Honeypots zur Verbesserung der Sicherheit von Industrieanlagen

Dominik Schlöder

Zusammenfassung—Industrieanlagen steuern alle unsere Systeme, sind sehr alt und waren früher isoliert vom restlichen Internet. Die Systeme priorisieren Ausfallsicherheit und bieten oftmals keine Authentifikation und Integritätssicherung. Mit der Zeit adaptieren Industrieanlagen öffentliche Protokolle wie Http und SMB. Dadurch wurde die Isolierung aufgehoben und viele Systeme sind von allen im Internet ansteuerbar und angreifbar. Es existieren viele Beispiele, in denen Industrieanlagen von Malware oder Angreifern kompromittiert wurden und beweisen immer wieder, wie wichtig es ist auch diese Systeme abzusichern. Auf Grund ihres Alters sind viele der Systeme aber nicht in der Lage aktuelle Sicherheitsmaßnahmen umzusetzen. Dennoch existieren mit Honeypots und Intrusion Detection Systemen Möglichkeiten, die Sicherheit der Industrieanlagen zu verbessern. In dieser Seminararbeit wird das HoneyPhy-Framework vorgestellt, welches die Erkennung von Honeypots durch Erweiterung der Simulationen erschweren soll. Ein weiteres Thema ist HosTaGe ein Honeypot, welcher automatisch Signaturen für Intrusion Detection Systeme generiert um bösartigen Netzwerkverkehr zu blockieren.

Index Terms—Honeypots, ICS, HoneyPhy, HosTaGe

1 Einleitung

Industrial Control Systems (ICS) ist ein allgemeiner Ausdruck für eine Vielzahl von unterschiedlichen Industrieanlagen bzw. von Systemarchitekturen, dessen einzelne Systeme oftmals über viele Kilometer verteilt sind[1, S.16]. Sie werden überall in der Industrie und kritischen Infrastrukuten genutzt, wie zum Beispiel im elektrischen Netz, Wasseraufbereitungsanlagen, Verkehrssysteme und Fabriken [1, S.16]. Abbildung 1 zeigt ein Beispiel für eine einfache Industrieanlage. In der Realität gibt es viele kleine Systeme dieser Art, die zusammen ein großen Netz von Systemen bilden. Systeme die für die Administration eines Netz zuständig sind, fallen auch unter den Begriff der ICS werden, aber häufig auch als *Supervisory Control and Data Acquisition* (SCADA) bezeichnet.

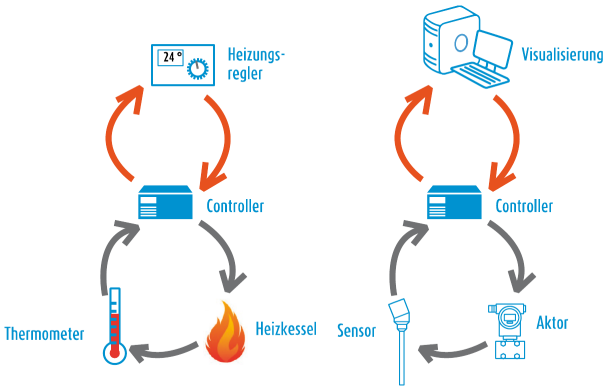

Abbildung 1. Links eine Ausprägung von einem kleiner Industrieanlage und rechts eine generalisierte Form[2, S.4]

Viele dieser Systeme wurden vor 1980 entwickelt, also bevor es IT-Systeme überhaupt gab, und wurden designt, um Ziele wie Wartbarkeit, Verfügbarkeit und Zuverlässigkeit zu erfüllen [1, S.29]. Schutz für Vertraulichkeit boten IC-Systeme nicht, lediglich proprietäre Protokolle und Isolation in einem internen Netz verhinderten Zugriff von Dritten[3, S.1]. Die später entwickelten IT-Systeme haben die gleichen Sicherheitsziele, priorisieren aber Vertraulichkeit (Abbildung 2). Weitere Gegensätze von ICS zu normalen IT-Systemen sind ihre sehr viel längere Lebenszeit mit bis zu 20 Jahren und ein manchmal fehlendes Patch-Management[2, S.5]. Trotzdem entwickelten sich auch Industrieanlagen und adaptierten öffentliche Protokolle von IT-Systemen[1, S.29]. Dadurch gelten die alten Sicherheitsannahmen, Isolation und Unbekanntheit der Protokolle, nicht mehr und müssen so wie Webserver, vor unbefugter Nutzung mit Firewalls, *Intrusion Detection Systems* (IDS) und Honeypots geschützt werden[3, S.3]. Unterschiede von ICS und IT-Systemen sind Alter, fehlendes Patchmanagement und berechnungsschwache Chips. Deswegen können Sicherheitsmaßnahmen aus technischen Gründen nicht übernommen werden[1, S.29].

Ein weiterer Aspekt der Anbindung von ICS ans Internet, ist die Auffindbarkeit durch Suchmaschinen, wie Google oder Shodan. Shodan ist eine Suchmaschine, die das Internet durchsucht und Banner indexiert, so ähnlich wie Google Webseiten indexiert[4, S.1]. Mit Shodan kann herausgefunden werden, können SSH-Server entdeckt werden oder welche Versionen von Internetservices am häufigsten genutzt werden. Es sind auch fortgeschrittenere Suchanfragen möglich. Zum Beispiel kann nach einer Veröffentlichung von einer Schwachstelle herausfinden, wie viele Server von dieser gefährdet sind[4, S.5]. Es ist mit Shodan daher möglich, nach ICS zu suchen. In 2012 wurden 50.000 Industrieanlagen von Shodan indexiert davon ungefähr 7.200 Anlagen für kritische Infrastruktur. Projekt Shine (*SHodan INtelligence Extraction*) fand von 2012-2014 schon 2 Millionen ICS[4, S.54]. Über Shodan werden zwar keine Angriffe durchgeführt, es könnte aber Angreifern erleichtern verwundbare Geräte auzunutzen.

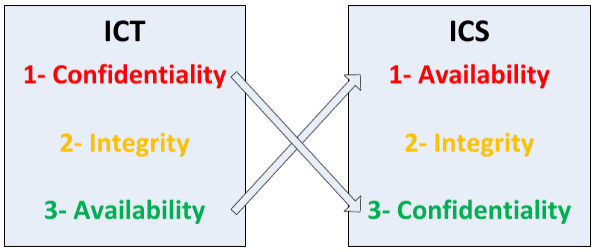

Abbildung 2. Unterschiedliche Sicherheitsziele bei ICS und herkömmlichen IT-Systemen (ICT) [3, S.4]

Wie ungesichert ICS sind, zeigen folgende 3 Vorfälle:

1) 2003 wurde ein Nuklearkraftwerk in Ohio, USA, durch einen *Denial-of-Service* Wurm befallen. Während der Wartungsarbeiten brachte ein Berater mit seinem Gerät den Wurm ins Netz. Da solche Anlagen nur selten gepatched werden, konnte der Wurm ein kritisches System befallen und das Nuklearkraftwerk 5 Stunden lang lahm legen [5, S.11].
2) 2008 gelang es einem Teenager mit einer präparierten TV-Fernbedienung 4 Straßenbahnen zur Entgleisung zu bringen, indem er Signale an die Schienensteuerungen sendete[5, S.14]. Folgen waren Verletzte und Materialschäden an den Straßenbahnen.
3) 2010 wurde einer der bekanntesten Würmer auf ICS bekannt: Stuxnet. Stuxnet verbreitete sich über 4 zero-day-exploits in bekannten Betriebssystemen. Ziel aber waren hauptsächlich Anreicherungsanlagen für Uran im Iran. Diese Malware manipulierte die ICS, sodass sie falsche Ergebnisse lieferten. Der Grund, weshalb Stuxnet lange unbemerkt blieb, war ein Trick um die Ingenieure zu täuschen. Die Industrieanlage wurde von Stuxnet so manipuliert, dass es während der Ausführung seines eigenen Schadcodes das zuletzt ausgeführte Programm anzeigte. Dieses Verhalten gleicht dem Anzeigen von Sicherheitsaufnahmen des gestrigen Tages, während eine Bank ausgeraubt wird[5, S.15].

Folgerungen dieser Vorfälle sind, dass normale Malware Industrieanlagen lahmlegen kann[5, S.11]. Selbst isolierte Industrieanlagen benötigen Schutz vor unerlaubter Nutzung und Staaten arbeiten an Malware für Cyberwarfare[5, S.14f].

1.1 Motivation

Industrieanlagen sind also ein leichtes Ziel für Viren und menschliche Angreifer und müssen entsprechend abgesichert werden. Aufgrund unterschiedlicher Sicherheitsziele zu normalen IT-Systemen ist es wichtig neue Methoden zu entwickeln, um Industrieanlagen abzusichern. Zero-Day-Exploits betreffen beide Klassen von Systemen. Es müssen Wege gefunden werden, solche Exploits zu finden. Weitere Fragestellungen betreffen das Profil der Angreifer [2, S.3]:

- Wer greift ICS an?
- Welche Expertise haben die Angreifer?
- Mit welchen Mitteln versuchen sie Anlagen zu kompromittieren?
- Welche Motivation haben die Angreifer bzw. können Rückschlüsse auf die Motivation gezogen werden?

1.2 Stand der Forschung

Um herauszufinden wie IT-Systeme angegriffen werden, werden normalerweise *Honeypots* eingesetzt. Ein *Honeypot* ist ein System, welches Angreifer anlocken soll und aufnimmt, wie ein Angreifer dieses System angreift[6, S.1]. Honeypots sind dementsprechend so designt, dass sie alles was mit ihnen geschieht persistieren um es anschließend zu analysieren. Vorzugsweise passiert dies ohne dass der Angreifer merkt, dass es sich um einen Honeypot handelt[6, S.1]. Die Daten von Honeypots können anschließend genutzt werden, um Sicherheitslücken zu schließen, Signaturen für Malware und IDS zu erstellen und den Angriffsvektor von Angreifern zu bestimmen. Honeypots eignen sich also auch, um die Fragestellungen für ICS zu beleuchten. Allerdings ist es notwendig spezielle Honeypots für diese zu entwickeln, weil auch andere Protokolle genutzt werden.

Viele Projekte widmen sich der Untersuchung von Angriffen auf ICS, wie zum Beispiel conpot[7] und das Honeytrain-Projekt von Koramis[2]. Beim Honeytrain handelt es sich um die Simulation eines Verkehrsbetriebs. Es wurden nicht nur Softwaresimulationen benutzt, sondern auch Hardware die in einem echten Verkehrbetrieb vorkommen würde[2, S.6]. So wurden Weboberfläche, Sicherheitskameras, ein Schienennetz und andere benötigte Systeme simuliert. Die Ergebnisse des Honeytrains sind, dass die meisten Angriffe automatisiert über Wörterbuchangriffe erfolgen und aus China kommen. Nur wenige Angreifer haben es aber tatsächlich geschafft, in das System einzudringen[2, S.16,18].

1.3 Aufbau der Arbeit

Als Erstes werden die Grundlagen von Honeypots erläutert. Es werden die Aufgaben, die Architekturen und die Klassifizierung von Honeypots erläutert. Anschließend werden Gegenmaßnahmen zur Entdeckung von Honeypots diskutiert. Danach werden neue Entwicklungen im Bereich von Honeypots betrachtet. Zunächst wird erörtert wie die Tarnung von ICS-Honeypots durch Simulation des physikalischen Prozesses verbessert werden kann und wie entsprechende Modelle entwickelt werden können. Anschließend wird betrachet, wie SCADA-Systeme in einer Kombination von Honeypot und Intrusion Detection System besser geschützt werden können. Zuletzt wird ein Ausblick auf weitere Arbeiten gegeben.

2 Grundlagen

Dieses Kaptiel erläutert als erstes die grundsätzliche Funktionalität von Honeypots. Anschließend wird eine mögliche Klassifikationen von Honeypots vorgestellt. Abschließend werden Möglichkeiten zur Entdeckung von Honeypots erläutert.

2.1 Grundlagen von Honeypots

Honeypots sind schwer zu definieren, da sie vielfältig in Zielen, Funktionalität und Aufbau sind[8, S.58]. Ziele von Honeypots können ganz unterschiedlich ausfallen. Im Gegensatz zu einer Firewall, die genau ein Problem löst und

zwar unerlaubte Verbindungen zu unterbinden, können Honeypots dazu genutzt werden, um Schwachstellen in Servern zu finden, Malware einzufangen und zu analysieren oder um Innenangreifer und Malware Infektionen offen zu legen. Wegen dieser Flexibilität, gibt es keinen einzigen Honeypot für alle diese Anwendungsfälle [8, S.59]. Die Einsatzzwecke sind also einerseits, um Sicherheit in produktiven Systemen zu erhöhen und, andererseits Wissen über Angreifer und Malware zu sammeln [8, S.60].

Honeypots bieten deshalb folgende Funktionalität. Sie emulieren eine Resource, dessen einzige Aufgabe es ist, auf unautorisierte Art und Weise benutzt zu werden[9, S.17].Eine Resource kann eine Datei, ein Programm, ein System oder eine Simulation von mehreren Systemen sein. Damit Sie für Angreifer interessant sind, emulieren sie echte produktiv eingesetzte Systeme [9, S.17]. Dementsprechend sind Honeypots keine Systeme, die eine Funktion für normale Benutzer erfüllen[8, S.58]. Daraus folgt eine wichtige Annahme: Alle Interaktionen mit einem Honeypot sind verdächtig [9, S.17] und somit für Untersuchungen interessant. Deswegen haben alle Honeypots eine Komponente um alle Interaktionen zu überwachen und zu speichern[9, S.17].

2.2 Klassifikationen

Der Aufbau von Honeypots fällt je nach Anwendungsfall unterschiedlich aus. Daher haben sich unterschiedliche Klassifikationen für Honeypots gebildet. Die vorgestellte Klassifikation ordnet Honeypots mit zwei orthogonalen Attributen. Das eine Attribut beschreibt, den Simulationsgrad des Honeypots, nochmals aufgeteilt in drei Unterattribute und das andere Attribut die Art der bereitgestellten Resource, auch nochmal aufgeteilt in drei Unterattribute[9, S.22].

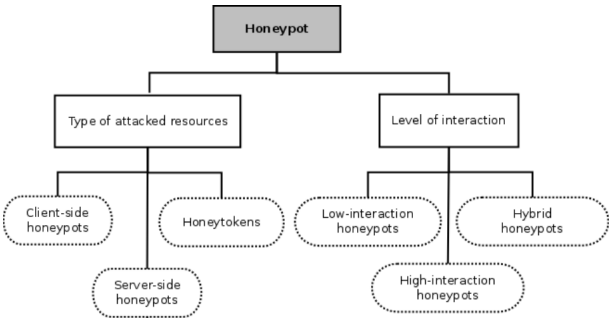

Abbildung 3. Klassifizierung von Honeypots nach [9, S.23]

Der Simulationsgrad von Honeypots wird in *low-*, *hybrid-* und *high-Interaction* eingeteilt. Bei einem *high-Interaction* Honeypot handelt es sich um ein System mit Hardware, Betriebssystem und laufenden Services. Ein Angreifer kann also mit dem Honeypot, wie mit einem echten System interagieren. Dies ist Vorteilhaft um 0-day Exploits, Schwachstellen, die noch nicht bekannt sind, aufzudecken[9, S.19]. *low-Interaction* Honeypots hingegen simulieren nur Services und schränken abhängig von der Güte der Simulation die Interaktion des Angreifers mit dem System ein. Ist die Simulation unvollständig, könnte ein Angriff vor der Aufnahme abbrechen. Dafür sind sie durch ihre geringere Komplexität einfacher zu kontrollieren [9, S.18]. Das letzte Attribut dieser Klassen sind die hybriden Honeypots, die Vorteile von beiden vorherigen Klasse vereinen, indem mehrere Honeypots beider Klassen benutzt werden[9, S.19].

Das andere Attribut beschreibt die Art der Resource, eingeteilt in *Client-Side*, *Server-Side* und *honeytoken*. *Client-Side* Honeypots sind anders, weil sie sich nicht passiv ausnutzen lassen, stattdessen sind es aktive Systeme, die sich durch Interaktion mit passiven bösartigen Systemen, also Servern oder Dateien, ausnutzen lassen[9, S.17]. Ein *Client-Side* Honeypot simuliert also einen unsicheren Webbrowser[9, S.18] oder einen Emailclient, der Anhänge öffnet[9, S.135]. *Server-Side* Honeypots sind die traditionellen Honeypots, die einen Service oder Services über offenen Ports simulieren. Sie eignen sich zum einfangen von autonomer Malware[9, S.18]. *Server-Side* Honeypots können viele Services gleichzeitig simulieren, obwohl es auch Honeypots gibt die sich auf einen Service spezialisieren und sich dafür besonders gut für diese Art von Angriffen eignen[9, S.23]. Bei der letzten Art handelt es sich um *honeytokens*, bei diesen Honeypots handelt es sich um eine Datei, eine Email oder einen Datenbankeintrag, der oder die normalerweise nicht verarbeitet wird[9, S.23].

Das erste Attribut betrifft also den Grad der möglichen Interaktion mit einem Honeypot. Daraus ergibt sich oftmals auch der mögliche Einsatzzweck. So kann mit einem low-Interaction Honeypot Malware gefangen werden, die bekannte Sicherheitslücken ausnutzen. Aufgrund von Unterschieden zwischen der Simulation und echten Systemen eignen sie sich allerdings nicht, um zero-day-exploits zu entdecken. Hierfür müsste dann ein High-Interaction Honeypot installiert werden, damit im eigentlichen System die Sicherheitslücke entdeckt werden kann. Das zweite Attribut betrifft die Art des Systems bzw. Resource, über die Information gesammelt werden soll.

2.3 Erkennung von Honeypots

Honeypots dienen dazu möglichst viele Informationen zu sammeln. Selbst wenn ein Angreifer erkennt, dass es sich um einen Honeypot handelt, ist es interessant wie er es herausgefunden hat [10, S.1]. Es muss aber beachtet werden, dass ein Honeypot nach seiner Entdeckung keine Informationen mehr über den Angreifer sammeln kann [10, S.1]Umgekehrt darf ein Honeypot nicht zu sehr geschlossen sein, sodass der Angreifer kaum mit dem Honeypot interagieren kann. Dementsprechend müssen Honeypots gut versteckt und unaufällig sein, aber auch nicht allzu versteckt und unaufällig [10, S.1].

Erkennung von Honeypots ist sehr implementierungs abhängig. Auf der einen Seite sind low-interaction Honeypots aufgrund ihres geringen Simulationsgrads leichter zu durchschauen als Honeypots die mit einer virtuellen Maschine simuliert werden oder echte physikalische Hardware haben. Auf der anderen Seite ist es sehr viel schwieriger, alle wichtigen Interaktionen bei high-interaction Honeypots zu speichern und zu

filtern, da sie sehr viel größere Datenmengen erzeugen und das System verlangsamen[10, S.5]. Im folgenden Teil wird betrachtet, wie unterschiedliche Systeme auf denen Honeypots aufsetzen, herausgeben, dass es sich um kein echtes System handelt.

```
UML Audio Relay (host dsp = /dev/sound/dsp,
          host mixer = /dev/sound/mixer)
Netdevice 0 : TUN/TAP backend -
divert: allocating divert_blk for eth0
...
Initializing software serial port version 1
 /dev/ubd/disc0: unknown partition table
```

Abbildung 4. UML spezifische Geräte verraten den Honeypot[10, S.2].

Bei User-Mode-Linux (UML) handelt es sich um eine Userspace Programm, dass einen weiteren Linux-Kernel über den echten Kernel simuliert[10, S.2]. Dies ist ähnlich zu virtuellen Maschinen. UML simuliert aber keine Hardware stattdessen nutzt es den Kontext des echten Kernel[10, S.3]. Ein Angreifer, der auf einem UML-Honeypot ist, würde am ungewöhnlichen Systemlog oder der ungewöhnliche Hardware Zusammenstellung wie Festplatte oder Netzwerkkarte bemerken, dass es sich um einen Honeypot handelt. Auch ein Blick in Dateien des proc-Verzeichnisses kann herausgeben, dass es sich um einen Honeypot handelt, wie Abbildung 5 zeigt.

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id       : User Mode Linux
model name      : UML
mode            : tt
[...]
```

Abbildung 5. Ungewöhnlicher Prozessor im UML ist leicht zu erkennen[10, S.2].

Virtuelle Maschinen bieten einen deutlich höheren Simulationsgrad als UML. Dementsprechend kann nicht schon am Kernel erkannt werden, dass es sich um einen Honeypot handelt, schließlich wird ein unveränderter Kernel benutzt. Nichts desto trotz kann ungewöhnliche Hardware genauso wie bei UML herausgeben, dass es sich um kein echtes System handelt. Leicht zu erkennen ist es an der MAC-Adresse der benutzten Netzwerkkarten. MAC-Adressen werden einzigartig vergeben, die vorderen 3 Bytes sind aber eine Identifikationsnummer des Herstellers. In einer virtuellen Maschine würde also ein Blick auf die MAC-Adresse reichen. Aber auch die restliche Hardware würde verdächtige Namen liefern [10, S.3]. Viele virtuelle Maschinen bieten an, mit dem Hostsystem zu interagieren. Das führt selbstverständlich auch zur Aufdeckung des Honeypots [10, S.3]. Heutzutage sind virtuelle Maschinen gerade im Unternehmensumfeld weit verbreitet. Durch Nutzung von virtuellen Maschinen kann Last besser auf Hardware verteilt werden. Außerdem bietet die Nutzung eine Separierung von Resourcen und verstärkt die Sicherheit der unterschiedlichen Maschinen. Daher reicht ein einfaches Identifizieren, dass es sich um eine virtuelle Maschine handelt, nicht mehr aus.

Der Aufruf von chroot, ändert das Root-Verzeichnis innerhalb eines Prozesses. Bei chroot handelt es sich nicht um einen Sicherheitsmechanismus, dennoch wird es in echten Systemen benutzt um sensitive Services oder Daten vom restlichen System abzukapseln[10, S.4]. Ein *ls -ialg /* liefert auf einem echten Dateisystem für das Root-Verzeichnis (.) und dessen Unterverzeichnis (..) jeweils die gleiche inode-Nummer 2. Bei Systemen in einem Chroot erhält man unterschiedliche Nummern. Des Weiteren versteckt chroot, keine Prozesse außerhalb des chroots. Ein Angreifer kann mit einem ptrace-Aufruf außerhalb vom chroot auf das darunterliegende System zugreifen [10, S.5].

Eine forgeschrittenere Möglichkeit des Erkennen von Honeypots ist das Messen von Systemcalldauern. So könnte ein Systemcall abhängig von den genutzten logging-Mechanismen oder eine langsamere Implementierung zu sehr viel langsamereren Aufrufen führen als auf echten Systemen [10, S.6]. Auch ein Aufdecken von Debuggern durch Software innerhalb eines Honeypots macht ihn erkennbar. Angenommen ein Honeypot würden Malware mit ptrace überwachen. Es existiert aber die Limitierung nur einen Debugger innerhalb eines Prozesses auszuführen. Falls die Malware ptrace aufrufen würde, würde dieser Aufruf fehlschlagen und der Malware zeigen, dass es sich um einen Honeypot handelt[10, S.6].

Auch übers Netzwerk können Honeypots aufgedeckt werden. Nmap ist ein Portscanner für Computer, die sich im Netzwerk befinden. Anhand von unterschiedlichen Implementierungen und Verhalten von Sockets kann Nmap das Betriebssystem erkennen[11]. Dies geschieht, indem Nmap unterschiedliche Attribute, wie verfügbare TCP-Optionen, Reihenfolge dieser und intitiale Packetgröße der IP-Packete, mit einer Datenbank abgleicht[11]. Eine low-Interaction Honeypot Implementierung könnte daher durch Nmap aufgedeckt werden, weil Nmap keinen passenden Eintrag dazu in der Datenbank finden würde.

Insgesamt führen diese Merkmale alle darauf zurück, dass Simulationen von Systemen, auf denen Honeypots beruhen, sich unterschiedlich oder merkwürdig zu echten Systemen verhalten. Schlecht implementierte Systeme hinterlassen also einen einzigartigen Abdruck, was als *fingerprinting* bezeichnet wird[8, S.70]. Dabei muss es sich nicht mal um ein technisches Merkmal handeln, Rechtschreibfehler in der Simulation von Protokollen können schon ausreichen, um einen Honeypot aufzudecken[8, S.70]. Manche Merkmale sind einfach zu beheben, wie zum Beispiel, die MAC-Adresse in virtuellen Maschinen, andere Erkennungsstrategien sind schwieriger zu verdecken, wie zum Beispiel, die Limitierung auf nur einen Debugger in einem Prozess oder Interaktionen zwischen Host und Guest in einer virtuellen Maschine. Der vermehrte Einsatz von Honeypots führt dazu, dass Angreifer und Schadsoftware Werkzeuge zur Erkennung von Honeypots benutzen, wie zum Beispiel die Malware Agobot [10, S.4]. Was Sicherheitsforscher dazu zwingt, bessere Honeypots zu entwickeln, um weiterhin neue Informationen zu sammeln

[10, S.10].

3 Glaubwürdige Simulation von physikalischen Prozessen

Wie im vorherigen Abschnitt besprochen entsprechen viele Gegenmaßnahmen dem Entdecken von unglaubwürdigen Eigenschaften. Es wurden viele Honeypots für Industrieanlagen entwickelt, die Protokolle oder einzelne Geräte vollständig emulieren und zunächst schwer von echten Systemen zu unterscheiden sind, wie zum Beispiel Conpot oder Gaspot [12, S.3]. Dennoch bietet keiner dieser Honeypots standardmäßig Simulation von physikalischen Prozessen, wie Interaktionsdauer oder konsistente Änderung nach Absatz eines Befehls[12, S.1]. Das HoneyPhy-Framework implementiert Lösungen für fehlende Interaktionsdauer und Vortäuschung von physikalischen Prozessen, die die Honeypots simulieren sollen [12, S.1]

Angenommen es soll ein Honeypot für das dargestellte System (links) aus Abbildung 1 modelliert werden. Zusätzlich soll die Steuerung dieses System, wie bei anderen IC-Systemen, aus dem Internet möglich sein. Der Einsatz eines low-Interaction Honeypots könnte unzureichend sein, weil ein Angreifer eben genannten Punkte erkennen würde. Als High-Interaction Honeypot könnte ein weiteres System eingesetzt werden. Dies ist aber bei komplexeren und verteilten Systemen nicht möglich, weil ein Angreifer tatsächlich die Kontrolle über das System erlangen und bösartige Kommandos absetzen könnte. Auch aus Kostengründen wäre dieses Vorgehen unwirtschaftlich [12, S.2]. Das HoneyPhy-Framework implementiert daher einen hybriden Honeypot mit simulierten physikalischem Prozess und echten oder simulierten Regelsystem[12, S.2].

3.1 Das HoneyPhy-Framework

Abbildung 6 zeigt die Schwächen von bisherigen Honeypots. Ein Angreifer auf dem beschriebenen ICS wird merken, dass es nicht nur keine Zeit benötigt um die Befehle umzusetzen, sondern auch die Befehle falsch umsetzt. Das HoneyPhy-Framework schlägt daher vor, die physikalischen Prozesse von ICS nach der Realität zu modellieren und dadurch glaubwürdiger zu machen. Die Architektur des HoneyPhy-Framework besteht aus folgenden Komponentenklassen: den Kommunikationskomponenten, die Gerätekomponenten, Interkommunikationskomponenten und einer Prozesskomponente [12, S.5]. Die Kommunikationskomponenten beschreiben die möglichen Protokolle über die der Honeypot erreichbar sein soll. Die Gerätekomponenten sind entweder echte Regelsysteme oder werden durch Software simuliert und kommunizieren mit und ändern die Prozesskomponente. Falls ein echtes Regelsystem genutzt wird, muss keine Interaktionsdauer modelliert werden, ansonsten muss die Simulationssoftware die Dauer berücksichtigen [12, S.5] Die Prozesskomponente muss den physikalischen Prozess simulieren, dies kann über gesammelte Daten des echten Systems geschehen bzw. ein Blackbox-Modell oder durch simulierende Whitebox-Modelle, wie linear dynamische oder autoregressive Modelle [12, S.5]. Die Kommunikation zwischen Geräten muss über die Kommunikationskomponenten geschehen, sodass der Angreifer diese mitlesen kann. Hingegen muss die Kommunikation zur Prozesskomponente über ein verborgenes Netz geschehen [12, S.5]

3.2 Modellierung des physikalischen Prozesses

Die Modellierung des physikalischen Prozess aus Abbildung 1 mit dem geschilderten Modell realisiert werden. Dieses Modell basiert auf gemessenen Daten eines echten Systems dieser Art[12, S.6]. Aus den Daten und dem Newtonschen Abkühlungsgesetz, Temperatur T zum Zeitpunkt t definiert als $T(t) = h \cdot A \cdot (T(t) - T_{env})$ mit h als Temperaturtransferkoeffizient, A als Temperaturtransferoberfläche und T_{env} als Umgebungstemperatur[13],können die Gleichungen erstellt werden für die Temperatur des geheizten Elements T_{bulb}und der Luft T_{air}[12, S.6]:

$$\begin{aligned} T_{bulb}(t) = T_{bulb}(t-\Delta t) + \Delta t \cdot [(S_{heater}) \cdot \\ (0.00775 \cdot T_{bulb}(t-\Delta t)^{1.04}) - 0.00084 \cdot \\ (T_{bulb}(t-\Delta t) - T_{air}(t-\Delta t))^{1.48775}] \\ T_{air}(t) = T_{air}(t-\Delta t) + \Delta t \cdot [(0.00084 \cdot \\ T_{bulb}(t-\Delta t) - T_{air}(t-\Delta t))^{1.085} - 0.00041 \cdot \\ (T_{air}(t-\Delta t) - T_{env}(t-\Delta t))^{1.225}] \end{aligned}$$

Das aufgeheizte Element T_{bulb} erwärmt sich, wenn der Zustand S_{heater} gleich 1 ist und kühlt sonst zur Umgebungstemperatur ab. Mit diesem Modell kann eine der beiden Temperaturen T_{bulb} oder T_{air} gesetzt werden und daraus ergibt sich dann anschließend die andere der beiden Variabeln.

3.3 Modellierung der Interaktionsdauer

Die Modellierung der Interaktionsdauer muss nicht notwendigerweise erfolgen, falls aber kein physikalisches Gerät im Honeypot genutzt wird, kann die Interaktionsdauer wieder mit einem Blackbox- oder Whitebox-Modell simuliert werden. Ein Blackbox-Modell wird folgendermaßen erstellt. Angenommen es wurden M Interaktionsdauern von einer Funktion *Starte Heizung* aufgenommen. Partitioniere die Menge M in B Intervalle, wobei eine Intervallbreite $t_i = i\frac{H}{B-i}$ und H die angenommene maximale Laufzeit ist. Aus diesen Intervallen und den Interaktionsdauern kann nun ein Histogramm bzw. ein Signaturvektor S erstellt werden. Die einzelnen Werte s_j des Signaturvektors S sind so definiert [14, S.4]:

$$s_j = \begin{cases} |\{m : t_{j-1} \leq m < t_j, m \in M\} & 0 < j < B \\ |\{m > H, m \in M\}| & j = B \end{cases}$$

Dieser Signaturvektor ist einzigartig für jedes Gerät oder zumindest jedes Gerät eines bestimmten Hersteller, da es einzigartige mechanische und pyhsikalische Attribute besitzt[14, S.8]. In diesem Fall könnte nun zufällig abhängig von der Verteilung, die durch den Signaturvektor bestimmt ist, eine Dauer gezogen werden, um eine Interaktionsdauer zu simulieren.

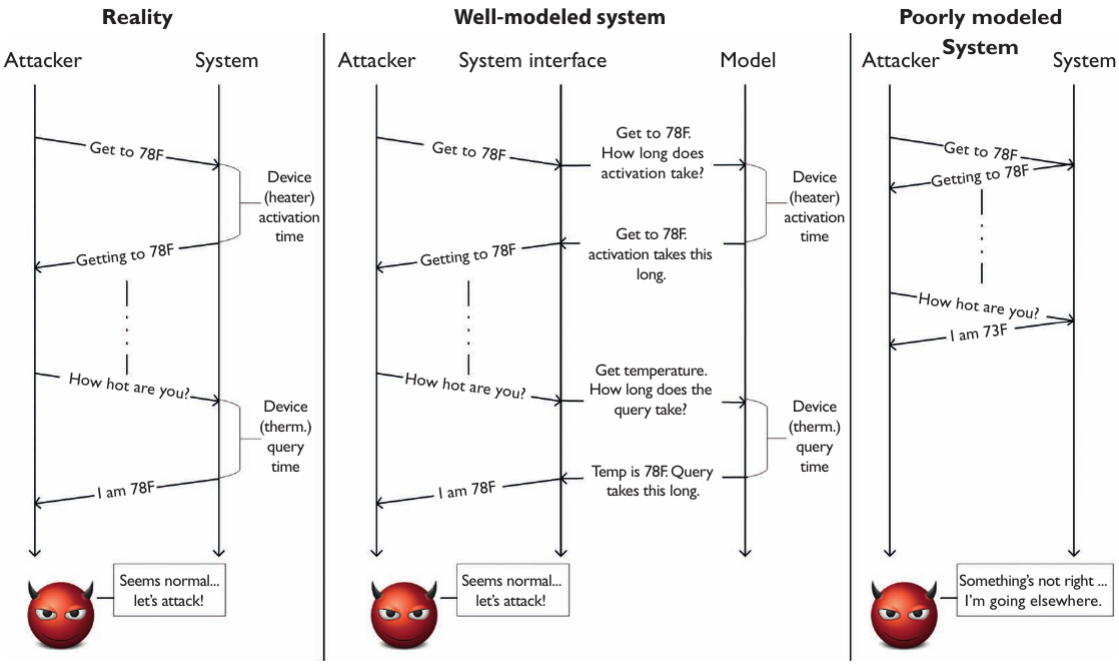

Abbildung 6. Links das Verhalten des realen Systems, Mittig ein Honeypot, welcher Interaktionsdauer und physikalischen Prozess modelliert und Rechts ein Honeypot mit fehlender Simulation [12, S.3].

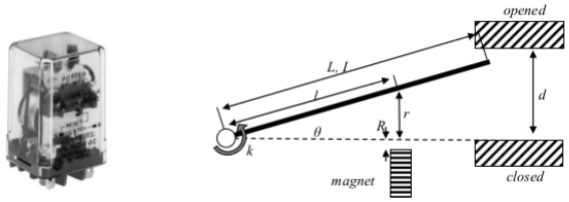

Abbildung 7. Das Gerät, wofür ein Blackbox- und Whitebox-Modell erstellt wurde[14, S.8]. Das Whitebox-Modell benötigt 4 Gleichungen und 11 Variabeln.

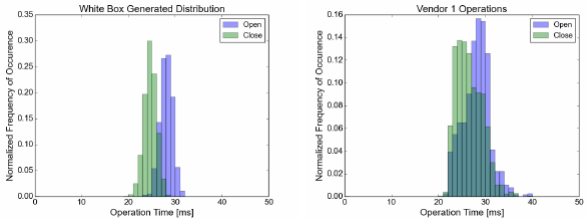

Abbildung 8. Links der Signaturvektor des Whitebox-Modells und rechts der des Blackbox-Modells[14, S.11].

Ein Whitebox-Modell wird durch die physikalischen Gleichungen eines Geräts erstellt. Als Gerät für das Whitebox-Modell wird ein Schalter, wie in Abbildung 7, in Stromnetzen genutzt, das starken Stromfluss durchlässt oder blockiert und durch schwache elektronische Signale gesteuert wird[14, S.8]. Es werden bereits mehrere physikalische Gleichungen für so ein einfaches Gerät benötigt. Daher wird hier auf die Quelle [14] verwiesen, die das Whitebox-Modell genauer erklärt. Wichtig ist allerdings, dass die Gleichungen, die dieses System beschreiben, erstellt wurden, indem das Gerät in seine Einzelteile zerlegt und die Dokumentation des Herstellers genutzt wurde[14, S.10]. Anschließend kann ein Signaturvektor für Interaktionsdauern erstellt werden, indem die Zeitkonstante über mehrere Simulationen zufällig aus einer Gaussverteilung gezogen wird[14, S.11].

Abbildung 8 zeigt, dass sowohl mit 1200 Messungen aus dem Whitebox-Modell als auch mit dem Blackbox-Modell gute Modelle für Interaktionsdauern erstellt werden können[14, S.11].

3.4 Validierung der Modelle

Der Vorteil eines Blackbox-Modells beruht darin, dass es lediglich eine Menge an gemessenen Interaktionsdauern benötigt um ein gutes Modell zu erstellen[14, S.10]. Es ist aber nicht immer möglich, eine entsprechende Menge von Daten zu sammeln, weil einige Operationen nur sehr selten ausgeführt werden. Zusätzlich wird ein phyiskalischer Zugriff zum Gerät benötigt, damit eine Messung erstellt werden kann[14, S.10]. Daher muss eventuell auf ein Whitebox-Modell zurück gegriffen werden[14, S.10]. Auch eine Kombination von beiden Modellen ist möglich um ein besseres Modell des Systems zu erstellen[14, S.12].

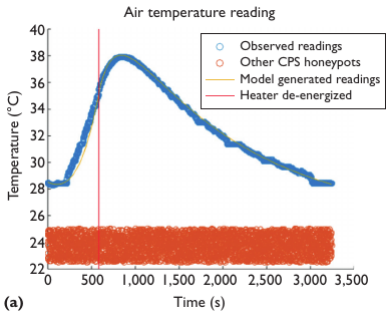

Abbildung 9. Blau die observierte Werte des Systems, Geld das vorgestellte Modell und Rot die Werte von bekannten ICS-Honeypots wie Conbot[12, S.7]

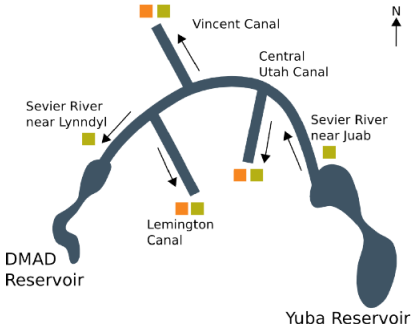

Abbildung 10. Diagramm des Flusses markiert mit den Systemen. Unten Links befindet sich der zentrale Server für das ICS [16, S.2].

Mit den so erstellten Modellen könnte ein Honeypot für das Beispielsystem aus Abbildung 1 erstellt werden. Abbildung 9 zeigt, dass die Modellierung des physikalischen Prozess, den aufgenommenen Daten entspricht. Für real eingesetzte Systeme ist eine Modellierung mit dem richtigen Abstraktionsmaß deutlich komplizierter. In [15] wird ein Algorithmus für Stromnetzte beschrieben, der automatisch ein Modell des physikalischen Systems erstellt. Dennoch ist diese Art der Modellierung sehr Aufwändig, da der Algorithmus zumindest mit allen möglichen Zuständen und Geräten parametrisiert ist[15, S.2]. Eventuell existieren aber bereits Modelle dieser Systeme, die wegen Risikomanagement erstellt wurden und die nach Anpassung mit dem HoneyPhy-Framework bentutzt werden könnten.

4 Signaturgenerierung durch Honeypots

Die erstellten Modelle für die Simulation von Interaktionsdauern können in Verbindung mit einem Klassifizierer, zum Beispiel einem neuronalen Netz, auch dazu genutzt werden um Signaturen von den Geräten zu erstellen [12, S.1]. Diese Zeitstempel-Signaturen können dann in Intrusion Detection Systemen (IDS) eingefügt werden um vor Delay-Attacken zu schützen. In diesem Abschnitt wird zunächst eine Delay-Attacke auf einer Industrieanlage erörtert. Anschließend wird ein Honeypot vorgestellt, der automatisch Signaturen für ein Intrusion Detection System erstellt. Dann wird die Güte der erstellten Signaturen untersucht.

4.1 Angriff auf eine Industrieanlage

Delay-Attacken erhöhen die Antwortzeiten von den Sensoren. Dadurch wird das System in einen instabilen Zustand gebracht, was gefährliche Auswirkungen haben oder großen finanziellen Schaden verursachen kann. Dazu folgendes Beispiel: Der Fluss Sevier, in UTAH USA, bewässert 286.000 Hekter in einer trockenen Region[16, S.2].

Der Wasserfluss wird durch ein ICS überwacht und angepasst. Abbildung 10 zeigt in Grün Sensoren für Wasserfluss und in Orange Sensoren für den Wasserstand der Schleusen [16, S.2]. Das System hat halbautomatische Schleusen. Die Sensorwerte werden automatisch an einen zentralen Server versandt[16, S.2]. Das Unternehmen dieses Systems nimmt Anfragen per Telefon oder SMS für Schleusenöffnungen von Bauern entgegen, wenn diese mehr Wasser für ihr Farmland benötigen[16, S.2]. Die größte Schwachstelle im System ist das Unternehmen, welches die Schleusen kontrolliert[16, S.5].

Um einen Delay-Angriff durchzuführen könnte ein Angreifer das interne ungeschützte Netzwerk der Sensoren infiltrieren oder eine *Man-In-The-Middle* Attacke auf die ungeschützte Webseite, auf der die Daten gesammelt und angezeigt werden, durchführen um fehlerhafte Daten anzuzeigen[16, S.5]. Angenommen der Wasserfluss hat bereits einen hohen Stand und die Sensorwerte werden um 100 Stunden verzögert. Der Steuerman würde dann einige Schleusen schließen. Da die Sensorwerte aber verzögert werden, würden mehr Schleusen geschlossen werden als benötigt. Wenn festgestellt wird, dass der Wasserstand niedrig ist, müssten viele Schleusen geöffnet werden um wieder normale Werte zu bekommen, durch die Verzögerung wird aber wieder eine zu starke Reaktion provoziert. Das System gerät in einen instabilen Zustand[16, S.5]. Ein Angriff dieser Art könnte einen Verlust von über 70 Millionen Dollar verursachen [16, S.6]. Ein IDS vor dem zentralen Server könnte so einen Angriff frühzeitig aufdecken, wenn anomale Interaktionsdauern von den Sensoren gemessen werden[12, S.2]. Ein solches Intrusion Detection System würde aber nicht davor schützen, wenn der zentrale Server mit Malware infiziert werden würden. Um Signaturen von Malware zu sammeln, wird ein Honeypot vorgestellt, der automatisch solche Signaturen erstellt.

4.2 Architektur von HosTaGe

HosTaGe ist ein neuer low-Interaction ICS-Honeypot, der die häufig genutzten ICS-Protokolle, wie Modbus, S7 und SMNP aber auch bekannte Protokolle wie HTTP, SMB und Telnet implementiert[17, S.1f]. HosTaGe steht für Honeypot-To-Go, weil die Implementierung eine Android-Applikation ist und ähnlich wie Conpot mehrere Geräte simuliert[18,

S.1]. Was HosTaGe von anderen Honeypots unterscheidet ist, dass es nach Detektierung von Angriffen, automatisch Signaturen für die Bro IDS erstellen kann[17, S.1]. HosTaGe erstellt zwei Arten von Signaturen. Einmal werden Signaturen mit dem erweiterten endlichen Automaten erstellt, diese bestehen aus den Ausgaben des Honeypots und beschreiben, wie der Angreifer mit dem Honeypot interagiert hat. Diese Signaturen können später zur Analyse der Malware genutzt werden. Die anderen Signaturen werden mit den kompletten Kommunikationspaketen zwischen Angreifer und dem Honeypot erstellt und werden in das Intrusion Detection System eingefügt.

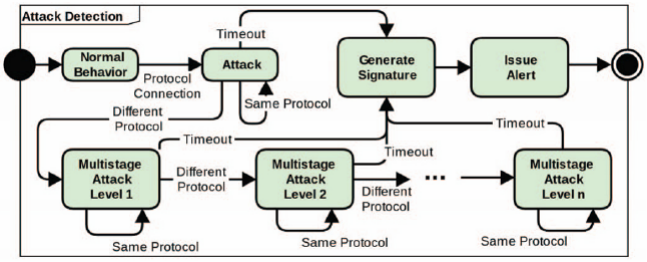

Abbildung 11. Der erweiterte endliche Automat für Multistage Angriffsdetektion von HosTaGe[17, S.3].

So wie andere low-Interaction Honeypots[19] basiert das Modell von HosTaGe auf einem erweiterten endlichen Automaten[17, S.3]. Im Gegensatz zu den endlichen Automaten können sie auch Zustandsänderungen durch Bedinungen (*if-statements*) durchführen. Der Automat M aus Abbildung 11 ist beschrieben durch den Tupel $M = (S, s_o, I, O, V, \delta, \lambda)$[17, S.3]. S ist die Menge der Zustände. Solange kein Angriff auf dem Honeypot durchgeführt wird bleibt er im Startzustand s_0. Sobald ein Angriff gestartet wird, wechselt der Honeypot in den nächsten Zustand. Hochentwickelte Malware wie Stuxnet oder dessen Weiterentwicklung Flame greifen Systeme auf vielen Protokollen an[17, S.1]. Um sinnvoll Signaturen für solche Viren zu erstellen, müssen deren gesendete Packete und *Payloads* (die bösartigen Daten, die auf dem System gespeichert werden) von allen Protokollen zusammen aufgenommen werden. Wenn also Innerhalb eines Zeitfensters tw von 15 Minuten x Protokolle angegriffen werden[17, S.4], wechselt der Automat in den Zustand *Multistage Attack Level x*, ansonsten wird das Ende des Angriffs angestoßen, eine Signatur generiert und eine Nachricht über den Angriff versendet[17, S.3]. Diese Signaturen bestehen aus der Menge der Outputs, die der Angriff generiert hat, und sind nicht für das Intrusion Detection System, sondern sollen aufzeigen, wie der Angriff versucht hat, dass System auszunutzen[17, S.4].

Die Mengen I, O, V und P sind die möglichen Eingaben, Ausgaben, Variabeln und Prädikate. Die Eingaben sind abhängig von den simulierten Protokollen. Eine Eingabe kann das Öffnen einer Verbindung zu einem Port sein oder eine spezielle Nachricht, die dem Honeypot geschickt wird und führt zu einer Zustandsänderung und eventuell zu einer Ausgabe o [17, S.3]. Außerdem kann eine Zustandsänderung stattfinden, wenn eine Boolsche Variable $v \in V$, sich ändert und darauf hin ein Prädikat aus P erfüllt wird. Ein Prädikat könnte also sein, dass ein weiteres Protokoll angegriffen wird, welches bis jetzt noch nicht angegriffen wurde. Die Funktion $\delta(s_i, i, p) = s_j$ ist die Zustandübergangsfunktion und $\lambda(s_j, i, p) = o$ die Funktion, die die Ausgabe aus einem Zustandswechsel generiert[17, S.3].

4.3 Angriffsdetektierung und Signaturgenerierung

HosTaGe ist mit diesem Automaten in der Lage, Angriffe mit einem Protokoll oder mehreren Protokollen aufzudecken. Des Weiteren werden auch die Payloads analysiert, die ein Angriff auf dem System zu speichern versucht[17, S.4]. Dies geschieht folgendermaßen, wenn Malware versucht, Daten auf dem System zu speichern, wird der Hash der Datei mit der Datenbank Virustotal abgeglichen, falls ein Treffer existiert, wechselt der Automat in einen entsprechenden Zustand, der diese Malware repräsentiert[17, S.4].

```
signature modbus-signature{
   ip-proto == tcp
   dst-port == 502
   payload
       /\x21\x00\x00\x00\x00\x06\x01\x04\x00
   event "Modbus attack"
   }
```

Abbildung 12. Beispiel einer Signatur für die Bro IDS generiert von HosTaGe [17, S.5]

Intrusion Detection Systeme lesen jeden Verkehr, der das Netzwerk betritt, und entscheiden anhand der Signaturen, ob es sich um bösartige Inhalte handelt[17, S.4]. Die Signaturen müssen abhängig von der Implementierung erstellt werden. HosTaGes *Signaturegeneration*-Module erstellt Signaturen für die Bro IDS. Zur Erstellung der Signaturen benutzt HosTaGe nicht nur die Packete, die einen Exploit durchführen, sondern die ganze Kommunikation zwischen Angreifer und dem Honeypot[17, S.4]. Dazu gehören auch Verbindungsanfragen, Menge der angegriffenen Protokolle und die Payloads der Malware. Ein Beispiel für solch eine erstellte Signatur ist Abbildung 12.

4.4 Test von HosTaGe

Um die Erkennungsrate von Angriffen und die Nutzbarkeit der generierten Signaturen zu evaluieren, wurden jeweils eine HosTaGe und Conpot Instanz über einen Zeitraum von 12 Wochen ins Internet gestellt[17, S.5]. Die Honeypots waren dabei im gleichen /24-Subnetz, erhielten dynamische IP-Adressen und es wurde keine Firewall eingesetzt [17, S.5]. Dies soll dafür sorgen, dass beide Honeypots die gleiche Wahrscheinlichkeit haben von Viren angesteuert zu werden. Die Erkennungsrate zwischen HosTaGe und Conpot ist beim Protokoll modbus ungefähr gleich, bei den Protokollen HTTP, S7 und SNMP hat HosTaGe aber eine deutlich bessere Detektierungsrate[17, S.5]. In Abbildung 13 spiegelt sich dieses Verhalten wieder. Während dieser Zeit wurden die beiden Honeypots auch von der Suchmaschine Shodan indexiert[17, S.5]. Shodan versucht mit dem HoneyScore[20]

herauszufinden, ob es sich um Honeypots handelt. Dies geschieht anscheinend mit einfachen Anfragen bei HTTP und SSH. Bei den Protokollen für IC-Systeme werden allerdings kompliziertere Nmap/Metasploit-Skripte benutzt[17, S.6]. HosTaGe ist in der Lage beim Honeyscore unerkannt zu bleiben, Conpot hingegen kann nicht sinnvoll auf alle Anfragen antworten oder stoppt mit einem Fehler, sodass Shodan ihn als Honeypot identifiziert[17, S.6]

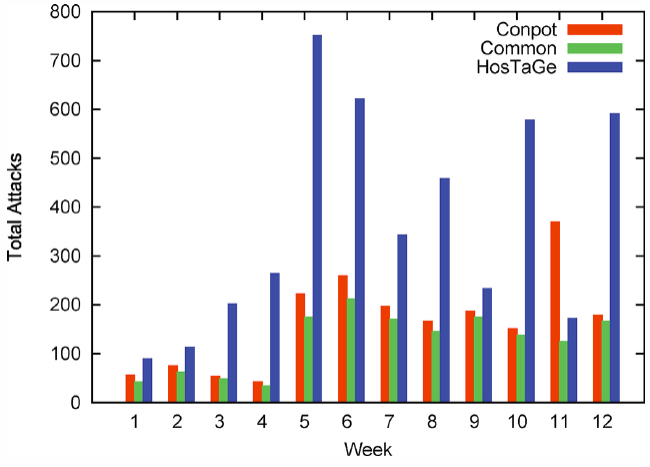

Abbildung 13. Die Anzahl der einzigartigen IP-Adressen über den Zeitraum in Wochen des Versuchs[17, S.5]. Es ist erkennbar, dass HosTaGe deutlich mehr Angriffe detektiert.

Um die Signaturen zu bewerten, wurden zwei weitere Tests durchgeführt. Zunächst wurden auf einer HosTaGe Instanz Multistage Angriffe ausgeführt und anschließend die generierten Signaturen in das IDS eingefügt. Aufgenommen wurde der Netzwerkverkehr mit Wireshark und die von HosTaGe generierten Signaturen. Mit diesen Daten und mehreren frei verfügbaren Daten von normalen und bösartigen Netzwerkverkehren wurde ein Test zur Detektierung von Fehlalarmen (*false positives*) und ein Test zur Detektierung der Angriffe durchgeführt[17, S.6]. Im ersten Test wird der Netzwerkverkehr mit den frei verfügbaren Daten im IDS simuliert, hier sollten keine Alarme auftreten. Im zweiten Test die gesammelten Daten mit den andere Netzwerkverkehren kombiniert und sollten dann Alarme auslösen. Beide Ergebnisse sind positiv, dementsprechend wurden im ersten Test keine Angriffe detektiert. Im zweiten wurden alle Angriffe die in die Netzwerkverkehre eingefügt wurden aufgedeckt[17, S.6]

Zusammenfassend ist festzustellen, dass ein solches System gegen bestimmte Malware erfolgreich detektieren und blocken kann. Allerdings müssen einige Vorraussetzungen erfüllt sein. Zum Beispiel muss bereits eine Signatur für eine bösartige Software existieren. Des Weiteren sind solche Signatur-Systeme schwach gegen polymorphen Code. Dieser Code ändert sich mit jeder Ausführung und kann dementsprechend nicht durch eine Signatur abgefangen werden[21, S.7]. Dennoch ist das System in der Lage auch gegen hochentwickelte Malware, wie Stuxnet, Schutz zu bieten.

5 Ausblick

Abschließend lassen sich die Sicherheitsprobleme von Industrieanlagen auf folgende Aspekte zurückführen. Diese Systeme wurden so aufgebaut, dass sie mit möglichst geringer Wartung und Fehlerrate lange Einsatzbereit sind. Das ist Vorteilhaft, da sie sich über weite Strecken im Land ausbreiten können und es unmöglich wäre so ein System wirtschaftlich zu betreiben, wenn es regelmäßig Ausfälle geben würde. Viele dieser Systeme sind dementsprechend sehr alt und sind nicht in der Lage die entsprechenden kryptographischen Funktionen für eine sichere Verbindung zu berechnen. Deswegen brauchen Industrieanlagen maßgeschneiderte Sicherheitsmaßnahmen, die effektiv umgesetzt werden können. Selbstverständlich gilt, wenn es möglich ist verschlüsselte Kanäle, Authentifikation und Integritätssicherung zu benutzen, sollten diese Maßnahmen auch umgesetzt werden. Nur weil der Einsatz von herkömmlichen Sicherheitsmaßnahmen nicht möglich ist, heisst das nicht, dass das viele Angreifer nicht durch andere Mittel abgehalten werden können. So ungeschützt manche Industrieanlagen auch sind, so wichtig ist oftmals auch ihre Funktionalität. Ein Ausfall solcher Systeme kann katastrophale Auswirkungen haben und deshalb sollten Anstrengungen unternommen werden diese entsprechend zu Schützen.

Um herauszufinden, wie Angreifer versuchen Industrieanlagen anzugreifen, können Honeypots genutzt werden. Honeypots sind Systeme die offentsichtlich ungeschützt im Netzwerk zu finden sind und keine echte Funktionalität erfüllen. Daraus ergibt sich die Schlüsselhypothese von Honeypots, jeder Zugriff ist grundsätzlich verdächtig. Des Weiteren simulieren sie echte Systeme und speichern jegliche Art von Interaktion. High-Interaction Honeypots schränken einen Angreifer nicht ein, wie er mit einem System interagiert. Außerdem ist es vermutlich schwieriger diese mit einem fingerprinting aufzudecken. Im Umfeld von Industrieanlagen sind solche Honeypots aber nicht benutztbar, weil es zu kostenintensiv wäre so einen Honeypot umzusetzen. In dieser Arbeit wurde das HoneyPhy-Framework vorgestellt. Dieses Framework beschreibt low-Interaction Honeypots mit Interaktionsdauern und physikalischen Prozessen, zwei Merkmalen die bekannte State-of-the-Art Honeypots nicht implementieren und dementsprechend leichter aufgedeckt werden können. Obwohl ein Honeypot Angriffe feststellen kann und frühzeitig einen Administrator benachrichtigen kann, sind aktive Sicherheitsmaßnahmen zu bevorzugen. HosTaGe zeigt wie ein Honeypot auch aktiv zur Sicherheit von Industrieanlagen beitragen kann, indem es automatisch Signaturen für ein IDS erstellt und einfügt. Limitierungen von low-Interaction Honeypots können aber vermutlich nicht komplett ausgemerzt werden. So könnten auch Implementierung von HoneyPhy unzureichend sein. Das physikalische Modell könnte sich in Spezialfällen komisch verhalten oder ungenau umgesetzt worden sein. Auch das Signaturverfahren von HosTaGe und IDS könnten durch polymorphen Code ausgehebelt werden.

Auf Grund ihrer Flexibilität können Honeypots in vielen

Feldern eingesetzt werden. In [22] werden Honeytokens eingesetzt um gegen Erpresser-Malware (*Ransomware*) zu schützen. Erpresser-Malware verschlüsselt zunächst Daten und verlangt anschließend ein Lösegeld um die Daten wieder zu entschlüsseln[22, S.1]. Es wird vorgeschlagen einen Ordner als Honeytoken einzusetzen und zu überwachen, sodass frühzeitig erkannt wird, wenn Ransomware die Dateien verschlüsselt[22, S.3].

In [23] wird ein Honeypot vorgestellt der eng an Webapplikationen gekoppelt ist, ähnlich wie in [21]. Dieser Honeypot lernt, dabei Antworten von den anderen Webapplikationen aus ihren Logdateien[23, S.4]. Ein Problem von diesen Honeypots ist allerdings, dass sie damit die Schlüsselhypothese brechen und auch echte Benutzer auf diesen System landen könnten[23, S.6]. In [21] ist dies kein Problem, weil der Honeypot eine angepasste Version der Applikation ist. Hingegen ist der Honeypot aus [23] in seinen Antworten limitiert. Die Honeypots in [21], genannt *Shadow Honeypot* sind interessant, da sie automatisch generiert werden können[21, S.5]. Es wird versucht, illegale Speicherzugriffe durch eine spezielle Allozierung zu erkennen und dadurch Malware aufzudecken[21, S.5]. Allerdings ist das sehr aufwändig, es müsste ein sichere Operation zur Speicher Allozierung geschrieben werden, im Kernel ein Systemcall hinzugefügt werden und dann die Applikation angepasst werden[21, S.5]. Der letzte Schritt geht durch Werkzeuge automatisch, dennoch würde eine hohe Expertise benötigt. Es könnte auch sein, dass nicht die Applikation selber von Malware angegriffen wird, sondern eine Schwachstelle in einer *shared library*, diese müssten auch angepasst werden. Eventuell ist der Aufwand dann zu Hoch. Dennoch könnten solche Honeypots effektiver sein, weil der Interaktionsgrad für den Angreifer nicht unterscheidbar zur echten Applikation ist.

Honeypots können also zur Sicherheit von Netzen auf verschiedene Art und Weise beitragen. Nachteile sind, dass sie ineffektiv sein können, wenn der Angreifer erkennt ‚dass es sich um einen Honeypot handelt. Ein Angreifer könnte sogar Angriffe so aussehen lassen, dass interne Systeme den Honeypot angreifen, um die Administrator so zu verwirren. Währendessen können dann echte Angriffe auf andere Systeme gestartet werden[8, S.70]. High-Interaction Honeypots können auch ein gewisses Risiko mit ins Netzwerk bringen, weil sie wirklich ausgenutzt werden können und anschließend andere Systeme bedrohen[8, S.70]. Aktive Sicherheitsmaßnahmen, wie Firewalls und IDS, werden also durch Honeypots nicht ersetzt. Sie erhöhen die Sicherheit, indem sie mit ihnen zusammenarbeitet[8, S.71].

Eine Untersuchung von Shodan und wie effektiv der HoneyScore ist interessant. Beim vorherigen Vergleich zwischen Conpot und HosTaGe wurde lediglich Conpot als Honeypot identifiziert. Es wäre interessant, welche Honeypots von Shodan als solche erkannt werden und welche nicht. Ich würde zunächst vermuten, dass Shodan Conpot identifizieren kann, weil er erstens eine State-of-the-Art Implementierung eines ICS-Honeypots ist und zweitens weil er schon deutlich älter ist. Dementsprechend könnte Shodans Identifizierungs-Werkzeug zu sehr auf Conpot abgestimmt worden sein und andere ICS-Honeypots deswegen nicht erkennen. Ein Vergleich zwischen nur zwei Honeypots ist aber nicht ausreichend um diese Aussage zu treffen. Über die Webseite [20] ist es zwar möglich einzelne Systeme zu überprüfen, allerdings kann ein kostenloser Account mit Suchergebnissen nicht auf diese Punktzahl zugreifen. Trotzdem ist so eine Funktion auch hilfreich, wenn ein low-Interaction Honeypot aufgesetzt werden soll. Es kann damit zumindest ein erstes Maß für die Güte der Implementierung bieten.

QUELLENVERZEICHNIS

[1] K. A. Stouffer, J. A. Falco und K. A. Scarfone, "Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc)", 2011.

[2] "Projekt honeytrain - aufbau, durchführung und ergebnisse", Koramis GmbH, Techn. Ber., 2015. Adresse: https://www.sophos-events.com/honeytrain/form.cfm.

[3] P. Simões, T. Cruz, J. Gomes und E. Monteiro, "On the use of honeypots for detecting cyber attacks on industrial control networks", in *Proc. 12th Eur. Conf. Inform. Warfare Secur. ECIW 2013*, 2013.

[4] J. C. Matherly, "Shodan the computer search engine", *Available at [Online]: Http://www. shodanhq. com/help*, 2009.

[5] E. P. Leverett, "Quantitatively assessing and visualising industrial system attack surfaces", *University of Cambridge, Darwin College*, Bd. 7, 2011.

[6] S. Sharma, "Detection and analysis of network & application layer attacks using maya honeypot", in *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*, IEEE, 2016, S. 259–262.

[7] (). Conpot. Stand: 01.11.2016, Adresse: http://conpot.org/.

[8] L. Spitzner, *Honeypots: Tracking hackers*. Addison-Wesley Reading, 2003, Bd. 1.

[9] T. Grudziecki, T. Grudziecki, P. Jacewicz, L. Juszczyk, P. Kijewski und P. Pawlinski, "Proactive detection of security incidents", 2012. Adresse: https://www.enisa.europa.eu/publications/proactive-detection-of-security-incidents-II-honeypots.

[10] T. Holz und F. Raynal, "Detecting honeypots and other suspicious environments", in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, IEEE, 2005, S. 29–36.

[11] (). Os detection. Stand: 04.01.2016, Adresse: https://nmap.org/book/man-os-detection.html.

[12] S. Litchfield, D. Formby, J. Rogers, S. Meliopoulos und R. Beyah, "Rethinking the honeypot for cyber-physical systems", *IEEE Internet Computing*, Bd. 20, Nr. 5, S. 9–17, 2016.

[13] (). Newton's law of cooling. Stand: 04.1.2016, Adresse: https://en.wikipedia.org/wiki/Newton's_law_of_cooling.

[14] D. Formby, P. Srinivasan, A. Leonard, J. Rogers und R. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems", in *NDSS, Feb*, 2016.

[15] S. Choi, B. Kim, G. J. Cokkinides und A. S. Meliopoulos, "Feasibility study: Autonomous state estimation in distribution systems", *IEEE Transactions on Power Systems*, Bd. 26, Nr. 4, S. 2109–2117, 2011.

[16] D. Grimsman, V. Chetty, N. Woodbury, E. Vaziripour, S. Roy, D. Zappala und S. Warnick, "A case study of a systematic attack design method for critical infrastructure cyber-physical systems", 2016.

[17] E. Vasilomanolakis, S. Srinivasa, C. G. Cordero und M. Mühlhäuser, "Multi-stage attack detection and signature generation with ics honeypots", 2016.

[18] E. Vasilomanolakis, S. Srinivasa und M. Mühlhäuser, "Did you really hack a nuclear power plant? an industrial control mobile honeypot", in *Communications and Network Security (CNS), 2015 IEEE Conference on*, IEEE, 2015, S. 729–730.

[19] J. G. Göbel, "Amun: A python honeypot", 2009.

[20] (). Honeypot or not? Stand: 04.01.2016, Adresse: https://honeyscore.shodan.io/.

[21] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, M. Polychronakis, A. D. Keromytis und E. P. Markatos, "Shadow honeypots", *International Journal of Computer and Network Security*, Bd. 2, Nr. 9, S. 1–16, 2010.

[22] C. Moore, "Detecting ransomware with honeypot techniques", in *Cybersecurity and Cyberforensics Conference (CCC), 2016*, IEEE, 2016, S. 77–81.

[23] S. Saito, S. Torii, K. Yoshioka und T. Matsumoto, "Wamber: Defending web sites on hosting services with self-learning honeypots", in *Information Security (AsiaJCIS), 2016 11th Asia Joint Conference on*, IEEE, 2016, S. 60–66.