Universität Rostock Traditio et Innovatio

800600 ROSTOCK

Masterarbeit zum Thema

Untersuchung der Möglichkeiten zum Betreiben von Honeypots zur Nachbildung von internetfähigen Steuergeräten in der Gebäudeautomation

Studiengang:	Informatik
Vorgelegt von:	Johann Bauer, B.Sc.
Matrikelnummer:	213204424
Bearbeitungszeitraum:	01. Mai 2018 – 18. September 2018
Betreuer:	Dr.-Ing. Thomas Mundt
Erstgutachter:	Prof. Dr. rer. nat. Clemens H. Cap
Zweitgutachter:	Prof. Dr. rer. nat. habil. Andreas Heuer

Lehrstuhl für Informations- und Kommunikationsdienste
Institut für Informatik
Fakultät für Informatik und Elektrotechnik
Universität Rostock

Gedruckt auf säurefreiem und elementar-chlorfrei gebleichtem Papier.

Stand: 12. November 2018

Danksagung

An dieser Stelle möchte ich all jenen danken, die durch ihre Unterstützung zum Gelingen dieser Masterarbeit beigetragen haben:

Bei Dr. Thomas Mundt, dem Betreuer meiner Masterarbeit, der sich das Thema erdacht hat und mich mit seinem fachkundigen Rat und hilfreichen Anregungen unterstützte.

Bei Johannes Goltz und Simeon Wiedenmann, die mir über den gesamten Entwicklungsstand der Masterarbeit durch konstruktiven Ideenaustausch sehr geholfen haben.

Bei meinen Eltern, die diese Arbeit korrekturgelesen hat.

Inhalt

Kurzzusammenfassung

Honeypots ermöglichen es, die Aktionen von Angreifern mit einem System detailliert nachzuvollziehen um daraus Kenntnisse über die Abwehr zukünftiger Angriffe zu gewinnen. In der vorliegenden Masterarbeit wird ein Konzept entwickelt, um einen Honeypot für ein bekanntermaßen unsicheres Gebäudeautomationssystem zu entwickeln und die aufgezeichneten Attacken auszuwerten.

Dieses Konzept wurde anschließend umgesetzt und der entwickelte Honeypot öffentlich verfügbar gemacht. Die dadurch gewonnenen Aufzeichnungen wurden untersucht und ausgewertet.

Abstract

Honeypots enable tracing the actions of attackers with a system in detail in order to gain knowledge about the defense against future attacks. In this master thesis a concept is established to develop a honeypot for a known insecure building automation system and to evaluate the recorded attacks.

This concept was subsequently implemented and the developed honeypot made publicly available. The records obtained in this way were further examined and evaluated.

Kapitel 1

Einleitung

Gebäudeautomationssysteme können vielfältige Vorteile eröffnen und werden darum oftmals in Neubauten integriert. Für die Nutzer des Gebäudes (beispielsweise Mitarbeiter oder Bewohner) ist dabei vor allem der mögliche Komfortgewinn interessant, so lässt sich beispielsweise in einem entsprechend ausgerüsteten Gebäude die Beleuchtung oder Klimatisierung automatisch auf Basis der aktuellen Außenhelligkeit und der Anzahl der Menschen in einem Raum steuern. Auch für die Besitzer des Gebäudes bringt dies Vorteile mit sich, so lassen sich durch automatisches Deaktivieren der Beleuchtung und Klimatisierung in nicht benutzten Räumen hohe Einsparungen bei den Energiekosten erzielen.

Daher ist es wenig erstaunlich, dass Techniken zur Gebäudeautomation immer mehr Verbreitung finden. So steigt etwa die Zahl der Mitglieder der KNX Association - die den KNX-Standard zur Feldbuskommunikation in Gebäudeautomationssystemen erarbeiten - seit 2005 kontinuierlich an (Abbildung 1.1). Auch in der Zukunft wird das Thema wohl immer beliebter werden, [Sta18] schätzt, dass sich der Umsatz im Bereich „Smart Home" bis zum Jahr 2022 um 296% auf 19 Mrd. Euro wachsen wird.

In Gebäudeautomationsssystemen werden sogenannte Direct Digital Controller (*DDC* oder *DDC-GA*) verwendet, um bei Bedarf die Komponenten im Gebäude zentral zu regeln und zu steuern. So können DDCs etwa programmiert werden, die Fenster zu öffnen, wenn die Temperatur im Gebäude einen bestimmten Sollwert übersteigt. Solche

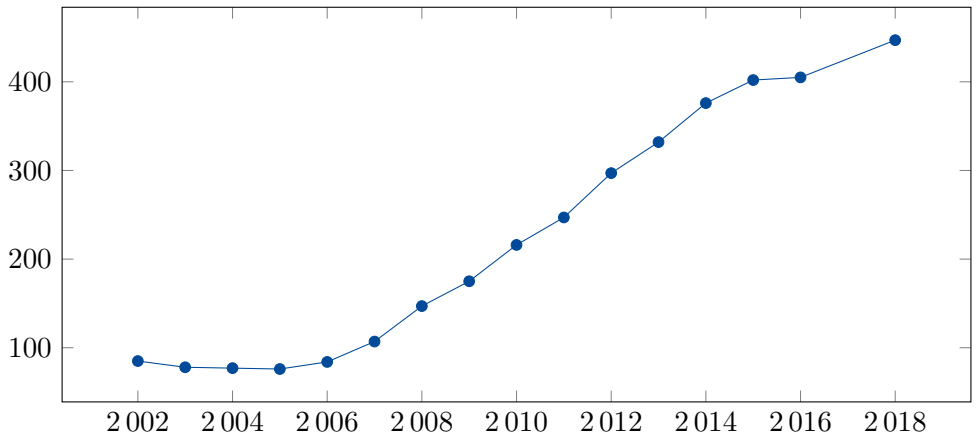

Abbildung 1.1: Zahl der KNX-Mitglieder seit 2002, Daten von [KNX17; KNX18]

Geräte verfügen in der Regel über eine Ethernet-Schnittstelle, sodass sie über IP-Netze konfiguriert werden können. Aufgrund der zentralen Rolle im Gebäudeautomationssystem hätte eine Sicherheitslücke, die sich über die IP-Schnittstelle entfernt ausnutzen lässt fatale Folgen.

Lamentablerweise räumen jedoch einige Hersteller der Absicherung dieser Schnittstellen keinen hohen Stellenwert ein, sodass für einige Geräte bereits bekannte Sicherheitslücken existieren. Umso überraschender ist in diesem Zusammenhang, dass mit geringem Aufwand zahlreiche solcher Geräte ungeschützt im Internet gefunden werden können.

Die möglichen Angriffsszenarien sind dabei im Rahmen der im Gebäude integrierten Infrastruktur unbegrenzt. Während das Ausschalten der Beleuchtung noch relativ harmlos wäre, hätte eine Manipulation der Klimatechnik schon weitreichendere Konsequenzen, die das Arbeiten im Gebäude unmöglich machen könnten. Zu den möglichen Angreifern gehören neben Hobbyhackern, die die Schwachstellen nur zum Spaß ausnutzen etwa Konkurrenzunternehmen oder feindliche Geheimdienste. Ein Angriffspotenzial ist somit vorhanden, in dieser Arbeit sollen daher Methoden entwickelt werden, um solche Angriffe zu untersuchen.

Um Angriffe auf Computersysteme zu untersuchen, eignen sich sogenannte *Honeypots*. Dabei handelt es sich um Systeme, die nach außen die Schnittstellen eines real existierenden Systems (beispielsweise ein SSH-Server oder ein DDC) nachbilden und sämtliche Zugriffsversuche umfangreich dokumentieren. Die dadurch gesammelten Daten

eignen sich zur Analyse der beobachteten Angriffe und für eine Abschätzung des Risikos eines Angriffes aus frei zugänglichen Systeme. Im Rahmen dieser Arbeit wurde daher exemplarisch ein Honeypot entwickelt, der die Schnittstellen eines DDC-4200 der Firma Kieback & Peter imitiert.

Durch den Einsatz des Honeypots sollen insbesondere folgende Forschungsfragen beantwortet werden:

- Wie kann man verhindern, dass Honeypots als solche erkannt werden?
- Welche Maßnahmen müssen getroffen werden um Honeypots für Angreifer präsent zu machen?
- Werden die bekannten Sicherheitslücken ausgenutzt?
- Was für Informationen lassen sich über Angreifer gewinnen?
- Wie lassen sich zielgerichtete Attacken erkennen?

Zur Beantwortung dieser Fragen soll ein entsprechender Honeypot entwickelt werden und 10 Wochen lang Daten von Angriffen sammeln. In Kapitel 2 werden daher zunächst die Grundlagen dieser Arbeit, Direct Digital Controls und Honeypots genauer vorgestellt. In Kapitel 3 werden dann mögliche rechtliche Probleme, die sich durch den Einsatz eines Honeypots ergeben können betrachtet und daraus abgeleitet Einschränkungen für den Honeypot festgelegt. In Kapitel 4 wird ein Konzept für einen Honeypot zur Nachbildung eines bestimmten DDC entwickelt, dieses Konzept wird dann in Kapitel 5 umgesetzt. Die durch den Einsatz des Honeypots gesammelten Daten werden schließlich in Kapitel 6 ausgewertet.

Kapitel 2

Stand der Technik

In diesem Kapitel

In diesem Kapitel werden die für das Verständnis der folgenden Kapitel notwendigen Grundlagen vorgestellt. Das beinhaltet zunächst Themen der Gebäudeautomation und dort insbesondere Direct Digital Controls. Darüber hinaus wird auch das Konzept der Honeypots eingeführt und verschiedene Möglichkeiten der Klassifizierung vorgestellt.

2.1 Gebäudeautomation

Die Gebäudeautomation wird in [Ver05] (zitiert nach [MHH10]) als „die digitale Mess-, Steuer-, Regel- und Leittechnik für die technische Gebäudeausrüstung" definiert. Dieser recht weit gefasste Definition wird oft dadurch von verwandten Gebieten wie der Gebäudesystemtechnik abgegrenzt, dass das System über zentrale Steuerungskomponenten verfügt, die die angeschlossenen Aktoren und Sensoren überwachen, regeln und steuern (Vgl. [Bau+13; MHH10]).

Durch den Einsatz von Gebäudeautomation lassen sich in verschiedenen Bereichen Vorteile erschließen [MHH10]:

- **Wirtschaftlichkeit und Energieeinsparung**: Durch eine zentrale und intelligente Steuerung von Licht- und Klimatechnik lassen sich Energiekosten sparen.
- **Komfort**: Für die Gebäudenutzer lassen sich Komfortfunktionen integrieren, wie etwa das automatische Schalten von Lampen abhängig von der Helligkeit und Bewegungsdaten.
- **Sicherheit**: Gebäudeautomationssysteme lassen sich prinzipiell einfach in Alarmanlagen integrieren, die etwa einen Alarm auslösen könnten, wenn im Gebäude ein Schalter ausgelöst wurde.
- **Flexibilität**: Während in konventionellen Elektroinstallationen ein Lichtschalter direkt mit den zugehörigen Lampen verbunden ist, ist ein Lichtschalter in einem Gebäudeautomationssystem mit allen Lampen verbunden und durch die Programmierung des Systems wird gesteuert, welche Lampen geschaltet werden. Falls in einem Gebäude also Räume vergrößert werden, ist keine aufwendige Änderung der Elektroninstallation mehr notwendig, da eine Neuprogrammierung des Systems genügt.

2.1.1 Übertragungsmethoden

Die zentrale Steuerung führt dazu, dass alle Teilnehmer des Gebäudeautomationssystems an ein mit der Steuerungskomponente gemeinsames Kommunikationsmedium angeschlossen sind. Häufig verwendete Systeme umfassen dabei unter anderem:

- **KNX**: Bei KNX handelt es sich um ein Feldbussystem, das bereits in den 1980er Jahren, ursprüngliche unter dem Namen „Europäischer Installationsbus" (EIB) entwickelt wurde [KSH09]. Bei KNX handelt es sich um einen offenen Standard, sodass prinzipiell jeder Hersteller KNX-kompatible Geräte entwickeln kann. Neben einem Zweidraht-Feldbus ist inzwischen auch die Kommunikation über IP, Powerline oder Funk spezifiziert.
- **ZigBee**: ZigBee ist ein drahtloses Kommunikationssystem, das auf dem IEEE 802.15.4-Standard aufbaut [Zig15]. Da eine Funkverbindung in Gebäuden aufgrund von Hindernissen üblicherweise nicht zwischen beliebigen Teilnehmern möglich ist, spezifiziert ZigBee ein Mesh-Kommunikationsprotokoll, bei dem Datenpakete über mehrere Funkverbindungen weitergeleitet werden können.

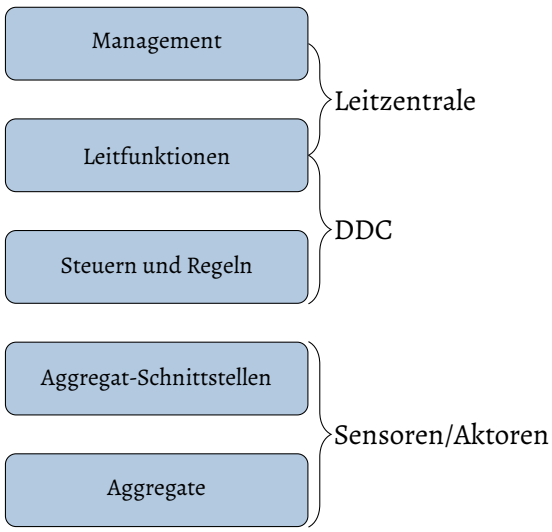

Abbildung 2.1: Hierarchische Struktur in der Gebäudeautomation

- **BACnet** (Building Automation and Control Networks): BACnet ist ein standardisiertes Kommunikationsprotokoll das für den Einsatz in der Gebäudeautomation entwickelt wurde. Da die Anzahl der unterstützten Geräte vergleichsweise gering ist, wird es hauptsächlich zur zentralen Steuerung von Gebäuden oder Gebäudeteilen eingesetzt, wobei nur die Verbindung zu einem Steuergerät über BACnet realisiert wird [MHH10].

2.1.2 Hierarchischer Aufbau

Zur Strukturierung von Gebäudeautomationssystemen gibt es zwei verschiedene Möglichkeiten: Ein Drei-Ebenen-Modell (Vgl. [Kar04]), das auf der Unterscheidung zwischen den verwendeten Geräten beruht und ein Fünf-Ebenen-Modell, das eine funktionale Trennung vornimmt.

Abbildung 2.1 stellt das Fünf-Ebenen-Modell nach [MHH10] dar, wobei den einzelnen Ebenen die zugehörigen Geräte zugeordnet wurden.

2.1.3 Direct Digital Control

Als Direct Digital Control (*DDC*, auch *DDC-GA*) werden Geräte bezeichnet, die umfangreiche Steuerungs- und Regelungsaufgaben in Gebäudeautomationssystemen ausführen [MHH10]. Somit agieren sie als Verbindungsstück zwischen den Sensoren und Aktoren im System und der Leitzentrale, in der zentral für beliebige angeschlossene Liegenschaften Einstellungen vorgenommen werden können. Die Verbindung zur Leitzentrale erfolgt häufig über das Internet, insbesondere wenn voneinander entfernte Gebäude gesteuert werden sollen. In der Regel wird dafür auf das BACnet-Protokoll zurückgegriffen, das für diesen Einsatzzweck entworfen wurde [Dip01]. Einige DDC verfügen jedoch auch über weitere Schnittstellen, wie etwa eine Webschnittstelle, mit deren Hilfe Konfigurationsänderungen durchgeführt werden können, etwa wenn keine spezialisierte Steuerungssoftware vorhanden ist, oder Änderungen durch einen ungeschulten Anwender durchgeführt werden.

Die Sensoren und Aktoren sind direkt mit einem DDC verbunden, etwa über ein Feldbusprotokoll (Vgl. Unterabschnitt 2.1.1). Bei einer Ansteuerung über KNX werden einzelne Gebäudeteile häufig von verschiedenen DDC bedient, dadurch lässt sich die Komplexität des Systems verringern und für den Fall eines Ausfalls eines DDC die dadurch entstehenden Auswirkungen vermindern. Da die einzelnen Systemteile in der Managementebene ohnehin zusammengeschaltet werden, ergibt sich dadurch in der Konfiguration kein Mehraufwand durch den Nutzer.

2.2 Honeypots

Als Honeypots werden spezielle Computersysteme bezeichnet, deren Zweck es ist, untersucht, angegriffen und kompromittiert zu werden [MA07; MHH10]. Üblicherweise sollen solche Aktionen verhindert werden und der Einsatz von Honeypots ist eine wichtige Maßnahme um das zu erreichen [MHH10]. So betreibt etwa die Deutsche Telekom ein Netz von über eintausend Honeypots mit dem Ziel sogenannte „Cyberangriffe" [HB17] abzuwehren, sehr zum Lob der Sicherheitsbehörden [HB17].

Der Name Honeypot ergibt sich aus dem Ziel das durch deren Einsatz verfolgt wird. Da Bären beispielsweise in Geschichten eine Vorliebe für Honig nachgesagt wird (Vgl. etwa

[MS04]), kann ein Honigtopf benutzt werden, um sie anzulocken. Analog benutzt man in Computersystemen Honeypots um Hacker anzulocken. Dies kann etwa den folgenden Zielen dienen [MA07]:

- Angreifer sollen von tatsächlich wertvollen Zielen (etwa ein Datenbankserver mit den Kundendaten) abgelenkt werden, um Schaden abzuhalten
- Ein Angriff auf einen Honeypot kann als Frühwarnsystem für Angriffe auf andere Systeme im Netz dienen. Wird ein Honeypot angegriffen, sollten die anderen Systeme genauer beobachtet werden.
- Durch detailliertes Protokollieren der Angriffe auf dem Honeypot ist es möglich, die Angriffe im Nachhinein zu analysieren und so wertvolle Informationen über die Abwehr von Attacken zu erhalten.

Honeypots sind davon abhängig, dass Angreifer mit ihnen interagieren. Da autorisierte Benutzer nicht über die protokollierte Schnittstelle mit den Honeypots arbeiten, ist prinzipiell davon auszugehen, dass jeder Zugriff mit boshafter Absicht geschieht. Die meisten Angriffe werden jedoch automatisiert durch bereits infizierte Systeme initiiert, sodass eine Strafverfolgung aller Attacken kaum durchführbar wäre.

Die Deutsche Telekom zog mit ihrem Netzwerk aus Honeypots[1] 2017 etwa durchschnittlich 6 Millionen Angriffe pro Tag auf sich [HB17]. Somit werden auch besondere Anforderungen an die Auswertung der Attacken gestellt.

2.2.1 Arten von Honeypots

Honeypots lassen sich - basierend auf dem dem Angreifer bereitgestellten Funktionsumfang - in 3 verschiedene Kategorien einteilen [MA07]:

[1] Solche Netzwerke werden auch als Honeynet bezeichnet.

1. **Low-Interaction Honeypots** akzeptieren lediglich eine Verbindung. Weitergehende Aktionen mit dem System sind nicht möglich, sodass für den Angreifer die Natur des Honeypots unverzüglich offenbart wird. Einzig die Tatsache, dass ein Zugriffsversuch erfolgte kann protokolliert werden.
2. **High-Interaction Honeypots** bilden eine komplette Systemumgebung nach, mit der ein Angreifer uneingeschränkt interagieren kann. Sämtliche Aktionen werden jedoch umfangreich protokolliert, sodass detaillierte Informationen zu den Angriffen gesammelt werden können.
3. **Medium-Interaction Honeypots** bieten eingeschränkte Aktionsmöglichkeiten, sind also zwischen den beiden anderen Kategorien einzuordnen. Üblicherweise werden hierbei Aktionen, die reale Systeme gefährden könnten, nicht gestattet. Beispielsweise könnte der Netzwerkzugriff vom Honeypot aus gesperrt sein, um zu verhindern, dass vom System DoS-Attacken gegen andere Systeme gestartet werden. Einige Systeme simulieren dabei bestimmte Aktionen, um die Erkennung des Honeypots zu erschweren, so könnte beispielsweise bei der Ausführung von PING oder NMAP nach angemessener Zeit eine Falschausgabe erfolgen, um eine erfolgreiche Ausführung zu simulieren.

Die Wahl eines bestimmten Typs von Honeypot hängt dabei maßgeblich davon ab, wie viele Informationen über die Angriffe gesammelt werden sollen. Üblicherweise sind hierbei so viele wie möglich gewünscht, sodass eine Abwägung zwischen dem Informationswunsch einerseits und dem Aufwand der Nachbildung beziehungsweise dem Risiko für andere Systeme andererseits getroffen werden muss.

2.2.2 Vor- und Nachteile

Bevor man Honeypots einsetzt, muss man die Vor- und Nachteile abwiegen um zu entscheiden, ob sie eine sinnvolle Integration in das jeweilige Sicherheitssystem darstellen. Zu den **Vorteilen** gehören dabei insbesondere [MA07; Spi03]:

- **Kleine Datensätze:** Da Honeypots nur unautorisierte Zugriffe protokollieren, enthalten die aufgezeichneten Datensätze keine insignifikanten Hintergrunddaten, wie etwa die Aktivität berechtigter Nutzer. Dadurch ist eine einfachere Analyse der Angriffe möglich.

- **Minimale Ressourcenanforderungen:** Für das Aufzeichnen der Daten und das Nachbilden der Systeme sind in den meisten Fällen nur wenig Ressourcen erforderlich, sodass geringe Hardwarekosten entstehen.
- **Einfachheit:** Im Vergleich etwa zu Firewalls oder Intrusion Detection Systemen sind Honeypots für Standarddienste wie etwa SSH leicht einzurichten.
- **Entdecken neuer Werkzeuge:** Mit Honeypots lassen sich die von Angreifern eingesetzten Werkzeuge analysieren, was das Verständnis der Angriffsvektoren vereinfacht.

Die **Nachteile** beinhalten [MA07; Spi03]:

- **Eingeschränkte Wahrnehmung:** Mit Honeypots kann man nur solche Daten erfassen, die direkt mit Angriffen auf den Honeypot zusammenhängen. Werden andere Dienste im selben Netzwerk angegriffen, kann ein Honeypot keine Erkenntnisse liefern.
- **Auffälligkeit und Erkennung:** Es existieren Techniken zur Entdeckung von Honeypots (Mehr dazu in Abschnitt 4.2). Der kleinste Fehler bei der Konfiguration kann dabei dazu führen, dass der Honeypot leicht von einem echten System zu unterscheiden ist.
- **Risiko einer Übernahme:** Falls die Honeypot-Software Sicherheitslücken besitzt, besteht das Risiko einer Übernahme des Hostsystems durch den Angreifer. Da sich dieses System möglicherweise bereits im internen Netz befindet, kann dadurch eine Gefährdung anderer Systeme im Netz entstehen. Aus diesem Grund ist es sehr wichtig, die eingesetzte Software aktuell zu halten, und Firewalls um den Honeypot korrekt zu konfigurieren.

Kapitel 3

Rechtliche Aspekte

In diesem Kapitel

In diesem Kapitel sollen rechtliche Fragen beantwortet werden, die sich aus der Entwicklung und aus dem Betrieb des Honeypots ergeben. Auch sollen einige Handlungsempfehlungen zur Vermeidung rechtlicher Probleme entwickelt werden.

Die Beantwortung der Fragen orientiert sich an den Gesetzen der Bundesrepublik Deutschland. Die Inhalte wurden mit größtmöglicher Sorgfalt recherchiert, da der Autor allerdings über keinerlei juristische Ausbildung verfügt, handelt es sich lediglich um eine unverbindliche Untersuchung der Rechtslage. Der Zweck dieses Kapitels ist nicht, allgemeine Rechtsberatung zum Einsatz von Honeypots zu geben, sondern lediglich mögliche Probleme im konkreten Fall dieser Arbeit zu untersuchen.

3.1 Aufzeichnung der Daten

Hierbei soll überprüft werden, ob es gestattet ist, die Aktionen der Angreifer auf dem Honeypot ohne deren Einverständnis zu protokollieren. Dazu gehören sämtliche getätigten Ein- und Ausgaben, heruntergeladene Programme und Änderungen am System.

Die einschlägige Richtlinie ist hierbei §202a „Ausspähen von Daten" des Strafgesetzbuchs [Deu71]. Dort heißt es:

> (1) Wer unbefugt sich oder einem anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
>
> (2) Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden.

Für diesen Straftatbestand ist demnach regelmäßig erforderlich, dass die Daten besonders gegen unberechtigten Zugriff gesichert sind. Ein solcher Schutz ist jedoch nicht vorhanden, da sämtliche Daten durch den Honeypot unmittelbar in eine Protokolldatei gespeichert werden.

Die Datenspeicherung scheint demnach zumindest im Falle nicht personenbezogener Daten rechtmäßig zu sein.

3.2 Personenbezogene Daten

Bei der Speicherung beziehungsweise dem Verarbeiten personenbezogenen Daten sind besondere datenschutzrechtliche Gesichtspunkte zu beachten. Im Sinne der dazu in der Europäischen Union gültigen Datenschutzgrundverordnung bezeichnet der Begriff in

Artikel 4 [Das16] „alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person (im Folgenden „betroffene Person“) beziehen; [...]”.

Zum Zweck dieser Einstufung ist daher nicht maßgeblich, ob eine Absicht zur Identifizierung der „betroffenen Person” besteht oder ob solch eine Identifizierung jemals durchgeführt wird. Allein die Möglichkeit einer Identifizierung genügt, um einen besonderen Schutz zu erwirken. Im Falle einer IP-Adresse ist zwar die Zuordnung nicht ohne weiteres möglich, hier wäre erforderlich, dass der zugehörige Internetprovider oder etwa eine dritte Internetseite die entsprechende Information erteilt. Diese mittelbare Identifizierbarkeit genügt allerdings um den Rechtsanspruch zu begründen (Vgl. [Bun17; Ger16]).

Dadurch muss nun eine der folgenden, in Artikel 6 [Das16] genannten Bedingungen erfüllt sein, um die IP-Adresse rechtmäßig zu verarbeiten:

(a) Die betroffene Person hat ihre Einwilligung zu der Verarbeitung der sie betreffenden personenbezogenen Daten für einen oder mehrere bestimmte Zwecke gegeben;
(b) die Verarbeitung ist für die Erfüllung eines Vertrags, dessen Vertragspartei die betroffene Person ist, oder zur Durchführung vorvertraglicher Maßnahmen erforderlich, die auf Anfrage der betroffenen Person erfolgen;
(c) die Verarbeitung ist zur Erfüllung einer rechtlichen Verpflichtung erforderlich, der der Verantwortliche unterliegt;
(d) die Verarbeitung ist erforderlich, um lebenswichtige Interessen der betroffenen Person oder einer anderen natürlichen Person zu schützen;
(e) die Verarbeitung ist für die Wahrnehmung einer Aufgabe erforderlich, die im öffentlichen Interesse liegt oder in Ausübung öffentlicher Gewalt erfolgt, die dem Verantwortlichen übertragen wurde;
(f) die Verarbeitung ist zur Wahrung der berechtigten Interessen des Verantwortlichen oder eines Dritten erforderlich, sofern nicht die Interessen oder Grundrechte und Grundfreiheiten der betroffenen Person, die den Schutz personenbezogener Daten erfordern, überwiegen, insbesondere dann, wenn es sich bei der betroffenen Person um ein Kind handelt.

Offensichtlich treffen die Bedingungen (b)-(f) hier nicht zu. Da das Einholen einer Einwilligung der „betroffenen Person" (des Angreifers) im Sinne der Geheimhaltung der Natur des Honeypots nicht zweckmäßig erscheint, ist es erforderlich, die IP-Adresse in einer anonymisierten Form zu speichern um keinen Rechtsbruch zu begehen. Dasselbe gilt für alle anderen personenbezogenen Daten die durch die Benutzung des Honeypots gespeichert werden könnten.

3.3 Aktionen des Angreifers

Hier soll untersucht werden, welche rechtlichen Folgen sich für den Betreiber des Honeypots ergeben, falls durch eine Sicherheitslücke in der Honeypot-Software der Angreifer die Kontrolle über das gesamte System gewinnt und diese nutzt um schadhafte Aktionen gegen Dritte auszuführen. Beispielsweise könnte das Honeypot-Gerät als Teil eines Botnetzes DoS-Angriffe ausführen und so dazu beitragen, Webseiten zu überlasten.

In § 27 „Beihilfe" des Strafgesetzbuches [Deu71] heißt es dazu:

> (1) Als Gehilfe wird bestraft, wer vorsätzlich einem anderen zu dessen vorsätzlich begangener rechtswidriger Tat Hilfe geleistet hat.
>
> (2) Die Strafe für den Gehilfen richtet sich nach der Strafdrohung für den Täter. Sie ist nach § 49 Abs. 1 zu mildern.

Ein Vorsatz ist in diesem Fall durch die Bereitstellung des Honeypots vermutlich nicht zu begründen, insbesondere wenn ein Medium-Interaction Honeypot eingesetzt wird, der die schadhaften Einflüsse eines Angreifers begrenzen kann.

Zivilrechtliche Ansprüche, etwa auf Schadensersatz zugunsten eines Geschädigten sind davon jedoch nicht ausgenommen. Dazu heißt es in § 823 Schadensersatzpflicht des Bürgerlichen Gesetzbuches []:

> (1) Wer vorsätzlich oder fahrlässig das Leben, den Körper, die Gesundheit, die Freiheit, das Eigentum oder ein sonstiges Recht eines anderen widerrechtlich verletzt, ist dem anderen zum Ersatz des daraus entstehenden Schadens verpflichtet.

In § 252 „Entgangener Gewinn" ebd. wird der Begriff des Schadens erweitert, sodass auch der Gewinnausfall etwa durch Nichtverfügbarkeit einer Webseite ersetzt werden muss:

> (1) Der zu ersetzende Schaden umfasst auch den entgangenen Gewinn.
>
> (2) Als entgangen gilt der Gewinn, welcher nach dem gewöhnlichen Lauf der Dinge oder nach den besonderen Umständen, insbesondere nach den getroffenen Anstalten und Vorkehrungen, mit Wahrscheinlichkeit erwartet werden konnte.

Allerdings ist für das Eintreten einer Schadensersatzpflicht ein Vorsatz oder fahrlässiges Handeln erforderlich. Ein Vorsatz liegt offenbar nur dann vor, wenn in irgendeiner Weise mit dem eigentlichen Angreifer zusammengearbeitet wurde. Fahrlässigkeit wird in § 276 „Verantwortlichkeit des Schuldners" ebd. wie folgt definiert:

> (2) Fahrlässig handelt, wer die im Verkehr erforderliche Sorgfalt außer Acht lässt.

Sofern der Honeypot ausreichend gegen ungewünschte Nutzung abgesichert ist, ist auch die erforderliche Sorgfalt gewährleistet und eine Fahrlässigkeit somit unbegründet. Es ist daher angeraten, dass die Konfigurationsschnittstelle, etwa durch ein VPN und den Einsatz starker Kryptographie abgesichert wird. Weiterhin sollte die eingesetzte Software vor und regelmäßiger während der Nutzung auf bekannte Sicherheitslücken überprüft werden.

3.4 Unerwünschte Daten

Ebenfalls möglich wäre es, dass der Angreifer urheberrechtlich geschützte Werke ohne die notwendigen Lizenzen erworben zu haben speichert - oder Material, dessen Besitz illegal ist, auf dem Gerät platziert. Das ist aufgrund der Funktionsweise des Honeypots, bei dem jede Aktion protokolliert wird, prinzipbedingt möglich.

Im Falle sogenannter Sharehoster, bei denen Nutzer beliebige Werke zum Download zur Verfügung stellen können, haben Gerichte entschieden, dass solche Dienste nicht als Täter, Mittäter, Anstifter oder Gehilfe haftbar für Uploads Dritter sind, sofern sie ihrer redaktionellen Aufsichtspflicht nachkommen, gemeldete illegale Werke entfernen und sie mit ihrem Geschäftsmodell keine Straftaten begünstigen (Vgl. [Bun13; LG 16; OLG17]).

Diese Prinzipien lassen sich vermutlich auch auf diesen Anwendungsfall übertragen, insbesondere da die gespeicherten Daten nur für einen eingeschränkten Personenkreis zugänglich sind, der mit der Auswertung der Protokolle befasst ist. Eine Verbreitung der Inhalte findet demnach nicht statt.

Für den unwahrscheinlichen Fall, dass tatsächlich urheberrechtlich geschützte oder illegale Werke gespeichert werden, sollten die Daten unverzüglich nach Kenntnisnahme gelöscht werden.

3.5 Geistiges Eigentum

Beim Erstellen des Honeypots ist es erforderlich, die Details der Originalschnittstellen so nachzubilden, dass für einen Nutzer kein Unterschied festzustellen ist. Da die Weboberfläche des DDC4200 sowohl clientseitigen Programmcode als auch ein Logo der Firma Kieback&Peter enthält, muss beides auch in der nachgebildeten Programmoberfläche des Honeypots enthalten sein.

Prinzipiell besitzt der Urheber des Logos beziehungsweise des Programmcodes auch das alleinige Verwertungsrecht (Vgl. §15 „Allgemeines" [Deu65]). Offenbar besitzt die Firma

Kieback&Peter ein vom jeweiligen Urheber erworbenes Nutzungsrecht für die entsprechenden Werke, das es ihr gestattet, sie mit ihren Geräten auszuliefern. Käufer (oder Entleiher) des Geräts besitzen dieses Recht grundsätzlich nicht, sodass eine weitere Verbreitung und Vervielfältigung eine entsprechende Lizenz erfordert.

Ein Anspruch auf Unterlassung oder Schadensersatz besteht daher von Seiten des Urhebers prinzipiell, auch wenn eine tatsächliche Inanspruchnahme unwahrscheinlich ist. Zur Absicherung wurde in diesem Zusammenhang jedoch bei der Firma Kieback&Peter eine Nutzungslizenz für den Einsatz im Honeypot angefragt.

Kapitel 4

Konzept

In diesem Kapitel

4.1 Bekanntmachen

Nach der Entwicklung des Honeypots ist es wichtig, dass der Honeypot für mögliche Angreifer auffindbar ist. Hier stellt sich daher die Frage, wie Angreifer ihre Ziele auffinden und wie der Honeypot als mögliches Ziel platziert werden kann.

Ein beliebtes Werkzeug zum Aufspüren bestimmter Ziele ist die Suchmaschine `Shodan.io`. Mithilfe von Shodan lassen sich mit dem Internet verbundene Geräte anhand flexibler Suchkriterien finden. So fördert etwa die Suche nach 'DEFAULT PASSWORD' PORT:23' ein Telnet-Banner zu Tage, in dem es heißt: „default username is 'admin' and password is

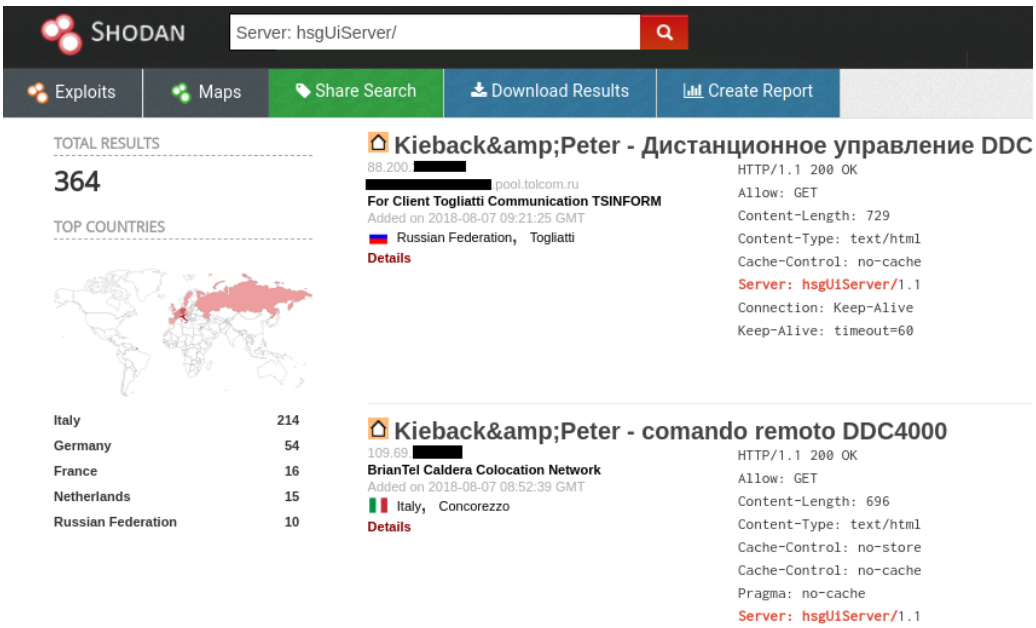

Abbildung 4.1: Mit dem richtigen Suchbegriff können in Shodan 364 DDC-Geräte von Kieback&Peter gefunden werden [Sho]

'password'. Please change them immediately." Bei der Suche nach COUNTRY:'DE' OS:'WINDOWS XP' PORT:'3389' finden sich mit Windows XP betriebene Rechner in Deutschland, die über das Remote-Desktop Protokoll erreichbar sind - selbst ein Screenshot des Anmeldebildschirms wird dargestellt.

Auch öffentlich erreichbare Weboberflächen des DDC4200 finden sich mit einer Suche nach SERVER: HSGUISERVER. HSG ist dabei die Abkürzung des Softwareunternehmens „holistische SoftwareGarage", das für die Entwicklung der Clientoberfläche verantwortlich ist [hol]. Der installierte Webserver gibt sich im HTTP-Header als hsgUiServer zu erkennen.

Der Honeypot sollte somit in einer späteren Installation mit diesem Suchbegriff ebenfalls gefunden werden. Welche Handlungen sind also erforderlich um eine Aufnahme in den Katalog von Shodan zu erwirken?

Von Shodan werden dabei mehrere Server betrieben die nach dem in Abbildung 4.2 beschriebenen Schema das komplette Internet scannen.

Dieses Vorgehen ist massiv parallelisierbar und prinzipiell nur durch den Datendurchsatz der Datenbank begrenzt. Es existiert keine öffentliche Dokumentation über die

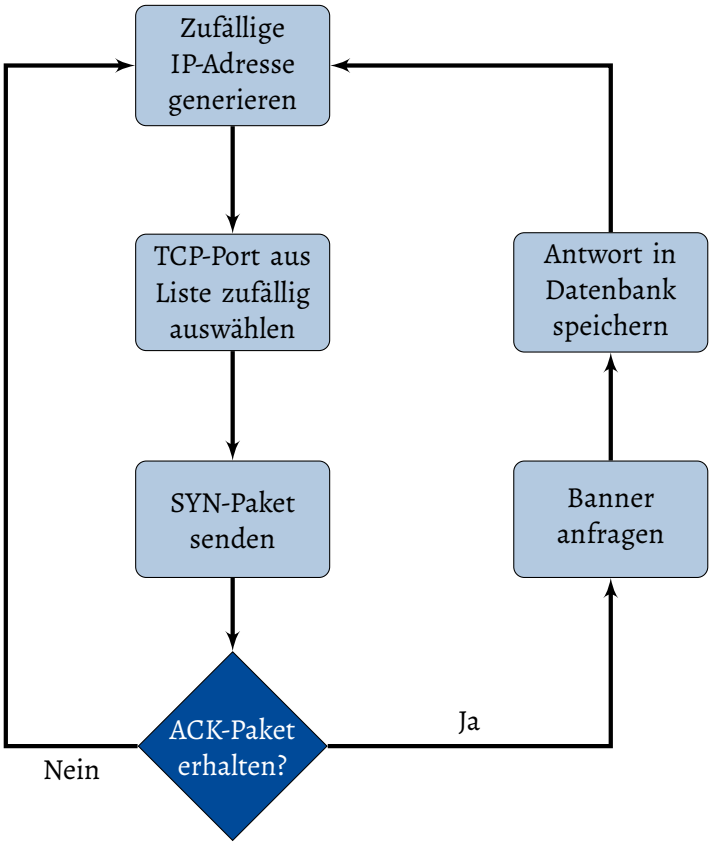

Abbildung 4.2: Shodans Scanroutine, nach [Bod+14]

durchsuchten Ports, allerdings lassen stichprobenartige Testanfragen den Schluss zu, dass zumindest alle TCP Ports im Bereich 1-10,000 gescannt werden. Um zu verhindern, dass das eigene Gerät in der Datenbank von Shodan auftaucht, müsste man den Zugriff so einschränken, dass Anfragen von Shodan nicht beantwortet werden.

Um den Honeypot auf Shodan zu listen, ist somit lediglich eine Installation mit einer öffentlichen IP-Adresse und einem von Shodan gescannten Port erforderlich. Zur Einfachheit werden die Ports 80 (alternativ 8080) und 22 beziehungsweise 2222 verwendet. Im Versuch von [Bod+14] wurden beispielsweise die Geräte ohne weiteres Zutun nach maximal 19 Tagen aufgeführt.

4.2 Honeypoterkennung

Der Honeypot sollte für potenzielle Angreifer möglichst nicht von einem echten Gerät zu unterscheiden sein. In diesem Abschnitt werden daher aus Angreiferperspektive Techniken zum Erkennen von Honeypots betrachtet und geeignete Gegenmaßnahmen entwickelt.

4.2.1 IP-Adresse

Einem Angreifer kann bereits die IP-Adresse viel über das mögliche Angriffsziel verraten. Da die IP-Adresse darüber hinaus kaum fälschbar ist, kann eine Untersuchung der IP-Adresse erste Anhaltspunkte bieten.

Zu einer IP-Adresse gibt es verschiedene Metainformationen die vom Besitzer der IP-Adresse, also nicht in jedem Fall vom Nutzer festgelegt werden:

- Geschätzte Teilnehmerposition („Geolocation")
- Eigentümerinformationen („Whois")
- Reverse-DNS Hostname

Mithilfe der **Geolocation-Daten** kann ein Angreifer bereits im Vorfeld lohnenswerte Ziele festmachen, indem er sich auf bestimmte Länder beschränkt. Falls das Ziel des Angreifers beispielsweise Erpressung ist, würde er sich auf Länder beschränken, deren Bewohner üblicherweise zahlungskräftig sind: ein Unternehmen in der Schweiz wäre also einem in Mosambik vorzuziehen.

Auch eine Plausibilitätsabschätzung könnte damit erfolgen. Wenn Geräte eines bestimmten Herstellers hauptsächlich in Europa eingesetzt werden, könnte eine Installation in Amerika verdächtig sein, vor allem dann wenn das eingesetzte Gerät gar nicht für die dortigen Bedingungen ausgelegt ist, etwa wenn es eine 230V-Stromversorgung voraussetzt und im Einsatzland 110V üblich sind.

Auch die **Eigentümerinformationen** sind ein mögliches Kriterium um zu entscheiden, ob ein Angriff erfolgen sollte. Eine entsprechende Recherche kann zum Einen verdächtige Akteure aufdecken, beispielsweise eine Universität, die möglicherweise die Sicherheit der eingesetzten Geräte überprüft und zum Anderen besonders attraktive Ziele, wie Großunternehmen hervorheben.

Da der zur IP-Adresse gehörende **Hostname** häufig zur Beschreibung der IP-Adresse genutzt wird, sollte auch er in dieser Hinsicht betrachtet werden. Völlig ungeeignet wäre ein Hostname der die Identität des Honeypots offenbart, beispielsweise HONEYPOT.INFORMATIK.UNI-ROSTOCK.DE. Qualifizierte Angreifer würden einen Dienst auf diesem Rechner vermutlich nicht angreifen. Ebenfalls ungeeignet wären IP-Adressen bei bekannten Hosting-Providern, da erkennbar wäre, dass es sich tatsächlich um einen Server handelt.

Um für den Honeypot passende IP-Adressen zu verwenden, werden folgende Strategien verwendet:

1. Ein Teil der Honeypots wird an gewöhnlichen privaten Internetanschlüssen betrieben. Der Großteil der bei Shodan gefundenen DDC4200-Installationen ist an solchen Anschlüssen zu finden [Sho18]. Die dabei verwendeten IP-Adressen sollten keinen Verdacht erregen.
2. Weitere Honeypots sollen über „PlanetLab Europe" [UPM+] bereitgestellt werden. Dabei handelt es sich um eine Organisation, deren Mitglieder Server in ihrer Infrastruktur bereitstellen um im Gegenzug Ressourcen auf den von den anderen Mitgliedern bereitgestellten Servern zu nutzen. Beim Großteil der Mitglieder handelt es sich um Universitäten. Hier ist lediglich darauf zu achten, dass die Hostnames zur IP-Adresse nicht nach dem Schema PLANETLAB-1.X.Y vergeben sind.

4.2.2 OS-Fingerprinting

Als Werkzeug für Netzwerkscans besitzt `nmap` [Gor] eine Funktion um aus den Metainformationen aus TCP-Antwortpaketen des Zielrechners Schlüsse auf das verwendete Betriebssystem zu ziehen. Dabei wird auch eine Version des Betriebssystems geschätzt, indem Änderungen im Antwortverhalten zwischen den Versionen verglichen werden [Lyo10].

Da eingebettete Geräte wie etwa der DDC4200 üblicherweise bereits vor der Auslieferung mit der Betriebssoftware ausgestattet werden und Änderungen an der Software nur durch Updates des Herstellers vorgenommen werden, kann üblicherweise auch die verwendete Betriebssystemversion eingegrenzt werden. Falls ein `nmap`-Scan des Geräts nun eine deutlich neueres Betriebssystem ermittelt, kann dies ein Anzeichen dafür sein, dass es sich etwa um einen Honeypot handelt.

Um dies zu vermeiden, soll zunächst überprüft werden, wie `nmap` das Betriebssystem des Zielrechners ermittelt und dann untersucht werden, wie verhindert werden kann, dass sich der Honeypot in dieser Hinsicht aufwändig verhält.

Um die für die Auswertung notwendigen Daten zu sammeln, werden zunächst insgesamt 16 TCP-, UDP- und ICMP-Pakete an bekannte offene und geschlossene Ports gesendet [Lyo10]. Die Anfragen sind dabei so gestellt, dass unklare Formulierungen in den Standards unterschiedliche Antworten erlauben.

Die Anfragen werden in [Lyo10] genau beschrieben und beinhalten etwa sogenannte „Christmas Tree Packets", bei denen jede einzelne Paketoption gesetzt ist oder Pakete mit bestimmten Werten im Timestamp- und Window-Size-Feld.

Die gesammelten Antwortpakete werden daraufhin auf bekannte Merkmale untersucht. Beispielsweise wird die initiale TCP-Sequenznummer in manchen TCP-Implementierungen in 64000er-Schritten erhöht, während andere Implementierungen den selben Wert für das ID-Feld im IP-Paket sowohl für TCP- als auch für ICMP-Pakete verwenden. Die gesammelten Ergebnisse werden dann zu einem Fingerabdruck zusammengesetzt, der in

create default set default personality "Linux 2.2.14ßet default default tcp action block add default udp port 53 "./scripts/dnstool.py"

einer in nmap enthaltenen Datenbank nachgeschlagen wird. Wird kein bekanntes Betriebssystem gefunden, kann der Nutzer den Eintrag selbst in die Datenbank einpflegen, wenn ihm das Betriebssystem bekannt ist.

Aufgrund der Fülle an durchgeführten Tests und dadurch, dass das Verhalten im Betriebssystemkernel definiert ist, ist eine Manipulation der Antwort mit umfangreichen Modifikationen verbunden. Prinzipiell wäre es natürlich möglich, auf dem Honeypot die selbe Kernelversion einzusetzen, die auch auf dem echten Gerät benutzt wird. Da diese allerdings bereits 8 Jahre alt ist[1], wäre ein solches Vorgehen aus Kompatibilitäts- und Sicherheitsgründen nicht ratsam.

Um dieses Problem zu lösen, wurden verschiedene Lösungen entwickelt. Eine davon ist ein Patch für den Linux-Kernel namens IP Personality [RS00], der die in nmap betrachteten TCP-Charakteristiken regelbasiert beispielsweise für bestimmte Quelladressen verändern kann. Mit verhältnismäßig geringem Konfigurationsaufwand lassen sich dadurch verschiedene Linux Kernel-Versionen vortäuschen, da die Bearbeitungsregeln auf der nmap-Datenbank basiert. Der Nachteil dieser Lösung ist allerdings, dass der Patch nur für Kernel der 2.4er-Serie verfügbar ist, also einen noch älteren Kernel erfordert, als ohnehin schon auf dem DDC4200 installiert ist.

Bei einem Kernel-Patch namens FingerPrintFucker [Pac01], der einen vergleichbaren Funktionsumfang für FreeBSD bietet, ergibt sich ein ähnliches Problem, da der Patch seit 2001 nicht mehr aktualisiert wurde. Das selbe gilt für den „Stealth Patch" [Tri02], dessen letzte Version immerhin 2002 erstellt wurde.

Eine Lösung die auch mit modernen Betriebssystemen eingesetzt werden kann ist honeyd [Pro08]. Statt das Verhalten des Betriebssystems zu verändern, wird hierbei ein Netzwerkgerät simuliert, das das gewünschte Verhalten an den Tag legt. Dadurch ist es möglich, dass honeyd als Userspace-Programm läuft und somit nicht vom verwendeten Kernel abhängt. Die Konfiguration ist dabei denkbar einfach [Pro03]:

[1] Auf dem Gerät wird laut uname Linux 2.6.34 eingesetzt

Mit honeyd lassen sich beliebig viele Netzwerkgeräte simulieren, die alle eine eigene IP-Adresse bekommen. Das macht den Einsatz vor allem für Unternehmensnetze interessant, da sich mit geringem Kostenaufwand ganze Netzwerke von Honeypots simulieren lassen. Im Anwendungsfall des DDC-Honeypots ist das allerdings nicht erforderlich, da bereits eine im Netzwerk gefundene DDC glaubhaft ist.

Eine Veränderung des Verhaltens des TCP-Stacks ist möglicherweise aber gar nicht notwendig, wenn der Honeypot wie in Unterabschnitt 4.2.1 beschrieben hinter einem handelsüblichen Router durch NAT betrieben wird. In dieser Konfiguration kommt der Großteil der von nmap gesammelten Daten vom Router selbst und nicht vom Honeypot, sodass ein OS-Fingerprinting entweder fehlschlägt oder Daten zum Router anzeigt. Da eine Shodan-Suche viele DDC mit einer dynamischen IP-Adresse liefert, entspricht solch eine Konfiguration auch einem Großteil der tatsächlichen Geräte, sodass dies eine im Sinne der Unauffälligkeit des Honeypots passende Strategie zu sein scheint.

4.2.3 Antwortzeit

Auch die Antwortzeit des Honeypots auf eine Anfrage kann dem Angreifer wichtige Informationen liefern. Solche Timing-Angriffe werden üblicherweise gegen Kryptosysteme verwendet (Vgl. etwa [Ber05; Koc96; Sch00; BB05]), sie könnten in diesem Fall allerdings auch dafür geeignet sein, den Honeypot zu enttarnen.

Der echte DDC4200 ist mit im Vergleich zu modernen Rechnern schlechter Hardware ausgestattet. Außerdem muss er, im Gegensatz zum Honeypot, je nach Eingabe unter Umständen Aktionen über den Feldbus durchführen. Das führt dazu, dass der Honeypot deutlich schneller auf Eingaben reagieren kann, als es dem echten Gerät möglich ist. Da dort die Reaktionszeit im Bereich mehrerer Sekunden liegen kann, ist eine solche Verzögerung auch problemlos über das Netzwerk messbar.

Es ist daher notwendig, dass der Honeypot in ähnlicher Weise langsam reagiert. Da eine Antwortzeit auf jede Anfrage von beispielsweise einer Sekunde recht auffällig wäre, sollte das System nach dem Beantworten einer Anfrage eine zufällige Zeitspanne warten, bevor die Antwort abgesendet wird. Bei einem durchgeführten Test mit 100 Anfragen lagen die Antwortzeiten des originalen Geräts zwischen 150ms und 2400ms, diese Parameter sollten daher auch für den Honeypot verwendet werden. Dadurch sollte das Enttarnen über die Antwortzeit verhindert werden können.

4.2.4 HTTP-Server

Neben den zur Verfügung gestellten Daten verrät eine HTTP-Server auch einige Metainformationen, die von einem Angreifer ausgewertet werden können, etwa:

- HTTP-Header
- Reihenfolge der HTTP-Header
- Unterstützte HTTP-Versionen
- Inhalt der Fehlerseiten (401, 404, 500...)
- Verhalten bei unterschiedlicher Groß- und Kleinschreibung

Diese Eigenschaften sollten nach Entwicklung des HTTP-Honeypots mit dem Webserver des DDC4200 verglichen werden. `nmap` besitzt ein Modul zum Fingerprinting von HTTP-Servern, das einige der oben angegebenen Metadaten auswertet.

4.2.5 SSH-Server

Auch SSH-Server geben einige Details über die Konfiguration preis, ohne dass eine Anmeldung erforderlich ist:

- SSH-Version-String (beispielsweise SSH-2.0-OpenSSH_5.1p1 Debian-5)
- Unterstützte Algorithmen für Public-Key-Authentisierung (beispielsweise Ed25519, RSA)
- Unterstützte Transportverschlüsselungsalgorithmen (beispielsweise Chacha20, AES, ...)
- Unterstützte Hashalgorithmen (beispielsweise MD5, SHA1, ...)
- Unterstützte Kompressionsalgorithmen (beispielsweise zlib, lz4, ...)

- Typ und Länge der verwendeten Hostkeys
- Unterstützte Authentisierungsverfahren (beispielsweise PUBLICKEY oder KEYBOARD-INTERACTIVE)

Diese Details müssen so konfiguriert werden, dass sie der Konfiguration des SSH-Servers auf dem DDC4200 gleichen, um zu vermeiden, dass der Honeypot bereits vor dem Login enttarnt wird. Die dafür verwendete Software muss also geeignete Konfigurationsparameter besitzen, um eine passende Nachbildung erzeugen zu können.

Nach dem Login geht es vor allem darum, das Entdecken des Honeypots so lange wie möglich hinauszuzögern. Im Fall von SSH scheint es unmöglich, eine komplett authentische Umgebung nachzubilden. Das liegt vor allem daran, dass der Betreiber sich gegen schadhafte Aktionen des Angreifers absichern muss (Vgl. Abschnitt 3.3) und somit bestimmter Netzwerkverkehr blockiert wird.

[Wan+10] beschreibt ein Vorgehen, mit dem ein Honeypot auf diese Weise enttarnt werden kann. Dazu wird schädlicher Traffic, beispielsweise Malware oder eine DoS-Attacke an einen dritten Rechner, den sogenannten Sensor geleitet. Eine Information, ob der Angriff unverändert beim Sensor ankam wird an den Angreifer gesendet. Wenn der Angriff das Ziel erreicht hat, kann der Angreifer davon ausgehen, dass es sich zumindest nicht um einen verantwortungsvoll eingerichteten Honeypot handelt. Das Vorgehen wird in Abbildung 4.3 genauer skizziert.

Dieses Vorgehen hat für den Angreifer jedoch auch Nachteile. Zum einen kann es falschnegative Ergebnisse geben, wenn eine Firewall den Angriff verhindert hat, der Zielrechner sonst jedoch frei zu benutzen ist. Zum anderen wäre es auch möglich, dass bei einem in Netzwerk angeschlossenen Intrusion Detection System durch den Angriff ein Alarm ausgelöst wird, der den Angreifer unnötigerweise enttarnt.

Somit scheint es sinnvoll, auch andere Methoden zur Honeypoterkennung zu betrachten, die einem Angreifer mit Kommandozeilenzugriff zur Verfügung stehen. Auch wenn ein minimaler Funktionsumfang der Laufzeitumgebung bereits ausreichen würde, um automatisierte Angriffe zu täuschen [Kra04].

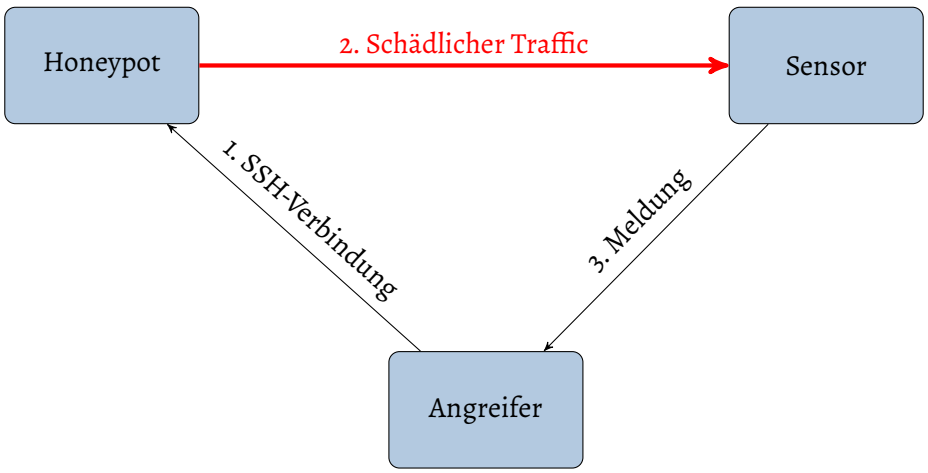

Abbildung 4.3: Methode zum Erkennen von Honeypots auf Basis eines Angriffs, nach [Wan+10]

Da manuelle Angriffe auch interessante Informationen liefern können, die im Rahmen dieser Arbeit untersucht werden sollten, sollte die Kommandozeile des Honeypots möglichst viele Möglichkeiten bieten, ohne Drittsysteme zu beschädigen. Im Rahmen der in Unterabschnitt 2.2.1 vorgestellten Hierarchie wäre somit ein Medium-Interaction Honeypot vorzuziehen.

Die dadurch zur Verfügung gestellten Programme bieten jedoch auch vielfältige Möglichkeiten, den Honeypot zu detektieren. In [HR05] werden unter anderem folgende Techniken vorgestellt:

1. Inhalt des Kernel Ring Buffers überprüfen (`dmesg`)
2. Prozessbaum mit `ps` überprüfen
3. In `ifconfig` nach virtualisierten Netzwerkadaptern Ausschau halten
4. CPU-Informationen in `/proc/cpuinfo` inspizieren
5. Hardwareinformationen etwa der Grafikkarte einsehen
6. Memory-Mapping verifizieren

[HR05] macht allerdings auch deutlich, dass eine perfekte Tarnung nicht unbedingt notwendig ist: „When deploying a honeypot, the goal is to capture lots of information about

the activity of the attacker. Even if he notices that he is on a honeypot, learning how he noticed it is supposed to be a valuable information, This means that honeypots need to be covert, but not too covert."

Insbesondere wenn eine bessere Tarnung Kompromisse beim Aufzeichnen der Aktionen erfordert, sollte der eigentliche Zweck des Honeypots bedacht werden.

4.2.6 Spezifisches Wissen

Durch den Einsatz von domänenspezifischem Wissen lassen sich - insbesondere im Bereich der Gebäudeautomation - sehr subtile Unstimmigkeiten finden, die auf das Vorhandensein eines Honeypots hindeuten können. Da der Honeypot ein Gebäude nicht in allen Einzelheiten simulieren kann, lassen sich solche Unzulänglichkeiten vom Angreifer ausnutzen.

Stellt das System etwa Messungen von Temperatur- oder Helligkeitssensoren zur Verfügung, können diese mit Wetterberichten am mutmaßlichen Standort des Gebäudes abgeglichen werden. Weniger subtil aber zumindest effektiv wäre eine Änderung der Zieltemperatur eines Raums im Gebäude. Die dadurch vollzogene Temperaturänderung könnte über das System beobachtet werden. Wenn eine entsprechende Änderung ausbleibt, handelt es sich entweder um einen Honeypot, oder um ein System das keine Änderungen zulässt. Beides wäre für einen Angreifer uninteressant.

Diese Möglichkeiten sind allerdings - zumindest im betrachteten System - erst nach einer erfolgreichen Anmeldung im System möglich. Wenn diese Hürde vom Angreifer überwunden wurde, wurden dabei schon wertvolle Informationen gesammelt. Auch die Information, wie der Angreifer das System nutzt, um den Honeypot zu erkennen, wäre von großem Interesse.

4.2.7 Online Honeypot-Tools

Über die Webpräsenz von Shodan wird auch ein „Honeyscore" genanntes Werkzeug bereitgestellt [Mat], von dem behauptet wird, dass es überprüfen kann, ob sich hinter einer IP-Adresse ein Honeypot oder ein reales Kontrollsystem verbirgt. Dafür werden die Informationen aus der Datenbank von Shodan verwendet. Auf der Basis von nicht näher bezeichneten Charakteristiken wird dann der namensgebende Honeyscore berechnet, der die Wahrscheinlichkeit angibt, dass es sich bei dem Gerät um einen Honeypot handelt.

Zwar gibt es keine weiteren Informationen, welche Daten genau als Berechnungsgrundlage verwendet werden, allerdings ist anzunehmen, dass hierfür solche Daten verwendet werden, die in den vorherigen Unterabschnitten thematisiert wurden. Eine Überprüfung durch dieses Werkzeug sollte nach der Installation des Honeypots dennoch vorgenommen werden.

4.3 Verfügbare Software

Honeypots sind ein beliebtes Werkzeug zum Erhöhen der Netzwerksicherheit. Aus diesem Grund gibt es eine Vielzahl fertiger Lösungen, die das Erstellen eines Honeypots vereinfachen. In [Ne] sind beispielsweise 271 verschiedene Programme zum Betrieb von Honeypots gelistet.

Da zwei verschiedene Dienste durch den Honeypot nachgebildet werden sollen, erscheint es sinnvoll, den Honeypot durch Nutzung zweier Komponenten aufzubauen:

4.3.1 SSH-Honeypot

Aufgrund der rechtlichen Rahmenbedingungen (Vgl. Abschnitt 3.3) und zum Verhindern einer zu leichten Entdeckung des Honeypots (Vgl. Abschnitt 4.2) wurde die Entscheidung getroffen, einen Medium-Interaction Honeypot zu verwenden. Im Folgenden werden daher Medium-Interaction Honeypots für das SSH-Protokoll vorgestellt, miteinander verglichen und auf ihre Eignung zum Einsatz für die Nachbildung des DDC4200 geprüft.

Honeypot	Entwickelt seit	Letzte Änderung	Codezeilen
MockSSH	06.01.2013	15.01.2017	1 657
Kippo	10.11.2009	30.09.2016	299 608
gohoney	12.12.2013	12.12.2013	562
Hornet	10.02.2014	30.04.2018	7909
Cowrie	10.11.2009	18.08.2018	318 277

Tabelle 4.1: Erster Überblick über die verfügbaren SSH-Honeypot-Lösungen [Cou13; Tam09; Pau13; Pan14; OT09], Stand: 22.08.2018

MockSSH

MockSSH [Cou13] ist ein in Python geschriebenes Framework zum Implementieren von Honeypots für das SSH-Protokoll. Standardmäßig werden nach erfolgreichem Login keine Kommandos zur Verfügung gestellt. Sämtliche für den Angreifer nutzbare Kommandos müssen daher vom Entwickler des Honeypots in Python oder einem LISP-Dialekt implementiert werden.

```
(mock-ssh  :users {"testuser" "1234"}
           :host "127.0.0.1"
           :port 2222
           :prompt "hostname>"
           :commands [
    (command :name "passwd"
             :type "prompt"
             :output "Password: "
             :required-input "1234"
             :on-success ["prompt" "hostname#"]
             :on-failure ["write" "Pass is 1234..."]))
```

Quellcodefragment 4.1: Beispielimplementierung eines Honeypots mit MockSSH in Hy-Lang, einem LISP-Dialekt [Cou13]

Das führt einerseits zu einer sehr starken Individualisierungsmöglichkeit, andererseits bedeutet dies auch einen beträchtlichen Konfigurationsaufwand, da selbst Standardkommandos, wie etwa `ls` selbst implementiert werden müssen. Das führt dazu, dass auch das Dateisystem des Honeypots manuell erfasst und in einer strukturierten Form

in einem Python- oder LISP-Modul geladen werden muss, um die grundlegendsten Kommandos zu implementieren.

Offenbar werden solche Kommandos in den meisten Honeypots benötigt, weshalb es wünschenswert wäre, wenn MockSSH diese bereits enthält. Positiv zu bewerten ist hingegen der Umstand, dass MockSSH den direkten Zugriff auf den zugrundeliegenden SSH-Server des Twisted-Projekts [Twi01] ermöglicht, was eine umfangreiche Konfiguration der benötigten Parameter, wie etwa des Version-Strings oder der unterstützten Verschlüsselungsalgorithmen ermöglicht. Da mit der Entwicklung von MockSSH erst nach Ende der Entwicklung des DDC4200 begonnen wurde, ist davon auszugehen, dass die dort unterstützen Algorithmen auch in MockSSH verwendet werden können.

Neben einer Protokollierung aller Anmeldeversuche zeichnet MockSSH auch die Inhalte aller Kommandozeilensitzungen auf, sodass jeder Tastendruck des Angreifers nachverfolgt und untersucht werden kann. Dafür wird das Logging-Framework von Kippo genutzt, das in Quellcodefragment 4.3.1 vorgestellt wird.

Prinzipiell ist MockSSH somit für den Einsatz als SSH-Honeypot geeignet, allerdings wird die starke Individualisierbarkeit mit einem hohen Konfigurationsaufwand erkauft. Hier wäre ein Kompromiss vorzuziehen, der eine stärkeren Fokus auf die Benutzbarkeit legt. Insgesamt entsteht der Eindruck, dass es sich eher um ein Proof-of-Concept handelt, als um einen verwendbaren Honeypot.

Kippo

Kippo [Tam09] ist ein Honeypot für das SSH-Protokoll der über einen vergleichsweise hohen Funktionsumfang verfügt, im Vergleich zu den anderen Lösungen besteht das Projekt daher auch aus vergleichsweise vielen Codezeilen.

Die meisten Funktionen können bequem über eine Konfigurationsdatei eingestellt werden, ohne dass Programmierkenntnisse erforderlich wären. Die damit konfigurierbaren Funktionen reichen von einem individuellen Hostnamen des simulierten Servers bis zum Verändern der vorgetäuschten Prozessorarchitektur.

Ein großer Vorteil von Kippo ist, dass für viele wichtige Kommandos, wie `ls`, `cd`, `cat` oder `ps` bereits täuschend echte Nachbildungen enthalten sind. Auch eigene Kommandos lassen sich integrieren, wenn das versäumt wurde sieht der Angreifer nur eine kryptische Fehlermeldung des Honeypots.

Um das Dateisystem des Zielrechners nachzubilden, stellt Kippo ein Programm namens `createfs` bereit, das die Informationen eines echten Dateisystems einliest und in ein Binärformat serialisiert. Die dadurch erstellte Datei enthält die wichtigsten Informationen des Dateisystems, die Inhalte der Dateien jedoch nicht. Falls der Angreifer den Inhalt ausführbarer Binärdateien anzeigen möchte, wird eine Dummy-Datei mit der eingestellten Prozessorarchitektur zurückgegeben. Die von Kippo bereitgestellten Scheinimplementierungen wichtiger Programme werden mit den im Dateisystem vorhandenen Binärdateien abgeglichen. Wenn etwa keine `ping`-Binärdatei in einem in der Umgebungsvariable `PATH` genannten Verzeichnis vorhanden ist, wird die `ping`-Funktionalität auch nicht bereitgestellt.

Möchte man Dateien (etwa Konfigurationsdateien) des realen Systems bereitstellen, müssen diese nur in ein bestimmtes Verzeichnis kopiert werden. Dadurch, dass die Metainformationen von den tatsächlichen Dateien getrennt sind, können auch schwer kopierbare Attribute wie etwa Inode-IDs oder Berechtigungen der Originaldateien nachgebildet werden.

Auch weitere Eigenschaften, wie erlaubte Kombinationen von Benutzername und Passwort, verwendete Hostkeys und die angezeigte Kernel-Version können problemlos konfiguriert werden.

Darüber hinaus scheint Kippo, jedenfalls den „Stern"-Markierungen auf Github und der vielen Suchergebnisse bei Google zufolge sehr beliebt zu sein. Das lässt vermuten, dass keine gravierenden Sicherheitsprobleme bestehen, die den Einsatz des Honeypots gefährlich machen. Als einziger Negativpunkt ist bei der Recherche aufgefallen, dass der Hauptentwickler sich 2015 aus dem Projekt zurückgezogen hat und seitdem keine größeren Änderungen mehr erfolgten. Zwar scheinen keine nötigen Funktionen zu fehlen, allerdings ist bei Problemen nicht auf die Unterstützung des Entwicklers zu hoffen.

Gohoney

Gohoney [Pau13] ist ein SSH-Honeypot mit sehr begrenztem Funktionsumfang. Zwar stellt er eine Kommandozeilensitzung zur Verfügung, sodass es nicht als Low-Interaction-Honeypot eingestuft werden kann, allerdings ist die Kommandozeilenumgebung sehr rudimentär. So wird beim Login zunächst eine Standard „Message of the day" eines Ubuntu 12.04 Servers ausgegeben. Sämtliche Kommandos werden daraufhin mit der Meldung `Could not read command` beantwortet.

Darüber hinaus werden vom Honeypot sämtliche Kombinationen aus Benutzername und Passwort sowie Benutzername und Private Key akzeptiert. Wichtige Konfigurationsparameter wie der SSH-Version-String lassen sich nur durch Modifikation des Quellcodes bearbeiten. Auch bei diesem Honeypot ist der Konfigurationsaufwand für die Nachbildung des DDC4200 sehr groß.

Hornet

Hornet [Pan14] ist ein aktiv entwickelter SSH-Honeypot. Wie Kippo stellt auch Hornet grundlegende Funktionen wie ein Dateisystem und einige Basiskommandos zur Verfügung. Der Umfang an zur Verfügung stehenden Kommandos ist zwar geringer als bei Kippo, das etwa auch Downloads mit `wget` erlaubt, dafür besitzt es eine interessante Implementierung des SSH-Kommandos:

In einer Konfigurationsdatei lassen sich mehrere sogenannte *Virtual Hosts* definieren. Diese verfügen dann jeweils über ein eigenes Dateisystem, erlaubte Benutzer und eine eigene IP-Adresse. Einer dieser *Virtual Hosts* wird als Standard definiert, welcher dann extern über das SSH-Protokoll erreichbar ist. Alle anderen *Virtual Hosts* lassen sich nur von einem der anderen *Virtual Hosts* erreichen, indem das SSH-Kommando auf gewohnte Weise verwendet wird. Dabei wird allerdings keine TCP-Verbindung hergestellt, sondern lediglich der Ausführungskontext geändert.

Dadurch lassen sich mit einem einzigen Honeypot und sehr geringen Ressourcenaufwand ganze Unternehmensnetzwerke mit mehreren Servern nachbilden. Dadurch lassen sich einem Angreifer möglicherweise mehr Informationen entlocken, als bei einem

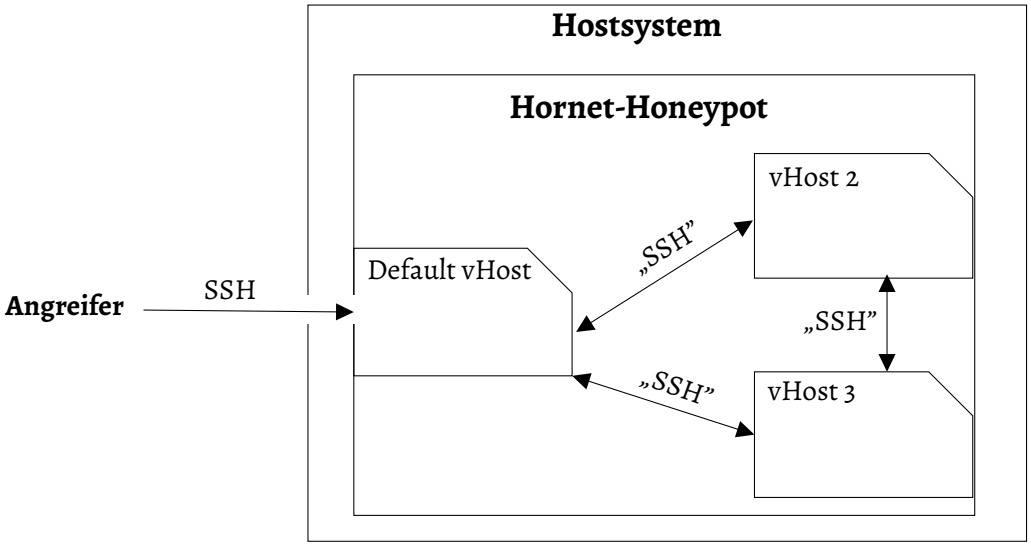

Abbildung 4.4: Konzept der Virtual Hosts in Hornet, nach [Pan14].

Honeypot, bei dem jede SSH-Verbindung ins Leere führt. Für den Einsatz als Nachbildung eines DDC4200 ist diese Funktion jedoch nicht zwingend notwendig. Würden andere Systeme wie etwa Datenbankserver simuliert werden, könnte das beispielsweise die Aufmerksamkeit des Angreifers von der Gebäudeautomation auf herkömmliche Angriffsziele lenken. Zwar könnte man andere DDCs im Netz simulieren, was in größeren Gebäuden durchaus üblich sein könnte, der Angreifer würde dort jedoch vermutlich keine Angriffe ausprobieren, die er nicht auch auf dem ersten Gerät ausprobieren könnte.

Durch den großen Funktionsumfang und die vergleichsweise einfache Konfiguration wäre Hornet für den Betrieb des SSH-Honeypots prinzipiell geeignet. Stark negativ fällt jedoch der Hinweis des Entwicklers am Ende der README-Datei auf: „Careful! Hornet is under development, and should not be used for production purposes yet. There are a fair amount of bugs, and perhaps security risks. Know what you're doing!" [Pan14] Ein öffentlicher Betrieb des Honeypots würde also Risiken mit sich bringen, andere Systeme zu gefährden.

Cowrie

Bei Cowrie [OT09] handelt es sich um eine Weiterentwicklung des Kippo-Honeypots, dessen Entwicklung 2015 eingestellt wurde. Da sämtlicher Code aus Kippo übernom-

men wurde, verfügt Cowrie auch über alle Funktionen, die Kippo bereitstellt. Der ursprüngliche Entwickler von Kippo empfiehlt inzwischen selbst die Nutzung von Cowrie für alle neuen Projekte [Tam09].

Neue Funktionen im Vergleich zum Funktionsumfang von Kippo umfassen unter anderem eine Unterstützung für Python 3, ein Docker Image zur Vereinfachten Bereitstellung und diverse neue Loggingschnittstellen wie etwa die Ausgabe von neuen Verbindungen in einem Slack Arbeitsplatz.

Wie Kippo ist selbstverständlich auch Cowrie zum Betrieb des DDC4200 SSH-Honeypots geeignet, da sämtliche Funktionen aus Kippo zur Verfügung stehen.

Fazit

Prinzipiell sind alle vorgestellten Lösungen geeigneten, um den SSH-Honeypot aufzubauen. Um einen möglichst geringen Konfigurationsaufwand bei gleichzeitig sehr realistischer Nachbildung zu erreichen, scheint Cowrie der geeignetstes Kandidat zu sein. Das liegt insbesondere an der leichten und trotzdem sehr flexiblen Konfiguration, den bereits enthaltenen hilfreichen Kommandos und der Stabilität der Software. Durch die weite Verbreitung werden offensichtliche Fehler schnell gefunden und vom Entwickler behoben.

Falls sich bei der späteren Entwicklung Gründe gegen die Nutzung von Cowrie ergeben, ist Hornet die beste Alternative. Die vom Entwickler ausgesprochene Nutzungswarnung stellt zwar neue Herausforderungen im Bezug auf die Sicherheit des Systems, diese ließen sich aber beispielsweise durch den Betrieb in einer mit einer Firewall gesicherten virtuellen Maschine lösen.

4.3.2 Web-Honeypot

Das Erstellen eines Honeypots für eine Webanwendung ist deutlich aufwendiger als das Erstellen eines Honeypots für einen SSH-Server. Während letzterer in den allermeisten Fällen eine Linux-Shell zur Verfügung stellt können für die Benutzerinteraktion mit einer Webseite verschiedene Technologien verwendet werden.

Besonders deutlich wird das am Beispiel der Vorgehensweise, die für die Anzeige der Nutzeroberfläche des DDC4200 genutzt wird. Statt gewöhnliche HTML-Elemente wie Formulare oder Schaltflächen zu verwenden, wird dem Nutzer ein vom DDC4200 generiertes Bild gezeigt. Auf diesem Bild sind serverseitig sämtliche Interaktionselemente gezeichnet worden. Animierte Elemente wie etwa blinkende Warnsymbole werden durch die Animationsfunktion des GIF-Bildformats umgesetzt. Klickt der Nutzer auf eine Schaltfläche, sendet der Browser die Koordinaten des Zielpunkts an den Server, der daraufhin eine neue Grafik generiert, die dann anstelle der alten angezeigt wird.

Um die Oberfläche glaubhaft nachzubauen, sollte die selbe Vorgehensweise verwendet werden. Dabei muss die Anwendungslogik in einem Maß nachgebaut werden, das dem Angreifer erlaubt, so mit dem System zu interagieren, dass Informationen über mögliche Attacken anfallen. Außerdem sollte das Aussehen der Oberfläche möglichst wenig vom Original abweichen. Folgende Anforderungen werden daher an die Implementierung der Oberfläche gestellt:

- Sämtliche erreichbaren Menüs sollen enthalten sein
- Die Loginfunktion mitsamt der Bildschirmtastatur soll Anmeldungen erlauben
- Interaktive Bildschirmelemente wie Textfelder oder Uhrzeitanzeigen sollen plausible Daten beinhalten

Diese Anforderungen sollten genügen um einerseits eine glaubhafte Kopie des Systems zu erstellen und andererseits ausreichende Interaktionsmöglichkeiten zu bieten, um Angriffe, wie etwa das Überwinden der Loginfunktion aufzuzeichnen. Auch kann so

nachvollzogen werden, welche Manipulationen der Angreifer am Gebäudeautiomationssystem durchführt, nachdem der Zugriffsschutz überwunden wurde. Diese Daten können wiederum Aussagen über das Ziel des Angreifers erlauben.

In [Ne] werden zwar einige Web-Honeypots gelistet, diese sind allerdings nur für bestimmte Applikationen geeignet. Da in diesem Fall sämtliche Interaktion ohnehin selbst implementiert werden muss, fehlt nur noch ein geeignetes Logging und ein HTTP-Server um den Honeypot zu betreiben. Für den Webserver wird `Flask` [Mön+10] mit `gunicorn` [Che09] im WSGI-Modus benutzt, da der Autor hiermit bereits Erfahrungen sammeln konnte.

Die Aufzeichnungen der Benutzerinteraktion sollten so erfolgen, dass auch der Bildschirminhalt zu jeder Interaktion nachvollzogen werden kann. Selbst wenn versteckte Funktionen im Honeypot nicht implementiert wurden, kann so nachvollzogen werden, worauf der Angreifer wann geklickt hat. Da das Bild ohnehin auf dem Server generiert wird, kann es dort auch gespeichert werden. Dabei kann die Stelle, auf die der Angreifer geklickt hat hervorgehoben werden. Neben dieser grafischen Darstellung die eine manuelle Auswertung des Angriffs erleichtert sollte die Interaktion jedoch auch in einem maschinenlesbaren Format festgehalten werden. Dadurch lassen sich große Datenmengen, durch die eine manuelle Auswertung unpraktisch wird automatisiert auswerten.

4.4 Auswertung der Angriffe

Nach Abschluss der Laufzeit der Honeypots müssen die protokollierten Angriffe ausgewertet werden. Dabei soll insbesondere überprüft werden, ob die in [HHT17] gefundenen Sicherheitslücken ausgenutzt wurden oder ob bisher unbekannte Sicherheitslücken ausgenutzt wurden. Wenn ein Zugriff auf den Honeypot erfolgt, soll zudem entschieden werden können, ob der Angreifer speziell Gebäudeautomationssysteme angreift, oder ob ganz allgemein Computersysteme ohne Betrachtung des Verwendungszwecks angegriffen werden. In diesem Abschnitt sollen daher Methoden entwickelt werden um diese Fragen systematisch zu beantworten.

4.4.1 Vorgehen bei der Auswertung

Um sicherzustellen, dass die Protokollaufzeichnungen auch die notwendigen Elemente enthalten wird im Voraus das grundlegende Vorgehen beim deren Auswertung festgelegt. Dabei werden zunächst grundlegende Parameter gesammelt und dann das spezielle Vorgehen für die einzelnen Honeypots festgelegt.

Da beide Honeypots eine Verbindung per TCP/IP ermöglichen, sind einige Verbindungsparameter bei beiden Honeypots verfügbar. Dazu gehört insbesondere die IP-Adresse des Angreifers. Hierbei kann mithilfe von Geolocation eine ungefähre Position des Angreifers festgestellt werden. Außerdem ist die IP-Adresse ein mögliches Kriterium zur Unterscheidung verschiedener Angreifer.

Zu Parametern, die unabhängig von der gewählten Verbindungsmethode anfallen gehört vor allem der Zeitpunkt des Angriffs. Dieser kann zur Einordnung der Herkunft des Angreifers hilfreich sein, wenn man davon ausgeht dass die Angriffe manuell erfolgen. Da die meisten Menschen nachts schlafen ist die Wahrscheinlichkeit hoch, dass der Angriff aus einer Region erfolgte, in der zum jeweiligen Zeitpunkt Tag war. Diese Information kann mit der Geolocation-Information der IP-Adresse verglichen werden, um die Plausibilität eines manuellen Angriffs einzuschätzen. Hierzu kann auch die Zeitspanne zwischen den einzelnen Aktionen ein wichtiges Kriterium sein. Wenn die Eingaben beziehungsweise Klicks in einer für Menschen ungewöhnlich hohen Geschwindigkeit erfolgen, ist die Wahrscheinlichkeit eines manuellen Angriffs hoch.

Die weiteren zu betrachtenden Eigenschaften sind spezifisch für den jeweiligen Honeypot. Für den SSH-Honeypot wäre das vor allem die Version des SSH-Clients und die zur Anmeldung angebotenen Typen von Schlüsselpaaren. Selbstverständlich sollte auch der eigentliche Interaktionsablauf aufgezeichnet werden. Da für den SSH-Honeypot (entsprechend der Einstellungen im DDC4200) nur Shellsitzungen und Kommandoaufrufe erlaubt wurden sind das vor allem die ausgeführten Befehle und die darauf gesendete Antwort. Beim Web-Honeypot wäre als weitere Metainformation der User-Agent des Angreifers von Interesse, da dieser Informationen über das vom Angreifer verwendete Werkzeug beziehungsweise den verwendeten Browser liefern könnte.

Die Auswertung selbst beinhaltet dann nur noch die Untersuchung der protokollierten Interaktionen des Angreifers mit dem Honeypot.

4.4.2 Absicht des Angreifers

Insbesondere beim SSH-Honeypot sind auch Angriffe zu erwarten, die gar nicht auf ein Gebäudeautomationssystem abzielen, sondern lediglich auf ein unsicheres Linuxsystem. So wird beispielsweise immer wieder davon berichtet, dass etwa mit der Mirai-Malware Botnetze aus verwundbaren Linux-Systemen aufgebaut werden [Den18]. Im Rahmen dieser Arbeit soll daher untersucht werden, wie sich solche Angriffe von Angriffen unterscheiden, die speziell auf das Gebäudeautomationssystem abzielen.

Zum Vergleich werden daher auch Angriffe gesammelt, die sich an Honeypots richten, die nicht als DDC getarnt sind. Diese Honeypots werden mit den Standardeinstellungen von Cowrie installiert und nur mit einer leicht zu erratenden Benutzername/Passwort-Kombination abgesichert.

Die dadurch gesammelten Daten lassen sich dann auf folgende Weise mit den vom DDC-Honeypot gesammelten Daten vergleichen:

- Welche Quelladressen haben die Angriffe durchgeführt?
- Wie häufig wurden die jeweiligen Honeypots angegriffen?
- Welche Kommandos wurden nach erfolgreichem Login ausgeführt?

Falls sich in den Daten entsprechend unterschiedliche Muster erkennen lassen, ließe sich daraus schließen, dass es auf Gebäudeautomationssysteme spezialisierte Angreifer gibt. Selbstverständlich würde ein Ausbleiben solcher Unterschiede andererseits nicht bedeuten, dass solche Angreifer nicht existieren.

Für den Web-Honeypot wäre ein solches Vorgehen nicht praktikabel, da ein Angriff auf die Weboberfläche ohnehin spezielles Wissen voraussetzt. Ein automatisierter Angriff müsste etwa die Bilder der Oberfläche laden, den Inhalt erkennen und Klick-Events an bestimmte Interaktionselemente senden. Falls Aktionen in der Weboberfläche des Honeypots ausgeführt werden, wird also entweder ein spezialisiertes Tool verwendet, das speziell für Angriffe auf den DDC4200 ausgelegt ist oder der Angriff erfolgt manuell.

Abbildung 4.5: In der Oberfläche des DDC4200 ist leicht zu erkennen, dass sie zum Steuern von Anlagen gedacht ist.

Falls ein manueller Angriff durchgeführt wird, ist der Angreifer spätestens nach Ansicht des in Abbildung 4.5 dargestellten Startbildschirms der Oberfläche des DDC4200 darüber informiert, dass die Anwendung zum Steuern von Anlagen gedacht ist. Insofern kann man davon ausgehen, dass der Angriff zielgerichtet ist, wenn er danach fortgeführt wird.

Somit liegt also im Fall der Weboberfläche, unabhängig davon, ob man erkennen kann, ob ein Angriff von einem automatisierten Werkzeug durchgeführt wurde in jedem Fall ein zielgerichteter Angriff vor.

Kapitel 5

Prototyp

In diesem Kapitel

5.1 SSH-Honeypot

Das Einrichten des Cowrie-Honeypots erwies sich wie erwartet als unkompliziert. Dabei ergab sich lediglich das Problem, dass sich die Nutzung von DSA-Schlüsseln zur Hostauthentifizierung nicht über die Konfigurationsdatei deaktivieren ließ. Da der DDC4200 selbst keine DSA-Hostkeys verwendet, sollte ein solcher jedoch nicht verwendet werden. Hierfür wurde die folgende Änderung am Quellcode der Honeypot-Software vorgenommen:

```
index 0e3693e..b9652f4 100644
--- a/cowrie/ssh/factory.py
+++ b/cowrie/ssh/factory.py
@@ -60,10 +60,12 @@ class CowrieSSHFactory(factory.SSHFactory):
         dsaPubKeyString, dsaPrivKeyString = cowriekeys.getDSAKeys()
         self.publicKeys = {
           b'ssh-rsa': keys.Key.fromString(data=rsaPubKeyString),
-          b'ssh-dss': keys.Key.fromString(data=dsaPubKeyString)}
+        }
         self.privateKeys = {
           b'ssh-rsa': keys.Key.fromString(data=rsaPrivKeyString),
-          b'ssh-dss': keys.Key.fromString(data=dsaPrivKeyString)}
+        }
--
```

Alle anderen notwendigen Einstellungen ließen sich in der Konfigurationsdatei `cowrie.cfg` vornehmen. Im Vergleich zur Standardkonfiguration wurden dabei folgende Werte geändert:

```
[honeypot]
sensor_name = honeypot
hostname = ddc4200
fake_addr = 192.168.23.2

[shell]
# Prozessorarchitektur des DDC4200:
arch = linux-powerpc-lsb

[ssh]
enabled = true
# Original Host-Keys vom DDC:
rsa_public_key = etc/ssh_host_rsa_key.pub
rsa_private_key = etc/ssh_host_rsa_key
version = SSH-2.0-dropbear_0.53.
```

```
# Diese Features werden vom SSH-Server auf dem Honeypot nicht unterstützt:
sftp_enabled = false
forwarding = false
forward_redirect = false
forward_tunnel = false
```

Da per SSH Root-Zugriff auf den DDC4200 bestand, konnte das Dateisystem mit dd in eine Image-Datei geschrieben werden. Diese wurde dann eingehängt und mit dem in Cowrie enthaltenen Werkzeug `createfs` eingelesen. Das erzeugt eine Datenstruktur im pythontypischen Pickle-Format, welche vom Honeypot beim Start eingelesen wird. Darin sind alle Daten aus der inode-Datenstruktur [KW10] enthalten, wie etwa Eigentümerinformationen, Zugriffsrechte und Attribute.

5.2 Web-Honeypot

Um die Weboberfläche des DDC4200 nachzubilden wurde zunächst untersucht, welche Menüansichten darzustellen sind und welche interaktiven Elemente diese enthalten. Dabei wurden die folgenden Ansichten ermittelt:

Startbildschirm

Verweise:

- Konfiguration
- Anmeldung
- Beenden

dynamische Elemente:

- Datum
- Uhrzeit

Passwort ändern

Verweise:

- Konfiguration

dynamische Elemente:

- Textfelder Passworteingabe

Anmeldung

Verweise:

- Startbildschirm
- Beenden

dynamische Elemente:

- Nutzerauswahl

Anmeldung Administratorkonto

Verweise:

- Startbildschirm
- Beenden

dynamische Elemente:

- Nutzerauswahl

Konfigurationsebene

Verweise:

- Passwort ändern
- Sicherheitseinstellungen
- Systeminformationen
- Systemzeit
- Datensicherung
- Service-Ebene Sprachauswahl
- Beenden

Datensicherung

Verweise:

- Konfigurationsebene

Anmeldeaufforderung

Verweise:

- Anmeldung
- Startbildschirm

Passworteingabe

dynamische Elemente:

- Tastatur mit Textfeld

Fehlermeldung Anmeldung

Verweise:

- Startbildschirm
- Passworteingabe

Sicherheitseinstellungen

Verweise:

- Konfigurationsebene

dynamische Elemente:

- Radio-Buttons für Serveroptionen
- Checkbox für BACnet-Zugriff von außerhalb

Startbildschirm Service-Ebene

Verweise:

- Konfigurationsebene
- Service-Ebene Objektgruppeninstallation

dynamische Elemente:

- Angeschlossene Zentralen

Service-Ebene Objektgruppeninstallation

Verweise:

- Startbildschirm Service-Ebene

dynamische Elemente:

- Auswahl zwischen BACnet, Feldbus und Schaltschrankbus

Service-Ebene Sprachauswahl

Verweise:

- Konfigurationsebene

dynamische Elemente:

- Auswahl zwischen Deutsch und Englisch

Systeminformationen

Verweise:

- Konfigurationsebene

Systemzeit

Verweise:

- Konfigurationsebene

dynamische Elemente:

- Datum und Uhrzeit
- Dropdownmenü für Zeitzone

Daraufhin wurde `mitmproxy` [Cor+10] genutzt, um für jede Ansicht die Bilder der grafischen Oberfläche, die zwischen dem DDC4200 und dem Browser ausgetauscht werden, zu speichern. Für jedes gespeicherte Bild wurden dann die Pixel-Koordinaten der Verweise und der dynamischen Inhaltselemente in einer YAML-Datei festgehalten. Eine beispielhafte Datei wird in Quellcodefragment 5.1 dargestellt.

Zum Einlesen dieser Daten wurde dann wiederum eine Python-Anwendung entwickelt, die die Navigation übernimmt und die gespeicherten Bilder mit den dynamischen Inhalten überschreibt. Für den Startbildschirm wird dies beispielhalft in Abbildung 5.1 dargestellt.

Für die Grafikoperationen auf den Pixelgrafiken wird die Python-Bibliothek Pillow [LC10] verwendet. Damit lassen sich beispielsweise Texte auf den Bildern überschreiben. Quellcodefragment 5.2 zeigt den dafür notwendigen Code.

Abbildung 5.1: Dynamische Inhalte wie Uhrzeiten werden zur Laufzeit in das Bild eingefügt

Das dadurch entwickelte Modul zur Simulation der Oberfläche wurde in einen Webserver integriert. Als Webframework in Python wurde Flask [Mön+10] gewählt. Dort wurden die statischen Dateien der DDC4200-Weboberfläche (HTML-Seiten, Javascript-Code

und Bilder) integriert und die dynamischen Aufrufe (`do_click.gif` und `ddc_main.gif`) mit dem Python-Modul verknüpft, das die Oberfläche nachbildet.

Damit war die Oberfläche zwar bereits lauffähig aber durch unterschiedliche HTTP-Header und eine merklich schnellere Reaktion vom Original zu unterscheiden. Die HTTP-Header können innerhalb von Flask gesetzt werden. Um die Ausführungsgeschwindigkeit auf langsameres Niveau zu senken, wurde die Ausführung vor dem Ausliefern einer Antwort jeweils um eine zufällige Dauer zwischen 150ms und 2400ms pausiert.

Um den Honeypot zu komplettieren fehlte damit nur noch ein Modul, um die Interaktionen mit dem System mitzuschneiden. Dafür wurden zwei Protokollformate gewählt, von denen eins menschenlesbar und das andere maschinenlesbar ist, um eine einfache automatische und manuelle Auswertung zu gewährleisten.

Für das menschenlesbare Protokoll wird dabei eine HTML-Datei erstellt, das anfangs die anonymisierte IP-Adresse des Angreifers und das Datum und die Uhrzeit der Attacke aufzeichnet. Anschließend wird für jede Interaktion das jeweils angezeigte Bild der Oberfläche dargestellt, wobei die Klickposition mit einem Symbol markiert ist. Das maschinenlesbare Protokoll wird im CSV-Format gespeichert, wobei jede Zeile folgende Informationen enthält:

1. Anonymisierte IP-Adresse des Angreifers
2. Zeitstempel der Interaktion
3. Koordinaten des Klick-Events
4. angezeigte Ansicht
5. resultierende Ansicht

Der Quellcode des Nachbaus der grafischen Oberfläche kann in [Bau18] eingesehen werden.

5.3 Evaluation

In diesem Abschnitt wird untersucht, ob der Honeypot für Angreifer auch als Honeypot erkennbar ist, oder ob er die Dienste des DDC4200 überzeugend genug nachbildet, um nicht erkannt zu werden. Dafür wird zunächst `nmap` verwendet, da es als eines der wichtigsten Werkzeuge in der Netzwerksicherheit vermutlich auch von Angreifern verwendet wird, um mehr Informationen über Ziele herauszufinden. Anschließend wird der Honeypot durch das „Honeyscore"-Werkzeug von Shodan überprüft, das angibt wie wahrscheinlich es ist, dass es sich beim Zielrechner um einen Honeypot handelt.

5.3.1 nmap

Um zu prüfen, wie ähnlich der Honeypot dem echten Gerät auf Netzwerkebene ist, wurde in `nmap` ein Scan des Honeypotsund des DDC4200 gestartet. Dabei wurden die folgenden Parameter verwendet:

- `-T4` parallelisiert die Portscans mit geringen Wartezeiten zwischen den einzelnen Scans.
- `-A` aktiviert „aggressive" Scan-Optionen wie etwa Traceroutes und die Betriebssytemerkennung.
- `-v` lässt mehr Meldungen in der Ausgabe erscheinen.

Die Ausgaben der beiden Scans wurden dann jeweils in eine Datei geleitet und mit `diff` miteinander verglichen. Im Versuchsaufbau wurden dabei die folgenden Unterschiede erkannt:

1. Die MAC-Adresse unterschied sich.
2. Die Ergebnisse der Betriebssystemerkennung (Kernel-Version, geschätzte Uptime) unterschieden sich.

Unterschied **1** ist für die Honeypoterkennung nicht relevant, da die MAC-Adresse des Zielgeräts über das Internet üblicherweise nicht ermittelt werden kann. Der zweite Unterschied kann durch den Einsatz von NAT oder von Diensten wie `honeyd` eliminiert werden.

5.3.2 Honeyscore

Das Honeyscore-Tool von Shodan ist deutlich einfacher zu bedienen als nmap. Als Eingabe wird lediglich die IP-Adresse des zu überprüfenden Geräts gefordert. Leider ist auch die Ausgabe ähnlich limitiert, nach einer kurzen Wartezeit erscheint lediglich der Hinweis „Looks like a real system!" (Vgl. Abbildung 5.2).

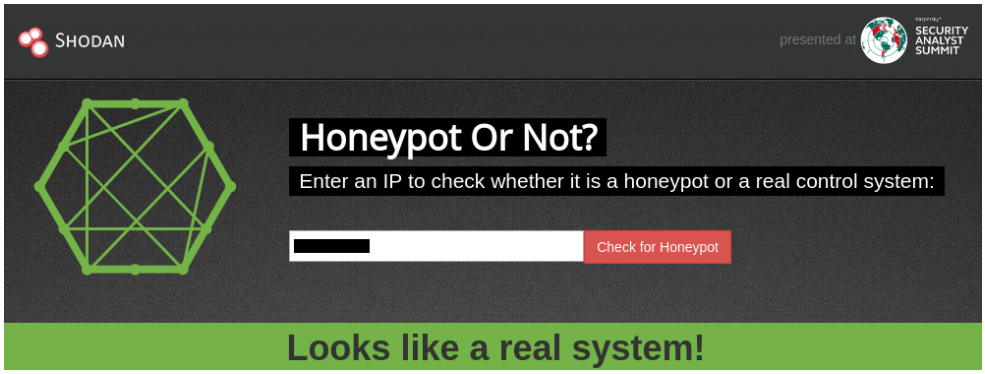

Abbildung 5.2: Ausgabe des Honeyscore-Werkzeug [Mat] zum Honeypot

Auf der Webseite wird angegeben, dass das Tool eine „Honeyscore" genannte Wahrscheinlichkeit ermittelt, nach der es sich bei dem Gerät um einen Honeypot handelt. Dieser taucht in der Ausgabe jedoch nicht auf, lediglich in der Entwicklerkonsole des Browsers wird die Wahrscheinlichkeit mit 0,0 angegeben. Offenbar ist das Werkzeug somit sehr überzeugt davon, dass es sich beim Honeypot um eine tatsächliches Kontrollsystem handelt. Das bedeutet entweder, dass der Honeypot besonders gut verschleiert wurde, oder dass das Werkzeug nicht präzise genug arbeitet.

5.4 Bereitstellung

Da ein Teil der Honeypots an Internetanschlüssen von Privatpersonen betrieben werden soll, muss ein Gerät bereitgestellt werden, das die Honeypot-Dienste im Netzwerk anbietet. Aus Kosten- und Energieeffizienzgründen wurde dazu ein Raspberry Pi 3 B+ verwendet.

Auf insgesamt 5 dieser Geräte wurden die Dienste installiert und im privaten Netz bereitgestellt. Nachdem eine entsprechende Portweiterleitung im Router konfiguriert wurde, waren die Dienste daraufhin aus dem Internet erreichbar.

Um die Logs leichter auswerten zu können und um einfacher Änderungen am System durchzuführen wurden die Geräte über ein VPN-Gateway miteinander verbunden. Dadurch konnten vom Gateway aus Konfigurationsänderungen über SSH vorgenommen werden. Außerdem wurde ein Dienst eingerichtet, der die Logs der Geräte mittels `rsync` in regelmäßigen Zeitabständen sichert und bei neuen Logdateien eine Benachrichtigung versendet.

Die über Planetlab [UPM+] bereitgestellten Server wurden so konfiguriert, dass der tatsächliche SSH-Port nur von dem System aus erreichbar ist, auf dem die Logs gesammelt werden. Dadurch wird verhindert, dass die Systeme durch 2 offene SSH-Ports, von denen einer zum Honeypot gehört unnötige Aufmerksamkeit erregen.

```
overwrite:
  - with: "{date} - {time}"
    x: 215
    y: 17
    background: "#cccccc"
    foreground: "#000000"
    size: 8
    bold: No
links:
  - to: configuration
    upper_left:
        x: 0
        y: 210
    lower_right:
        x: 34
        y: 240
  - to: login
    params:
      after_login: overview
    upper_left:
        x: 34
        y: 210
    lower_right:
        x: 68
        y: 240
  - to: exit
    upper_left:
        x: 68
        y: 240
    lower_right:
        x: 102
        y: 240
```

Quellcodefragment 5.1: YAML-Objekt das zusammen mit dem zugehörigen Bild die Startseite der Benutzeroberfläche definiert

```
for o in page["overwrite"]:
    fnt = "./p_arial" + ("-bold" if o["bold"] else "") + ".ttf
    fnt = ImageFont.truetype(fnt, o["size"])
    text = o["with"].format(**self.state)
    w, h = fnt.getsize(text)
    d = ImageDraw.Draw(image)
    d.rectangle([(o["x"], o["y"]), (o["x"] + w, o["y"] + h)],
                fill=o["background"])
    d.text((o["x"], o["y"]), text, font=fnt, fill=o["foreground"])
```

Quellcodefragment 5.2: Beispiel der Nutzung von Pillow zum Ersetzen von Text auf einem Bild

Kapitel 6

Angriffe auf den Honeypot

Die Auswertung der Angriffe soll nach Möglichkeit insbesondere die folgenden Fragen beantworten:

1. Welche Angriffe wurden manuell ausgeführt?
2. Welche Angriffe zielten speziell auf Gebäudeautomationssysteme ab?
3. Waren Angriffe in der Lage, die Natur des Honeypots zu offenbaren?

In Unterabschnitt 4.4.1 wurden die Daten festgelegt, deren Sammlung in den Protokolldateien des Honeypots sinnvoll erscheint. Auf Basis der dort festgelegten Kriterien wurden zwei Softwarekomponenten entwickelt, die die Diense des DDC4200 als Honeypot nachbilden (Vgl. Kapitel 5). Die fertigen Honeypots wurden daraufhin vom 09. Juli. bis zum 14. September, also insgesamt 10 Wochen lang im Internet bereitgestellt. Die Honeypots waren nach jeweils maximal 3 Tagen im Index von Shodan gelistet.

Im Folgenden werden die jeweils aufgezeichneten Protokolldaten ausgewertet und untersucht.

6.1 Angriffe auf den SSH-Honeypot

Beim Betrachten der angesammelten Protokolle des SSH-Honeypots fällt zunächst auf, dass lediglich eine Aufzeichnung einer Terminalsitzung vorhanden ist. Diese wurde vom Autor selbst zum Test des Honeypots durchgeführt. Eine weitere Untersuchung der Protokolldateien zeigt, dass zwar insgesamt 62.611 Versuche getätigt wurden, einen Zugang zum System zu erlangen, jedoch keiner davon auch nur den richtigen Benutzernamen verwendet hat. Offenbar ist die Kombination aus dem Benutzernamen `kesroot` mit dem gleichlautenden Passwort sicherer als erwartet, zumindest scheint sie in den einschlägigen Passwortlisten nicht enthalten zu sein.

Da die Protokolle des Honeypots im JSON-Format vorliegen, ist eine Weiterverarbeitung der Daten zur Auswertung problemlos möglich. Ein Skript das die verwendeten Benutzernamen und Passwörter zählt fördert die in Tabelle 6.1 dargestellte Liste der jeweils am häufigsten verwendeten Daten zutage. Diese erweckt den Eindruck, dass mit einem Wörterbuch aus häufig verwendeten Logindaten gearbeitet wurde, da die Daten jeweils für einen Linux-Server plausibel erscheinen.

Nutzername	**Versuche**
'root'	14607
'admin'	12385
'user'	2729
'ubnt'	2153
'test'	1581
'ftp'	1534
'oracle'	1515
'mysql'	1066
'support'	1063
'pi'	808

Passwort	**Versuche**
''	2350
'admin'	1694
'password'	1338
'132456'	1260
'12345'	1059
'1234'	1008
'root'	808
'ubnt'	695
'test'	683
'qwerty'	543

Tabelle 6.1: Top 10 der Login-Versuche auf dem SSH-Honeypot

Insgesamt am häufigsten wurde der Nutzername `admin` mit gleichlautendem Passwort zur Anmeldung versucht. Interessant ist hierbei, dass die selben Logindaten 674 Mal erfolglos versucht wurden. Der kleinste Abstand zwischen zwei Loginversuchen mit dieser Kombination beträgt nur etwa 17 Minuten. Offensichtlich ist es nicht sehr wahrscheinlich, dass innerhalb dieser Zeitspanne das entsprechende Passwort gesetzt wurde. Dies

deutet darauf hin, dass die Loginversuche nicht zentral koordiniert wurden, möglicherweise gehen die Angriffe daher von verschiedenen Akteuren aus.

Eine Darstellung der zeitlichen Verteilung der Angriffe in Abbildung 6.1 zeigt, dass die Angriffe über den ganzen Tag verteilt durchgeführt wurden. Zusammen mit der großen Zahl an Loginversuchen legt dies die Vermutung nahe, dass der Angriff durch ein Programm automatisiert durchgeführt wird. Auf den ersten Blick fällt eine Häufung der Loginversuche in der Zeit von ca. 18:00-6:00 UTC (also 20:00-8:00 MESZ) auf. Allerdings ist der relative Unterschied zwischen den beiden Zeiträumen gering (jeweils 40% und 60% der Angriffe), sodass diese Eigenschaft möglicherweise nicht signifikant ist. Als Erklärung wäre jedoch denkbar, dass es Malware gibt, die nur Server angreift, an deren Position gerade Nacht ist um zu verhindern, dass der Angriff durch aufmerksame Systemadministratoren sofort bemerkt wird.

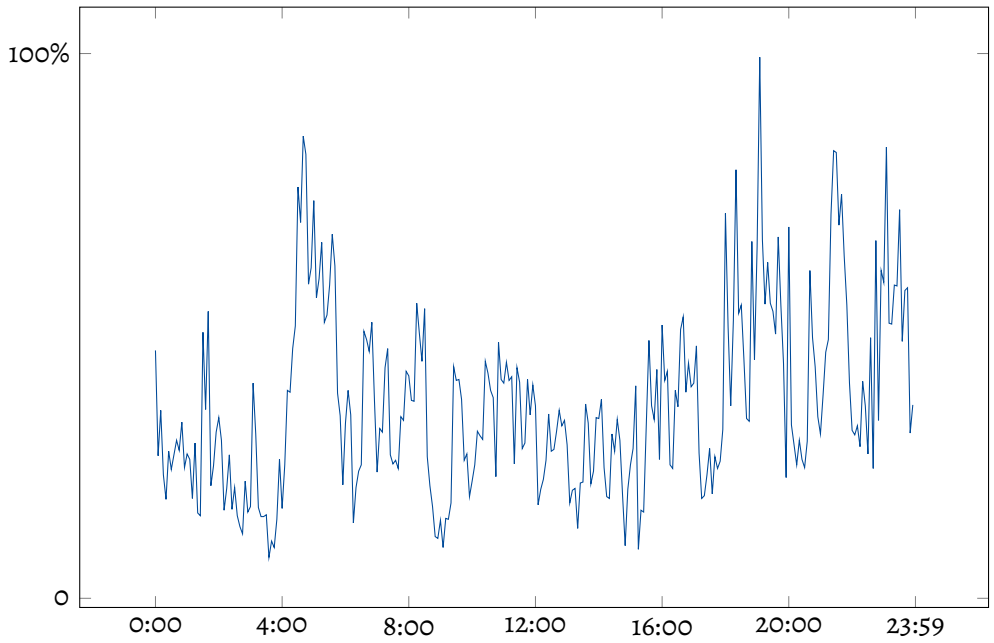

Abbildung 6.1: Verteilung der Loginversuche nach Tageszeiten über den gesamten Messzeitraum (Zeit in UTC)

Insgesamt gingen Loginversuche von 2506 verschiedenen IP-Adressen aus, von denen allerdings 1702 IP-Adressen lediglich einen einzigen Loginversuch durchführten. Die in den Protokolldaten am häufigsten auftauchende IP-Adresse ist mit insgesamt 12.394 Loginversuchen für etwa 20% aller Loginversuche verantwortlich. Da die IP-Adressen

anonymisiert wurden, indem das letzte Oktett mit einer 1 ersetzt wurde, ist es möglich dass hier mehrere IP-Adressen zusammengefasst wurden, dennoch ist der zugehörige Netzbereich offenbar an vielen SSH-Angriffen beteiligt. Dieser Netzbereich wurde einem in Russland ansässigen Hostingunternehmen zugewiesen. In Abbildung 6.2 wird die Herkunftsverteilung der Angriffe als Heatmap dargestellt.

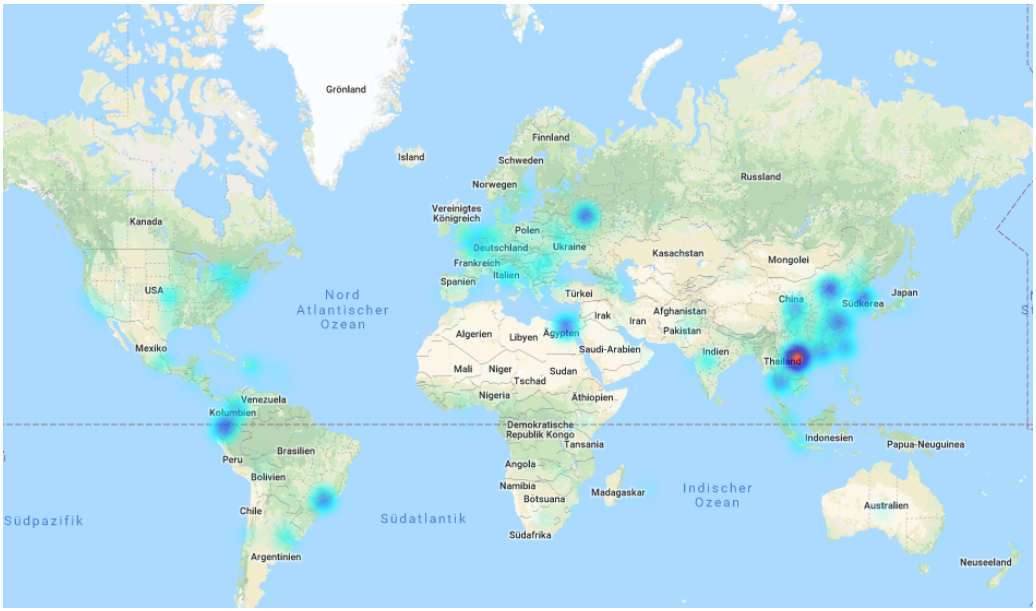

Abbildung 6.2: Heatmap der Geolocation-Informationen der IP-Adressen von denen Loginversuche ausgingen. Kartendaten © 2018 Google, INEGI. Geolocation-Daten von MaxMind, Inc.

Da das für den DDC4200 vorkonfigurierte Passwort `kesroot` nicht verwendet wurde, liegt der Schluss nahe, dass kein speziell für den DDC4200 entwickeltes Tool versucht hat, den Honeypot anzugreifen. Da der Versuchszeitraum mit 10 Wochen jedoch relativ kurz war, kann keine Aussage über die Nichtexistenz eines solchen Werkzeugs getroffen werden. Möglicherweise hat auch eine vom Autor selbst nicht gefundene Schwachstelle in der Tarnung des Honeypots dazu geführt, dass solche Werkzeuge den Honeypot ignoriert haben.

6.2 Angriffe auf den Web-Honeypot

Während der Laufzeit des Experiments ist lediglich ein einziger Zugriff auf den Web-Honeypot erfolgt. Der zugehörige IP-Adressbereich gehört zu einem ungarischen Internetanbieter, als User-Agent wird `Mozilla/5.0 (Windows NT 10.0; Win64; x64;`

`rv:61.0) Gecko/20100101 Firefox/61.0` angegeben, möglicherweise erfolgte der Zugriff von einem Windows-Rechner mit einem Firefox-Browser. Insgesamt war die Oberfläche des Honeypots 11 Sekunden geöffnet, es wurde keine Interaktion mit der Benutzeroberfläche durchgeführt.

Es wurden weitere Anfragen an den Webserver ausgeführt, diese betrafen jedoch ausnahmslos statische Dateien, sodass vom Python-Modul, das die Oberfläche des DDC4200 nachbildet keine Logdatei angelegt wurde. Diese Anfragen wurden somit nicht von Webbrowsern, sondern von HTTP-Clients ohne JavaScript-Unterstützung durchgeführt. Auch dieser Honeypot liefert somit kein Indiz dafür, dass auf die DDC4200 spezialisierte Angriffe durchgeführt werden.

Kapitel 7

Fazit

In diesem Kapitel werden die Ergebnisse der Arbeit zusammengefasst und mögliche Fehlerquellen aufgezeigt. Darüber hinaus werden offene Forschungsfragen, die sich bei der Bearbeitung ergeben haben dokumentiert.

7.1 Zusammenfassung

In der Arbeit wurden die Grundlagen zu Gebäudeautomationssystemen und dort insbesondere die Rolle von DDCs erläutert. Als zentrale Steuerungskomponente in Gebäudeautomationssystemen bieten im Internet verfügbare DDCs ideale Angriffsziele für Individuen mit schadhaften Absichten. Da die Sicherheit solcher Systeme nicht in jedem Fall gegeben ist, sollte untersucht werden, inwiefern bekannte Sicherheitslücken ausgenutzt werden. Dazu wurde das Konzept von Honeypots erklärt und Möglichkeiten zur Kategorisierung von Honeypots dargelegt.

Im Weiteren wurden rechtliche Fragen aufgeworfen, die im Vorfeld des Experiments beantwortet werden mussten. Dazu wurden dann die einschlägigen Gesetzestexte analysiert und der Versuch unternommen, rechtliche Zusammenhänge zu ermitteln. In diesem Zusammenhang ergaben sich einerseits datenschutzrechtliche Probleme, die sich

lösen ließen, indem der Honeypot auf das Speichern personenbezogener Daten verzichtet. Andererseits wurden jedoch auch urheberrechtliche Misshelligkeiten ermittelt, die weitere Vorkehrungen erforderten.

Auf dieser Basis wurde dann ein Konzept zum Entwickeln eines geeigneten Honeypots erarbeitet. Dabei war zunächst zu klären, welche Maßnahmen getroffen werden müssen, um mögliche Angreifer auf den Honeypot aufmerksam zu machen. Hierbei wurde festgestellt, dass keine solche Maßnahmen erforderlich waren, da sich der Honeypot in dieser Hinsicht wie ein wahrhaftiges Gerät verhält. Um sicherzustellen, dass der Honeypot auch in anderer Hinsicht ein Verhalten vorweist, das sich von dem des echten DDC4200 nicht unterscheidet wurden daraufhin Möglichkeiten gesucht um Honeypots aus Angreiferperspektive zu erkennen und dafür geeignete Abwehrmaßnahmen ermittelt. Dafür wurden Voraussetzungen an einen Honeypot gestellt, um eine Detektion zu verhindern.

Um diese Anforderungen für den SSH-Honeypot zu erfüllen, wurden mögliche Softwarelösungen aufgespürt, miteinander verglichen und auf die Eignung für den Einsatz als Honeypot für den DDC4200 geprüft. Dabei stellte sich die Honeypotlösung Cowrie [OT09] als beste Variante heraus, auch wenn andere Bibliotheken prinzipiell ebenfalls geeignet waren. Aufgrund der vielfältigen Interaktionsmöglichkeiten mit einer Webseite gab es keinen Web-Honeypot der ohne Weiteres für die Nachbildung der grafischen Oberfläche des DDC4200 geeignet war. Das bedingte die Entwicklung einer eigenen Lösung, die zunächst konzeptionell skizziert wurde. Auch die für die Auswertung relevanten Daten wurden ermittelt und zusammengestellt.

Daraufhin wurde die Honeypotlösung mit den beiden Teilkomponenten eines Web-Honeypots und eines SSH-Honeypots auf Basis des zuvor festgelegten Konzepts entwickelt. Der fertige Prototyp wurde mit dem Netzwerkscanwerkzeug nmap [Gor] und dem in Shodan integrierten Honeyscore-Werkzeug [Mat] evaluiert. Dabei wurden mit beiden Werkzeugen keine signifikanten Unterschiede zum echten DDC4200 festgestellt, sodass damit keine Erkennung des Honeypots möglich ist. Die entwickelte Lösung wurde dann auf Kleinstcomputern vom Typ Raspberry Pi B sowie auf virtuellen Maschinen von PlanetLab [UPM+] installiert und im Internet verfügbar gemacht.

Nach einer Laufzeit des Experiments von 10 Wochen wurden die auf den Honeypots angefallenen Protokolldaten ausgewertet. Für den SSH-Honeypot wurde dabei festgestellt,

dass lediglich automatisierte Angriffe erfolgten, die nicht imstande waren das Passwort des DDC4200 zu erraten. Diese Angriffe wurden daraufhin genauer untersucht und ausgewertet. Der Web-Honeypot konnte gar keine Angriffe verzeichnen. Lediglich ein einziger Zugriff wurde protokolliert, bei diesem wurde jedoch keine Interaktion mit dem System vorgenommen, sodass es sich nicht um einen Angriff handeln kann.

7.2 Mögliche Fehler

Da der Honeypot keine für den DDC4200 geeigneten Angriffe verzeichnen konnte, liegt der Schluss nahe, dass aktuell keine entsprechenden Angriffe ausgeführt werden. Im Folgenden werden jedoch andere mögliche Ursachen untersucht, die für eine Erklärung des Ergebnisses geeignet sind.

7.2.1 Unzureichende Tarnung

Falls der Honeypot auch ohne Zugriff auf eine SSH-Shell und die grafische Oberfläche des DDC4200 als Honeypot zu erkennen war, würden Angreifer eine protokollierte Interaktion mit dem System, die Informationen über deren Vorgehen verraten könnte vermeiden. Das könnte etwa durch das Aufrufen einer im Honeypot nicht vorhandenen Webressource geschehen. Eine Prüfung der im Zugriffsprotokoll verzeichneten Anfragen die mit einem Fehlercode 404 beantwortet wurden zeigte allerdings, dass der Webserver des DDC4200 diese Anfragen ebenfalls mit demselben Fehler beantwortet. Möglich wäre weiterhin, dass der auf dem DDC4200 eingesetzte SSH-Server bestimmte Anfragen auf eine Weise beantwortet, die ihn von anderen SSH-Servern unterscheidet. Aufgrund der zahllosen Möglichkeiten, SSH-Anfragen zu stellen wurde diese Möglichkeit jedoch nicht weiter untersucht.

7.2.2 Zu geringe Laufzeit

Der Honeypot war insgesamt 10 Wochen im Internet verfügbar. Möglicherweise wurden lediglich in diesem Zeitraum keine entsprechenden Angriffe durchgeführt.

7.2.3 Uninteressante IP-Adressen

Falls spezialisierte Angriffe gegen Gebäudeautomationssysteme ausgeführt werden, geschieht dies möglicherweise nur bei besonders lohnenswerten Zielen. Da die Honeypots unter IP-Adressen privater Internetanschlüsse und von Universitäten erreichbar waren, wurden sie möglicherweise nicht als lohnenswertes Ziel für kriminelle Aktivitäten betrachtet.

7.2.4 Fehlerhafte Protokollroutine

Möglicherweise wurden Angriffe auf den Honeypot durchgeführt, diese aber aufgrund eines Fehlers nicht im Protokoll verzeichnet. Da vor und nach dem Experiment ein Test beider Honeypot-Dienste durchgeführt wurde, erscheint diese Möglichkeit jedoch unwahrscheinlich.

Eine abschließende Aussage darüber, warum die Angriffe ausblieben kann also mit den vorhandenen Informationen nicht getroffen werden.

7.3 Offene Fragestellungen

In diesem Abschnitt werden weiterführende Forschungsfragen vorgestellt, die im Laufe dieser Arbeit aufgekommen sind.

Eine weiterführende Untersuchung sollte weitere Protokolldaten erheben. Insbesondere sollten TCP-Pakete, die auf geschlossenen Ports ankommen protokolliert werden. Dadurch ließe sich ein Systemscan etwa durch nmap oder Shodan detektieren. Mit den dadurch gewonnenen Daten könnte zum einen die Shodan-Suchroutine genauer untersucht werden und zum anderen könnte ein durchgeführter Scan mit nmap auf ein Interesse eines Angreifers hindeuten.

Ebenfalls interessant wäre eine Wiederholung des Experiments mit einem anderen DDC. Falls ein Gerät verwendet wird, das häufiger genutzt wird, könnte das die Wahrscheinlichkeit von Angriffen erhöhen.

Abbildungen

Tabellen

Quellcodefragmente

QUELLCODEFRAGMENTE

Literatur

[Bau+13] Siegfried Baumgarth, Elmar Bollin, Manfred Büchel, Burkhard Fromm, Alfred Karbach, Dieter Otto, Hartmuth Paerschke, Peter Ritzenhoff, Georg-Peter Schernus, Frank Sokollik u. a. *Digitale Gebäudeautomation*. Springer-Verlag, 2013 (siehe S. 5).

[Bau18] Johann Bauer. *ddc-ui*. 2018. URL: `https://git.informatik.uni-rostock.de/jb592/ddc-ui` (besucht am 02. 09. 2018) (siehe S. 52).

[BB05] David Brumley und Dan Boneh. „Remote timing attacks are practical“. In: *Computer Networks* 48.5 (2005). PII: S1389128605000125, S. 701–716. ISSN: 13891286. DOI: 10.1016/j.comnet.2005.01.010 (siehe S. 28).

[Ber05] Daniel J. Bernstein. „Cache-timing attacks on AES“. In: (2005). URL: `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf` (besucht am 10. 08. 2018) (siehe S. 28).

[Bod+14] Roland Bodenheim, Jonathan Butts, Stephen Dunlap und Barry Mullins. „Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices“. In: *International Journal of Critical Infrastructure Protection* 7.2 (2014). PII: S1874548214000213, S. 114–123. ISSN: 18745482. DOI: 10.1016/j.ijcip.2014.03.001 (siehe S. 23, 24).

[Bun13] Bundesgerichtshof, Hrsg. *GEMA gegen Rapidshare*. I ZR 80/12. Urteil. 15. Aug. 2013. URL: `http://juris.bundesgerichtshof.de/cgi-bin/rechtsprechung/document.py?Gericht=bgh&Art=en&nr=65241` (besucht am 02. 08. 2018) (siehe S. 18).

[Bun17] Bundesgerichtshof, Hrsg. *Patrick Breyer gegen Bundesrepublik Deutschland*. VI ZR 135/13. Urteil. 16. Mai 2017. URL: http://juris.bundesgerichtshof.de/cgi-bin/rechtsprechung/document.py?Gericht=bgh&Art=en&nr=78741 (besucht am 31. 07. 2018) (siehe S. 15).

[Che09] Benoit Chesneau. *gunicorn*. 2009. URL: http://gunicorn.org/ (besucht am 27. 08. 2018) (siehe S. 41).

[Cor+10] Aldo Cortesi, David Weinstein, Doug Freed, Thomas Kriechbaumer, Pietro Francesco Tirenna, Maximlian Hils und Ujjwal Verma. *mitmproxy*. 2010. URL: https://mitmproxy.org/ (besucht am 31. 08. 2018) (siehe S. 51).

[Cou13] Nicolas Couture. *MockSSH*. 2013. URL: https://github.com/ncouture/MockSSH (besucht am 22. 08. 2018) (siehe S. 34, 73).

[Das16] Das Parlament und der Rat der Europäischen Union - gestützt auf den Vertrag über die Arbeitsweise der Europäischen Union, insbesondere auf Artikel 16 - auf Vorschlag der Europäischen Kommission. *Verordnung 2016/679 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung)*. DSGVO. Version 27.04.2016. 2016. URL: https://eur-lex.europa.eu/legal-content/DE/TXT/HTML/?uri=CELEX:32016R0679&from=DE (besucht am 31. 07. 2018) (siehe S. 15).

[Den18] Dennis Schirrmacher. „Botnetz: Mirai-Malware gefährdet durch Cross-Compiling noch mehr Systeme“. In: *heise Security* (27. Aug. 2018). URL: https://www.heise.de/security/meldung/Botnetz-Mirai-Malware-gefaehrdet-durch-Cross-Compiler-noch-mehr-Systeme-4144912.html (besucht am 28. 08. 2018) (siehe S. 43).

[Deu65] Deutscher Bundestag. *Gesetz über Urheberrecht und verwandte Schutzrechte*. UrhG. Version 1.9.2017. 9. Sep. 1965 (siehe S. 18).

[Deu71] Deutscher Bundestag. *Strafgesetzbuch*. StGB. Version 30.10.2017. (15.05.1871). URL: https://www.gesetze-im-internet.de/stgb/index.html (besucht am 28. 07. 2018) (siehe S. 14, 16).

[Dip01] Christian Dipl.-Ing. Müller VDI. *BACnet als StandardBussystem in der Gebäudeautomation*. Hrsg. von Honeywell AG. 2001. URL: https://www.ta.hu-berlin.de/res/co.php?id=14081 (besucht am 25. 07. 2018) (siehe S. 8).

[Ger16] Gerichtshof der Europäischen Union, Hrsg. *Patrick Breyer gegen Bundesrepublik Deutschland*. C-582/14. Urteil nach Vorabentscheidungsersuchen. 19. Okt. 2016. URL: http://curia.europa.eu/juris/document/document.jsf;?docid=184668&cid=254785 (besucht am 31. 07. 2018) (siehe S. 15).

[Gor] Gordon Lyon. *nmap*. URL: https://nmap.org/book/osdetect-methods.html#osdetect-probes (besucht am 08. 08. 2018) (siehe S. 26, 66).

[HB17] Arne Hell und Anja Bröker. *Ein Zentrum gegen Cyberattacken. Telekom investiert*. Hrsg. von Tagesschau. 2017. URL: https://www.tagesschau.de/inland/cyberangriff-abwehr-telekom-101.html (besucht am 25. 07. 2018) (siehe S. 8, 9).

[HHT17] Kevin Hagemeister, Danilo Heide und Florian Thonig. *Penetrationstest von Steuerungskomponenten für die Gebäudeautomation. Komplexe Software Systeme*. 2017 (siehe S. 41).

[hol] holistische SoftwareGarage Andreas Fahrig & Christian Stamm IT-Dienstleistungen GbR. *Kunden & Partner*. URL: http://www.hologarage.de/Partner.html (besucht am 07. 08. 2018) (siehe S. 22).

[HR05] T. Holz und F. Raynal. „Detecting honeypots and other suspicious environments". In: *Proceedings from the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2005*. Proceedings from the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2005. (West Point, NY, USA,). IEEE, 15.-17. Juni 2005, S. 29–36. ISBN: 0-7803-9290-6. DOI: 10.1109/IAW.2005.1495930 (siehe S. 31).

[Kar04] Alfred Karbach. „Gebäudeautomation und technisches Gebäudemanagement". In: *Digitale Gebäudeautomation*. Hrsg. von Siegfried Baumgarth, Elmar Bollin, Manfred Büchel, Burkhard Fromm, Alfred Karbach, Dieter Otto, Hartmuth Paerschke, Peter Ritzenhoff, Georg-Peter Schernus, Frank Sokollik, Friedbert Tiersch und Wilfried Treusch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 315–371. ISBN: 978-3-642-62176-5. DOI: 10.1007/978-3-642-18582-3_9 (siehe S. 7).

[KNX17] KNX Association International. *KNX Der weltweite STANDARD für Haus- und Gebäudesystemtechnik*. 2017. URL: https://www2.knx.org/media/docs/downloads/Marketing/Presentations/HVAC-For-System-Integrators/HVAC-For-System-Integrators_de.pdf (besucht am 18. 07. 2018) (siehe S. 2).

[KNX18] KNX Association International. *KNX Manufacturers list*. 2018. URL: https://www.knx.org/knx-en/for-manufacturers/members/index.php (besucht am 18. 07. 2018) (siehe S. 2).

[Koc96] Paul C. Kocher. „Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology — CRYPTO '96*. Hrsg. von Gerhard Goos, Juris Hartmanis, Jan van Leeuwen und Neal Koblitz. Bd. 1109. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, S. 104–113. ISBN: 978-3-540-61512-5. DOI: 10.1007/3-540-68697-5_9 (siehe S. 28).

[Kra04] N. Krawetz. „Anti-honeypot technology". In: *IEEE Security & Privacy Magazine* 2.1 (2004), S. 76–79. ISSN: 1540-7993. DOI: 10.1109/MSECP.2004.1264861 (siehe S. 30).

[KSH09] Werner Kriesel, Frank Sokollik und Peter Helm. „KNX/EIB für die Gebäudesystemtechnik in Wohn-und Zweckbau". In: (2009) (siehe S. 6).

[KW10] Michael Kerrisk und Jakub Wilk. *inode(7). Linux Programmer's Manual*. 2010. URL: http://man7.org/linux/man-pages/man7/inode.7.html (besucht am 31. 08. 2018) (siehe S. 47).

[LC10] Frederik Lundh und Alex Clark. *Pillow*. 2010. URL: https://pillow.readthedocs.io/en/latest/ (besucht am 31. 08. 2018) (siehe S. 51).

[LG 16] LG München I, Hrsg. *Sharehoster als Gehilfe einer Urheberrechtsverletzung durch unbekannte Dritte*. 21 O 6197/14. Urteil. 10. Aug. 2016. URL: http://www.gesetze-bayern.de/Content/Document/Y-300-Z-BECKRS-B-2016-N-14540 (besucht am 02. 08. 2018) (siehe S. 18).

[Lyo10] Gordon Lyon. *Nmap network scanning. Official Nmap project guide to network discovery and security scanning*. eng. Zero-day release: May 2008. Lyon, Gordon (VerfasserIn). Sunnyvale, CA: Insecure.Com LLC, 2010. 434 S. ISBN: 978-0-9799587-1-7 (siehe S. 26).

[MA07] Iyatiti Mokube und Michele Adams. „Honeypots: concepts, approaches, and challenges". In: *Proceedings of the 45th annual southeast regional conference*. ACM. 2007, S. 321–326 (siehe S. 8–11).

[Mat] John Matherly. *Honeyscore*. URL: https://honeyscore.shodan.io/ (besucht am 21. 08. 2018) (siehe S. 33, 54, 66).

[MHH10] Hermann Merz, Thomas Hansemann und Christof Hübner. *Gebäudeautomation. Kommunikationssysteme mit EIB/KNX, LON und BACnet*. 2., neu bearb. Aufl. München: Fachbuchverl. Leipzig im Carl-Hanser-Verl., 2010. 300 Seiten. ISBN: 9783446421523 (siehe S. 5, 7, 8).

[Mön+10] Adrian Mönnich, Armin Ronacher, Davi Lord und Markus Unterwaditzer. *Flask*. Pallets Project, 2010. URL: http://flask.pocoo.org/ (besucht am 27. 08. 2018) (siehe S. 41, 51).

[MS04] Alan Alexander. Milne und Ernest H. Shepard. *Winnie-the-Pooh*. London: Egmont Children's, 2004. 145 Seiten, gebunden. ISBN: 9781405249430 (siehe S. 9).

[Ne] José Nazario und et al. *Awesome Honeypots*. URL: https://github.com/paralax/awesome-honeypots/blob/195b1fc1247e4e1a09c3bcbdf040388334f709dc/README.md (besucht am 21. 08. 2018) (siehe S. 33, 41).

[OLG17] OLG München, Hrsg. *Voraussetzungen der Teilnehmer- und Störerhaftung eines Sharehostingdienstes*. 29 U 3735/16. Urteil. 2. März 2017. URL: http://www.gesetze-bayern.de/Content/Document/Y-300-Z-BECKRS-B-2017-N-106250 (besucht am 02. 08. 2018) (siehe S. 18).

[OT09] Michel Oosterhof und Uppi Tamminen. *Cowrie*. 2009. URL: https://github.com/micheloosterhof/cowrie (besucht am 22. 08. 2018) (siehe S. 34, 38, 66).

[Pac01] Packet Storm. *FingerPrintFucker*. 2001. URL: https://packetstormsecurity.com/files/24208/bsdfpf.tar.gz.html (besucht am 09. 08. 2018) (siehe S. 27).

[Pan14] Aniket Panse. *Hornet*. 2014. URL: https://github.com/czardoz/hornet (besucht am 22. 08. 2018) (siehe S. 34, 37, 38).

[Pau13] Paul Maddox. *gohoney*. 2013. URL: https://github.com/PaulMaddox/gohoney (besucht am 22. 08. 2018) (siehe S. 34, 37).

[Pro03] Niels Provos. *Honeyd Sample Configurations*. 2003. URL: http://www.honeyd.org/configuration.php (besucht am 09. 08. 2018) (siehe S. 27).

[Pro08] Niels Provos. *Honeyd*. 2008. URL: http://www.honeyd.org/index.php (besucht am 09. 08. 2018) (siehe S. 27).

[RS00] Gaël Roualland und Jean-Marc Saffroy. *IP Personality*. 2000. URL: http://ippersonality.sourceforge.net/ (besucht am 09. 08. 2018) (siehe S. 27).

[Sch00] Werner Schindler. „A Timing Attack against RSA with the Chinese Remainder Theorem“. In: *Cryptographic Hardware and Embedded Systems — CHES 2000*. Hrsg. von Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Çetin K. Koç und Christof Paar. Bd. 1965. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, S. 109–124. ISBN: 978-3-540-41455-1. DOI: 10.1007/3-540-44499-8_8 (siehe S. 28).

[Sho] Shodan. *Suchergebnis*. URL: https://www.shodan.io/search?query=Server%3A+hsgUiServer%2F (besucht am 07. 08. 2018) (siehe S. 22).

[Sho18] Shodan. *Shodan-Suche nach DDC4200-Installationen*. 10.08.2018. URL: https://www.shodan.io/search?query=holistische+SoftwareGarage (siehe S. 25).

[Spi03] Lance Spitzner. *Honeypots: tracking hackers*. Bd. 1. Addison-Wesley Reading, 2003 (siehe S. 10, 11).

[Sta18] Statista. *Smart Home Report. Statista Digital Market Outlook - Market Report*. 2018. URL: https://de.statista.com/outlook/279/100/smart-home/weltweit (besucht am 18. 07. 2018) (siehe S. 1).

[Tam09] Uppi Tamminen. *Kippo*. 2009. URL: https://github.com/desaster/kippo (besucht am 22. 08. 2018) (siehe S. 34, 35, 39).

[Tri02] Sean Trifero. *Stealth Patch*. 2002. URL: http://web.archive.org/web/20090921072120/http://www.innu.org/~sean/ (besucht am 09. 08. 2018) (siehe S. 27).

[Twi01] Twisted Matrix Laboratories. *Twisted Python*. 2001. URL: https://twistedmatrix.com/trac/ (besucht am 23. 08. 2018) (siehe S. 35).

[UPM+] UPMC Sorbonne Universités, Institut National de Recherche en Informatique et en Automatique, University of Pisa und Hebrew University of Jerusalem. *PlanetLab Europe*. URL: https://planet-lab.eu/ (besucht am 10. 08. 2018) (siehe S. 25, 55, 66).

[Ver05] Verein Deutscher Ingenieure GmbH, Hrsg. *Gebäudeautomation (GA) - Grundlagen*. Berlin: Beuth Verlag, 2005. URL: https://www.vdi.de/3814 (siehe S. 5).

[Wan+10] Ping Wang, Lei Wu, Ryan Cunningham und Cliff C. Zou. „Honeypot detection in advanced botnet attacks“. In: *International Journal of Information and Computer Security* 4.1 (2010), S. 30. ISSN: 1744-1765. DOI: 10.1504/IJICS.2010.031858 (siehe S. 30, 31).

[Zig15] ZigBee Alliance, Hrsg. *ZigBee PRO 2015 Specification*. 5. Aug. 2015. URL: http://www.zigbee.org/download/zigbee-pro-2015-spec/ (besucht am 19. 07. 2018) (siehe S. 6).

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Die Arbeit ist noch nicht veröffentlicht und ist in ähnlicher oder gleicher Weise noch nicht als Prüfungsleistung zur Anerkennung oder Bewertung vorgelegt worden.

Rostock, den 12. November 2018