eigene Darstellung)

Fängt man den Datenverkehr des Browsers beim Aufrufen des Warenkorbes mittels Burp Suite ab, so wird wie in Abbildung 5.5 zu sehen ein GET-Request angezeigt, welcher an den Endpunkt /rest/basket/1 verweist und vermutlich über eine Rest-API den Warenkorb von Benutzer 1 anfordert.

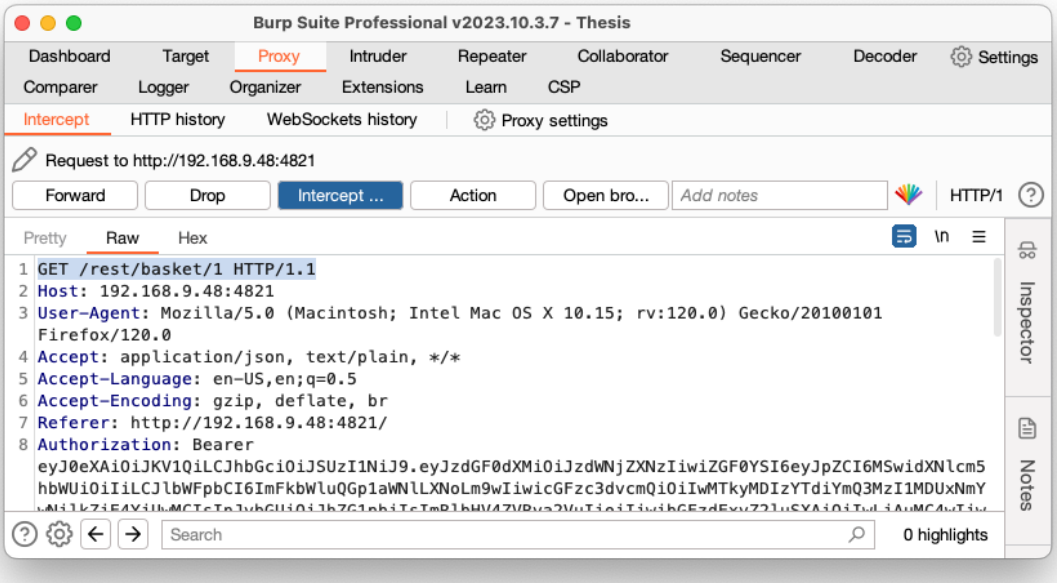

Abbildung 5.5: Abgefangener GET-Request in der Software Burp Suite (Eigene Darstellung)

Wird nun die ID am Ende des Requests von 1 auf 2 geändert und der Request anschließend an das Backend weitergeleitet, so wird als Ergebnis der Warenkorb des Benutzers mit der ID 2, wie in Abbildung 5.6 zu sehen ist zurückgemeldet.

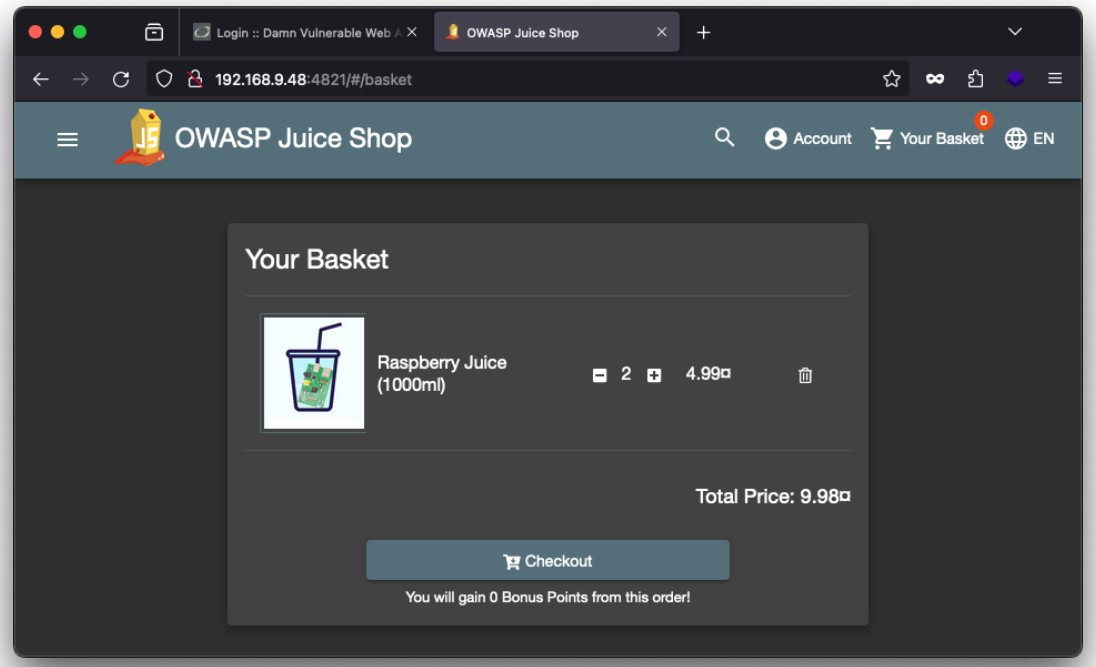

Abbildung 5.6: Warenkorb des Benutzers mit der Benutzer-ID 2 des fiktiven Onlineshops OWASP Juice Shop (Eigene Darstellung)

Es ist also möglich, ohne die Zugangsdaten von Benutzer 2 zu kennen, Informationen aus dessen Benutzerkonto einzusehen. Dies ist eindeutig ein Fehler in der Zugriffskontrolle und damit eine Schwachstelle in der Webanwendung.

Injections sind eine der häufigsten Schwachstellen in Webanwendungen. Mittels Injections lässt sich schädlicher Code wie beispielsweise JavaScript in eine Webanwendungen einschleusen, oder es können etwa SQL-Befehle bei einer SQL-Injection ausgeführt werden.

```
qwert')) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9'
    FROM Users--
```

SQL-Injection Payload [55]

Als Ergebnis dieser Abfrage erhält man eine Liste aller Benutzer innerhalb der Datenbank inklusive deren Passwörter in Form eines MD5-Hashs.

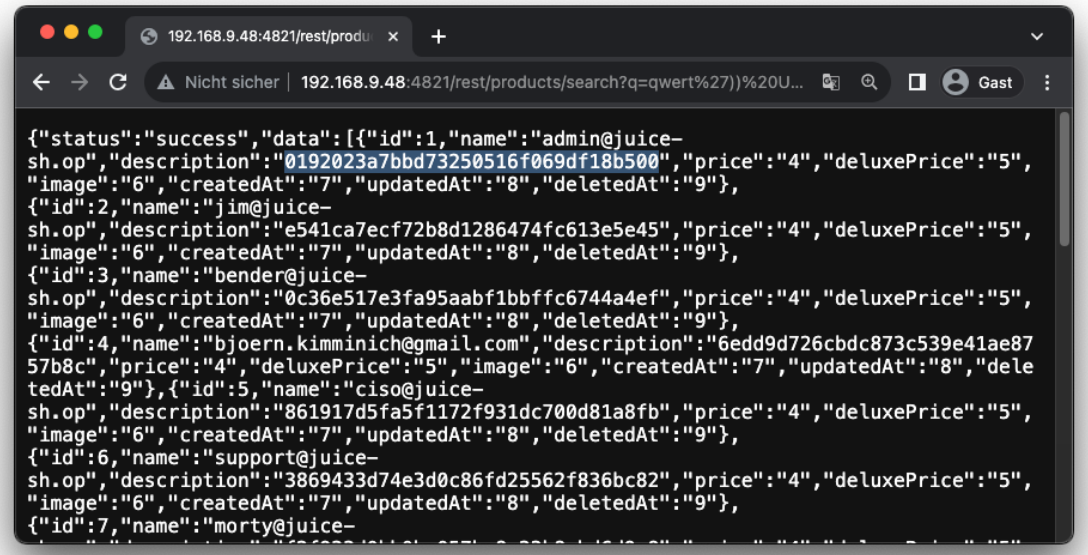

Abbildung 5.7: Erfolgreiche SQL-Injection in OWASP Juice Shop auf dem Endpunkt der Produktsuche (Eigene Darstellung)

Die angezeigten MD5-Hashes führen zu einer weiteren Schwachstelle in Webanwendungen. Den sogenannten **Cryptographic Failures (Kryptographische Fehler)**. Bei dieser Schwachstelle werden Fehler bei der Implementierung kryptographischer Verfahren begannen, oder schwache Kryptographie verwendet. Hash-Verfahren werden unter anderem dafür genutzt, Passwörter nicht im Klartext abspeichern zu müssen, sondern mittels einer Einwegfunktion einen Hash des Passwortes zu erzeugen, welcher sich nicht zum ursprünglichen Passwort zurückrechnen lässt.[30] Übergibt man die Hash-Werte aus Abbildung 5.7 an ein Programm zum Knacken (cracken) von Passwörtern wie beispielsweise JohnTheRipper oder Hashcat, so erhält man innerhalb von kürzester Zeit eine Klartextversion des Passwortes. Der Befehl zum knacken eines MD5-Hashes in JohnTheRipper lautet:

```
1 "john --format=raw-md5 --wordlist=[path_to_wordlist] 0192023
    a7bbd73250516f069df18b500"
```

Im Fall des Benutzers *admin@juice-sh.op* lautet das Passwort im Klartext: *admin123*. Schwache Algorithmen, die nicht mehr als ausreichend sicher gelten und von deren Verwendung abgesehen werden sollte, sind beispielsweise MD2, MD4, MD5 und SHA-1.[31]

Ein weiteres Beispiel für kryptografische Fehler lässt sich anhand einer Überwachungskamera des Herstellers *ieGeek* erläutern. Bei dieser Kamera wird für die Weboberfläche das

[30] Vgl. 9, S. 240 f.
[31] Vgl. 9, S. 241 f.

HTTP-Protokoll anstelle des HTTPS-Protokolls verwendet. Für die Authentifizierung des Benutzers an der Weboberfläche wird Basic-Auth verwendet. Bei Basic-Auth werden Benutzername und Passwort mit Base64 kodiert und lassen sich daher leicht dekodieren. Nachfolgend ist ein GET-Request zu sehen, welcher die Authentifizierung an der Weboberfläche zeigt und mittels des interception-Proxies Burp Suite abgefangen wurde.

```
GET /cgi-bin/hi3510/param.cgi?cmd=getlanguage HTTP/1.1
Host: 192.168.24.73
Authorization: Basic YWRtaW46YWRtaW4=
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
    KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36
Accept: */*
Referer: http://192.168.24.73/
Accept-Encoding: gzip, deflate, br
Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

Authentifizierung an der Weboberfläche der Kamera mittels Basic-Auth

Dort ist die in Base64 kodierte Zeichenkette *YWRtaW46YWRtaW4=* zu erkennen. Dekodiert lautet diese Zeichenkette *admin:admin* und ergibt die Standardlogindaten, welche von Werk eingestellt sind. Diese Standardlogindaten sind zudem auf einem Aufkleber auf dem Außengehäuse der Kamera angebracht. Diese Logindaten stellen ein Sicherheitsrisiko dar, da sie nicht nur außen auf dem Gehäuse angebracht sind, sondern auch leicht zu erraten sind und sollten umgehen geändert werden. Ein Mitlesen des Datenverkehrs mittels Burp Suite macht allerdings deutlich, dass auch die neuen Zugangsdaten im Klartext übertragen werden. Im Folgenden ist ein abgefangener POST-Request zu sehen, welcher der Kamera die neuen Zugangsdaten mitteilt.

```
POST /web/cgi-bin/hi3510/param.cgi HTTP/1.1
Host: 192.168.24.73
Content-Length: 266
Cache-Control: max-age=0
Authorization: Basic YWRtaW46YWRtaW4=
Upgrade-Insecure-Requests: 1
Origin: http://192.168.24.73
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
    KHTML, like Gecko) Chrome/120.0.6099.71 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
    image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.24.73/web/user.html
Accept-Encoding: gzip, deflate, br
```

```
13 Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
14 Connection: close
15
16 cmd=setuserattr&cururl=http%3A%2F%2F192.168.24.73%2Fweb%2Fuser.html&-
     at_username=admin&-at_newname=lars&-at_password=password&cmd=
     setuserattr&-at_username=user&-at_newname=user&-at_password=user&cmd=
     setuserattr&-at_username=guest&-at_newname=guest&-at_password=guest
```

Verändern des Standardpasswortes der Kamera

Insecure Design (Unsicheres Design) - Schwachstellen können sehr vielseitig sein und lassen sich am besten durch intensives Nutzen und Untersuchen der Anwendung feststellen. Währenddessen ist es sinnvoll, ausführliches Threat Modeling zu betreiben, um nicht nur Schwachstellen rein technischer Natur festzustellen, sondern auch Schwachstellen in der Architektur und Logik der Anwendung.[32] Threat Modeling ist eine Methode die Architektur einer Anwendung zu analysieren und dabei strukturiert Sicherheitsrisiken zu identifizieren und zu bewerten.[33] Zudem werden für Angreifer attraktive Ziele identifiziert sowie die Applikationslogik inklusive des Datenflusses modelliert, um somit mögliche Bedrohungen aufzudecken.[34] Basierend auf den daraus gewonnenen Informationen lassen sich anschließend praktische Penetrationstest-Methoden anwenden, um Schwachstellen zu finden. Ein gängiges Beispiel für ein unsicheres Design wäre etwa ein Prozess zum Zurücksetzen von Passwörtern, welcher auf dem Frage-Antwort-Prinzip besteht, da ein potenzieller Angreifer die Antworten auf die Sicherheitsfragen kennen oder mit Techniken wie einer Open-Source Intelligence (OSINT)-Recherche, oder Social Engineering herausfinden könnte.[35]

Security Misconfiguration (Sicherheitsrelevante Fehlkonfiguration)-Schwachstellen beruhen auf einer unsicheren Konfiguration der Anwendung, wie etwa nicht deaktivierten Funktionen welche nicht benötigt werden, falsch konfigurierten Berechtigungen, oder etwa Fehlermeldungen welche mehr Informationen preisgeben als nötig.[36] Auch diese Sicherheitslücken lassen sich wie zuvor gut durch ein intensives Erkunden der Anwendungen feststellen. So ist es ratsam nach dem Auskundschaften der in der Webanwendung verwendeten Technologien und Softwareversionen herauszufinden, ob beispielsweise Standardbenutzerkonten noch aktiv sind oder Standardkennwörter nicht geändert wurden.[37] Auch ein Portscanner wie *nmap* kann bei der Suche nach sicherheitsrelevanten Fehlkonfigurationen verwendet werden, um Ports zu finden, welche unbeabsichtigt von außen zugänglich sind oder auf dem zu testenden System verfügbare und möglicherweise angreifbare Dienste zu identifizieren. Beim Testen verursachte Fehlermeldungen sollten zudem auf nützliche Informationen untersucht werden. Denn oftmals werden Fehler nicht sauber abgefangen und geben Angreifern zu viele Informationen preis. Eine Fehlermeldung, welche nicht korrekt abgefangen wurde

[32] Vgl. 42.
[33] Vgl. 9, S. 786 f.
[34] Vgl. 9, S. 786 f.
[35] Vgl. 45.
[36] Vgl. 46.
[37] Vgl. 46.

und zu viele Informationen preisgibt, lässt sich mittels OWASP Juice Shop demonstrieren. Wie in Abbildung 5.8 zu sehen ist, wurde mittels Burp Suite ein Request zum Login eines Nutzers abgefangen und der Benutzername wurde so manipuliert, dass er einen Fehler bei der Datenbankabfrage auslöst.

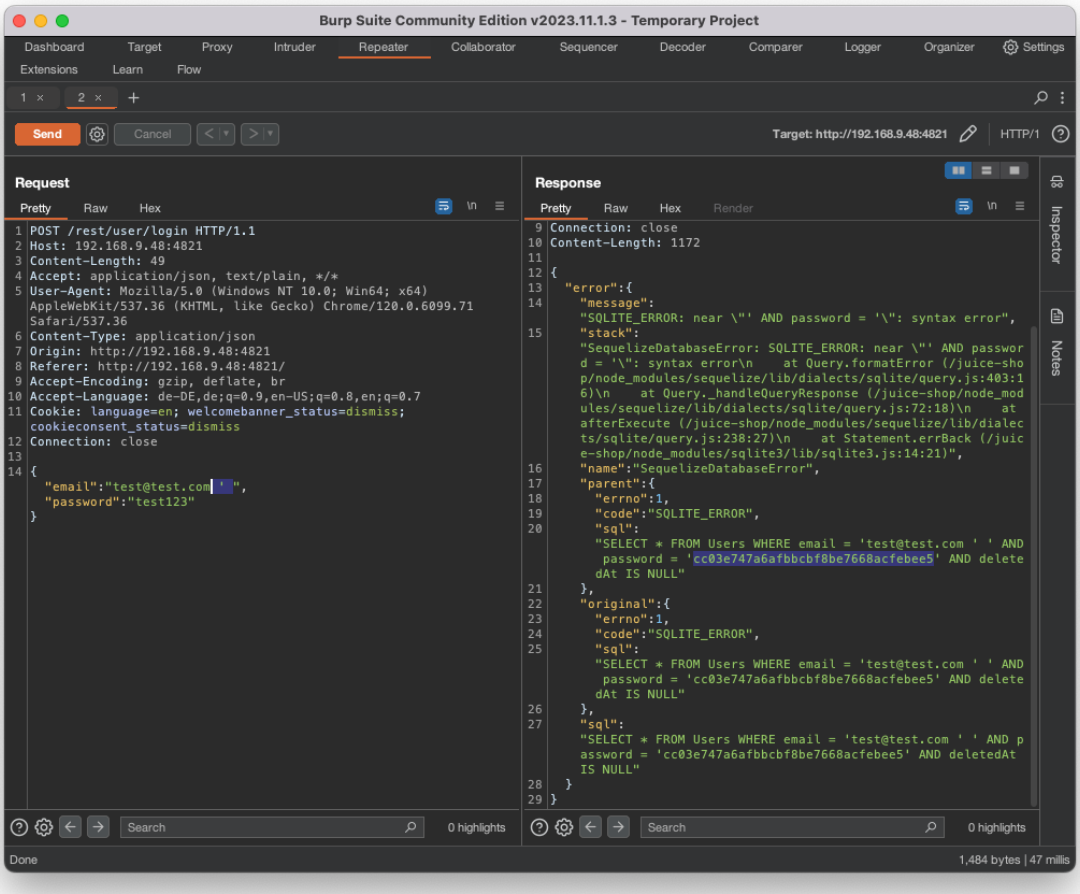

Abbildung 5.8: Response mit enthaltener Fehlermeldung, welche sensible Informationen preisgibt (Eigene Darstellung)

In der Abbildung kann man gut erkennen, dass der genaue Aufbau der internen Datenbankabfragen preisgegeben wird, sowie die verwendete Datenbanktechnologie und Version (SQLite 3). Zudem ist ersichtlich, dass das Passwort des Nutzers sehr wahrscheinlich in einen Hash des Typs MD5 umgewandelt wird, welcher wie zuvor bereits erwähnt nicht mehr als sicher gilt. Diese Informationen könnten einem Angreifer dabei helfen, einen SQL-Injection-Angriff durchzuführen. Die demonstrierte Schwachstelle lässt sich nicht nur als sicherheitsrelevante Fehlkonfiguration einordnen, sondern auch als **Security Logging and Monitoring Failures (Fehler bei Logging und Monitoring)**, da durch diese spezifische

Fehlkonfiguration Logdaten nach außen gelangen, welche nur intern ersichtlich sein sollten.

Vulnerable and Outdated Components (Angreifbare und veraltete Komponenten) betreffen viele IoT-Geräte, da diese häufig keine regelmäßigen Sicherheitsupdates erhalten. Im allgemeinen lassen sich IoT-Geräte mit veralteten Softwareversionen über die IoT-Suchmaschine shodan.io ausfindig machen. Während eines Penetrationstestes für ein spezielles Gerät können die verwendeten Bibliotheken und Module im Quellcode analysiert werden. Steht der Quellcode nicht zur Verfügung, kann der Einsatz eines Schwachstellenscanners wie beispielsweise Nessus in Betracht gezogen werden, um verwundbare Softwarekomponenten zu identifizieren.

Identification and Authentication Failures (Identifikations- und Authentifizierungsfehler) können in unterschiedlichen Ausprägungen auftreten. Diese Schwachstelle tritt auf, wenn es Benutzern gestattet ist, unsichere Passwörter zu verwenden, die Passwörter in Klartext abgespeichert werden, Sitzungs-IDs in der Uniform Resource Locator (URL) preisgegeben werden, keine Zwei-Faktor-Authentifizierung vorhanden ist oder kein Schutz vor Brute-Force-Angriffen implementiert ist.[38] Während eines Penetrationstests kann daher gezielt auf diese Schwachstellen getestet werden. Brute-Force-Angriffe auf Secure Shell (SSH)-Zugänge lassen sich beispielsweise mit dem Programm *hydra* durchführen. Bei Brute-Force-Angriffen auf Webanwendungen können Programme wie *wfuzz*, oder *burp suite* zum Einsatz kommen. Es sollte dabei nicht nur darauf geachtet werden, ob dabei erfolgreich Zugangsdaten ermittelt werden können, sondern auch ob grundsätzlich ein Brute-Force-Schutz implementiert ist, da dieser Angriffe erschwert. Die Software Burp Suite bringt zudem ein Modul namens sequencer mit, mit dem sich testen lässt, ob IDs wie Sitzungs-IDs zufällig generiert sind oder nach einem Muster erstellt werden. Sind die IDs nicht zufällig, kann ein Angreifer dies ausnutzen und versuchen, eigene IDs zu erstellen, um sich damit Zugang zu sensiblen Daten zu verschaffen.

Software and Data Integrity Failures (Software- und Datenintegritätsmängel) entstehen häufig durch unsichere oder schädliche Abhängigkeiten von Software, wie beispielsweise externe Module, Bibliotheken oder Plugins.[39] Ein gezielter Penetrationstest auf diese Art von Schwachstellen ist ohne den Quellcode der Anwendung nur schwer durchzuführen. Für eine genaue Analyse ist es notwendig alle Abhängigkeiten der Software zu recherchieren und anschließend durch eine intensive Recherche herauszufinden, ob diese Abhängigkeiten Schwachstellen mit sich bringen. Sollte es sich bei dem Penetrationstest um einen White-Box-Test handeln, empfiehlt es sich eine Software wie *OWASP Dependency Check* zur Hilfe zu nehmen, um die Abhängigkeiten automatisiert zu prüfen.[40]

Server-Side Request Forgery veranlasst eine Webanwendung dazu, Anfragen an entfernte Ressourcen durchzuführen, auf die ein Anwender keinen Zugriff haben sollte. Beispielhaft wären etwa Anfragen an Server aus dem internen Netzwerk, auf die ein Nutzer keinen Zugriff haben sollte, aber mittels Ausnutzung einer Schwachstelle einen anderen Server dazu bringt die gewünschte Anfrage erfolgreich auszuführen.[41] Durchführbar ist dies beispielsweise

[38] Vgl. 47.
[39] Vgl. 48.
[40] Vgl. 48.
[41] Vgl. 49.

durch das gezielte Mitlesen des Datenverkehrs zwischen Webseiten und Browser mittels einer Software wie Burp Suite und dem anschließenden austauschen der darin enthaltenen Serveradressen.[42]

5.2.3 Mobile Anwendungen

Zur Steuerung von IoT-Geräten kommen oftmals auch Smartphone Anwendungen zum Einsatz. Auch diese sollten in einen Penetrationstest mit einbezogen werden. Ein Ansatz beim Testen von Smartphone Apps ist das Mitschneiden des Datenverkehrs zwischen App und IoT-Gerät mittels Anwendungen wie *Burp Suite*, *OWASP ZAP*, oder *Wireshark*, um eventuell unverschlüsselte Datenübertragungen, oder Kommunikationsziele aufzudecken.[43] Dieses Vorgehen bietet sich vor allem bei WLAN-Verbindungen an, da hierfür keine spezielle Hardware benötigt wird, sondern nur ein handelsüblicher Laptop oder Desktop Computer, mit eingebauter oder externer WLAN-Karte. Steht der Quellcode einer App zur Verfügung, so sollten dieser und dessen Abhängigkeiten bei einem Penetrationstest untersucht werden. Steht der Quellcode nicht zur Verfügung, so kann man versuchen, mittels sogenanntem Reverse Engineering Nützliche Informationen oder Teile des Quellcodes aus der Binärdatei zu extrahieren.[44] Unter Reverse Engineering versteht man in diesem Kontext das Rekonstruieren des Quellcodes aus bereits kompilierten Binärdateien.

Die einfachste Möglichkeit Android Apps zu analysieren ist es, die APK-Datei aus dem Internet herunterzuladen und ähnlich wie ein ZIP-Archiv zu entpacken. Dadurch wird ein großer Teil der App-Struktur, wie sie auch in der Android Manifest-Datei beschrieben wird offengelegt. Für eine tiefere Analyse bietet sich ein Reverse Engineering des Quellcodes mittels Programmen wie *apktool* an, auf diese Weise ist es möglich mittels eines kurzen Kommandozeilenbefehls nach dem Schema *apktool d <Pfad/AppName.apk>* kompilierte Java-Dateien der App zu dekompilieren, um diese anschließend genauer zu betrachten.[45]

Da sich der Quellcode von Smartphone Apps oft nicht vollständig rekonstruieren lässt und die Analyse komplex sein kann, ist es unter Umständen sinnvoll eine Anwendung wie das *Mobile Security Framework* zu verwenden, um schelle aber nicht sehr tiefgreifende statische und dynamische Analysen von Android und iOS Apps zu erstellen.[46] Zum Erstellen einer Analyse reicht es aus, die APK-Datei einer Android App, oder die IPA-Datei einer iOS-App in die Weboberfläche zu ziehen und auf den Analyze-Button zu klicken.[47] Ein Ausschnitt einer solchen Analyse ist in Abbildung 5.9 zu erkennen.

[42] Vgl. 56.
[43] Vgl. 27, S. 957.
[44] Vgl. 27, S. 958.
[45] Vgl. 27, S. 959 ff.
[46] Vgl. 27, S. 962.
[47] Vgl. 27, S. 963.

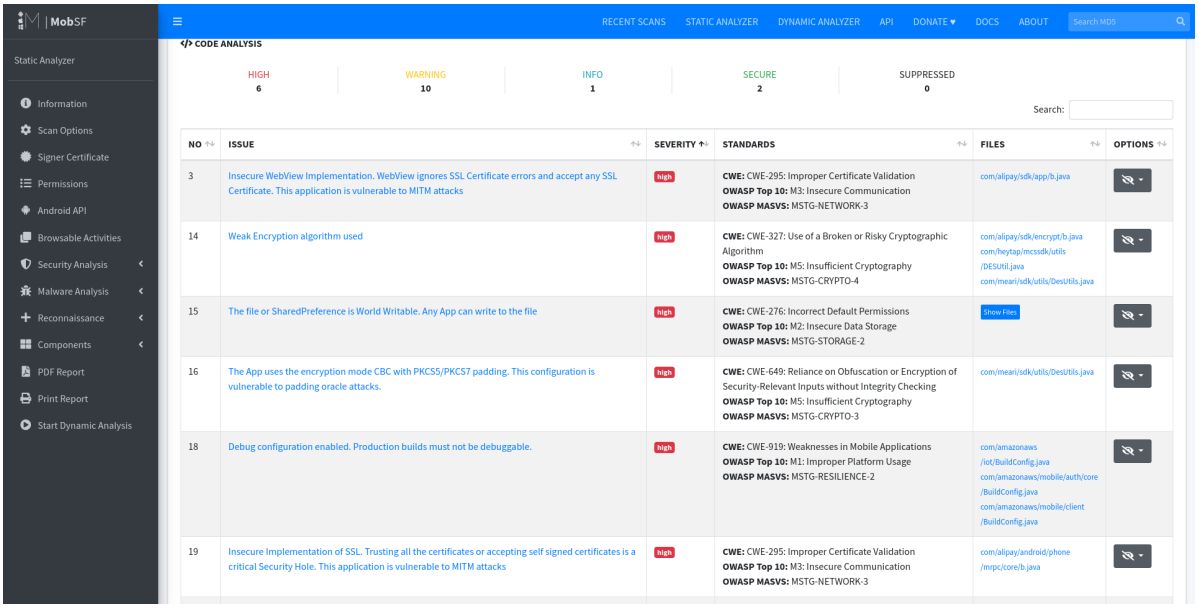

Abbildung 5.9: Ergebnisse der statischen App-Analyse einer App zur Steuerung von ieGeek Überwachungskameras mittels MobSF (Eigene Darstellung)

Die Abbildung zeigt eine statische Analyse der App zur Steuerung von *ieGeek* Überwachungskameras. Zu erkennen sind potenzielle im Quellcode der App vorhandene Schwachstellen wie schwache kryptografische Algorithmen, die verwendet wurden, oder aktiviertes Debugging, welches nur während der Entwicklung verwendet werden sollte. Zudem werden die Schwachstellen gängigen Standards wie Common Weakness Enumeration (CWE) einer Liste gängiger Schwachstellentypen, OWASP Mobile Application Security Verification Standard (MASVS) welches ein Standard für die Sicherheit mobiler Anwendungen ist und der OWASP Top 10 zugeordnet. Aber nicht nur der Quellcode wird in Mobile Security Framework (MobSF) analysiert, auch die von der App verlangten Berechtigungen werden geprüft. Zudem wird die Android Manifest Datei untersucht und Internet Domains und URLs, zu denen sich die Anwendung verbinden kann werden aufgelistet. Nützlich ist auch die Auflistung aller in der App hinterlegten Variablen und deren Werte und die automatische Hervorhebung möglicher fest einprogrammierter Geheimnisse wie etwa API-Schlüssel.

Der Vorteil einer solchen Analyse mittels MobSF ist, dass die Analyse sehr schnell und automatisiert erfolgt. Dies ist von Vorteil, wenn bei einem Penetrationstest nicht ausreichend Zeit eingeplant ist, um eine ausführliche manuelle Analyse einer App durchzuführen. Zudem hilft eine solche Analyse Penetrationstestern, die keine tiefgreifenden Kenntnisse in der Analyse von Smartphone-Apps haben, sich einen groben Überblick und Verständnis über die Anwendung zu verschaffen, ohne sich zuvor tiefgreifendes Wissen aneignen zu müssen. Aufgrund der Geschwindigkeit und Anwenderfreundlichkeit dieser Automatisierung, bietet es sich zudem an Apps auch dann in einen Penetrationstest mit einzubeziehen, wenn diese nicht Teil der eigentlichen Planung sind. So lassen sich zumindest grobe und offensichtliche Mängel schnell identifizieren.

5.3 Netzwerk

Ein zentraler Aspekt von IoT-Geräten ist die Netzwerkverbindung. Diese wird bei IoT-Geräten für gewöhnlich drahtlos über Technologien wie ZigBee, Bluetooth oder WLAN aufgebaut. Die Arten der Kommunikation unterscheiden sich dabei von Technologie zu Technolgie. So ist die Kommunikation über WLAN und ZigBee nur innerhalb eines Netzwerkes möglich, wohingegen Bluetooth eine direkte Kommunikation von Gerät zu Gerät erlaubt.[48]

Allerdings müssen **Bluetooth**-Geräte für eine Kommunikation erst miteinander gekoppelt werden, um eine verschlüsselte Kommunikation zu ermöglichen.[49] Eine zusätzliche Sicherheitsmaßnahme von BLE ist das sogenannte *Channel Hopping*, bei dem die Geräte nach einem individuell ausgewählten Muster die Bluetooth-Kanäle wechseln und somit ein Mitlesen verhindern sollen.[50] Mittels Programmen wie *Ubertooth One* lassen sich die Verbindungen allerdings überwachen und das Muster für das Wechseln der Kanäle kann automatisiert berechnet werden.[51] Um einen Angriff auf ein Bluetooth-Gerät durchzuführen, muss zuerst die physische Adresse des Gerätes ermittelt werden. Dies lässt sich mit dem Programm *hcitool* erledigen, welches bei vielen Linux Distributionen bereits vorinstalliert ist.[52] Mittels des Befehls *hcitool lescan* lassen sich beispielsweise die Hardwareadressen aller BLE-Geräte in Reichweite des Computers anzeigen.[53] Anschließend lassen sich Verbindungen über das Programm *Ubertooth* unter Verwendung des Befehls *ubertooth-btle -f -t<Hardwareadresse> -c <dateiname>.pcap* mitlesen.[54] Der Parameter *-f* sorgt dafür, dass dem Datenverkehr auch bei Kanalwechseln weiter gefolgt wird und der Parameter *-c* speichert die Datenpakete in einer Datei, welche später mit dem Programm *Wireshark* eine bessere Analyse ermöglicht.[55] Mittels des Programmes *gatttool*, welches Bestandteil vieler Linux Distributionen ist, lassen sich einzelne Datenpakete zum Durchführen eines Replay Angriffes anschließend erneut an das Gerät senden, oder verändern und an das Gerät senden.[56] Um Angriffe auf Bluetooth LE komfortabler durchzuführen, existiert das Framework *BtleJuice (https://github.com/DigitalSecurity/btlejuice)*, welches eine grafische Oberfläche besitzt und viele Funktionen für Penetrationstests von BLE-Verbindungen bündelt.

ZigBee lässt sich mittels Replay Angriffen angreifen, bei denen bereits verschlüsselte Datenpakete mitgelesen und zu einem späteren Zeitpunkt noch einmal gesendet werden, woraufhin das Zielgerät den mitgelesenen Befehl noch einmal durchführt.[57] Da die Datenpakete eine einzigartigen IDs enthalten, sondern nur Werte zwischen 0 und 255, sind solche Angriffe nicht sonderlich komplex in der Ausführung.[58] Eine weitere Angriffstechnik auf

[48] Vgl. 12, S. 197.
[49] Vgl. 12, S. 178.
[50] Vgl. 12, S. 181.
[51] Vgl. 12, S. 181.
[52] Vgl. 54, S. 296 f.
[53] Vgl. 54, S. 296 f.
[54] Vgl. 54, S. 297 f.
[55] Vgl. 54, S. 296.
[56] Vgl. 54, S. 305 f.
[57] Vgl. 12, S. 200.
[58] Vgl. 12, S. 200.

ZigBee Netzwerke ist das Abfangen des Netzwerkschlüssels. Dieser Schlüssel ist bei vielen ZigBee Koordinatoren fest einprogrammiert, lässt sich nicht verändern und wird zudem immer im Klartext übertragen, wodurch ein Angreifer nur darauf warten muss, bis ein neues Gerät dem Netzwerk beitritt und der Schlüssel beim Beitritt mitgelesen werden kann.[59][60] Solche Angriffe lassen sich mit geeigneter ZigBee Hardware welche mit einer speziellen *KillerBee (https://github.com/riverloopsec/killerbee)* Firmware bespielt wurde und dem zugehörigen Framework durchführen, welches für Penetrationstests von ZigBee-Geräten entwickelt wurde.[61] Damit lässt sich ZigBee Datenverkehr mitlesen und für eine einfachere Analyse an das Programm *Wireshark* weiterleiten, welches für gewöhnlich zum Mitlesen von WLAN Datenverkehr genutzt wird.[62]

Das Protokoll **Z-Wave** lässt sich auf ähnliche Art und Weise mit dem Framework *EZ-Wave* (https://github.com/AFITWiSec/EZ-Wave) angreifen, welches allerdings spezielle Hardware namens *HackRF* des Herstellers *Great Scott Gadgets* (https://greatscottgadgets.com/hackrf/one/) voraussetzt.[63] Mit dieser Kombination von Hardware und Software können Penetrationstester Z-Wave Geräte aufspüren, den Datenverkehr mitlesen und Aktionen wie Replay-Angriffe oder Überlagerungen und Störungen des Signals durchführen.[64]

Für Penetrationstests von klassischem **Ethernet und WLAN** Netzwerken kommen oftmals Programme wie *nmap* und *Wireshark* zum Einsatz. Das Programm *nmap* sendet IP-Pakete an alle möglichen Netzwerkadressen eines Netzwerkes, um basierend auf den erhaltenen Antworten alle im Netzwerk aktiven Geräte zu identifizieren.[65] Zudem erlaubt nmap eine weitere Analyse der gefundenen Netzwerkgeräte, indem es gezielt nach offenen Netzwerkports sucht und anschließend versucht herauszufinden, welche Dienste und deren Version auf den gefundenen Ports laufen.[66] Basierend auf diesen Informationen lassen sich weitere Angriffe vorbereiten, indem beispielsweise bereits bekannte Schwachstellen der gefundenen Dienste mittels Schwachstellendatenbanken ermittelt werden. Die Abbildung 5.10 zeigt das Ergebnis eines einfachen nmap Scans, welcher die IP-Adresse einer Überwachungskamera als Ziel hatte.

[59] Vgl. 12, S. 201.
[60] Vgl. 12, S. 201 ff.
[61] Vgl. 26, S. 276.
[62] Vgl. 26, S. 278.
[63] Vgl. 26, S. 280 f.
[64] Vgl. 26, S. 280 ff.
[65] Vgl. 27, S. 116.
[66] Vgl. 27, S. 116.

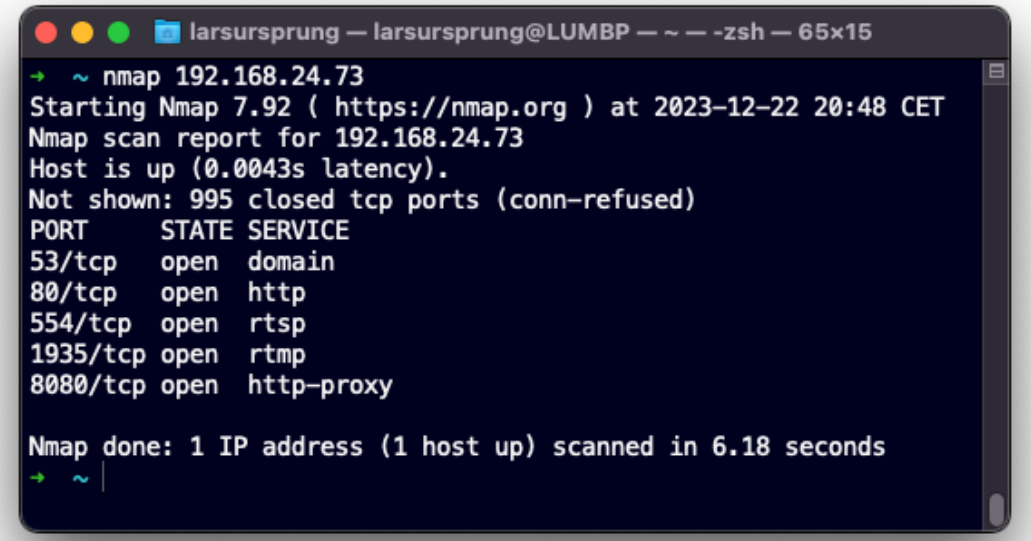

Abbildung 5.10: Ergebnis eines nmap Scans an einer ieGeek Überwachungskamera (Eigene Darstellung)

Auf dem Screenshot ist anhand der offenen Ports erkennbar, dass zusätzlich zu den Diensten Real-Time Streaming Protocol (RTSP) und Real-Time Messaging Protocol (RTMP), welche zur Videoübertragung bei Überwachungskameras genutzt werden, auch ein Domain Name System (DNS)-Server auf der Kamera läuft, als auch ein Webserver und ein Proxy-Server. Somit ist leicht erkennbar, dass auf der Kamera eine Weboberfläche für die Administration verfügbar ist. Da zudem nur der HTTP-Port mit der Nummer 80 erreichbar ist und nicht der HTTPS-Port mit der Nummer 443, ist zu erwarten, dass die Weboberfläche nur über das unsichere HTTP-Protokoll erreichbar ist. Mit durch den Parameter *-sV* aktiviertem Fingerprinting, erkennt *nmap* zudem, dass der verwendete Webserver vom Typ *Mongoose httpd* ist und RTSP über einen *Hipcam RealServer rtspd in Version 1.0* bereitgestellt wird. Weiterhin wird als Betriebssystem *DD-WRT* in Version 3.0 mit den Linux Kernel in Version 4.4.2 erkannt, sowie *Shenzengtong BO Weitechnology* als Hersteller der Netzwerkkarte identifiziert. Die Verwendung von *nmap* liefert demnach also eine gute Informationsgrundlage, um das weitere Vorgehen während eines Penetrationstests zu planen.

Für intensivere Analysen von Netzwerkgeräten und dem Datenverkehr, der von diesen ausgeht, ist das Programm Wireshark weit verbreitet. Es bietet eine grafische Benutzeroberfläche und kann den gesamten Datenverkehr einer Netzwerkschnittstelle mitschneiden und analysieren.[67] Für die Analyse des Datenverkehrs bei IoT-Geräten bietet sich daher

[67] Vgl. 27, S. 135.

an, einen Computer auf dem Wireshark installiert ist als WLAN-Zugangspunkt für das zu testende Gerät zu konfigurieren, um so den gesamten Datenverkehr analysieren zu können.[68] Da die Datenmenge schnell sehr groß und damit unübersichtlich werden kann, ist es ratsam die Filtermöglichkeiten von Wireshark anzuwenden und damit nur bestimmte Geräte und Protokolle anzeigen zu lassen, sowie Datenpakete in nicht zu großen Intervallen aufzunehmen.[69]

5.4 Cloud

Häufig sind IoT-Geräte mit Clouddiensten verbunden, um Daten zentral abzulegen und verfügbar zu machen, oder um Geräte von außerhalb des internen Netzwerkes aus zu steuern und zu verwalten. Ein weit verbreiteter Clouddienst zum Speichern von größeren Datenmengen ist Amazon S3. S3 erlaubt es Daten in sogenannten *Buckets* entweder öffentlich oder privat abzulegen.[70] Bei einem Penetrationstest ist es also sinnvoll den Netzwerkverkehr mittels Anwendungen wie *Burp Suite* oder *Wireshark* mitzulesen und zu überprüfen, ob Verbindungen zu S3-Buckets aufgebaut werden. Anschließend sollte überprüft werden, ob Daten in diesen Buckets vor Zugriffen von außen geschützt und Berechtigungen korrekt konfiguriert wurden. Die Namensgebung für S3-Buckets folgt den gleichen Regeln wie für Hostnahmen, bei denen nur Kleinbuchstaben und wenige Sonderzeichen gestattet sind.[71] Dies verleitet Administratoren und Entwickler dazu, für Buckets ähnliche Namen wie für die eigenen Domainnamen zu verwenden.[72] Da die Namen für S3-Buckets in den meisten Fällen nach dem Muster *http://s3.amazonaws.com/<bucket name>/* oder *http://<bucket name>.s3.amazonaws.com/* aufgebaut sind, können potenzielle Angreifer mögliche Adressen mit Anwendungen wie *Gobuster* oder *DirBuster* erraten und anschließend analysieren.[73] *Gobuster* bietet neben Möglichkeiten zum enumerieren von Verzeichnissen und Domains auch einen Modus zum Auffinden von Amazon S3-Buckets. In Abbildung 5.11 ist ein solcher Scan zu erkennen.

[68] Vgl. 27, S. 136.
[69] Vgl. 27, S. 137 ff.
[70] Vgl. 27, S. 851 f.
[71] Vgl. 27, S. 852.
[72] Vgl. 27, S. 852.
[73] Vgl. 27, S. 858.

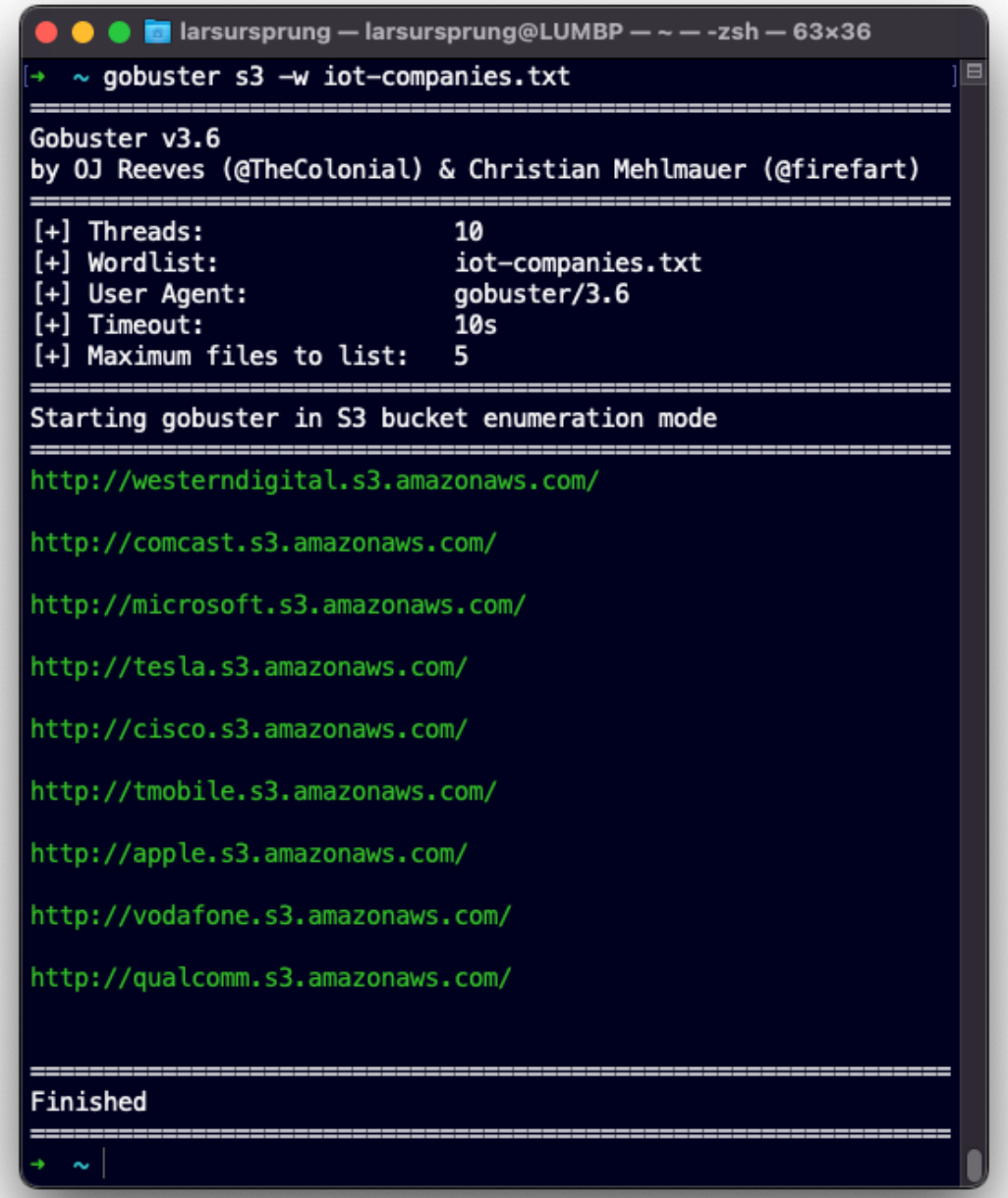

Abbildung 5.11: Ergebnis eines Gobuster Scans nach Amazon S3-Buckets (Eigene Darstellung)

Für diesen Scan wurde mittels einer schnellen Internetrecherche eine Liste mit 25 Unternehmen aus dem Bereich IoT angefertigt. Bei neun Unternehmen, also mehr als einem Drittel waren S3-Buckets auffindbar. Sobald der Name eines Buckets bekannt ist, können mittels

der AWS Konsolenanwendung und dem Befehl *aws s3 ls s3://<bucket name>/ –recursive > directory.txt* alle Dateien und Verzeichnisse rekursiv aufgelistet und in eine Textdatei exportiert werden.[74] Eine Möglichkeit öffentlich zugängliche S3-Buckets automatisch aufzuspüren und zu durchsuchen, bieten Anwendungen wie *Warfare*, *bucket-stream*, *AWSBucketDump*, oder *Bucket finder*.[75] Bei Penetrationstests von Amazon Web Services ist dabei zu beachten, dass für Penetrationstests zuvor die Erlaubnis von Amazon eingeholt werden muss, welche sich über ein Onlineformular innerhalb von wenigen Tagen beantragen lässt.[76] Auch andere Anbieter von Clouddiensten haben Richtlinien für Penetrationstests aufgestellt, die es jeweils zu beachten gilt, um rechtlich einwandfrei zu handeln.[77]

[74] Vgl. 27, S. 858.
[75] Vgl. 27, S. 859.
[76] Vgl. 12, S. 236 f.
[77] Vgl. 12, S. 234 f.

6 Methode zur Durchführung von Penetrationstests an IoT-Geräten

Basierend auf den Erkenntnissen der vorherigen Kapitel, wird nachfolgend eine geeignete Methode entwickelt, um effektiv Penetrationstests an IoT-Geräten durchzuführen. Dabei werden zunächst bestehende Penetrationstestmethoden evaluiert, um anschließend kritische Elemente für eine Penetrationstestmethodik zu bestimmen. Mit den daraus resultierenden Anforderungen wird im Nachhinein eine Penetrationstest-Methodik entwickelt, die sich von Unternehmen und Penetrationstestern auf unterschiedlichste IoT-Geräte anwenden lassen soll. Die neu entwickelte Methodik wird abschließend mit den zuvor erläuterten bestehenden Methoden verglichen und es wird geklärt, wie die entwickelte Methodik an zukünftige Entwicklungen im Bereich IoT angepasst und damit langfristig relevant bleiben kann.

6.1 Bestehende Penetrationstest-Methoden für IoT-Geräte

Zunächst gilt es zu evaluieren, welche bestehenden Penetrationstest-Methoden für IoT-Geräte existieren und wo deren Stärken, als auch deren Schwächen liegen. Dies dient dazu, den aktuellen Stand in dem Bereich der IoT-Penetrationstests aufzuzeigen. Hierfür werden nachfolgend einige verbreitete Methoden betrachtet. Dies bildet zudem die Grundlage für die spätere Identifizierung kritischer Elemente bei der Entwicklung einer eigenen Methodik.

Der **OWASP IoT Security Testing Guide** ist ein Leitfaden speziell entwickelt für die Durchführung von Penetrationstests im IoT-Bereich. Der Leitfaden ist darauf ausgelegt Schwachstellen in IoT-Geräten zu identifizieren und zu mitigieren und versucht dabei möglichst alle Bereiche von IoT abzudecken und auch bei zukünftigen Entwicklungen stets vergleichbare Ergebnisse zu liefern.[1] Enthalten sind eine Reihe praktisch umsetzbarer Techniken und Methoden, die von Penetrationstestern in ihren Tests direkt umgesetzt werden können. Der Leitfaden teilt IoT-Geräte in unterschiedliche Komponenten auf, wobei jeder Komponente neben einer umfangreichen Sammlung an Informationen, ein Katalog an Testszenarien inklusive Checklisten zur Verfügung gestellt wird.[2] Unter anderem werden die Bereiche Authentifizierung, Verschlüsselung, Firmware-Aktualisierungen und Netzwerkkommunikation abgedeckt. Durch diese breite Aufstellung wird ein ganzheitlicher Blick auf die IoT-Sicherheit gewährleistet. Die stetige Aktualität wird dabei durch die Beiträge aus der OWASP-Gemeinschaft sichergestellt, welche neue Informationen zusammenträgt und veröffentlicht. Dieser gemeinschaftsorientierte Ansatz kann allerdings die Konsistenz beein-

[1] Vgl. 57.
[2] Vgl. 58.

trächtigen, da zwar oft ein breites Spektrum an Informationen zusammengetragen wird, diese aber in Tiefe und Konsistenz der Inhalte schwanken können, wodurch die Einheitlichkeit und Qualität der Richtlinien Schwankungen unterliegen. Ein weiterer Nachteil des Leitfadens ist das Fehlen einer empfohlenen Reihenfolge oder Struktur in, welcher die verschiedenen Test durchgeführt werden sollten. Dies kann eine Rolle spielen, da bei der Untersuchung der Hardware diese zerstört werden kann und anschließend ein Test der Software nicht mehr möglich wäre, sofern nicht weitere Gerät für Tests zur Verfügung stehen. Ein weiteres Beispiel wäre das Ausleiten der Firmware während einer initialen Aktualisierung der Firmware. Sollten keine anderen Möglichkeiten zur Verfügung stehen, die Firmware zu extrahieren und die initiale Aktualisierung bei Inbetriebnahme des Gerätes ist bereits durchgeführt, so entfällt diese Möglichkeit an die Firmware zu gelangen damit. Daher kann die Reihenfolge, in der die einzelnen Tests durchgeführt werden eine wichtige Rolle für einen effizienten Penetrationstest spielen. Auch darauf wie eine effektive Zusammenarbeit von Penetrationstestern untereinander aussehen kann, findet in dem Leitfaden keine Erwähnung. Der OWASP IoT Security Testing Guide ist daher als umfassendes Nachschlagewerk anzusehen, auf dessen Grundlage Unternehmen eine individuelle Methodik zunächst noch entwickeln müssen.

Das **BSI-Penetrationstest-Modell** beschäftigt sich mit der Durchführung von Penetrationstests von vernetzten IT-Systemen im Allgemeinen.[3] Es handelt sich hierbei um eine Studie, welche vom Bundesamt für Sicherheit in der Informationstechnik (BSI) durchgeführt wurde und den Fokus auf die strukturelle Durchführung von Penetrationstests legt, statt auf die praktische Anwendung von spezifischen Techniken.[4] Dadurch kann das Modell als Informationsgrundlage zur Entwicklung eigener, spezifischer Methoden verwendet werden, unter anderem auch für den IoT-Bereich. Das BSI-Modell geht dabei nicht nur ausschließlich auf die Struktur von Penetrationstests ein, sondern auch auf die Rahmenbedingungen, wie gesetzliche Vorschriften, welche bei der Durchführung beachtet werden sollten.[5] Doch auch organisatorische Rahmenbedingungen wie mögliche Systemausfälle oder personelle und technische Voraussetzungen, die bei der Planung von Penetrationstests beachtet werden sollten, lässt das BSI-Modell nicht außer Acht.[6] Die Stärke des BSI-Modells liegt in seiner strukturierten Vorgehensweise, die in fünf Phasen unterteilt ist: Vorbereitung, Informationsgewinnung, Bewertung, aktive Eindringversuche und Abschlussanalyse.[7] Diese systematische Herangehensweise kann auch beim Testen von IoT-Umgebungen von Vorteil sein, da sie aufgrund intensiver Vorbereitung und wohldefinierter Struktur eine detaillierte und umfassende Analyse ermöglicht. Allerdings ist das BSI-Modell aufgrund seines großen Umfangs und den detaillierten Modulbeschreibungen mit spezifischen Checklisten für die einzelnen Penetrationstestbestandteile nur bedingt für die Durchführung von IoT-Systemen geeignet.[8] Als Grund lässt sich hier unter anderem der starke Fokus aus den Bereich Software anführen, denn die physische Untersuchung von Hardware wie sie bei Penetrationstests im Bereich des

[3] Vgl. 59, S. 4.
[4] Vgl. 59, S. 4.
[5] Vgl. 59, S. 18 f.
[6] Vgl. 59, S. 36 ff.
[7] Vgl. 59, S. 45 ff.
[8] Vgl. 59, S. 53 ff.

IoT notwendig ist, fehlt gänzlich. Somit ist das BSI-Modell zwar nicht als alleinige Grundlage für die Durchführung von Penetrationstests im IoT-Bereich geeignet, bietet allerdings eine solide Grundlage, auf der Penetrationstests in diesem Bereich aufbauen können.

IoT Pentesting Guide by Aditya Gupta Der IoT Pentesting Guide von Aditya Gupta liefert eine weitere Methode zur Durchführung von Penetrationstests im Bereich IoT und soll vor allem als Nachschlagewerk für Anfänger und Fortgeschrittene Penetrationstester dienen. Der Leitfaden konzentriert sich dabei vornehmlich auf die physischen Komponenten von IoT-Geräten, sowie auf die Firmware und Netzwerkkommunikation.[9] Derzeit befindet sich der Leitfaden noch in einer frühen Phase der Entwicklung und führt alle Themenbereiche nur kurz und stichpunktartig aus. Aufgrund dessen ist der Leitfaden im aktuellen Zustand nicht für den praktischen Einsatz geeignet. Des Weiteren Fehlt die Definition einer strukturierten Vorgehensweise und zur Planung von IoT-Penetrationstests.

Auch das Buch **Praktische Einführung in Hardware Hacking** von Marcel Mangel und Sebastian Bicchi bietet viele Informationen zum Thema IoT-Penetrationstests. Es enthält Informationen zu unterschiedlichsten Themen und geht teilweise auch darauf ein, in welcher Reihenfolge ein Penetrationstest an IoT-Geräten durchzuführen ist. Es wird etwa speziell darauf hingewiesen, dass bei der ersten Inbetriebnahme eines Gerätes der Datenverkehr mitgeschnitten werden sollte, um währenddessen eventuell wichtige Informationen abzufangen.[10]

Das **IoT Security Assurance Framework** welches von der IoT Security Foundation geschaffen wurde, stellt eine umfassende Sammlung an Richtlinien und Praktiken dar, welche speziell auf die Absicherung von IoT-Geräten und deren Ökosystem ausgerichtet sind. Das Framework bietet einen strukturierten Ansatz für die Bewertung der Sicherheit von IoT-Umgebungen, der detaillierte Verfahren und Empfehlungen enthält, die sich an verschiedenen Interessengruppen im IoT Umfeld richten, darunter Hersteller von IoT-Geräten, Entwickler und auch Händler.[11] Ein großer Vorteil gegenüber anderen eher allgemein gehaltenen Frameworks stellt dabei die Spezialisierung auf das IoT Ökosystem dar. Das Framework bietet dabei praktische Anleitungen und Richtlinien, die von Unternehmen direkt anwendbar sind. Dabei werden Themenbereiche wie Datenschutz, Gerätesicherheit, Netzwerksicherheit, sowie Cloud-Schnittstellen mit abgedeckt. Somit kann die Sicherheit von IoT-Geräten in Gänze abgebildet werden. Dabei lässt sich das Framework auf eine Vielzahl unterschiedlicher Geräte anwenden und ist auf IoT Landschaften unterschiedlichster Größe anpassen. Ein Nachteil des Frameworks ist allerdings der Mangel an tiefen Informationen unter anderem im Bereich Netzwerktechnik. Somit ist das Framework nicht als alleiniges Nachschlagewerk geeignet und bei spezifischen Fragestellungen müssen weitere Informationen aus weiteren Quellen hinzugezogen werden. Aufgrund der detaillierten Empfehlungen und Richtlinien ist das Framework zwar flexibel auf unterschiedlichste Gerätetypen anwendbar, muss aufgrund der raschen Entwicklungen im IoT Umfeld stetig aktualisiert werden und implementiert die aktuellsten Entwicklungen unter Umständen erst verzögert. Zudem birgt eine ausführliche

[9] Vgl. 60.
[10] Vgl. 12, S. 20 ff.
[11] Vgl. 61, S. 4 f.

Sammlung von Richtlinien das Risiko, dass diese Richtlinien unbedacht und eventuell unpraktikabel umgesetzt werden, anstelle eine umfassende und angepasste Sicherheitsstrategie zu entwickeln, die für das Unternehmen praktikabel ist. Es stellt zudem keine dedizierte Methode zur Durchführung von Penetrationstests bereit, sondern dient vielmehr dazu fundierte Entscheidungen in Bezug auf die Sicherheit von IoT-Geräten zu treffen, indem es eine Liste mit Voraussetzungen für eine gute Sicherheit bereitstellt.[12] Das Framework bildet daher eine bessere Grundlage für Unternehmen die IoT-Geräte entwickeln möchten, oder die vor der Kaufentscheidung von IoT-Geräten stehen, als für solche die Penetrationstests durchführen möchten.

Die **NIST Special Publication 800-183 (Networks of 'Things')** ist ein Framework, welches darauf ausgelegt ist die Komplexität moderner Computernetze vor allem im IoT Umfeld verständlich zu machen. Die Veröffentlichung bietet einen Überblick über die grundlegenden Aspekte vernetzter Geräte und Systeme, einschließlich ihrer Zusammensetzung, Verbindungsarten und Eigenschaften. Das Hauptaugenmerk liegt hierbei darauf, Grundlagen zu definieren, um ein gemeinsames Verständnis für diese und deren Begriffe zu schaffen.[13] Dieser Ansatz kann für IoT-Penetrationstests von Nutzen sein, da er ein breites Spektrum an vernetzten Geräten und deren Möglichkeiten zur Interaktion untereinander umfasst und eine Informations- und Kommunikationsgrundlage für eine gründliche Bewertung potenzieller Schwachstellen über verschiedene Netzwerkschichten von IoT hinweg ermöglicht. Vor allem die Interoperalität von IoT-Geräten steht dabei im Fokus, was bei der Identifikation von Schwachstellen, die während der Interaktion von Geräten entstehen können hilfreich ist, ein kritischer Aspekt, der bei gerätezentrierten Methoden übersehen werden kann. Zudem genießt das Framework eine hohe Glaubwürdigkeit und Reputation, da es von einer Bundesbehörde der USA stammt und in Zusammenarbeit mit Forschern der ganzen Welt entwickelt wurde. Allerdings liegt der Fokus nicht auf der Durchführung von Penetrationstests, weshalb das Framework nicht als eigenständige Penetrationstestmethode zu sehen ist. Auch der exklusive Fokus auf Netzwerktechnologien stell hier einen Nachteil dar. Jedoch stellt das Framework ein fest definiertes Vokabular und Grundlagenwissen für den Bereich der IoT-Netzwerke zu Verfügung. Dieses kann während Penetrationstests verwendet werden, um ein gemeinsames Verständnis aller Beteiligten für verschiedene Begriffe zu festigen.[14] Das Framework ist daher eher als Informationsquelle oder Nachschlagewerk anzusehen, welches bei Penetrationstests unterstützend angewendet werden kann.

6.2 Kritische Elemente bei der Entwicklung einer Penetrationstest-Methodik für IoT-Geräte

Im Folgenden Abschnitt werden auf Grundlage der zuvor bei der Durchführung von Penetrationstests und bei der Evaluierung von verschiedenen Penetrationstestmethoden erlangten gesammelten Erkenntnisse genutzt, um kritische Elemente für die Entwicklung einer Methode

[12] Vgl. 61, S. 5.
[13] Vgl. 62, S. 1 f.
[14] Vgl. 62, S. 22.

für Penetrationstests bei IoT-Geräten zu benennen. Dies ist notwendig, da das IoT-Ökosystem eine Vielzahl komplexer und miteinander verbundener Elemente umfasst und Penetrationstests genau strukturiert werden sollten.

Als erstes wichtiges Element für eine Penetrationstestmethodik ist die **Testreihenfolge** zu nennen, in der Tests durchgeführt werden sollten. Die Reihenfolge ist von entscheidender Bedeutung, da sie sowohl die Wirksamkeit des Testes, als auch die Integrität des zu testenden Gerätes beeinflussen kann. Die Reihenfolge sollte daher strategisch so gewählt werden, dass sie die Wahrscheinlichkeit mit der Schwachstellen entdeckt werden maximiert und das Risiko minimiert das Gerät zu einem ungünstigen Zeitpunkt zu beschädigen, oder den Betrieb des Gerätes zu verhindern. Es ist daher ratsam, mit nicht-invasiven Tests zu beginnen. Ein sinnvoller Beginn wäre etwa die Untersuchung des Netzwerkverkehrs bei Inbetriebnahme des Gerätes, da hierbei Schwachstellen wie unverschlüsselte Kommunikation für den Schlüsselaustausch oder initiale Konfigurationen abgefangen werden können. Des Weiteren führen Geräte bei der erstmaligen Inbetriebnahme häufig Softwareaktualisierungen durch, wobei entweder die Übertragung der Firmware mitgelesen werden kann, oder zumindest von welchem Endpunkt beim Hersteller dieser bezogen wird. Anschließend können weitere Tests an der Software durchgeführt werden, wie etwa das Suchen nach Schwachstellen und eventuell vorhandenen Weboberflächen, oder mobilen Apps. Anschließend können invasivere Analysen wie die Untersuchung der Sicherheit des Gehäuses und das Identifizieren von Debugging-Schnittstellen wie UART. Bei diesen Untersuchungen besteht das Risiko das Gerät irreversibel zu beschädigen und die Durchführung von weiteren Tests zu verhindern, oder zu verzögern sofern es dann notwendig wird Ersatz zu beschaffen. Aus diesen Gründen ist eine sorgfältige Wahl der Reihenfolge für eine effektive und effiziente Durchführung von Penetrationstests unabdinglich.

Noch bevor Penetrationstester eigene praktische Tests und Untersuchungen durchführen, sollte eine umfassende **OSINT-Recherche** durchgeführt werden. Hierbei können vorab bereits wichtige Informationen gewonnen werden, die Penetrationstestern vorab ein besseres Verständnis für das zu testende Gerät vermitteln. So gibt es für eine Vielzahl von Geräten auf Webseiten wie etwa *iFixit* (https://www.ifixit.com/Teardown) bereits detaillierte Anleitungen für das Auseinanderbauen der Hardware. Somit kann bereits vorab ein Verständnis dafür erlangt werden, wie ein Gerät aufgebaut ist, notwendige Werkzeuge können vorab beschafft werden und Beschädigungen aufgrund falscher Herangehensweisen können vermieden werden. Auch Firmwares können in einigen Fällen bereits vorab heruntergeladen und untersucht werden. In vielen Fällen lassen sich Firmwares sogar beim Hersteller selbst herunterladen. Sollte dies nicht möglich sein, gibt es Firmwaredatenbanken, in denen sich viele Firmwares auffinden lassen. Somit ist auch ein Vergleich von unterschiedlichen Firmwareversionen möglich, um eventuelle Auffälligkeiten aufzudecken. Auch lassen sich Standardpasswörter von vielen Geräten im Internet auffinden, sowie die Position der Debugging-Schnittstellen auf der Platine. Eine umfassende Suche mittels Suchmaschinen wie Google (https://google.de) oder Plattformen wie GitHub (https://github.com) kann also vorab wichtige Erkenntnisse oder nützliche Vergleichsdaten liefern.

Ein weiteres kritisches Element ist die kontinuierliche **Dokumentation** von Zwischenergeb-

nissen und durchgeführten Aktionen. Dies stellt sicher, dass durchgeführte Tätigkeiten zu jedem Zeitpunkt reproduzierbar sind und andere Penetrationstester den aktuellen Stand eines Tests aufgreifen können. Es sollte dabei auf eine saubere und logische Struktur der Dokumentation und Daten geachtet werden. So sollten interessante Mitschnitte des Netzwerkverkehrs so abgelegt werden, dass sie zu einem späteren Zeitpunkt leicht auffindbar sind. Dies könnte etwa notwendig sein, um Requests miteinander zu vergleichen und Auswirkungen von Änderungen der Konfiguration zu identifizieren. Es ist daher ratsam eine sinnvolle Projektstruktur für Penetrationstests festzulegen, in der sich alle Beteiligten jederzeit zurechtfinden.

Zu Beginn eines Penetrationstestes ist es notwendig den genauen **Umfang** zu definieren. Ohne einen zuvor festgelegten Umfang ist es kaum möglich einen Penetrationstest innerhalb eines zuvor bestimmten Zeitrahmens effizient durchzuführen. Dieser ist daher ein weiterer kritischer Aspekt eine Penetrationstestmethodik. Der Umfang sollte alle relevanten Komponenten eines IoT-Systems abdecken. Relevante Komponenten sind Hardware, Software, Netzwerk und Cloud-Dienste, sowie deren Teilbereiche. Sollte für einen Penetrationstest nur eine begrenzte Zeit oder begrenzte Ressourcen zu Verfügung stehen, so sollte der Umfang entsprechend angepasst werden. Der Umfang sollte daher auf Ressourcen wie Zeit, Kenntnisse der Penetrationstester und zur Verfügung stehende Werkzeuge und Hardware angepasst werden.

Threat Modeling ist eine Technik, bei der alle Funktionen, technischen Abhängigkeiten und der Datenfluss von Systemen modelliert werden, um dabei möglichst alle Angriffsvektoren eines Systems zu identifizieren und zu grafisch dokumentieren.[15] Sobald die Angriffsvektoren identifiziert wurden, werden Angriffsszenarien, auch Threats genannt, mittels Methoden wie Spoofing Identity, Tampering with Data, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege (STRIDE) identifiziert und klassifiziert, um das zugehörige Risiko festzulegen.[16] Aus dem Threat Modeling resultiert das sogenannte *Threat Model*, auf welchem die Durchführung des Penetrationstests aufbauen kann. Ein Threat Model kann bei einem IoT Penetrationstest also eine wichtige Rolle spielen. Daher wird das Threat Modelling bei der Methodenentwicklung ausführlicher behandelt.

Das Übergreifende Ziel eines Penetrationstestes ist es einen umfassenden Bericht über die Sicherheit des zu testenden Systems zu erstellen. Dies wird auch **Reporting** genannt und ist ein weiterer kritischer Bestandteil von Penetrationstests. Der Ergebnisbericht dient nach dem Penetrationstest als Entscheidungsgrundlage für weitere Handlungen, oder als Grundlage zur Behebung der gefundenen Schwachstellen. Dieser Bericht richtet sich an unterschiedliche Zielgruppen wie Entwickler, Führungskräfte oder andere Penetrationstester und sollte daher verwertbare Informationen für jede dieser Zielgruppen aufweisen.

6.3 Methodenentwicklung

Im Folgenden werden die zuvor identifizierten kritischen Elemente von IoT Penetrationstests, sowie die in den vorherigen Kapiteln gewonnenen Kenntnisse dazu genutzt, eine Methode für

[15] Vgl. 26, S. 32.
[16] Vgl. 26, S. 32.

Penetrationstests von IoT-Geräten zu entwickeln. Als kritisches Element hat sich eine sinnvolle Reihenfolge für die Durchführung der einzelnen Analysen der Komponenten von IoT-Geräten erwiesen. Steht beispielsweise nur ein einzelnes Gerät physisch für einen Penetrationstest zur Verfügung und ein Penetrationstester beginnt den Test mit der Analyse der Hardware, so besteht die Möglichkeit, dass das Gerät bei der Analyse zerstört wird und der Test somit nicht vollständig durchgeführt werden kann.

6.3.1 Threat Modeling

Sollte nicht bereits ein Threat Model existieren, so sollte dieses zu Beginn des Tests erstellt werden. Dies dient dazu mögliche Angriffsvektoren im Voraus zu identifizieren und den Penetrationstest effektiv gestalten zu können. Ein Threat Model sollte konstant aktualisiert werden, um zukünftige Entwicklungen, oder neue Erkenntnisse einzubringen.[17] Sobald mögliche Angriffsvektoren identifiziert sind, werden für die einzelnen Angriffsvektoren mittels Methoden sie STRIDE konkrete Bedrohungsszenarien modelliert und die Gefahr die von diesen ausgeht mittels Methoden wie Damage Reproducibility Exploitability Affected Discoverability (DREAD) oder CVSS bewertet.[18] Wichtig ist vor allem einen einheitlichen Standard für die Bewertung auszuwählen, um die interne Vergleichbarkeit der Modelle zu gewährleisten. Für die Erstellung eines Threat Models eignen sich Programme für die Erstellung von Diagrammen im Allgemeinen, oder spezialisierte Programme für die Erstellung von Threat Models, wie beispielsweise das quelloffene *OWASP Threat Dragon (https://github.com/OWASP/threat-dragon)*.

Im Folgenden wird beispielhaft ein Threat Model für eine Überwachungskamera des Herstellers *ieGeek* erstellt. Zu Beginn werden alle relevanten Komponenten der Kamera identifiziert und in tabellarischer Form festgehalten.[19] Eine solche Auflistung ist in Tabelle 6.1 zu sehen.

[17] Vgl. 26, S. 32.
[18] Vgl. 26, S. 32 f.
[19] Vgl. 26, S. 43.

ID	Komponente	Beschreibung
1	Gehäuse	Das Gehäuse der Kamera beinhaltet die Technik der Kamera. Es stellt einen Steckplatz für Micro SD Karten, sowie einen Stromanschluss bereit. Das Gehäuse lässt sich über zwei Schrauben an der Vorderseite der Kamera öffnen.
2	Interne Hardware	Innerhalb des Gehäuses befinden sich zwei Platinen, sowie die Linse der Kamera. Auf einer der Platinen befinden sich mögliche UART-Konnektoren.
3	Weboberfläche	Die Kamera lässt sich über eine Weboberfläche administrieren, welche lokal auf der Kamera ausgeführt wird. Über die Weboberfläche lässt sich zudem das Videobild der Kamera anzeigen.
4	Firmware	Auf der Firmware der Kamera laufen interne Anwendungen und Konfigurationen die zum Betrieb der Kamera notwendig sind.
5	WLAN	Die Kamera besitzt eine WLAN-Schnittstelle, für eine Einbindung in Drahtlose Netzwerke.

Tabelle 6.1: Komponenten einer *ieGeek* Überwachungskamera

Anschließend wird eine vereinfachte grafische Darstellung der Architektur des IoT-Gerätes erzeugt, um den Datenfluss zu visualisieren. in Abbildung 6.1 ist eine solche Darstellung beispielhaft dargestellt.

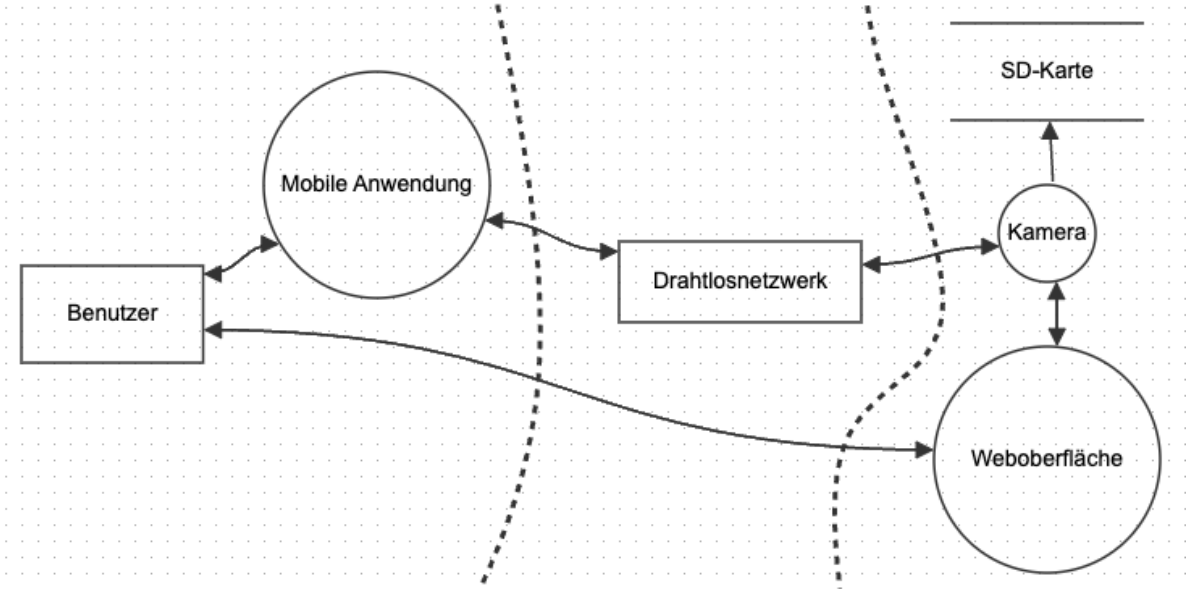

Abbildung 6.1: Vereinfachte grafische Darstellung der Architektur einer ieGeek Überwachungskamera (Eigene Darstellung)

Auf Basis der Erkenntnisse von Komponenten und Architektur, sowie Datenfluss, kann eine Übersicht möglicher Angriffsziele erstellt werden.[20] Diese sollten zu einem späteren Zeitpunkt noch vor der praktischen Durchführung eines Penetrationstests individuell bewertet werden. Die nachfolgende Tabelle 6.2 zeigt eine solche Auflistung

[20] Vgl. 26, S. 46 f.

ID	Komponente	Beschreibung
1	Gehäuse	Das Gehäuse der Kamera beinhaltet die Technik der Kamera. Es stellt einen SD-Kartensteckplatz, sowie einen Stromanschluss bereit. Das Gehäuse lässt sich über zwei Schrauben an der Vorderseite der Kamera öffnen.
2	Interne Hardware	Innerhalb des Gehäses befinden sich zwei Platinen, sowie die Linse der Kamera. Auf einer der Platinen befinden sich mögliche UART-Konnektoren.
3	Weboberfläche	Die Kamera lässt sich über eine Weboberfläche administrieren, welche lokal auf der Kamera ausgeführt wird. Über die Weboberfläche lässt sich zudem das Videobild der Kamera anzeigen.
4	Firmware	Auf der Firmware der Kamera laufen interne Anwendungen und Konfigurationen die zum Betrieb der Kamera notwendig sind.
5	WLAN	Die Kamera besitzt eine WLAN-Schnittstelle, für eine Einbindung in Drahtlose Netzwerke.
6	Mobile Anwendung	Die Kamera lässt sich mittels einer mobilen Anwendung steuern.
7	SD-Kartensteckplatz	Die Kamera besitzt einen SD-Kartensteckplatz, für die Speicherung von Videodaten.

Tabelle 6.2: Mögliche Angriffsziele einer *ieGeek* Überwachungskamera

Im nächsten Schritt können mögliche Bedrohungen (Threats) identifiziert werden. Dies kann mittels Methoden wie STRIDE ausführlich in tabellarischer Form geschehen, oder auf oberflächlicher Ebene als stichpunktartige Liste.[21] Einige Bedrohungen für eine Überwachungskamera könnten etwa das Abfangen des Videosignals zum Ausspähen von Nutzern sein, ein DoS-Angriff zum Verhindern der Videoüberwachung oder das Entwenden der SD-Karte und des darauf befindlichen Videomaterials.

Im Anschluss daran werden die einzelnen Bedrohungen mit Beschreibungen, möglichen Ziele, Angriffstechniken und Gegenmaßnahmen ausführlich dokumentiert und mittels einer Methode wie DREAD bewertet.[22] Mit diesen Informationen kann nun der Umfang des Penetrationstests geplant und festgelegt werden.

[21] Vgl. 26, S. 51.
[22] Vgl. 26, S. 51 ff.

6.3.2 Umfang

Die Festlegung des Umfangs (Scope) bildet die Grundlage für die Durchführung eines Penetrationstestes. Der Umfang legt die Grenzen, als auch die Ziele für einen erfolgreichen Penetrationstest fest. Durch den definierten Umfang wird die effektivität eines Penetrationstests direkt beeinflusst. Wird ein zu großer Umfang festgelegt, können die einzelnen Komponenten mangels ausreichender Zeit möglicherweise nicht intensiv genug getestet werden. Wird ein zu geringer Umfang festgelegt, können Schwachstellen in ungetesteten Komponenten unentdeckt bleiben. Durch eine gezielte Festlegung des Umfangs werden zudem auch Störungen im Unternehmen vermieden, indem etwa Produktivsysteme ausgeschlossen und der Test auf Testsysteme beschränkt wird.

Um den Umfang in geeigneter Weise zu bestimmen, sollte die Dauer des Penetrationstestes festgelegt sein. Der Zeitraum sollte möglichst so gewählt werden, dass der laufende Betrieb des Unternehmens so unterbrechungsfrei und störungsfrei wie möglich bleibt. Anschließend lässt sich basierend auf der zur Verfügung stehenden Zeit in Kombination mit den Erkenntnissen aus dem zuvor erstellten Threat Model der Umfang bestimmen. In Kontrast zu den zu testenden Komponenten (In-Scope), sollten auch Komponenten vom Test ausgeschlossen werden (Out-of-Scope). Hierzu können etwa Nicht-IoT-Systeme und -Netzwerke zählen, als auch die Bewertung der physischen Sicherheit, da es sinnvoll ist diese separat zu testen, um den Umfang pro Test nicht zu groß zu definieren. Auch Testszenarien die rechtlich problematisch sein könnten, etwa aufgrund von Datenschutzbedenken, oder aufgrund fehlender Genehmigungen Dritter, wie Cloud-Anbietern oder Dienstleistern.

Neben der Festlegung des reinen Umfangs, sollten auch Verhaltensregeln für den Penetrationstest festgelegt werden (Rules of Engagement). Dies dient dazu sicherzustellen, dass der Penetrationstest kontrolliert und ethisch einwandfrei durchgeführt wird. Es muss zu jederzeit sichergestellt sein, dass alle Beteiligten sich an Gesetze halten und all e nötigen Genehmigungen eingeholt wurden. Zudem sollten auch alle störenden Aktivitäten in den Regeln bedacht werden, um Schäden und Unterbrechungen des laufenden Betriebs zu vermeiden. Gängige Praxis es beispielsweise Angriffe aus dem Testumfang auszuschließen, die nur die Unterbrechung des Betriebs von Systemen und Geräten zum Zweck haben, sogenannte DoS.

6.3.3 OSINT Recherche

Im Anschluss an das Erstellen eines ausführlichen Threat Models und der Definition des Umfanges, kann auf dieser Grundlage eine umfangreiche OSINT Recherche durchgeführt werden, um bereits vorab wichtige Informationen zu gewinnen. Unter OSINT versteht man das Zusammentragen und Analysieren von frei zugänglichen Daten.[23] Im Rahmen von IoT-Penetrationstests kann eine OSINT-Recherche dazu verwendet werden, Informationen und Daten wie Bedienungsanleitungen, Firmwares, Informationen über Hardware und Elektronik oder bereits bekannte Schwachstellen mit relativ geringem Aufwand zu ermitteln, ohne selbst große Mengen an Arbeit zu investieren.[24] Dies spart Zeit und hilft bei einer guten

[23] Vgl. 12, S. 45.
[24] Vgl. 12, S. 45 f.

Vorbereitung des Penetrationstests, ohne dass das Gerät physisch zur Verfügung stehen muss.[25]

Geeignete Quellen für eine OSINT Recherche können Plattformen wie *GitHub (https://github.com)* sein, auf denen unter anderem Entwickler und Sicherheitsforscher Informationen teilen und Mitarbeit von anderen Nutzern der Plattform zulassen. Speziell im IoT-Umfeld angesiedelt ist etwa die zuvor bereits erwähnte Suchmaschine *shodan.io (https://shodan.io)*, mit der sich unter Zuhilfenahme verschiedener Filter IoT-Geräte im Internet suchen lassen. Aber auch Herstellerwebseiten und die Verwendungen von Suchmaschinen wie *Google (https://google.de)* können hilfreiche Informationen liefern. Im Bereich der mobilen Apps sind vor allem Android Applikationen eine nützliche Informationsquelle, da diese sich mit geringem Aufwand dekompilieren lassen und so viele Informationen preisgeben.[26] Bei der Dokumentation der Ergebnisse der OSINT-Analyse sollte zwingend darauf geachtet werden die Quellen der Informationen anzugeben, da diese zu einem späteren Zeitpunkt sonst schwer nachvollziehbar sein können.[27]

Speziell im Bereich Hardware kann eine OSINT Recherche zudem hilfreich sein, um für den Penetrationstest benötigte Werkzeuge bereits im Vorfeld zu identifizieren und zu beschaffen.[28] Dies können beispielsweise Schraubendreher für spezielle Schraubentypen sein, welche sich bei sogenannten *Teardowns*, dem gezielten Zerlegen von Hardware, über Webseiten wie *iFixit (https://ifixit.com)* herausfinden lassen. Hier finden sich häufig auch Fotos der Platinen von Geräten, über welche sich einzelne darauf befindliche Bauteile identifizieren lassen.

6.3.4 Testreihenfolge

Nach dem Festlegen des Umfangs, kann die Reihenfolge der einzelnen praktischen Tests geplant werden. Die richtige Struktur für einen Penetrationstest von IoT-Geräten zu finden kann eine Herausforderung darstellen, da dieses Feld relativ jung ist und die Menge an Erfahrungen noch nicht so groß ist, wie etwa bei Penetrationstests für Webanwendungen.[29] Je nach Größe des Teams und geplantem Umfang, kann es sinnvoll sein, mehrere Tests gleichzeitig durchzuführen. So kann etwa ein Team, welches auf die Analyse von mobilen Anwendungen spezialisiert ist zum gleichen Zeitpunkt testen, wie ein Team welches auf die Analyse von Webanwendungen spezialisiert ist.[30] Somit wird ein Penetrationstest zeit-effizienter durchgeführt und die Aufgaben können nach Expertise an die jeweiligen Tester verteilt werden. Es sollte aber vor allem auf eine sinnvolle Reihenfolge bei der Testplanung geachtet werden, um den Erfolg des Penetrationstests zu maximieren. Da bei IoT-Penetrationstests auch physische Hardware involviert ist, sollte darauf geachtet werden diese so lange wie notwendig vor Zerstörung und Beschädigung zu schützen. Denn sollte nur eine begrenzte Menge an physischen Geräten, oder gar nur ein einziges Gerät zur Verfügung stehen, so kann eine Beschädigung

[25] Vgl. 12, S. 45 f.
[26] Vgl. 12, S. 46.
[27] Vgl. 12, S. 47.
[28] Vgl. 12, S. 46.
[29] Vgl. 54, S. 33 f.
[30] Vgl. 54, S. 35 f.

den Penetrationstest vorzeitig beenden und das Auffinden von Schwachstellen verhindern. Aus diesem Grund sollten zunächst weniger destruktive Tests durchgeführt werden, wie solche an der Software des Gerätes.

Aber auch beim Testen der Software von IoT-Geräten kann die richtige Reihenfolge von wichtiger Bedeutung sein. So kann es sein, dass Geräte bei erstmaliger Inbetriebnahme wichtige Konfigurationsdateien aus dem Internet herunterladen, ein unverschlüsselter Schlüsselaustausch stattfindet, oder aber Software- oder Firmware-Aktualisierungen stattfinden. Diese möglicherweise nur einmalig stattfindenden Aktivitäten können wichtige Informationen enthalten und sollten nicht verpasst werden. Nachdem die Tests der Softwarekomponenten abgeschlossen sind, kann mit der Untersuchung der Hardware fortgefahren werden. Dies kann die Demontage des Gerätes sein, oder aber auch das Auslöten von Speicherbausteinen, bei denen das Gerät unbenutzbar gemacht werden könnte. Die Reihenfolge des Penetrationstests sollte aus diesen Gründen im Vorfeld definiert und basierend auf in der Vergangenheit gemachten Erfahrungen stetig angepasst und verbessert werden.

6.3.5 Dokumentation

Eine umfassende Dokumentation während der Durchführung eines Penetrationstestes ist essenziell, um jederzeit den aktuellen Stand abfragen zu können. Zudem bietet die Dokumentation die wichtigste Grundlage für das spätere Erstellen des Penetrationstestberichtes. Es ist daher wichtig, eine Struktur für die laufende Dokumentation festzulegen, mit welcher alle Beteiligten gut arbeiten können. Eine mögliche Ordnerstruktur könnte daher wie folgt aussehen:

```
IoT-Pentest-Vorlage/
├── OSINT-Recherche/
│   ├── Hardware/
│   │   ├── Gehäuse
│   │   └── Platine
│   ├── Software/
│   │   ├── Firmware
│   │   ├── Webinterface
│   │   └── Mobile Apps
│   ├── Netzwerk/
│   │   ├── Mitschnitte
│   │   └── Schnittstellen
│   └── Cloud
├── Penetrationstest-Daten/
│   ├── Hardware/
│   │   ├── Gehäuse
│   │   └── Platine
│   ├── Software/
│   │   ├── Firmware
│   │   ├── Webinterface
│   │   └── Mobile Apps
│   ├── Netzwerk/
│   │   ├── Mitschnitte
│   │   └── Schnittstellen
│   └── Cloud
└── Report/
```

Diese Struktur gliedert die Informationen nach den verschiedenen Komponenten von IoT-Systemen. Diese Komponenten werden für gewöhnlich separat getestet und somit sind alle Teiltests strukturell voneinander abgegrenzt. Diese Struktur wirkt bei der späteren Erstellung des Berichtes unterstützend, da auch der Bericht nach den einzelnen Komponenten strukturiert werden sollte. Auch der Ort an dem die Dokumentation abgelegt wird, kann eine Rolle spielen. Hier ist es sinnvoll, die Ordnerstruktur in einem Versionskontrollsystem wie Git abzulegen. Somit können mehrere Penetrationstester gemeinsam an einem Penetrationstest arbeiten und dabei jederzeit auf den gleichen Informationsstand zugreifen. Dies kann hilfreich sein, wenn wie zuvor erwähnt verschiedene Teams gleichzeitig unterschiedliche Komponenten testen, da somit jedes Team Einsicht in die Ergebnisse der anderen Teams hat. Zudem wird eine Historie über alle Änderungen abgespeichert, wodurch der genaue Ablauf des Penetrationstests stets nachvollziehbar bleibt und fehlerhafte Änderungen an der Dokumentation rückgängig gemacht werden können. Des Weiteren ist es bei gängigen Versionskontrollplattformen wie GitHub (http://github.com) möglich eine zuvor definierte Projektstruktur für Penetrationstests als Vorlage einzurichten und damit eine stets einheitliche und einfach zu verwendende Struktur sicherzustellen. Als Teil dieser Vorlage können soge-

nannte *Markdown*-Dateien in der Ordnerstruktur abgelegt werden, um für jede Teilstruktur sinnvolle Informationen wie Referenzen zu Informationsquellen, nützliche Programme zum Testen von Komponenten, oder aber auch Vorgehensweisen und Erfahrungen aus vergangenen Penetrationstests zur Verfügung zu stellen. Sofern diese Dateien unter dem Dateinamen *README.md* abgelegt und in der Auszeichnungssprache Markdown geschrieben werden, zeigen die Weboberflächen von Versionskontrollsystemen diese unterhalb der Ordnerstruktur als formatierten Text an, ohne dass diese Dateien zuvor geöffnet werden müssen. Wie ein solches Repository im Browser dargestellt wird, ist beispielhaft in der folgenden Abbildung 6.2 zu sehen.

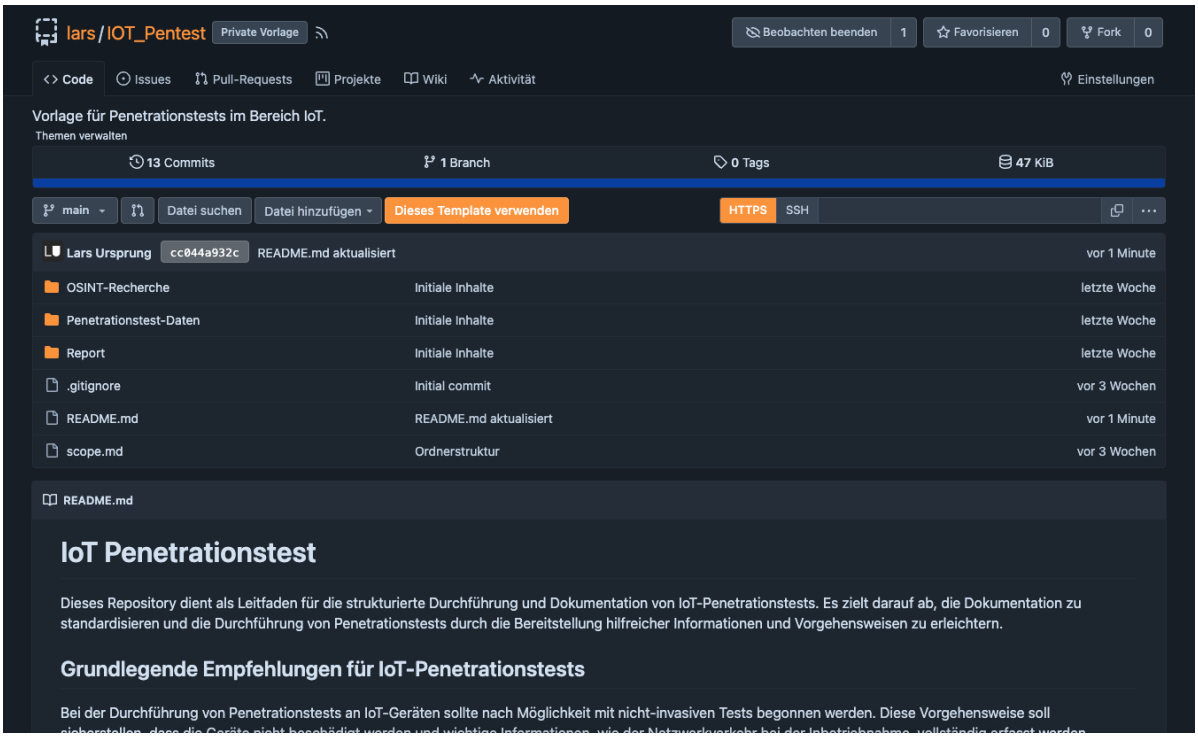

Abbildung 6.2: Darstellung des Git-Repositories im Browser unter Verwendung des Versionskontrollsystems Forgejo (https://forgejo.org/) (Eigene Darstellung)

So können eine README-Dateien innerhalb des OSINT-Verzeichnisses wichtige OSINT-Informationsquellen aggregieren, die bei der Recherche hilfreich sein könnten. Vereinfachte Beispiele für solche README-Dateien finden sich im Anhang unter *README-Dateien*, mit der jeweiligen Pfadangabe für die zuvor definierte Ordnerstruktur. Mittels dieses Vorgehens lassen sich Informationen, Vorgehensweisen und Vorlagen an den richtigen Stellen zentral bündeln und damit eine Fragmentierung verhindern. Somit entfällt das Suchen von Informationen an unterschiedlichen Quellen und sorgt zudem dafür, dass auch im Nachhinein nachvollzogen werden kann, welcher Informationsstand zum Zeitpunkt des Penetrationstests zur Verfügung stand.

6.3.6 Berichterstellung

Die letzte Phase eines Penetrationstests ist die der Berichterstellung (Reporting). Im Penetrationstestbericht werden alle wichtigen Erkenntnisse des Penetrationstests festgehalten. Zudem wird dokumentiert, welche Methoden und Techniken während des Tests verwendet wurden. Allen gefundenen Schwachstellen wird mittels eines Bewertungsstandards wie CVSS ein Schweregrad zugeordnet. Auch werden Empfehlungen gegeben, wie die gefundenen Schwachstellen behoben werden können und die Sicherheit des getesteten Systems oder Gerätes im Allgemeinen verbessert werden kann, wenn dies möglich ist. Der Penetrationstestbericht wird aus den Informationen der zuvor erstellten Dokumentation geschrieben.

Es empfiehlt sich den Bericht in die folgenden Abschnitte zu unterteilen:

- Zusammenfassung in einfacher Sprache (Executive Summary)
  - In der Management-Zusammenfassung, der sogenannten Executive Summary, werden die Ergebnisse des Penetrationstests kurz in einfacher Sprache zusammengefasst. Die Sprache wird dabei so gewählt, dass auch Personen mit geringen technischen Kenntnissen sie verstehen und bewerten können.
- Methodik
  - In diesem Teil des Berichtes werden alle während des Penetrationstests verwendeten Methoden aufgelistet und erläutert. Hierbei soll auch nachvollzogen werden können, warum die Methoden verwendet wurden.
- Gefundene Schwachstellen (Findings)
  - Dieser Teil des Berichtes listet alle gefundenen Schwachstellen und Auffälligkeiten detailliert auf. Hierbei ist es wichtig, dass die Schwachstellen genau beschrieben werden und alle Schritte zur Reproduktion verständlich und lückenlos enthalten sind. Zudem wird die Art der Schwachstelle klassifiziert, sowie der Schweregrad der Schwachstelle wird angegeben. Für den Schweregrad empfiehlt sich die Verwendung eines Standards wie CVSS, um eine Vergleichbarkeit der Schwachstellen untereinander zu gewährleisten.
- Empfehlungen
  - Unter Empfehlungen sollten die empfohlenen Schritte zur Behebung der einzelnen Schwachstellen oder Auffälligkeiten gegeben werden. Dieser Teil richtet sich explizit an Personen wie Entwickler oder Administratoren, die für die Behebung verantwortlich sind.
- Anhang
  - Im Anhang können alle Daten aufgeführt werden, die ergänzende Informationen liefern, aber die Übersicht des eigentlichen Berichtes reduzieren würden. Auch Logdaten, oder Hintergrundinformationen können hier angeführt werden.

6.3.7 Anpassung der Methodik an unterschiedliche IoT-Geräte

Die IoT-Landschaft ist sehr vielfältig und besteht aus unterschiedlichsten Geräten in den verschiedensten Ausprägungen. Dies spiegelt sich auch auf die Durchführung von Penetrationstests in diesem Bereich wider. Ein Standard für Penetrationstests im Bereich IoT sollte daher flexibel genug sein, um auf unterschiedlichste Geräte wie smarte Lautsprecher, oder vernetzte Autos anwendbar zu sein. Ein vollständig einheitlicher und spezifischer Ansatz ist in diesem Umfeld also nicht sinnvoll, denn je spezifischer die Methodik wird, umso unflexibler wird sie auch. Es sollte daher ein eher genereller Ansatz gewählt werden, der ein Rahmenwerk vorgibt, welches sich auf möglichst viele Geräte anwenden lässt.

Aus diesem Grund wurde für die zuvor entwickelte Methode ein eher generischer Ansatz gewählt, welcher Penetrationstests in die Grundkomponenten der meisten IoT-Geräte aufteilt. Diese Grundkomponenten sind Hardware, Software, Netzwerk und Cloud. Festgelegt wird dies als Ordnerstruktur innerhalb eines Git-Repositories samt ergänzenden Informationen. Dieses Repository wird anschließend als Vorlage festgelegt und kann stetig wiederverwendet werden. Für einzelne Penetrationstests kann diese Vorlage geklont und verwendet werden, ohne dabei das Original zu verändern. Während eines Penetrationstests kann die Ordnerstruktur abgestimmt auf das zu testende Gerät innerhalb der Grundkomponenten erweitert werden. So ließe sich die Grundkomponente Netzwerk mit Netzwerktechnologien oder Protokollen wie etwa ZigBee bei Bedarf erweitern. Innerhalb aller Komponenten können mittels der sogenannten README-Dateien Informationen über Vorgehensweisen, häufige Schwachstellen, nützliche Programme und mehr abgelegt werden. Informationen sollten dabei nach Möglichkeit nur referenziert werden, um die Übersichtlichkeit zu gewährleisten. Dies hat zudem den Vorteil, dass nur Referenzen zu Informationen gepflegt werden müssen, statt einer ausführlichen Sammlung an Informationen und Vorgehensweisen. Neue Informationen können anschließend per Pull Request in die Vorlage übernommen werden, wodurch diese stetig aktualisiert wird und für immer mehr Gerätetypen verwendbar wird. Durch die Zentralität von Informationen und Vorgehensweisen, bleiben die Penetrationstests stets konsistent und eine Kollaboration aller Beteiligten wird vereinfacht. Die Methode ist somit effektiv von Unternehmen jeglicher Größe und Penetrationstestern mit unterschiedlichen Kenntnissständen einsetzbar und anpassbar. Auch die Effizienz der Methode bleibt durch das zuvor beschriebene Vorgehen konstant, da weder zu große und komplexe Mengen an Informationen vorhanden sind, noch zu wenige Informationen.

6.3.8 Vergleich der entwickelten Methode mit bestehenden Methoden

Im Folgenden wird die zuvor entwickelte Methodik mit den anfangs vorgestellten bereits existierenden Methoden verglichen. Dies dient dazu, Vorteile und Nachteile der eigens entwickelten Methodik durch den Vergleich mit bestehenden Methoden aufzudecken und zu evaluieren.

Ähnlich wie der **OWASP IoT Security Testing Guide** deckt die zuvor entwickelte Methode verschiedene Komponenten aus dem Bereich IoT ab und lässt sich flexibel auf verschiedene IoT-Geräte anwenden und bei zukünftigen Entwicklungen gut anpassen. Auch enthalten

sind praktisch umsetzbare Techniken und Methoden, die dabei helfen einen Penetrationstest effektiv durchzuführen. Allerdings ist die Aufteilung in die verschiedenen Komponenten von IoT-Geräten weniger granular als im OWASP IoT Security Testing Guide. Somit ist die Methode leichter anzuwenden und bleibt bei eventuellen zukünftigen Veränderungen im IoT-Bereich leichter anpassbar. Zudem sind keine detaillierten Checklisten enthalten, sondern generisch gehaltene Hilfestellungen. Dies ist der Fall, da detaillierte Checklisten zum stumpfen abarbeiten anregen, ohne die Vorgehensweisen zu hinterfragen oder bei Bedarf anzupassen. Generisch gehaltene Hilfestellungen fördern das Entwickeln eigener auf das zu testende System angepasste Vorgehensweisen.

Wie auch der OWASP IoT Security Testing Guide verfolgt die zuvor entwickelte Methode einen gemeinschaftsorientierten Ansatz, bei dem Penetrationstester gemeinsam an der Weiterentwicklung beteiligt sind. Somit können auch hier die enthaltenen Informationen in Tiefe und Konsistenz schwanken. Ein Vorteil gegenüber dem OWASP IoT Security Testing Guide ist allerdings das Vorgeben einer sinnvollen Reihenfolge. Diese sorgt für eine effizientere Durchführung von Penetrationstests und beugt einer zu frühen Beschädigung des zu testenden Gerätes und dem Verlust von wichtigen Informationen vor. Zudem kann der gemeinschaftsorientierte Ansatz mit der Zeit zu einer Ansammlung von großen Mengen an Informationen und damit einhergehend hoher Komplexität führen. Hier kann eine empfohlene Reihenfolge dabei helfen, schneller durch die Menge an Informationen zu navigieren, ohne die Penetrationstester gleich zu beginn zu überfordern.

Verglichen mit dem **BSI-Penetrationstest-Modell** ist die eigens entwickelte Methode weniger umfangreich und spezifisch in der Anwendung. Das BSI-Modell enthält umfangreiche Checklisten und organisatorische Empfehlungen für die Durchführung von Penetrationstests und benötigt damit einen größeren Aufwand, für die Integration in Unternehmen. Zudem verfolgt es einen generischen Ansatz, der versucht die Vorgehensweise von Penetrationstests im Allgemeinen abzubilden, wobei der IoT-Bereich gänzlich außer acht gelassen wird. Auch dem Testen von Hardware wird im BSI-Modell demzufolge keine Beachtung geschenkt. Dennoch gibt es einige Gemeinsamkeiten, mit der eigens entwickelten Methode. So wird bei beiden Methoden großer Wert auf eine strukturierte und systematische Herangehensweise an Penetrationstests gelegt. Auch den Modularen und flexiblen Aufbau, mit Handlungsempfehlungen und dem möglichen Einsatz von Checklisten für einzelne Bereiche von Penetrationstests haben beide Methoden miteinander gemein. Allerdings hat die eigens Entwickelte Methode im Gegensatz zum BSI-Modell nicht den Anspruch ein möglichst breites Spektrum an IT-Systemen abzudecken, sondern fokussiert sich speziell auf IoT-Systeme und geht im Gegensatz zum BSI-Modell auch auf die praktische Durchführung und ein. Da die eigens entwickelte Methode viel Wert auf eine gute und einfache Anpassbarkeit legt, lassen sich Bestandteile des BSI-Modells bei Bedarf jedoch leicht integrieren, wobei sich hier die Gemeinsamkeiten beider Methoden positiv bemerkbar machen.

Im Gegensatz zum **IoT Pentesting Guide by Aditya Gupta** dient die eigens entwickelte Methode nicht nur als statisches Nachschlagewerk. Sie dient als interaktiver Arbeitsbereich, welcher alle nötigen Informationen an den Stellen zur Verfügung stellt, an denen sie gebraucht werden. Zudem kann die Methode eigenständig je nach Bedarf angepasst und erweitert

werden, wodurch die Abhängigkeit von einer zentralen Verwaltungsinstanz verringert wird. Derzeit befindet sich der IoT Pentesting Guide by Aditya Gupta noch in einer frühen Phase der Entwicklung und es ist nicht absehbar, wann neue Informationen hinzugefügt werden und welche Informationen hinzugefügt werden. Zudem wird auch hier im Gegensatz zur eigens entwickelten Methode noch keine Struktur vorgegeben, mit der sich Penetrationstests effektiv planen lassen.

Das **IoT Security Assurance Framework** fokussiert sich darauf, eine umfassende Sammlung an Richtlinien und Praktiken für die Bewertung der Sicherheit von IoT-Geräten bereitzustellen. Die Zielgruppe dieses Frameworks sind daher nicht explizit Penetrationstester, sondern vor allem Manager, sowie Entwickler und Hersteller von IoT-Geräten. Das Framework bietet einen umfassenden Fragenkatalog und Vorgehensweisen zur Bewertung der Sicherheit, ohne diese dabei praktisch wie in einem Penetrationstest zu untersuchen. Zudem ist das Framework als statisches Portable Document Format (PDF)-Dokument organisiert, was einen interaktiven Einsatz und die Anpassung auf die eigenen Bedürfnisse erschwert. Die eigens entwickelte Methode ist hier flexibler und lässt sich in der Praxis schneller zum Einsatz bringen. Das IoT Security Assurance Framework ließe sich allerdings als ergänzende Informationsquelle unterstützend in die eigene Methode einbinden, um bei statischen Analysen der Sicherheit von IoT-Geräten zu unterstützen. Der Fokus beider Methoden ist allerdings aufgrund der unterschiedlichen Zielsetzungen recht unterschiedlich und daher nur bedingt vergleichbar.

Auch die **NIST Special Publication 800-183 (Networks of 'Things')** lässt sich nur bedingt mit der eigens entwickelten Methodik vergleichen, da auch hier der Fokus nicht ausschließlich auf der Durchführung von Penetrationstests liegt. Der Fokus liegt vor allem darauf, die grundlegenden Aspekte vernetzter Geräte und Systeme zu beschreiben und eine standardisierte Nomenklatur für diese Systeme zu schaffen. Ein direkter Vergleich ist dabei recht schwierig, allerdings lassen sich die darin enthaltenen Informationen in die eigens entwickelte Methodik einbinden oder referenzieren, um während der Penetrationstests ein gemeinsames Verständnis für verwendete Begrifflichkeiten zu haben und Missverständnissen vorzubeugen. Das Framework eignet sich also bestens dafür, in andere Methoden für das Penetrationstesten von IoT-Geräten eingebunden zu werden, um Grundlagenwissen und ein gemeinsames Vokabular zu etablieren, was die Effektivität der Penetrationstests steigern kann.

Besser für einen Vergleich eignen sich Bücher zum Thema Penetrationstests von IoT-Geräten, wie das **IoT Penetration Testing Cookbook by Aaron Guzman and Aditya Gupta**. In diesem Buch wird ein breites Wissen zum Thema Penetrationstesten von IoT-Geräten vermittelt. Das Buch deckt viele praktische Techniken ab, die in IoT Penetrationstests Anwendung finden ab. Allerdings geht das Buch nicht auf den organisatorischen Ablauf von Penetrationstests ein und wie IoT Penetrationstests effektiv in einem Unternehmen implementiert werden.[31] Daher ist dieses Buch auch nur als Nachschlagewerk während der Durchführung von IoT Penetrationstests anzusehen. Besser ist hier das Buch **Praktische Einführung in Hardware Hacking** von Marcel Mangel und Sebastian Bicchi. Dieses Buch enthält nicht nur umfassende Informationen zum Thema IoT Penetration Testing, sondern geht auch darauf ein wie sich der Ablauf eines IoT Penetration Test sinnvoll gestalten lässt. So wird etwa explizit erklärt,

[31] Vgl. 54, S. 7 ff.

dass es sinnvoll ist den Datenverkehr der ersten Inbetriebnahme aufzuzeichnen, um eventuelle Aktualisierungen oder die Registrierung des Gerätes mit dem Server des Herstellers aufzuzeichnen.[32] Im Allgemeinen ist das Buch eine gute Informationsgrundlage und bietet umfassende Informationen zum Thema IoT Penetrationstests. Als Nachteile sind lediglich das Fehlen von einer Methodik für eine effektive Zusammenarbeit im Team zu erwähnen, sowie die statische Natur von Büchern im allgemeinen. Die stetige Aktualisierung der Informationen ist hier nicht gesichert, oder kann längere Zeit in Anspruch nehmen, bis eine aktualisierte Ausgabe herausgegeben wird.

Die statische Natur von Büchern und das Risiko, dass diese mangels Nachfrage nicht mit Sicherheit regelmäßig aktualisierte Auflagen erscheinen, ist ein Nachteil von Büchern als Grundlage für eine langfristig effektive Penetrationstest-Methodik. Bücher können allerdings als Informationsgrundlage dienen, um eine eigene Methodik zu entwickeln. Des Weiteren eignen sich gut recherchierte Bücher hervorragend als Nachschlagewerk während der Durchführung eines Penetrationstests.

Im Vergleich zu bestehenden Methoden spielt die selbst entwickelte Methode also vor allem ihre Vorteile im Bereich der Flexibilität und der einfachen praktischen Anwendbarkeit aus. Die Methode lässt sich leicht erweitern und bestehende Informationen können referenziert werden und sind nicht statisch in die Methodik eingebunden. Somit lässt sich eine große Sammlung an Informationen einbinden, ohne Penetrationstester zu überfordern oder die Übersichtlichkeit zu gefährden. Die Verwendung einer GitHub-Vorlage als Grundlage schafft eine wohldefinierte Struktur und sorgt für eine gute Vergleichbarkeit der Penetrationstests untereinander. Des Weiteren wird die Zusammenarbeit im Team erleichtert, was besonders in Anbetracht der Komplexität von IoT-Geräten von Vorteil ist. Da diese Geräte sich aus unterschiedlichsten Komponenten von Webanwendungen bis hin zu physischen Hardwarekomponenten zusammensetzen, die sich somit technologisch stark voneinander unterscheiden können, ist die effektive Zusammenarbeit im Team von großer Bedeutung. Die Organisation spielt in bestehenden Penetrationstestmethoden für IoT-Geräte kaum eine Rolle, obwohl diese von entscheidender Bedeutung ist. Dennoch hat die entwickelte Methodik nicht den Anspruch anderen Methoden überlegen zu sein, sondern ermöglicht es diese einfach zu integrieren und so alle benötigten Informationen dort bereitzustellen wo sie gebraucht werden und gleichzeitig flexibel zu bleiben. Somit lassen sich die Vorteile bestehender Methoden leicht übernehmen und auf die eigens entwickelte Methode übertragen, ohne diesen Prozess kompliziert und unflexibel zu gestalten.

6.4 Anpassbarkeit der Methode an zukünftige Entwicklungen

Angesichts der stetigen Weiterentwicklungen von IoT-Technologien muss die zuvor entwickelte Penetrationstest-Methode kontinuierlich weiterentwickelt werden, um auch zukünftig sinnvoll einsetzbar zu sein. Zukünftige Entwicklungen könnten beispielsweise neue Netzwerkprotokolle und Funkstandards sein, die bei Penetrationstests mit eingezogen werden müssen. Aber auch neue Hardwarearchitekturen oder bisher ungenutzte Anwendungsbereiche von

[32] Vgl. 12, S. 20 ff.

IoT müssen bedacht werden. Ein weiteres beispiel könnte generative künstliche Intelligenz (KI) sein. Diese findet immer häufiger Einsatz in unterschiedlichsten Szenarien und könnte auch bei IoT Penetrationstests zum Einsatz kommen. So eignet sich generative KI etwa für den Einsatz bei statischen Quellcode Analysen und könnte somit auch in IoT Penetrationstests genutzt werden.[33] So wäre vorstellbar, dass Repository mit KI-Fragestellungen (Prompts) für effektive Quellcode Analysen zu erweitern, um dabei möglichst effektiv zu sein. Hierfür bietet die Entwickelte Methodik eine solide Grundlage, da sie eine klare, aber anpassbare Struktur vorgibt, die ohne großen Aufwand und Verlust von Struktur erweitert werden kann. Gleichzeitig ist die Methodik nicht zu spezifisch und detailliert, um Anpassungen für zukünftige Entwicklungen zu komplex zu machen, oder IoT-Geräte auszuschließen. Die Methodik versucht nicht die Rolle eines umfassenden Nachschlagewerkes einzunehmen, sondern referenziert vorhandene Informationen und bündelt sie sinnvoll an den Stellen, an denen diese Informationen von Penetrationstestern benötigt werden. Fehlende Informationen können aufgrund des Aufbaus als Git-Repositories mit geringem Aufwand mittels *Pull-Request* in den Standard mit aufgenommen werden. Dies könnte entweder zentral in einem generell zugänglichem Git-Repository geschehen oder aber auch in kleinerem Maßstab innerhalb von unternehmensinternen Git-Umgebungen individuell pro Unternehmen. Dies macht die Methode für Unternehmen jeglicher Größe, als auch für einzelne Personen anwendbar. Zudem wird durch die Basis eines Git-Repositories die Zusammenarbeit im Team erleichtert. Dies ist vor allem bei umfangreichen IoT-Penetrationstests von entscheidender Bedeutung, da IoT-Geräte durch die Verbindung von spezieller Hardware und individueller Software oftmals nicht von einzelnen Experten untersucht werden können, sondern mehrere Experten verschiedener Fachgebiete während eines Penetrationstests zusammenarbeiten müssen.

[33] Vgl. 63, S. 183.

7 Fazit

7.1 Zusammenfassung

(Keine erneute Argumentation, oder Beweisführung, Kommentierung!) In der vorliegenden Arbeit wurde die Sicherheit von IoT-Geräten analysiert, sowie Methoden für die Durchführung von Penetrationstests untersucht, sowie mit den gewonnenen Erkenntnissen eine neue Methodik für die Durchführung von Penetrationstests entwickelt. Ausgehend von einer grundlegenden Einführung in das Thema Penetrationstests und in das Thema IoT im Allgemeinen, wurden zunächst die wichtigsten Grundlagen beider Themengebiete vermittelt. Bei der Vermittlung der Grundlagen von Penetrationstests, wurde zunächst auf die verschiedenen Arten von Penetrationstests eingegangen. Diese bestehen aus Black-, White- und Grey Box Tests. Diese verschiedenen Arten von Penetrationstests, unterscheiden sich in der Informationsmenge die Penetrationstester über das zu testende System besitzen. Bei Black Box Tests sind keine bis wenig Informationen über das zu testende System verfügbar, wohingegen bei White Box Tests weitgehende Informationen wie etwa der Quellcode verfügbar sind. Anschließend wurde der generelle Ablauf von Penetrationstests erläutert.

Dieser ist üblicherweise in die Phasen der Planung, Ausführung und abschließenden Tätigkeiten aufgeteilt. In der Planungsphase wird ein Penetrationstest umfassend vorbereitet, um einen reibungslosen Ablauf zu gewährleisten. Dies umfasst das Beschaffen von Zugängen und Berechtigungen, die eventuell benötigt werden, als auch das Festlegen des Umfangs. Ein festgelegter Umfang ist notwendig, um einen Penetrationstest effektiv durchzuführen und nur Teile des Systems zu überprüfen, die für das Unternehmen relevant sind. Nach abgeschlossener Planung folgt die Ausführungsphase. In dieser wird der Penetrationstest praktisch durchgeführt. Dies geschieht mittels einer umfassenden Untersuchung des zu testenden Systems, unter Einsatz verschiedener Programme und Hilfsmittel, die teilweise speziell für das Durchführen von Penetrationstests geschaffen wurden. Während der Ausführungsphase werden alle gefundenen Schwachstellen und relevanten Informationen für die Phase der abschließenden Tätigkeiten dokumentiert. Diese ist die letzte Phase und dient dazu die Ergebnisse des Penetrationstests in einem ausführlichen Bericht festzuhalten und zu bewerten. Der Bericht ist an verschiedene Zielgruppen, wie Führungskräfte, Systemverantwortliche und Entwickler gerichtet und bildet die Grundlage für die Behebung der Schwachstellen. Aus dem Bericht werden konkrete Aufgabenstellungen abgeleitet, die zur Behebung der Schwachstellen innerhalb eines angemessenen Zeitrahmens führen und in einer Verbesserung der IT-Sicherheit des Unternehmens resultieren.

Nachdem der Vermittlung der Grundlagen zu Penetrationstests, wurden zusätzlich noch die Grundlagen des Internet of Things vermittelt, um ein solide Grundlage für das Verständnis der weiteren Kapitel der Arbeit zu vermitteln. Hier wurden IoT-Geräte zunächst als vernetzte

physische Objekte aus dem Alltag definiert, welche Daten sammeln und verarbeiten können. Dies können Geräte in Privathaushalten sein, wie etwa Fernseher, Überwachungskameras, Drucker oder Glühbirnen. Es können aber auch Geräte aus der Industrie sein, wie vernetzte Produktionsanlagen oder Systemen zur intelligenten Verkehrsdatenerfassung und Steuerung. In Privathaushalten wird der Einsatz von IoT-Geräten üblicherweise als *Smart-Home* bezeichnet, wohingegen der Einsatz von IoT-Geräten in Industrieunternehmen als *Industrie 4.0* bezeichnet wird.

Da IoT-Geräte oftmals über andere Netzwerkprotokolle und Dienste kommunizieren als klassische IT-Hardware, wurde anschließend ein Überblick über die wichtigsten Netzwerkprotokolle und Dienste aus dem Bereich IoT gegeben. Diese wurden zusätzlich den unterschiedlichen Schichten des TCP/IP Referenzmodells zugeordnet.

Anschließend an die Erläuterung der wichtigsten Netzwerktechnologien, wurde ein weiterer elementarer Bestandteil von IoT-Geräten, nämlich Firmwares und Betriebssysteme aufgezeigt. Bei der Firmware handelt es sich um eine zentral im Gerät abgespeicherte Software, welche die Kommunikation zwischen Anwendungssoftware und Hardware ermöglicht. Diese ist für den Endanwender nicht zugänglich, da sie eine technische Notwendigkeit ist, die keine direkte Interaktion mit dem Anwender benötigt. Ebenso wie die Firmwares von IoT-Geräten, sind auch die Betriebssysteme selbiger genau auf den Einsatz im IoT-Umfeld zugeschnitten. Dies resultiert aus den oft limitierten Ressourcen für Hardware und Energiebedarf solcher Geräte. So existiert mit den Betriebssystemen *TinyOS* und *Contiki* beispielsweise zwei Betriebssysteme, welche auf den Einsatz für Geräte in Sensornetzwerken ausgerichtet sind. Sie zeichnen sich vor allem durch einfache Programmierbarkeit in der Programmiersprache *NesC* aus. Aber auch das Betriebssystem RIOT ist aufgrund seiner Echtzeit-Funktionalität und des Mikrokernels, welche in besonders geringen Latenzen resultieren weit verbreitet.

Abgeschlossen wurde die Erläuterung der Grundlagen mit der Vermittlung von Basiswissen über IoT-Hardware. Diese hat häufig eine geringe physische Größe und muss vor allem bei Batteriebetriebenen Geräte äußert energieeffizient sein. Dies erreichen Hersteller oftmals unter Verwendung sogenannter SoCs, welche Bestandteile von klassischen Computern wie Prozessor, Funkmodule, Speicherbausteiner und weitere Peripherie auf einem einzigen Chip von geringer Größe bündeln. Aufgrund der unterschiedlichen Einsatzgebiete von IoT-Geräten, unterscheidet sich die Hardware von Gerät zu Gerät allerdings oft deutlich. Zudem ist auf der Hauptplatine von IoT-Geräten oft eine Schnittstelle namens UART integriert, welche sich mit der klassischen seriellen Schnittstelle von Computern vergleichen lässt und Entwicklern und Herstellern Debugging-Funktionalitäten bereitstellt. Auf diese Schnittstelle und ihre Relevanz in Penetrationstests wird zu einem späteren Zeitpunkt ausführlicher eingegangen.

Auf die erfolgreiche Vermittlung des nötigen Grundlagenwissens, wurden häufig vorkommende Schwachstellen im Bereich des Internet of Things vorgestellt. Als Grundlage hierfür wurde unter anderem die Liste der zehn häufigsten IoT-Schwachstellen nach OWASP verwendet. Zudem wurden unterschiedliche Schwachstellen in den verschiedenen Kategorien von IoT-Geräten voneinander abgegrenzt. Diese Abgrenzung war notwendig, da IoT-Geräte in unterschiedlichen Ausprägungen mit unterschiedlichen Komponenten vorkommen können. So haben einige IoT-Geräte integrierte Weboberflächen für die Administration, welche eine Reihe

von Schwachstellen mit sich bringen können, die nur in Webanwendungen vorhanden sind. Dies eröffnet eine Reihe neuer Möglicher Schwachstellen, wie solche aus der OWASP Web Top 10, welche die zehn häufigsten Schwachstellen in Webanwendungen auflistet. Andere Geräte haben vielleicht keine integrierte Weboberfläche zur Administration, aber dafür eine notwendige Cloud-Anbindung, die wiederum neue Schwachstellen aus dem Cloud-Bereich mit sich bringt. Durch diese Abgrenzung wurde aufgezeigt, dass nicht alle IoT-Geräte die gleichen Schwachstellen und Angriffsvektoren mit sich bringen, sondern diese sich von Gerät zu Gerät voneinander unterscheiden können. Somit konnte in diesem Kapitel nicht nur die Forschungsfrage nach den häufigsten Schwachstellen im Bereich IoT umfassend beantwortet werden, sondern auch die Forschungsfrage wie sich die Schwachstellen zwischen den verschiedenen IoT-Geräten unterscheiden. So wurden auch die wichtigsten übergreifenden Komponenten von IoT-Geräten identifiziert, in denen unterschiedliche Schwachstellen auftreten können. Diese bestehen aus der Hardware des Gerätes, der Software des Gerätes, welche sich zudem in die Unterkategorien Firmware, Webanwendungen und mobile Anwendungen aufteilen lässt, als auch Netzwerktechnologien und Cloud. Dies sind die wichtigsten Bestandteile für Penetrationstests an IoT-Systemen mit jeweils eigenen Schwachstellen.

Im Anschluss an die Identifizierung von möglichen Schwachstellen und in welchen Kategorien von IoT-Geräten diese auftauchen können, wurde vermittelt wie sich das zuvor erlangte theoretische Wissen praktisch einsetzen lässt. Hierfür wurde erläutert, wie die Durchführung von Penetrationstests an Hardware, Software, Netzwerk und Cloud umgesetzt werden kann. In diesem Zuge wurde zunächst die Untersuchung der Sicherheit der Hardware beschrieben. Hierbei wurde auch auf weniger offensichtliche Indikatoren für die Sicherheit von IoT-Hardware eingegangen, wie etwa die Verpackung des Gerätes und Aufkleber auf dem Gehäuse. Hier kann es nämlich vorkommen, dass Hersteller Details zur Netzwerk-Hardware für Dritte leicht ersichtlich außen anbringen. Mit solchen Informationen, wie etwa der MAC-Adresse lassen sich Geräte von potenziellen Angreifen leicht auffinden. Dieses Aufspüren von Geräten mittels der MAC-Adresse wurde zudem mittels der IoT-Suchmaschine *shodan.io* demonstriert. Als weitere Indikatoren dafür, wie ernst es ein Hersteller mit der Sicherheit seiner Geräte nimmt, wurden Merkmale wie Spezialschrauben, spezielle Verklebungen oder holographische Siegel benannt, welche eine eventuelle Manipulation ersichtlich machen sollen. Neben dieser eher optischen und weniger komplexen Untersuchung der Hardware wurden auch fortgeschrittene und invasive Techniken vorgestellt, wie etwa das Auffinden und Benutzen der Debugging-Schnittstellen wie UART. In diesem Zuge wurde nicht nur theoretisch erläutert, wie sich diese Schnittstellen unter Einsatz eines Multimeters auf der Platine des Gerätes auffinden lassen, sondern dieses Vorgehen wurde auch praktisch an einer Netzwerkkamera und einer Smart Home Bridge demonstriert. Dabei wurden die Geräte physisch untersucht und es wurde mittels eines USB zur UART Konverters eine Konsolenverbindung auf einem Computer mit den Geräten hergestellt, um über eine Kommandozeile mit diesen zu interagieren und nützliche Informationen auszuleiten.

Darauf folgend wurde die Untersuchung der Softwarekomponenten im Detail erläutert. Neben Erläuterungen zur Firmwareanalyse, wurde ein besonderes Augenmerk auf die Untersuchung von Webanwendungen gerichtet. Diese stellen ein eigenes umfangreiches

Feld im Bereich der Penetrationstests dar und benötigen spezielles Fachwissen. Auch hier wurden einige wichtige Schwachstellen und Techniken zu deren Identifizierung praktisch demonstriert. So wurde etwa unter Zuhilfenahme der Software *Burp Suite* demonstriert, wie sich Fehler in der Zugriffskontrolle einer Anwendung identifizieren lassen. Aber auch Schwachstellen wie Structured query language (SQL)-Injektionen und der Einsatz schwacher Kryptografie wurden praktisch demonstriert. Als Grundlage für diese Demonstrationen wurde die absichtlich verwendbare Nachbildung eines Online-Shops *OWASP Juice Shop* verwendet. Aber auch die Netzwerkkamera deren Hardware zuvor bereits untersucht wurde, wurde erneut herangezogen. Diese verfügte über eine Weboberfläche, welche aufgrund von schwacher, beziehungsweise Fehlender Kryptografie leicht angreifbar war. Die Weboberfläche verzichtet auf eine verschlüsselte Datenübertragung, wodurch sich Anmeldevorgänge oder das Ändern von Passwörter von Dritten im Netzwerk im Klartext mitlesen lassen.

Da viele IoT-Geräte zu Steuerung auch Smartphone Anwendungen benötigen, wurde auch deren Untersuchung Beachtung geschenkt. Dabei wurde demonstriert, wie sich mittels der *Software Mobile Security Framework* APK-Dateien von Android Anwendungen analysieren lassen. Diese Software erleichtert die Untersuchung von Mobilen Anwendungen von iOS und Android Systemen und kann automatisiert auf einfache Schwachstellen und fest einprogrammierte API-Schlüssel und Passwörter hinweisen. Somit lassen sich Penetrationstests an mobilen Anwendungen schnell und ohne umfassende Vorkenntnisse durchführen.

Ein zentraler Aspekt von IoT-Geräten ist die Netzwerkkommunikation. Daher wurden auch mögliche Angriffstechniken auf diese beschrieben. In diesem Zuge wurden Techniken zum Attackieren von Bluetooth-Verbindungen erwähnt. So wurde etwa aufgezeigt, dass sich die Sicherheitsmaßnahme des Channel Hoppings mittels Programmen wie *Ubertooth One* aushebeln lassen oder, dass sich Replay-Angriffe mit dem Programm *gatttool* durchführen lassen, welches unter Linux meist vorinstalliert ist. Auch wurde geklärt, dass sich ZigBee angreifen lässt, indem beim Hinzufügen eines neuen Gerätes zum ZigBee-Netzwerk der Netzwerkschlüssel mitgelesen kann. Da dieser Schlüssel in den ZigBee Koordinatoren oft fest einprogrammiert ist, ist das Netzwerk dadurch dauerhaft kompromittiert. Aber auch ein reales Beispiel wurde in diesem Kapitel wieder herangezogen. So wurde die zuvor bereits mehrmals verwendete Netzwerkkamera auch hier wieder für einen praktischen Test genutzt. So wurden offene Ports und damit verbundene Dienste der Netzwerkkamera mittels des verbreiteten Netzwerkscanners *nmap* identifiziert.

Abschließend wurde auch der Untersuchung von Cloud-Diensten Beachtung geschenkt. Explizit wurde hier auf die Sicherheit von Amazon S3-Buckets eingegangen, welche häufig zur Speicherung von großen Datenmengen verwendet werden. Hier wurde praktisch aufgezeigt, wie sich mittels der Software *Gobuster* S3-Buckets von Unternehmen aufspüren lassen. Solche S3-Buckets können aufgrund falscher Konfiguration der Zugriffskontrolle öffentlich zugängliche Daten enthalten, welche eigentlich nicht einsehbar sein sollten. In diesem Zuge wurde auch erwähnt, dass Cloud-Dienste häufig nicht zum eigenen Unternehmen gehören und eventuell eine Erlaubnis für die Durchführung von Penetrationstests eingeholt werden muss.

Auf Grundlage des zuvor erlangten theoretischen Wissens, sowie der Erlangung von prak-

tischen Erfahrungen im Bereich des Penetrationstestens von IoT-Geräten, wurde im letzten Kapitel eine Methode zur Durchführung von Penetrationstests an IoT-Geräten entwickelt. Zu diesem Zweck wurden zunächst bestehende Methoden zur Durchführung von Penetrationstests an IoT-Geräten ermittelt und evaluiert. Zu diesen bestehenden Methoden zählt etwa der *OWASP IoT Security Testing Guide*, welcher IoT-Geräte granular in einzelne Komponenten aufteilt und einen Katalog an Testszenarien samt Checklisten zur Verfügung stellt. Zudem wird dieser Leitfaden kontinuierlich durch die OWASP-Gemeinschaft aktualisiert und erweitert. Zwar werden durch diesen Leitfaden reichlich Informationen und Checklisten zur Verfügung gestellt, eine einfach nachvollziehbare Vorgehensweise und sinnvolle Reihenfolge für die Durchführung von Penetrationstests wird allerdings nicht gegeben. Zudem dient der Leitfaden aufgrund des Umfanges eher als Nachschlagewerk und lässt sich nicht ohne umfangreiche Vorbereitung einsetzen. Eine weiterer Leitfaden ist der *IoT Pentesting Guide* von Aditya Gupta, welcher auch als Nachschlagewerk für Anfänger und fortgeschrittene Penetrationstester dienen und nützliche Informationen für die Durchführung von IoT Penetrationstests bereitstellen soll. Allerdings befindet sich dieser Leitfaden noch in der Entwicklungsphase und ist noch von geringem Umfang. Von größerem Umfang ist das *IoT Security Assurance Framework*, der IoT Security Foundation, welches eine umfassende Sammlung an Richtlinien und Praktiken zum Thema IoT-Sicherheit enthält. Es richtet sich dabei allerdings weniger an Penetrationstester, als an Führungskräfte, sowie Hersteller und Entwickler von IoT-Geräten, da die enthaltenen Richtlinien Voraussetzungen für die Sicherheit von IoT-Geräten darstellen, aber nicht darauf eingeht, wie die Sicherheit praktisch überprüft werden kann. Als Grundlage für Penetrationstests ist das Framework daher weniger geeignet. Auch die *NIST Special Publication 800-183 (Networks of 'Things')* ist ein Framework, welches sich nicht als praktische Methode für die Durchführung von IoT-Penetrationstests verwenden lässt. Dies liegt daran, dass dieses Framework den Zweck hat die Komplexität von IoT-Netzwerken verständlich zu machen und dabei ein gemeinsames Verständnis für Begriffe in diesem Umfeld zu schaffen. Daher kann das Framework allerdings in andere Methoden zur Durchführung von Penetrationstests und IoT-Geräten integriert werden, um ein gemeinsames Verständnis für Begriffe im Bereich des IoT zu etablieren. Mit der Evaluierung dieser bestehenden Methoden für Penetrationstests im IoT-Umfeld konnte die Forschungsfrage welche bestehenden Penetrationstest-Methoden für IoT-Geräte derzeit existieren ausführlich beantwortet werden.

Im nächsten Abschnitt des Kapitels konnte mit diesem und dem Wissen aus vorherigen Kapiteln zudem die Forschungsfrage nach kritischen Elementen für die Entwicklung einer Penetrationstest-Methodik für IoT-Geräte beantwortet werden. Als erstes und wichtigstes Element konnte die Testreihenfolge identifiziert werden. Diese ist von elementarer Bedeutung, da sie einen direkten Einfluss auf die Wirksamkeit des Penetrationstests hat und die Integrität des Gerätes beeinflussen kann. Denn sofern bereits zu Beginn des Penetrationstests invasive Methoden wie die Analyse der Hardware und internen Elektronik des Gerätes erfolgen, kann das Gerät irreversibel beschädigt werden. Sollte anschließend eine Neubeschaffung des Gerätes notwendig sein, kostet dies wertvolle Zeit. Doch nicht nur physische Beschädigungen stellen ein Problem dar. Denn bei Inbetriebnahme tauschen IoT-Geräte oft Sicherheits- oder API-Schlüssel aus, oder führen eine Aktualisierung der Software durch. Wird der Netzwerk-

verkehr also nicht von Beginn an mitgeschnitten, so kann es sein, dass sich dieses Vorgehen zu einem späteren Zeitpunkt nicht nachholen lässt.

Ein weiteres kritisches Element stellt die OSINT-Recherche dar, bei der bereits vor der praktischen Durchführung des Penetrationstests wichtige Informationen über das Gerät erlangt werden können. So lassen sich etwa Anleitungen für das Zerlegen des Gerätes auf der Herstellerwebseite oder anderen Plattformen erlangen. Dies kann dabei helfen das Gerät zerstörungsfrei zu öffnen, oder bereits vorab Werkzeuge zu beschaffen, die im späteren Verlauf des Tests notwendig sind. Des Weiteren können unter Umständen auch Softwarekomponenten wie die Firmware des Gerätes über das Internet bezogen werden. Somit ist es nicht notwendig diese unter hohem Aufwand manuell vom Gerät zu extrahieren. Dies spart Zeit und macht den Penetrationstest damit effizienter.

Auch die Dokumentation stellt ein kritisches Element dar. Diese sollte kontinuierlich während des gesamten Prozesses gepflegt werden, um alle durchgeführten Tätigkeiten und Ergebnisse reproduzierbar zu halten. Die Struktur der Dokumentation sollte dabei logisch und nachvollziehbar sein, sodass alle Daten auch zu einem späteren Zeitpunkt leicht auffindbar sind. Eine festgelegte und konstante Struktur der Dokumentation hilft demnach allen Beteiligten sich jederzeit zurechtzufinden.

Als weiteres kritisches Element kann das Festlegen des Umfangs angesehen werden. Dieser sollte in Anbetracht der zur Verfügung stehenden Zeit geplant werden. Wird der Umfang zu weitreichend gelegt, ist es möglich, dass Tests nicht ausführlich genug durchgeführt werden können, da nicht ausreichend Zeit zur Verfügung steht. Wird der Umfang zu gering definiert, so können wichtige Schwachstellen übersehen werden. Der Umfang sollte daher den zur Verfügung stehenden Ressourcen wie Zeit, Werkzeugen und Kenntnissen der Penetrationstester entsprechen.

Ein weiteres kritisches Element, welches zudem der Planung des Penetrationstests zuträglich ist, ist das Threat Modeling. Dieses modelliert alle Funktionen, technischen Abhängigkeiten und den Datenfluss von Systemen und dokumentiert diese grafisch. Dies hilft der Identifikation von Angriffsvektoren und der Bewertung dieser. Ein Threat Model sollte möglichst bereits vor dem Penetrationstest bestehen, da es hilfreich bei der Planung des Tests ist.

Wichtigstes Element beim Penetrationstest von IoT-Geräten ist der Bericht, welcher zum Abschluss des Testes erstellt wird. Dieser Test dient als Entscheidungsgrundlage zum Umgang mit den identifizierten Schwachstellen. Der Bericht sollte verschiedene Zielgruppen wie Entwickler, Führungskräfte und Penetrationstester bedienen und für alle diese Zielgruppen verwertbare Informationen zur Verfügung stellen.

Auf Basis der Identifikation der kritischen Elemente konnte anschließend eine Methodik für die Durchführung von Penetrationstests entwickelt werden. Diese Methodik integriert alle kritischen Elemente und lässt sich mit geringem Aufwand und unabhängig vom Kenntnisstand der beteiligten Penetrationstester anwenden. Grundlage der Methodik ist ein Git-Repository, welches eine sinnvolle Ordnerstruktur vorgibt. Innerhalb der Ordnerstruktur werden mittels README-Dateien empfohlene Vorgehensweisen beschrieben, sowie nützliche Informationen referenziert. Dieses Repository dient zudem als Vorlage, welches sich mittels Plattformen

wie GitHub leicht verwenden lässt. Auch die empfohlene Testreihenfolge wird vorgegeben und empfiehlt nicht-invasive Tests zu beginn durchzuführen. Durch die Verwendung eines Git-Repositories wird zudem die Zusammenarbeit im Team mit mehreren Penetrationstestern erleichtert. Diese findet in anderen Penetrationstest-Methoden keine Beachtung. Da IoT-Geräte aber aus unterschiedlichen Komponenten von Hardware und Software bestehen, ist eine Zusammenarbeit im Team mit Penetrationstestern unterschiedlicher Kenntnisse essenziell. Die Verwendung eines Git-Repositories zentralisiert die während des Tests entstehenden Daten und sorgt für eine geordnete Struktur. Somit lässt sich die entwickelte Methode mit geringem Aufwand von Unternehmen und Penetrationstestern praktisch anwenden, womit die Forschungsfrage beantwortet wurde, wie sich die Methode effektiv von Unternehmen und Penetrationstestern anwenden lassen kann. Ein weiterer Vorteil der entwickelten Methodik ist, dass umfassende Informationen nicht direkt integriert sind, sondern nur referenziert werden. Eine Herausforderung im Bereich der IoT Penetrationstests ist nämlich, dass Methoden für die Durchführung mit Zunahme an Informationen zu diesem komplexen Thema sehr unübersichtlich und wartungsintensiv werden können. Die Referenzierung von Informationen sorgt dafür, dass Informationen nicht aufwändig aktuell gehalten werden müssen. Zudem bleibt das Repository stets übersichtlich und dient nicht als umfangreiches Nachschlagewerk. Eine nicht zu granulare Ordnerstruktur innerhalb des Repositories sorgt zudem für eine leichte Anpassbarkeit an verschiedene IoT-Geräte, da bei Bedarf weitere Unterordner hinzugefügt werden, oder überschüssige Ordner entfernt werden können. Verbesserung und Weiterentwicklungen können wie bei Git üblich mittels Pull Requests in die Vorlage übernommen und somit für zukünftige Penetrationstests nutzbar gemacht werden. Mit dieser Erkenntnis wurde die Forschungsfrage nach der Anpassbarkeit der Methode an unterschiedliche Geräte und zukünftige Entwicklungen beantwortet, sowie der Frage welche Herausforderungen auftreten und wie diese gelöst werden können. Abschließend wurde die entwickelte Methode mit den zuvor evaluierten bestehenden Methoden verglichen, um die Forschungsfragen zu beantworten wie die entwickelte Methode im Vergleich zu bestehenden Methoden abschneidet und welche Vorteile im Vergleich zu diesen zu erwarten sind. Beantwortet werden konnte dies damit, dass die entwickelte Methode aufgrund der Verwendung einer Git-Vorlage besonders schnell und einfach einzusetzen ist. Zudem ist der Wartungsaufwand gering, da Informationen nur referenziert werden, was zudem der Übersichtlichkeit zugutekommt. Die Methode ist also flexibler als bestehende Methoden, bei einer zu erwartenden Effektivität, die den anderen Methoden ebenbürtig ist. Diese Arbeit trägt somit nicht nur zum akademischen Verständnis von IoT-Sicherheit und Penetrationstests im Bereich des IoT bei, sondern liefert auch praktische Ansätze zur Verbesserung der Sicherheit und Penetrationstestmethoden, in einer immer stärker vernetzten Welt.

7.2 Kritische Reflexion der Ergebnisse

Die vorliegende Arbeit hat aufgezeigt, dass die Sicherheit von IoT-Geräten und die Durchführung von Penetrationstests in diesem Bereich eine komplexe Herausforderung darstellen. Es wurde eine neue Methodik für Penetrationstests im IoT-Bereich entwickelt, welche die

bereits bestehenden Ansätze aufgreift und eine praktikablere Herangehensweise bietet. Diese Methode adressiert die übergeordneten Komponenten von IoT-Systemen, um möglichst flexibel zu bleiben und bietet durch eine empfohlene Testreihenfolge, sowie eine Fokussierung auf die Zusammenarbeit im Team eine leicht umzusetzende Struktur als Grundlage. Die Methode ist dahingehend einzigartig, dass sie einen schnellen Einstieg in die Welt der IoT-Penetrationstests bietet und sich unabhängig vom Kenntnisstand des Anwenders schnell und praktisch umsetzen lässt.

Trotz dieser Stärken ist die Arbeit nicht ohne Einschränkungen. Die Komplexität und Vielfalt von IoT-Systemen bedingt, dass nicht alle möglichen Umstände und Szenarien abgedeckt werden konnten. Zudem konnte die praktische Anwendung der Methode aufgrund zeitlicher und ressourcenbedingter Grenzen nur in einem begrenzten Rahmen getestet werden.

Zur weiteren Verbesserung der Arbeit wäre es empfehlenswert, die Methodik in einer breiteren Palette von IoT-Umgebungen zu testen und die Erfahrungen von weiteren Sicherheitsexperten zu integrieren. Die fortlaufende Entwicklung im Bereich der IoT-Technologien erfordert zudem eine kontinuierliche Aktualisierung und Anpassung der Methodik, um ihre Relevanz und Effektivität zu gewährleisten.

7.3 Ausblick

Die stetig zunehmende Vernetzung und der rasante technologische Fortschritt, insbesondere im Bereich der künstlichen Intelligenz stellen sowohl Chancen als auch Herausforderungen für den Bereich IoT dar. Vor allem generative künstliche Intelligenz hält Einzug in mehr und mehr Produkte. Die mögliche Integration von generativer künstlicher Intelligenz in IoT-Geräte könnte Angriffsvektoren aus diesem Bereich auch auf IoT-Geräte übertragen. Sollte diese Entwicklung Einzug in die Welt der IoT-Geräte halten, so müssen Angriffsmöglichkeiten aus diesem Bereich bei Penetrationstests berücksichtigt werden. Eine weitere zukünftige Herausforderung entsteht durch die zunehmende Adaption der Mobilfunktechnologie 5G. Das Marktforschungsunternehmen *Gartner* geht davon aus, dass die Anzahl an IoT-Geräten sich zukünftig vervielfachen wird, wobei Überwachungskameras und Fahrzeuge mit 5G Verbindung den größten Anteil ausmachen könnten.[1] Diese zunehmende Verwendung von 5G-Technologie sorgt für neue Herausforderungen, da IoT-Geräte damit zunehmend aus internen Netzen in öffentlich erreichbare Netze übergehen werden. Diese Entwicklung sollte genau verfolgt und bei Bedarf in der weiteren Entwicklung von Penetrationstestmethoden Beachtung finden.

Schließlich ist die zunehmende Vernetzung und Komplexität von IoT-Systemen eine fortwährende Herausforderung. Die Entwicklung von Sicherheitskonzepten muss mit dieser Entwicklung Schritt halten, um wirksamen Schutz gegen zukünftige Bedrohungen zu bieten. Dies erfordert eine kontinuierliche Forschung und Anpassung der Sicherheitsstrategien und Methoden im Bereich des IoT.

Insgesamt bietet die Arbeit einen umfassenden Einblick in die aktuellen Herausforderungen der IoT-Sicherheit und legt den Grundstein die Durchführung von effektiven Penetrationstests

[1] Vgl. 64.

in diesem dynamischen und immer relevanter werdenden Feld.

Literatur

[1] *Botnetze – Auswirkungen und Schutzmaßnahmen*. URL: https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/Botnetze/botnetze_node.html#:~:text=Im%20Fachjargon%20ist%20mit%20Bot,zu%20bestimmten%20Aktionen%20missbraucht%20werden. (besucht am 02. 01. 2024).

[2] A. Meyer. *Softwareentwicklung: ein Kompass für die Praxis*. De Gruyter Praxishandbuch. Berlin ; München ; Boston: De Gruyter Oldenbourg, 2018. 271 S. ISBN: 978-3-11-057580-4.

[3] *firmware*. URL: https://csrc.nist.gov/glossary/term/firmware (besucht am 25. 01. 2024).

[4] *Framework*. URL: https://csrc.nist.gov/glossary/term/framework (besucht am 02. 01. 2024).

[5] *git –local-branching-on-the-cheap*. URL: https://git-scm.com/ (besucht am 02. 01. 2024).

[6] *IEEE Standard for Low-Rate Wireless Networks*. URL: https://standards.ieee.org/ieee/802.15.4/7029/ (besucht am 02. 01. 2024).

[7] P. René und S. Bjørn. *Git : Dezentrale Versionsverwaltung im TeamGrundlagen und Workflows*. Bd. 5th ed. dpunkt.verlag, 2019. ISBN: 978-3-86490-649-7. URL: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2253700&site=ehost-live.

[8] C. Eckert. *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. 10. Auflage. De Gruyter studium. München: De Gruyter Oldenburg, 2018. ISBN: 978-3-11-055158-7.

[9] R. Aigner, K. Gebeshuber, T. Hackner, S. Kania, P. Kloep, M. Kofler, F. Neugebauer, T. Scheible, M. Widl, M. Wübbeling und A. Zingsheim. *Hacking & Security: das umfassende Handbuch*. ger. 3., aktualisierte und erweiterte Auflage. Rheinwerk Computing. Bonn: Rheinwerk, 2023. ISBN: 978-3-8362-9164-4.

[10] *Internet of Things und Industrie 4.0*. ger. 1st ed. 2023. OCLC: 1372478739. Wiesbaden, Wiesbaden: Springer Fachmedien Wiesbaden Springer Vieweg, 2023. ISBN: 978-3-658-39901-6.

[11] S. Cheruvu, A. Kumar, N. Smith und D. Wheeler. *Demystifying Internet of Things security: successful IoT device/edge and platform security deployment*. eng. OCLC: 1117469508. New York: Apress, 2020. ISBN: 978-1-4842-2896-8.

[12] M. Mangel und S. Bicchi. *Praktische Einführung in Hardware Hacking: Sicherheitsanalyse und Penetration Testing für IoT-Geräte und Embedded Devices*. ger. 1. Auflage. Frechen: mitp, 2020. ISBN: 978-3-95845-816-1.

[13] *Black Hat*. en. URL: https://www.blackhat.com/about.html (besucht am 20. 09. 2023).

[14] *Black Hat Asia 2020*. URL: https://www.blackhat.com/asia-20/briefings/schedule/#track/internet-of-things (besucht am 20. 09. 2023).

[15] *About the OWASP Foundation | OWASP Foundation*. en. URL: https://owasp.org/about/ (besucht am 20. 09. 2023).

[16] *OWASP IoT Security Testing Guide | OWASP Foundation*. en. URL: https://owasp.org/www-project-iot-security-testing-guide/ (besucht am 20. 09. 2023).

[17] *OWASP IoT Security Testing Guide*. original-date: 2023-08-24T16:47:39Z. Sep. 2023. URL: https://github.com/OWASP/owasp-istg (besucht am 20. 09. 2023).

[18] *OWASP Internet of Things Project - OWASP*. URL: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10 (besucht am 12. 07. 2023).

[19] R. Svensson. *From Hacking to Report Writing - An Introduction to Security and Penetration Testing*. New York: Apress, 2016. ISBN: 978-1-484-22283-6.

[20] K. A. Scarfone, M. P. Souppaya, A. Cody und A. D. Orebaugh. *Technical guide to information security testing and assessment*. Techn. Ber. 2008. DOI: 10.6028/nist.sp.800-115. URL: https://doi.org/10.6028/nist.sp.800-115.

[21] J. Ruohonen. „A look at the time delays in CVSS vulnerability scoring“. In: *Applied Computing and Informatics* 15.2 (2019), S. 129–135. ISSN: 2210-8327. DOI: https://doi.org/10.1016/j.aci.2017.12.002. URL: https://www.sciencedirect.com/science/article/pii/S2210832717302995.

[22] B. für Sicherheit in der Informationstechnik. *Sicherheit von Geräten im Internet der Dinge*. de. Okt. 2017. URL: https://www.allianz-fuer-cybersicherheit.de/SharedDocs/Downloads/Webs/ACS/DE/BSI-CS/BSI-CS_128.html?nn=133710 (besucht am 12. 11. 2023).

[23] *Heightened DDoS Threat Posed by Mirai and Other Botnets | CISA*. en. Okt. 2017. URL: https://www.cisa.gov/news-events/alerts/2016/10/14/heightened-ddos-threat-posed-mirai-and-other-botnets (besucht am 12. 11. 2023).

[24] *Heightened DDoS Threat Posed by Mirai and Other Botnets | CISA*. 17. Okt. 2017. URL: https://www.cisa.gov/news-events/alerts/2016/10/14/heightened-ddos-threat-posed-mirai-and-other-botnets (besucht am 25. 01. 2024).

[25] A. Gerber und J. R. U. 3. J. 2. |. P. 2. Mai 2017. *Connecting all the things in the Internet of Things*. URL: https://developer.ibm.com/articles/iot-lp101-connectivity-network-protocols/ (besucht am 12. 11. 2023).

[26] A. Guzman und A. Gupta. *IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices*. Packt Publishing, 2017. ISBN: 9781787285170. URL: https://books.google.de/books?id=rEFPDwAAQBAJ.

[27] R. Aigner, M. Kofler, K. Gebeshuber, A. Zingsheim, T. Hackner, M. Widl, S. Kania, P. Kloep und F. Neugebauer. *Hacking & Security: das umfassende Handbuch*. Bonn: Rheinwerk Verlag, 2018. ISBN: 978-3-836-24548-7.

[28] S. Wendzel. *IT-Sicherheit für TCP/IP- und IoT-Netzwerke: Grundlagen, Konzepte, Protokolle, Härtung*. ger. 2., aktualisierte und erweiterte Auflage. Lehrbuch. Wiesbaden [Heidelberg]: Springer Vieweg, 2021. ISBN: 978-3-658-33423-9.

[29] P.-B. Bök, A. Noack, M. Müller und D. Behnke. *Computernetze und Internet of Things: technische Grundlagen und Spezialwissen*. ger. Lehrbuch. Wiesbaden [Heidelberg]: Springer Vieweg, 2020. ISBN: 978-3-658-29409-0.

[30] N. Nikolov. „Research of MQTT, CoAP, HTTP and XMPP IoT Communication protocols for Embedded Systems“. In: *2020 XXIX International Scientific Conference Electronics (ET)*. 2020, S. 1–4. DOI: 10.1109/ET50336.2020.9238208.

[31] J. Olsson. „6LoWPAN demystified“. In: *Texas Instruments* 13 (2014), S. 1–13. URL: https://www.ti.com/lit/wp/swry013/swry013.pdf?ts=1710054244074.

[32] O. Iova, P. Picco, T. Istomin und C. Kiraly. „RPL: The Routing Standard for the Internet of Things... Or Is It?“ In: *IEEE Communications Magazine* 54.12 (2016), S. 16–22. DOI: 10.1109/MCOM.2016.1600397CM.

[33] R. Gessler und T. Krause. *Wireless-Netzwerke für den Nahbereich: eingebettete Funksysteme: Vergleich von standardisierten und proprietären Verfahren*. ger. 2., aktualisierte u. erw. Aufl. Wiesbaden: Springer Fachmedien Wiesbaden, 2015. ISBN: 978-3-8348-2075-4.

[34] L. Chettri und R. Bera. „A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems“. In: *IEEE Internet of Things Journal* 7.1 (2020), S. 16–32. DOI: 10.1109/JIOT.2019.2948888.

[35] C. Baun. *Computernetze kompakt*. ger. IT kompakt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-41653-8.

[36] O. Hahm, E. Baccelli, H. Petersen und N. Tsiftes. „Operating Systems for Low-End Devices in the Internet of Things: A Survey“. In: *IEEE Internet of Things Journal* 3.5 (2016), S. 720–734. DOI: 10.1109/JIOT.2015.2505901.

[37] F. Javed, M. K. Afzal, M. Sharif und B.-S. Kim. „Internet of Things (IoT) Operating Systems Support, Networking Technologies, Applications, and Challenges: A Comparative Review“. In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), S. 2062–2100. DOI: 10.1109/COMST.2018.2817685.

[38] F. Hüning. *Embedded Systems für IoT*. ger. Berlin [Heidelberg]: Springer Vieweg, 2019. ISBN: 978-3-662-57901-5.

[39] P. Ferrara, A. K. Mandal, A. Cortesi und F. Spoto. „Static analysis for discovering IoT vulnerabilities“. In: *International Journal on Software Tools for Technology Transfer* 23 (2021), S. 71–88. DOI: 10.1007/s10009-020-00592-x.

[40] T. Xu, J. B. Wendt und M. Potkonjak. „Security of IoT systems: Design challenges and opportunities“. In: *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2014, S. 417–423. DOI: 10.1109/ICCAD.2014.7001385.

[41] *OWASP Internet of Things Project - OWASP*. URL: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10 (besucht am 12. 07. 2023).

[42] *OWASP Top 10:2021*. URL: https://owasp.org/Top10/ (besucht am 24. 11. 2023).

[43] *A02 Cryptographic Failures - OWASP Top 10:2021*. URL: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/ (besucht am 24. 11. 2023).

[44] *A03 Injection - OWASP Top 10:2021*. URL: https://owasp.org/Top10/A03_2021-Injection/ (besucht am 24. 11. 2023).

[45] *A04 Insecure Design - OWASP Top 10:2021*. URL: https://owasp.org/Top10/A04_2021-Insecure_Design/ (besucht am 24. 11. 2023).

[46] *A05 Security Misconfiguration - OWASP Top 10:2021*. URL: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ (besucht am 24. 11. 2023).

[47] *A07 Identification and Authentication Failures - OWASP Top 10:2021*. URL: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/ (besucht am 24. 11. 2023).

[48] *A08 Software and Data Integrity Failures - OWASP Top 10:2021*. URL: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/ (besucht am 24. 11. 2023).

[49] *A09 Security Logging and Monitoring Failures - OWASP Top 10:2021*. URL: https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/ (besucht am 24. 11. 2023).

[50] *A10 Server Side Request Forgery (SSRF) - OWASP Top 10:2021*. URL: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/ (besucht am 24. 11. 2023).

[51] *OWASP Mobile Top 10*. URL: https://owasp.org/www-project-mobile-top-10/ (besucht am 28. 01. 2024).

[52] F. A. Alaba, M. Othman, I. A. T. Hashem und F. Alotaibi. „Internet of Things security: A survey“. In: *Journal of Network and Computer Applications* 88 (2017), S. 10–28. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2017.04.002. URL: https://www.sciencedirect.com/science/article/pii/S1084804517301455.

[53] *OWASP Cloud-Native Application Security Top 10 | OWASP Foundation*. en. URL: https://owasp.org/www-project-cloud-native-application-security-top-10/ (besucht am 24. 11. 2023).

[54] A. Gupta. *The IoT hacker's handbook: a practical guide to hacking the internet of things*. New York, NY: Apress, 2019. 320 S. ISBN: 978-1-4842-4300-8.

[55] *Challenge solutions - Pwning OWASP Juice Shop*. URL: https://help.owasp-juice.shop/appendix/solutions.html (besucht am 06. 01. 2024).

[56] *Server-side request forgery (SSRF)*. URL: https://portswigger.net/web-security/ssrf#:~:text=Server%2Dside%20request%20forgery%20is,services%20within%20the%20organization's%20infrastructure. (besucht am 07. 03. 2024).

[57] *Testing Methodology - OWASP IoT Security Testing Guide*. URL: https://owasp.org/owasp-istg/02_framework/methodology.html (besucht am 02. 01. 2024).

[58] *Introduction - OWASP IoT Security Testing Guide*. URL: https://owasp.org/owasp-istg/01_introduction/index.html (besucht am 02. 01. 2024).

[59] *Study - A Penetration Testing Model*. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1 (besucht am 07. 03. 2024).

[60] *IoT Pentesting Guide*. URL: https://www.iotpentestingguide.com/ (besucht am 02. 01. 2024).

[61] *IoTSF IoT Security Assurance Framework Release 3.0 Nov 2021*. 2021. URL: https://iotsecurityfoundation.org/wp-content/uploads/2021/11/IoTSF-IoT-Security-Assurance-Framework-Release-3.0-Nov-2021-1.pdf (besucht am 02. 01. 2024).

[62] *NIST Special Publication 800-183 Networks of 'Things'*. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-183.pdf. (Besucht am 02. 01. 2024).

[63] F. Alwahedi, A. Aldhaheri, M. A. Ferrag, A. Battah und N. Tihanyi. „Machine learning techniques for IoT security: Current research and future vision with generative AI and large language models". In: *Internet of Things and Cyber-Physical Systems* 4 (2024), S. 167–185. ISSN: 2667-3452. DOI: https://doi.org/10.1016/j.iotcps.2023.12.003. URL: https://www.sciencedirect.com/science/article/pii/S2667345223000585.

[64] *Gartner Predicts Outdoor Surveillance Cameras Will Be Largest Market for 5G Internet of Things Solutions Over Next Three Years*. URL: https://www.gartner.com/en/newsroom/press-releases/2019-10-17-gartner-predicts-outdoor-surveillance-cameras-will-be (besucht am 28. 01. 2024).

Anhang

README-Dateien

```
# IoT Penetrationstest

Dieses Repository dient als Leitfaden für die strukturierte Durchführung und Dokumentation von IoT-Penetrationstests.
Es zielt darauf ab, die Dokumentation zu standardisieren und die Durchführung von Penetrationstests durch die Bereitstellung hilfreicher Informationen und Vorgehensweisen zu erleichtern.

## Grundlegende Empfehlungen für IoT-Penetrationstests
Bei der Durchführung von Penetrationstests an IoT-Geräten sollte nach Möglichkeit mit nicht-invasiven Tests begonnen werden.
Diese Vorgehensweise soll sicherstellen, dass die Geräte nicht beschädigt werden und wichtige Informationen, wie der Netzwerkverkehr bei der Inbetriebnahme, vollständig erfasst werden können.

### Empfohlene Testreihenfolge:
0. **Planung**: Erstellen oder Beschaffen eines Threat Models und Festlegung des Umfangs.
1. **OSINT-Recherche**: Sammeln von öffentlich zugänglichen Informationen über das Zielgerät.
2. **Netzwerkanalyse bei Inbetriebnahme**: Erfassen des Netzwerkverkehrs, um mögliche geheime Schlüssel beim Schlüsselaustausch und Daten bei Firmwareaktualisierungen abzufangen.
3. **Software- und Firmware-Analyse**: Untersuchen der Softwarekomponenten des Geräts auf Schwachstellen.
4. **Invasive Tests**: Diese sollten nur durchgeführt werden, wenn alle anderen Methoden ausgeschöpft sind, da sie das Gerät beschädigen können. Hierzu zählen beispielsweise das Auseinanderbauen des Gerätes und die Manipulation der Elektronik im Inneren des Gerätes.

## Verwendung des Repositories
- Für die Verwendung dieses Repositories den Button "Dieses Template verwenden" anklicken und einen Namen für den Penetrationstest angeben.
```

```
- Sicherstellen, dass alle beteiligten Penetrationstester in dem neu erstellten Repository schreiben können.
- In den Unterordnern dieses Repositories befinden sich ‘README‘-Dateien, in denen hilfreiche Informationen und empfohlene Vorgehensweisen beschrieben sind.
- Nicht benötigte Unterordner und Dateien können zur Verbesserung der Übersichtlichkeit gelöscht werden.

## Verbesserungen / Ergänzungen
Verbesserungen oder Ergänzungen für dieses Template können über Pull Requests eingereicht werden. Hierzu muss das Repository zunächst mittels ‘git clone‘ geklont werden. Anschließend können änderungen und Ergänzungen per Pull Request an das Template weitergereicht werden.

## Struktur des Repositories
- **OSINT-Recherche**: In diesem Verzeichnis werden alle Informationen aus der OSINT-Recherche abgelegt.
- **Penetrationstest-Daten**: In diesem Verzeichnis werden alle während des Penetrationstests angefallenen Daten und Informationen abgelegt.
- **Report**: In diesem Verzeichnis werden alle zur Erstellung des Reports notwendigen Daten abgelegt, sowie der Report selbst.
```

/README.md

```
# Definition des Umfangs

## Details
- **Projekt Name:** [Name des Penetrationstests]
- **Kunde:** [Name des Kunden (sofern anwendbar)]
- **Datum:** [Datum]

---

## Zielsetzung
Die zu erreichenden Ziele des Penetrationstests
1. Bewertung der Sicherheitslage der IoT-Geräte und -Infrastruktur des Kunden.
2. Identifizierung von Schwachstellen und Risiken im IoT-Ökosystem.
3. Abgabe von Empfehlungen zur Mitigierung der identifizierten Schwachstellen und Risiken.

---
```

```
17
18 ## Umfang
19 ### In-Scope
20 Die folgenden Aspekte sind Teil des Penetrationstests:
21
22 1. **IoT Geräte:** [Sicherheitskameras, Sensoren, ...]
23 2. **Netzwerk Infrastruktur:** [Gateways, Router, Switche, ...]
24 3. **Protokolle:** [ZigBee, ...]
25 4. **Firmware:** [Cloud Plattform, ...]
26 5. **Benutzerschnittstellen:** [Weboberfläche, Mobile App, ...]
27 6. **Authentifizierung und Autorisierung:** [Mechanismen zur
     Authentifizierung und Autorisierung]
28 7. ...
29
30 ### Out-of-Scope
31 Die folgenden Aspekte sind **<u>nicht</u>** Teil des Penetrationstests:
32
33 1. Gehäuse der Geräte
34 2. Physische Gegebenheiten, wie Zugang zum Gerät
35 3. Netzwerke die nicht in Verbindung mit der IoT-Landschaft stehen
36
37 ---
38
39 ## Methoden
40 Der Penetrationstest wird unter Anwendung der folgenden Methoden durchgefü
     hrt:
41 - **OSINT Recherche:** Initiale Gewinnung von Informationen unter
     Verwendung öffentlich zugänglicher Informationsquellen
42 - **Schwachstellenscan:** Identifizierung von Schwachstellen unter Einsatz
      eines Schwachstellenscanners
43 - **Hardwareanalyse:** Analyse der Hauptplatine zum Auffinden von
     Debugging-Schnittstellen
44 - ...
45
46 ---
47
48 ## Regeln (Rules of Engagement)
49 Während des Penetrationstests werden die folgenden Regeln befolgt:
50 - Alle Testaktivitäten finden in Übereinstimmung mit dem Kunden statt
51 - Der Penetrationstest darf Produktivsysteme nicht beeinflussen
52 - Während dem Penetrationstest werden keine illegalen Aktivitäten durchgef
     ührt
```

```
- Schützenwerte Daten werden nicht an unbefugte weitergegeben

---

## Berichterstellung
Bei Fertigstellung des Penetrationstests wird ein detaillierter Bericht ü
    bermittelt. Der Bericht enthält folgende Informationen:
1. Management Zusammenfassung
2. Methodik
3. Findings
4. Empfehlungen
5. Anhänge

---

## Zeitraum
Der voraussichtliche Zeitplan für den IoT-Penetrationstest sieht wie folgt
     aus:
### Vorbereitung
- **Kickoff Meeting:** [Datum]
### Durchführung
- **Testdurchführung:** [Start] - [Ende]
### Dokumentation
- **Fertigstellung des Berichtes:** [Datum]
```

/scope.md

```
# Penetrationstestbericht [Name]

## Überblick
Dieses Verzeichnis ist für die Erstellung und Speicherung des
    Abschlussberichtes des Penetrationstests vorgesehen. Der Bericht fasst
    die Ergebnisse, Analysen und Empfehlungen zusammen, die im Verlauf des
    Tests erarbeitet wurden.

## Struktur des Berichts
- **Einleitung**: Beschreibung des Projektziels und des Umfangs der Arbeit.

- **Methodik**: Darstellung der verwendeten Methoden und Ansätze.
- **Ergebnisse**: Detaillierte Präsentation der erzielten Ergebnisse und
    Erkenntnisse.
- **Diskussion**: Analyse und Interpretation der Ergebnisse.
```

```
- **Schlussfolgerungen und Empfehlungen**: Zusammenfassung der wichtigsten
    Erkenntnisse und Vorschläge für zukünftige Maßnahmen.
- **Anhänge**: Zusätzliche Dokumente, Daten, Grafiken etc.

## Richtlinien für die Berichterstellung
- **Klarheit und Genauigkeit**: Der Bericht muss klar und verständlich
    formuliert sein.
- **Nachvollziehbarkeit**: Alle Schritte innerhalb des Berichts sollten
    reproduzierbar und verständlich sein.
- **Vertraulichkeit beachten**: Einhaltung der Datenschutz- und
    Vertraulichkeitsrichtlinien.

## Checkliste vor der Veröffentlichung
- [] Überprüfung von Rechtschreibung und Grammatik
- [] Validierung aller Daten und Quellen
- [] Einhaltung aller relevanten Richtlinien und Standards
- [] Abschlussprüfung durch eine unabhängige Partei oder ein Review-Team

# Management Zusammenfassung

# Methodik

# Findings

# Empfehlungen

# Anhänge
```

/Report/README.md

```
# IoT Penetrationstest-Daten

In diesem Verzeichnis werden alle Daten abgelegt, die während der Durchfü
    hrung des Penetrationstests angefallen sind.

Die Daten werden in den Unterverzeichnissen dieses Verzeichnisses
    einsortiert. Nicht benötigt Verzeichnisse können zugunsten der Ü
    bersicht entfernt werden. Bei Bedarf können zusätzliche Verzeichnisse
    Angelegt werden.

In den Unterverzeichnissen befinden sich jeweils ‘README‘-Dateien, mit nü
    tzlichen Informationen und Referenzen zu weiterführenden Informationen.
```

```
8
9 ## Hinweis
10 Sollte bei der Durchführung des Penetrationstests auffallen, dass Daten
     Fehlen oder vorhandene Daten fehlerhaft sind, so kann das Template
     mittels ‘Pull Request‘ erweitert werden. Dies stellt sicher, dass das
     Template stehts aktuell ist und zukünftige Entwicklungen integriert
     werden.
```

/Penetrationstest-Daten/README.md

```
1 # Software Penetrationstest
2
3 ## Überblick
4 In diesem Verzeichnis befinden sich Informationen und Leitfäden für das
     Testen der Software von IoT-Geräten, einschließlich Firmware,
     Webinterfaces und mobilen Apps.
5
6 ## Vorgehensweise
7 1. **Firmware**: Untersuchung der Firmware auf Schwachstellen, Ausnutzung
     von Firmware-Updates. Sollte es notwendig sein die Firmware vom Gerät
     zu extrahieren und könnte das Gerät dabei beschädigt werden, so ist
     dieser Punkt als letztes durchzuführen.
8 2. **Webinterface**: Überprüfung auf Schwachstellen der OWASP Top 10.
9 3. **Mobile Apps**: Analyse der Sicherheit von mobilen Anwendungen.
10
11 ## Nützliche Tools und Ressourcen
12 - Firmware-Analysewerkzeuge wie Binwalk
13 - Interception Proxies wie Burp Suite
14 - Tools zur Analyse von Mobilen Apps wie MobSF
15
16 ### Referenzen zu weiterführenden Informationen
17 - [OWASP Mobile Security Testing Guide](https://mobile-security.gitbook.io
     /mstg/)
18 - [OWASP Web Application Security Testing](https://owasp.org/www-project-
     web-security-testing-guide/)
```

/Penetrationstest-Daten/Software/README.md

```
1 # Hardware Penetrationstest
2
3 ## Überblick
4 Dieser Ordner enthält Ressourcen und Informationen für Penetrationstests
```

```
    im Bereich der Hardware von IoT-Geräten. Dies umfasst physikalische
    Komponenten wie Gehäuse und Platinen.

## Vorgehensweise
0. **Beschaffung von Werkzeugen**: Beschaffung der für die Analyse
    notwendigen Werkzeuge.
1. **Inspektion des Gehäuses**: Untersuchung auf physische
    Sicherheitsmerkmale, wie Spezialschrauben, Holographische Siegel und
    Verklebungen.
2. **Analyse der Platine**: Ermittlung von Komponenten und Schnittstellen,
     Reverse Engineering von Schaltkreisen und Identifikation von möglichen
     Angriffspunkten.

## Nützliche Tools und Ressourcen
- Multimeter
- Oszilloskop
- UART zu USB Adapter
- Schraubendreher
- Lötkolben

## Referenzen zu weiterführenden Informationen
- [Hackaday](https://hackaday.com/)
- [Hardware Hacking Subreddit](https://www.reddit.com/r/hardwarehacking/)
```

/Penetrationstest-daten/Hardware/README.md

```
# Netzwerk Penetrationstest

## Überblick
Dieser Ordner beinhaltet Ressourcen für Netzwerk-Penetrationstests,
    einschließlich der Analyse von Netzwerkverkehr und Schnittstellen.

## Vorgehensweise
1. **Mitschnitte analysieren**: Überwachung und Analyse des
    Netzwerkverkehrs zur Identifikation verdächtiger Aktivitäten. Nach Mö
    glichkeit den Netzwerkverkehr bei erster Inbetriebnahme des Gerätes
    mitschneiden.
2. **Schnittstellen prüfen**: Sicherheitstests von Netzwerkschnittstellen
    und Kommunikationsprotokollen.

## Nützliche Tools und Ressourcen
- Netzwerk-Sniffer wie Wireshark
```

```
- Netzwerk-Scanner wie Nmap

## Referenzen zu weiterführenden Informationen
- [Wireshark User Guide](https://www.wireshark.org/docs/wsug_html_chunked/)
- [Nmap Network Scanning](https://nmap.org/book/)
```

/Penetrationstest-daten/Netzwerk/README.md

```
# Cloud Penetrationstest

## Überblick
Informationen für Penetrationstests von IoT-bezogenen Cloud-Diensten.

## Vorgehensweise
1. **Zustimmung einholen**: Falls notwendig Zustimmung für den Penetrationstest vom Cloud-Anbieter einholen.
2. **API-Tests**: Untersuchung der Sicherheit von APIs und der Datenübertragung zwischen IoT-Geräten und Cloud-Diensten.

## Nützliche Tools und Ressourcen
- Postman für API-Tests

## Referenzen zu weiterführenden Informationen
- [OWASP Cloud-Native Application Security Top 10](https://owasp.org/www-project-cloud-native-application-security-top-10/)
- [OWASP Docker Top 10](https://owasp.org/www-project-docker-top-10/)
- [OWASP Kubernetes Top 10](https://owasp.org/www-project-kubernetes-top-ten/)
```

/Penetrationstest-Daten/Cloud/README.md

```

# OSINT-Recherche

## Überblick
Dieses Verzeichnis enthält Informationen zur OSINT-Recherche, sowie Erkenntnisse welche aus dieser gewonnen werden.

## Informationen zur Verwendung dieses Verzechnisses
Dieses Verzeichnis enthält mehrere Unterverzeichnisse, in denen sich jeweils eine ‘README‘-Datei befindet. Jede dieser Dateien enthält
```

```
    Referenzen zu Tools und Informationsquellen. Sollten
    Informationsquellen oder Tools fehlen, so sind diese mittels ‘Pull
    Request‘ in das Template aufzunehmen.
9
10 ## Referenzen zu weiterführenden Informationen
11 - Maltego - Tool zur Visualisierung und Verknüpfung von Daten aus
    verschiedenen Quellen
12 - [Shodan.io](https://shodan.io)
```

/OSINT-Recherche/README.md