HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Thomas Ringhof

Implementierung und Vergleich verschiedener Honeypotsysteme anhand von Angriffsszenarien

2020

Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

Implementierung und Vergleich verschiedener Honeypotsysteme anhand von Angriffsszenarien

Autor:
Thomas Ringhof

Studiengang:
Angewandte Informatik

Seminargruppe:
IF15wI-B

Matrikelnummer:
40336

Erstprüfer:
Prof. Dr. Dirk Pawlaszczyk

Zweitprüfer:
Stefan Schildbach

Mittweida, August 2020

Bibliografische Angaben

Ringhof, Thomas: Implementierung und Vergleich verschiedener Honeypotsysteme anhand von Angriffsszenarien, 55 Seiten, 22 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2020

Referat

Die vorliegende Bachelorarbeit vergleicht die zwei Honeypotsysteme T-Pot und Modern Honey Network miteinander. Hierfür zeigen Angriffsszenarien gegen die Honeypotsysteme, wie Angreifer diese wahrnehmen würden und wie die Honeypots die Angriffe protokollieren. Die ersten Kapitel dienen als Einführung in die Thematik der Honeypots und erläutern die Problemstellung und Vorgehensweise. Kapitel 4 und 5 beleuchten den Entwurf und die Durchführung der Angriffsszenarien. In Kapitel 7 wird der Vergleich anhand selbst aufgestellter Kriterien durchgeführt. Das Honeypotsystem T-Pot hat in diesem Vergleich knapp besser abgeschnitten. Jedoch lässt sich Schlussfolgern, das je nach individuellen Zielen und Einsatzzwecken beide Honeypotsysteme mit ihren Stärken überzeugen können.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

1 Einführung

1.1 Motivation

Im Laufe meines Studiums habe ich viele Themengebiete der IT-Sicherheit kennengelernt. Jene des Moduls „Abwehr von IT-Angriffen“ im 5. Semester haben mich besonders angesprochen. Mein Praxissemester absolvierte ich daher bei der IT-SEAL GmbH, um weiterführende Einblicke in diesem Bereich erhalten und praktische Erfahrungen zu sammeln.

Für die Entwicklung sinnvoller Strategien und Konzepte zur Abwehr gegen IT-Angriffe ist es aus meiner Sicht notwendig, sich in die Rolle des Angreifers zu versetzen. Honeypots und Honeypotsysteme ermöglichen es, Angreifern eine Falle zu stellen und bieten damit einzigartige Einblicke in die Vorgehensweise von Angreifern sowie Malware [1]. Dieses Konzept finde ich äußerst spannend und hat mich zur Wahl dieser Thematik für diese Bachelorarbeit bewogen.

1.2 Aufgabenstellung

Ziel dieser Bachelorarbeit soll es sein, Honeypotsysteme miteinander zu vergleichen. Hierbei werden zunächst die Grundlagen zum Thema Honeypots behandelt, welche Arten von Honeypots zum Stand der Bachelorarbeit entwickelt worden sind und wie sich ein Honeypotsystem von anderen IT-Sicherheitskonzepten abhebt. Anschließend werden in einer Laborumgebung verschiedene Honeypotsysteme installiert und konfiguriert, deren Vor- und Nachteile abgewägt. Eigens entwickelte Angriffsszenarien prüfen im Anschluss die Funktionalität und Effektivität der beiden Honeypotsysteme.

2 Grundlagen zu Honeypots

Im ersten Kapitel wird zunächst die Definition von Honeypots erläutert, wie Honeypots zu kategorisieren sind und wie diese sich in Netzwerken platzieren lassen, um unterschiedliche Ziele zu erfüllen.

2.1 Definition Honeypot

Zunächst soll erläutert werden, um was es sich bei einem Honeypot handelt. Unter einem „Honeypot“, auf Deutsch „Honigtopf“, im weiteren „Honeypot“ genannt, versteht man in der Netzwerksicherheit ein System, welches durch bewusst implementierte Sicherheitslücken Angreifer dazu bewegen soll, dieses zu befallen. Ein Angreifer kann hier automatisierte Malware sein oder ein gezielter Hackerangriff. Je nach Art des Honeypots werden Aktivitäten aufgezeichnet und geben einzigartige Einblicke in die Vorgehensweise von Hackern und Malware. Ein Honeypotsystem ist somit eine Art Lockvogel, dessen Wert darin liegt, dass es attackiert, untersucht und kompromittiert wird [2]. Da ein Honeypotsystem mit keinen Produktionsdaten oder ähnlichem arbeitet, ist jeder versuchte Zugriff oder Netzwerkscan als Anomalie zu betrachten und kann Hinweise auf mögliche Angriffe auf das Netzwerk preisgeben und zeigt, wie der Angreifer vorgeht. [2]

2.2 Vergleich von Honeypots zu anderen IT-Sicherheitskonzepten

Bevor im weiteren Verlauf auf Honeypots eingegangen wird, soll zunächst elaboriert werden, wie und wo Honeypots im Vergleich zu anderen Sicherheitskonzepten der IT-Sicherheit einzordnen sind und welche Rolle diese dabei spielen können. Nach Bruce Schneier [3] lassen sich Sicherheitskonzepte in drei Kategorien einteilen:

- **Prävention:** Beschreibt die Umsetzung von Maßnahmen, welche präventiv und vorbeugend gegen Angriffe und Malware agieren.
- **Detektion:** Ist der aktive Prozess des Untersuchens laufender Systeme um Anomalien zu entdecken, die durch Angriffe oder Malware ausgelöst wurden.
- **Reaktion:** Alle Maßnahmen welche umgehend nach der Feststellung eines Angriffes durchgeführt werden fallen in diese Kategorie. Ergebnisse aus reaktiven Sicherheitskonzepten können die anderen beiden Kategorien sinnvoll ergänzen, um ein sich ständig weiterentwickelndes Sicherheitskonzept aufzubauen. In diese Kategorie fallen oft forensische Analysen von bereits kompromittierten Systemen.

Die unterschiedlichen Sicherheitskonzepte erfüllen die Ziele dieser drei Kategorien unterschiedlich stark. Um sich ein gutes Bild eines Sicherheitskonzepts zu verschaffen, hilft es daher, es hinsichtlich dieser Kategorien zu analysieren. Präventive Sicherheitskonzepte bilden zum Beispiel Firewalls ab, welche den Netzverkehr mit statischen Regeln überwachen und kontrollieren. Zur Kategorie der Detektion gehören Intrusion-Detection-Systeme(IDS),und deren Erweiterung, Intrusion-Prevention-Systeme(IPS). IDS analysieren die gesamte Struktur von Paketen im Netzverkehr, also den Header und den Payload eines Pakets. Finden sich Signaturen welche auf bekannte Malware deutet, können verschiedene Ereignisse ausgelöst werden und Warnungen ausgegeben werden. IPS können darüber hinaus dynamisch neue Regeln hinzufügen und bilden damit eine Überschneidung zwischen Detektion und Reaktion. Antivirensysteme bedienen sich ebenfalls eines Signaturverfahrens, agieren allerdings auf Dateiebene und versuchen „bösartige“ Dateien zu erkennen und in eine sogenannte Quarantäne zu verschieben, damit diese nicht weiter ausgeführt werden können. [6]

Honeypots hingegen erfüllen nahezu keine Maßnahmen zur Prävention oder Detektion von Malware. Im Falle der Prävention ist genau genommen der Gegenteil der Fall: Durch das bewusste Präsentieren von Sicherheitslücken laden diese eher dazu ein, einen Angriff zu wagen. Auch im Bereich der Detektion von Malware sind Honeypots als eher schwach einzustufen. Lediglich die Tatsache, dass jede Interaktion mit einem Honeypot direkt als Anomalie oder Angriff gewertet werden kann, hilft im Sinne der Detektion. Anders verhält es sich im Bereich der Reaktion. Angriffe auf Honeypotsysteme lassen sich deutlich einfacher analysieren als Produktionssysteme, da jeder Netzwerkverkehr ein- oder ausgehend vom Honeypotsystem als verdächtig eingestuft werden kann. Produktionssysteme erzeugen einen konstanten Strom von Netzwerkverkehr, was die Unterscheidung zwischen normalen und anomalen Verbindungen deutlich schwieriger gestaltet. Weiterhin ist es möglich, Honeypotsysteme für unbestimmte Zeit vom Netz zu trennen, um forensische Analysen durchzuführen. Ergebnisse aus solchen Analysen helfen dabei, das eigentliche Produktionssystem weiter abzuhärten und das Netzwerk zu schützen. [6]

Somit lässt sich abschließend zusammenfassen, dass Honeypots reaktive Maßnahmen implementieren, während IDS, IPS und Antivirensysteme größtenteils detektive Maßnahmen bereitstellen und Firewallsoftware präventive Maßnahmen bereitstellt. [6]

2.3 Taxonomie von Honeypots

Honeypotsysteme lassen sich auf unterschiedliche Weise kategorisieren. In diesem Unterkapitel werden die gängigsten Kategorisierungsmethoden vorgestellt und erläutert. Darunter fallen:

- Einsatzgebiet
- Interaktionsgrad
- Interaktionsfluss
- Physikalität

2.3.1 Kategorisierung nach Einsatzgebiet

Eine der ersten Kategorisierungen von Honeypotsoftware erfolgte durch Lance Spitzner und Marty Roesch, dem Entwickler des IDS Snort. Dieser teilt Honeypots in zwei Kategorien ein. [2]

- Produktionsorientiert
- Forschungsorientiert

Ein produktionsorientierter Honeypot wird, wie der Name suggeriert, in einem Produktionsumfeld eines Unternehmens installiert, um die Sicherheitsmaßnahmen zu erweitern. Der Honeypot ist isoliert vom Produktionsnetz und bietet dem Unternehmen die Möglichkeit, im Falle eines Angriffes schnell zu agieren und das Produktionsnetz zu schützen. Charakteristisch sind diese Honeypots schnell in das Netzwerk integriert und bedürfen geringer administrativer Pflege. Dafür bieten sie allerdings nur eine geringe Quantität und Qualität der gesammelten Informationen. [2]

Im Gegenzug dazu bieten forschungsorientierte Honeypots eine sehr ausführliche Bandbreite an Informationen über die Angriffe, benötigen aber einen höheren administrativen Verwaltungsaufwand und sind höchst spezialisiert aufgesetzt. Das Ziel dieser Honeypots ist es, eine Grundlage für die Forschung zu dienen. Um eine Bedrohung zu analysieren, muss diese zunächst ermittelt werden. Hierbei spielen die Honeypots eine große Rolle, denn nur Honeypots bieten die Möglichkeit, sogenannte Zero-Day-Exploits, also Sicherheitslücken über die bisher keinerlei Information vorliegt, zu untersuchen. Weiterhin ist es für Forscher auch interessant zu verfolgen, was Hacker oder Malware mit dem Honeypotsystem machen, nachdem dieses kompromittiert wurde. Es könnte beispielsweise analysiert werden, mit welchen Systemen der Server Verbindungen aufbaut oder welche Dateien verändert werden. Da diese Form der Honeypots darauf ausgelegt sind, der Forschung zu dienen, tragen sie nur wenig zur Sicherheit des Netzwerkes bei. [2]

2.3.2 Kategorisierung nach Interaktionsgrad

Moderne Betrachtungen eines Honeypots erfolgen häufig aus der Sicht des Interaktionsgrades, bei dem sie in Low-Interaction und High-Interaction Honeypots eingeteilt werden(LIHP und HIHP). Honeypots, welche nur Services in kleiner Anzahl emulieren - wie SSH - und geben dem Angreifer nur wenig Interaktionsmöglichkeit mit dem Honeypotsystem. Diese Honeypots werden häufig in Produktionsnetzen eingesetzt, da sie leicht zu installieren und zu verwalten sind. Allerdings ist der Informationsgehalt begrenzt, häufige Login versuche oder Portscans lassen sich damit jedoch bereits aufzeichnen. High-Interaction Honeypots emulieren ein komplettes System und sind deutlich aufwendiger zu installieren. Das Risiko, dass ein Angreifer das System komplett kompromittiert, ist damit im Vergleich zu Low-Interaction Honeypots sehr hoch und muss beachtet werden. Im Gegenzug ermöglichen High-Interaction Honeypots einzigartige Einblicke in die Vorgehensweise eines Angreifers oder automatisierter Malware. Es kann auch beispielsweise mitverfolgt werden, wenn der Angreifer einen Zero-Day Exploit ausnutzt. Da die Analyse von High-Interaction Honeypotdaten ebenfalls sehr komplex sein kann, werden diese häufiger in wissenschaftlichen Umgebungen genutzt. [6]

2.3.3 Kategorisierung nach Interaktionsfluss

Honeypots müssen nicht zwangsläufig auf einem Server installiert werden. Es existieren einige Honeypots, welche ebenfalls auf Clients installiert werden können. Daher lässt sich auch zwischen serverseitigen Honeypots und clientseitigen Honeypots unterscheiden. Die in den vorigen Unterkapiteln angesprochenen Honeypots wären nach dieser Kategorisierungsart serverseitige Honeypots, da sie passiv sind, bis ein Angreifer mit dem Server interagiert. Clientseitige Honeypots dagegen suchen aktiv nach möglicher Malware, indem sie zufällige Seiten aufrufen und verdächtiges Verhalten aufzeichnen. Diese Unterscheidung zwischen server- und clientseitigen Honeypots kann ebenfalls in Low- oder High-Interaction aufgeteilt werden. So emulieren Low-Interaction Honeypots clientseitig nur einige Komponenten eines Clients, während High-Interaction Honeypots komplette Browser emulieren, um Anfragen an Internetseiten zu stellen. [6]

2.3.4 Kategorisierung nach Physikalität

Die letzte, eher selten benutzte Kategoriesierungsart, geht nach der Physikalität des Honeypots. So können Honeypots entweder auf virtuellen Maschinen installiert und gewartet werden, oder auf einem physikalisch vorhandenem System. Häufig handelt es sich bei virtuellen Honeypots ebenfalls um High-Interaction Honeypots. [6]

2.4 Strategien zur Verteilung von Honeypots in Netzwerken

Es gibt unterschiedliche Strategien, um Honeypots in einem Netzwerk zu platzieren. Ebenfalls kann es gewollt sein, mehrere Honeypots in einem Netzwerk einzubinden, damit diese wie Sensoren unterschiedliche Bereiche des Netzwerks abdecken. In diesem Unterkapitel sollen diese Strategien in einem beispielhaften Unternehmensnetzwerk vorgestellt werden.

2.4.1 Honeypot außerhalb einer Firewall

In dieser Variante befindet sich der Honeypot außerhalb der Firewall und ist vollständig vom Internet extern erreichbar. Diese Variante ist gängig, da mit ihr auch das Potential für externe Angriffe am höchsten ist, und somit können hier die meisten Daten gesammelt werden. Ein beispielhaftes Netzwerk wird in Abbildung 2.1 gezeigt. Hieraus bilden sich folgende Vor- und Nachteile: [12]

Vorteile:

- Der Honeypot bietet selbst bei Kompromittierung des Systems keine Angriffsmöglichkeiten auf das eigentliche Unternehmensnetzwerk.
- Externe Angriffe lenkt der Honeypot auf sich, die Firewall wird weniger belastet.

Nachteile:

- Ein Honeypot in dieser Position wird viel Netzwerkverkehr ausgesetzt sein, und die Chance auf False-Positive Zugriffe durch Webcrawler wie Shodan [13] ist hoch.
- Da der Honeypot vom internen Netz abgeschnitten ist, können keine Angreifer aus dem internen Netz gefangen werden.

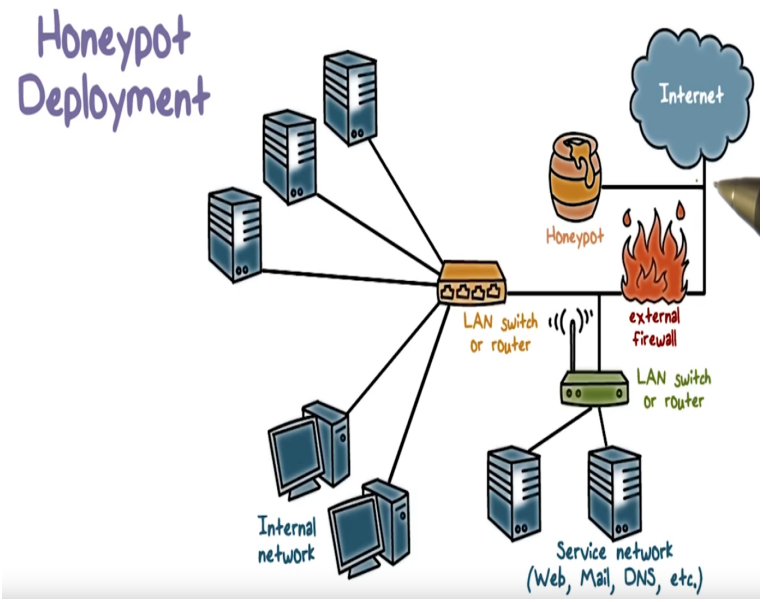


Abbildung 2.1: Platzierung eines Honeypots außerhalb der Firewall Quelle: Ausschnitt eines Webvideos des Kanals „Udacity“. Titel: „Honeypot Deployment“ [12]

2.4.2 Honeypot als Teil der DMZ

Wird der Honeypot in der demilitarisierten Zone (DMZ) des Netzwerkes platziert, in dem sich zum Beispiel auch öffentlich zugängliche Server befinden, wie zum Beispiel Mail- oder Webserver, kann dieser nicht ohne weiteres sinnvoll Daten sammeln. Da sich die DMZ hinter der eigentlichen Firewall befindet und diese den Zugang zu den Servern reguliert, wäre eine Anpassung der Firewall-Regeln notwendig, damit überhaupt extern auf den Honeypot zugegriffen werden kann.

Dies erhöht das Sicherheitsrisiko signifikant, da sich der Angreifer im Falle einer Kompromittierung des Systems in einem sensiblen Netzwerkbereich befindet, wie in Abbildung 2.2 gezeigt wird. [12]

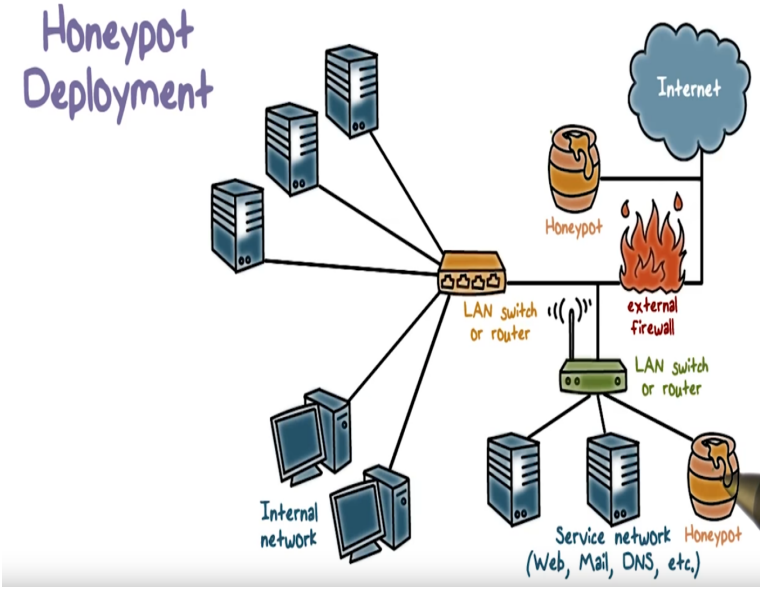


Abbildung 2.2: Platzierung eines Honeypots innerhalb der DMZ Quelle: Ausschnitt eines Webvideos des Kanals „Udacity“. Titel: „Honeypot Deployment“ [12]

2.4.3 Honeypot als Teil des internen Netzwerks

Der Honeypot kann ebenso im internen Netzwerk platziert werden, also in jenem Netzwerkbereich, in dem auch die Clientcomputer der Mitarbeiter und interne Server erreichbar sind. [12]

Vorteile:

- Der Honeypot ist in der Lage, interne Angriffe zu protokollieren.
- Sollte es Fehler in der Konfiguration der Firewall geben, kann ein Honeypot diese aufdecken, sollte ein externes Gerät sich mit diesem verbinden können.

Nachteile:

- Sollte der Honeypot kompromittiert werden, stellt dies ebenfalls ein hohes Sicherheitsrisiko dar.

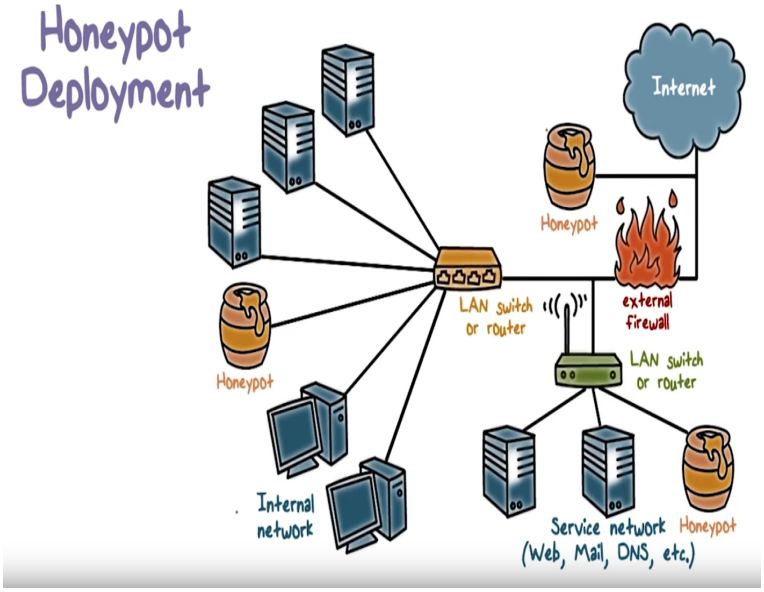


Abbildung 2.3: Platzierung eines Honeypots innerhalb der Firewall Quelle: Ausschnitt eines Webvideos des Kanals „Udacity“. Titel: „Honeypot Deployment“ [12]

Eine sinnvolle Kombination dieser Strategien wäre wohl ein Honeypot im internen Netzwerk, um interne Angriffe detektieren zu können, sowie ein externer Honeypot außerhalb der Firewall. Somit würde man von beiden Varianten die Vorteile ausnutzen können und die Nachteile gering halten. Je nach Honeypotsystem muss allerdings auch der Wartungsaufwand, welcher sich mit jedem weiteren System erhöht, mit einbezogen werden.

2.5 Trendentwicklung von Open-Source Honeypotsystemen

Während der Recherche an dieser Arbeit wurde eine gewisse Trendentwicklung von Honeypotsystemen über die letzten Jahre bemerkbar. Im Verlauf des letzten Jahrzehnts schien die Zahl der einsetzbaren und noch gepflegten Open-Source High-Interaction Honeypotsysteme abzunehmen, während Open-Source Low-Interaction Honeypotsysteme noch sehr weit verbreitet sind.

Das Whitepaper „A Survey on Honeypot Software and Data Analysis“ [6] aus dem Jahre 2016 bietet eine ausführliche Übersicht von Low- und High-Interaction Honeypotsystemen. Von den vorgestellten High-Interaction Honeypotsystemen wird keines mehr aktiv gepflegt. Argos [7], Sebek [8], Honeywall [9] und HIHAT [10] werden seit einigen Jahren nicht mehr aktualisiert oder können von ihren ursprünglichen Quellen nicht mehr heruntergeladen werden. Das Internetportal Honeynet.org [11], welches sich der Entwicklung von Honeypotsoftware widmet und in diesem Bereich erforscht, bietet bei den aktuellen

Projekten keine High-Interaction Honeypotsysteme an.

Diese Entwicklung bedeutet allerdings nicht, dass der Bedarf an High-Interaction Honeypotsystemen gesunken ist oder das Interesse der Forschung sich von diesem Bereich abgewandt hat. Wie bereits bei der Kategorisierung von Honeypotsystemen angesprochen, sind High-Interaction Honeypots deutlich komplexer aufzusetzen und zu administrieren als Low-Interaction Honeypots und decken im Bereich der Forschung eine Nische ab. Die Komplexität von High-Interaction Honeypots ergibt sich durch die intensive Planung und Vorbereitung einer Implementierung eines solchen Systems, da mögliche Fehler in der Konfiguration bereits fatale Folgen für das gesamte Netzwerk haben können. Ein High-Interaction Honeypot ist, anders als bei einem Low-Interaction Honeypot, kein einzelner Server oder Netzwerkdienst, sondern beschreibt ein komplettes, abgeschlossenes System.

Daraus ergibt sich, dass der High-Interaction Honeypot für das Netzwerk, in dem er eingesetzt werden soll, maßgeschneidert und der Situation entsprechend implementiert werden muss. Ein bereits fertiges System könnte hierfür bereits zu unflexibel sein. Im Jahre 2019 wurde von dem Unternehmen „Trend Micro“ ein derartiges High-Interaction Honeypotsystem aufgesetzt, und dieses über Monate andauernde Projekt verdeutlicht, wie hoch der Zeitaufwand sowie das Investment in eine gute Planung und Durchführung ist [4]. Hier wurde nicht auf ein fertiges System gesetzt, sondern es wurde vielmehr ein eigenes, der Situation entsprechendes System entworfen und genutzt. Ein weiteres Argument könnte sein, das Open-Source High-Interaction Honeypotsysteme auch möglichen Angreifern zur Verfügung stehen. Dies könnte Angreifern die Möglichkeit geben, Malware zu entwerfen, welche eben diese Honeypotsysteme versucht zu erkennen und sich im Falle einer Untersuchung anders zu verhalten.

Eine derartige Entwicklung aus der Sicht der Angreifer hat es bereits im Rahmen von virtuellen Maschinen gegeben, bei dem Malware versucht zu erkennen, ob es eine virtuelle Umgebung und damit in einem Testsystem befallen hat, und sich schließlich anders verhält oder gar selbst löscht [5].

3 Problemstellung und Vorgehensweise

In dem vorherigen Kapitel wurden die Grundlagen zu Honeypots dargelegt und erläutert. Wie in der Aufgabenstellung zu Beginn erwähnt, werden in dieser Arbeit zwei Honeypotsysteme miteinander verglichen. Außerdem wird analysiert, wie Netzwerkanalysten - so genannte Penetrationstester oder gar Hacker - vorgehen würden, wenn diese ein System auf Schwachstellen prüfen bzw. angreifen.

Für die Entwicklung passender Angriffsszenarien zum Testen der Honeypots wird daher ebenfalls ein System aufgesetzt. In diesem Kapitel wird zunächst auf die Konfiguration und Installation der Laborumgebung eingegangen, um anschließend auf die zwei zu vergleichenden Honeypotsysteme einzugehen.

Die beiden Honeypotsysteme, sowie der Client werden in einer virtuellen Umgebung installiert, welche durch einen Proxmox VE Server [17] bereitgestellt wird. Über die Weboberfläche des Proxmox-Servers können die virtuellen Maschinen bedient und gesteuert werden. Der Aufbau des Netzwerks wird im Kapitel 3.2.3 erläutert.

3.1 Wahl der Honeypotsysteme für das Experiment

High-Interaction Honeypots wären am geeignetsten für das Experiment, da diese durch ihre vielseitige Interaktionsweise am meisten Daten zu Angriffen erzeugen können. Wie bereits in der Sektion 2.5 erläutert wurde, waren die gängisten Open-Source High-Interaction Honeypots nicht mehr verfügbar oder sind nur unzureichend gepflegt. Daher wird der Fokus auf Low-Interaction Honeypots zu gelegt.

Da einzelne Low-Interaction Honeypots nur wenig Vergleichsmöglichkeiten bieten, wurde auf Honeypotsysteme oder Frameworks gesetzt, welche mehrere Low-Interaction Honeypots miteinander verbinden oder gleich mehrere einsetzen. Der Vergleich bezieht sich daher auch auf die Honeypotsysteme und nicht auf die einzelne Honeypotsoftware. Außerdem wird untersucht, wie die geplanten Angriffsszenarien protokolliert werden können oder welche Möglichkeiten bestehen, Messungen zu tätigen.

Als zu vergleichende Honeypotsysteme wurde das Open-Source Projekt „Modern Honey Network“ [15], sowie das Open-Source Honeypotsystem der Deutschen Telekom AG, „T-POT“ [14] ausgewählt. Beide Honeypotsysteme haben den Vorteil, ausreichend Analysewerkzeuge der gesammelten Daten bereitzustellen. Weiterhin bieten beide Honeypotsysteme die Möglichkeit, mehrere Honeypots gleichzeitig zu installieren. Ausschlaggebend war ebenfalls, dass es sich bei beiden Systemen um Open-Source Software handelt und sich daher beide frei nutzen lassen.

3.2 Installation und Konfiguration der Laborumgebung

In diesem Kapitel wird die genutzte Hardware, Software und der Aufbau des Netzwerks der Testumgebung erläutert. Am Schluss des Kapitels werden die beiden Honeypotsysteme T-Pot und MHN, sowie der Kali Linux [16] Client grundlegend erklärt. Der Client, von welchem aus die Angriffsszenarien auf die Honeypots durchgeführt werden, wird mit dem Betriebssystem Kali Linux ausgestattet.

3.2.1 Hardwarespezifikation

Die folgenden Tabellen bieten eine Übersicht der eingesetzten Hardware. Die virtuellen Machinen teilen sich die Ressourcen des Proxmox Servers. Die angegebenen Hardwarespezifikationen bei den virtuellen Machinen geben an, wieviel Ressourcen den virtuellen Machinen zugeteilt wurde. Da es nicht nötig ist, alle virtuellen Machinen zum gleichen Zeitpunkt zu starten, muss nicht darauf geachtet werden, das die Obergrenze an Ressourcen durch alle virtuellen Machinen überschritten wird. Da keine großen Datenmengen erzeugt werden, ist besonders großer Festplattenspeicher ist nicht notwendig. Die Ressourcen für die virtuellen Machinen wurden derart gewählt, dass diese die minimalen Anforderungen der Software und der Betriebssysteme ausreichend decken.

Proxmox VE (Mac mini Server)	
Prozessor	Core i7 (I7-2635QM)
RAM	8192 GB
Festplatte	128GB SSD
Betriebssystem	Proxmox VE 6.2-6

Tabelle 3.1: Proxmox VE Spezifikation

T-Pot VM	
Prozessor	Core i7(2 Kerne)
RAM	4096 GB
Festplatte	32GB SSD
Betriebssystem	Debian 10.5

Tabelle 3.2: T-Pot VM Spezifikation

MHN Server VM	
Prozessor	Core i7(2 Kerne)
RAM	2048 GB
Festplatte	16GB SSD
Betriebssystem	Ubuntu 18.04.4

Tabelle 3.3: MHN Server VM Spezifikation

MHN Client VM	
Prozessor	Core i7(2 Kerne)
RAM	2048 GB
Festplatte	16GB SSD
Betriebssystem	Ubuntu 18.04.4

Tabelle 3.4: MHN Client VM Spezifikation

Kali Linux VM	
Prozessor	Core i7(2 Kerne)
RAM	2048 GB
Festplatte	32GB SSD
Betriebssystem	Kali Linux 2020.2

Tabelle 3.5: Kali Linux VM Spezifikation

3.2.2 Softwarespezifikation

Die nachfolgenden Tabellen auf der nächsten Seite geben einen Überblick der eingesetzten Software auf den virtuellen Machinen. Die zweite Spalte beschreibt die betroffenen Protokolle oder Services.

T-Pot VM	
Honeypotsoftware	**Service/Protokoll**
ADBHoney	Android Debug Bridge
Cisco ASA	Cisco Adaptive Security Appliance
Citrix Honeypot	Citrix ADC
Conpot	ICS/SCADA
Cowrie	SSH/Telnet
Dicompot	DICOM
Dionaea	FTP, HTTP, BlackHole, EPMAP, Memache, Mirror, MongoDB, MQTT, MSSQL, MySQL, nfq, PPTP, SIP(VoIP), SMB, TFTP, UPNP
Elasticpot	Elasticsearch
Heralding	FTP, Telnet, SSH, RDP, HTTP, HTTPS, POP3, POP3S, IMAP, IMAPS, SMTP, VNC, PostgreSQL, socks5
HoneySAP	SAP
Honeytrap	TCP/UDP Services
Mailoney	SMTP
Medpot	HL7/FHIR
RDPY	RDP
Snare/Tanner	Web Applikationen

Tabelle 3.6: Softwarespezifikation T-Pot VM.

Weitere installierte Softwares, welche die Bedienung von T-Pot ergänzen, sind wie folgt:

- Cockpit
- Cyberchef
- Docker
- Elasticsearch & Elasticsearch Head
- Logstash
- Kibana
- Spiderfoot
- IDS Suricata

Modern Honey Network VM (MHN)	
Honeypotsoftware	**Service/Protokoll**
Cowrie	SSH/Telnet
Dionaea	FTP, HTTP, BlackHole, EPMAP, Memache, Mirror, MongoDB, MQTT, MSSQL, MySQL, nfq, PPTP, SIP(VoIP), SMB, TFTP, UPNP
Glastopf	Web Applikationen

Tabelle 3.7: Softwarespezifikation MHN VM.

Darüber hinaus nutzt MHN noch das IDS Snort.

3.2.3 Netzwerkkonfiguration

Die Laborumgebung wurde in einem privaten Heimnetzwerk aufgebaut. Der Proxmox VE Server diente als Virtualisierungsplattform. Die einzelnen Honeypotserver waren virtuelle Machinen, welche mit Proxmox erzeugt wurden. Jede dieser virtuellen Machinen, sowie der Proxmox Server, erhielten eine eigene IP-Adresse, welche vom Router per Dnyamic Host Configuration Protocol(DHCP) zugeteilt wurde. Über diese IP-Adressen lassen sich die virtuellen Machinen und der Proxmox Server erreichen und ansprechen. Die Grafik 3.1 zeigt den Aufbau des Netzwerkes.

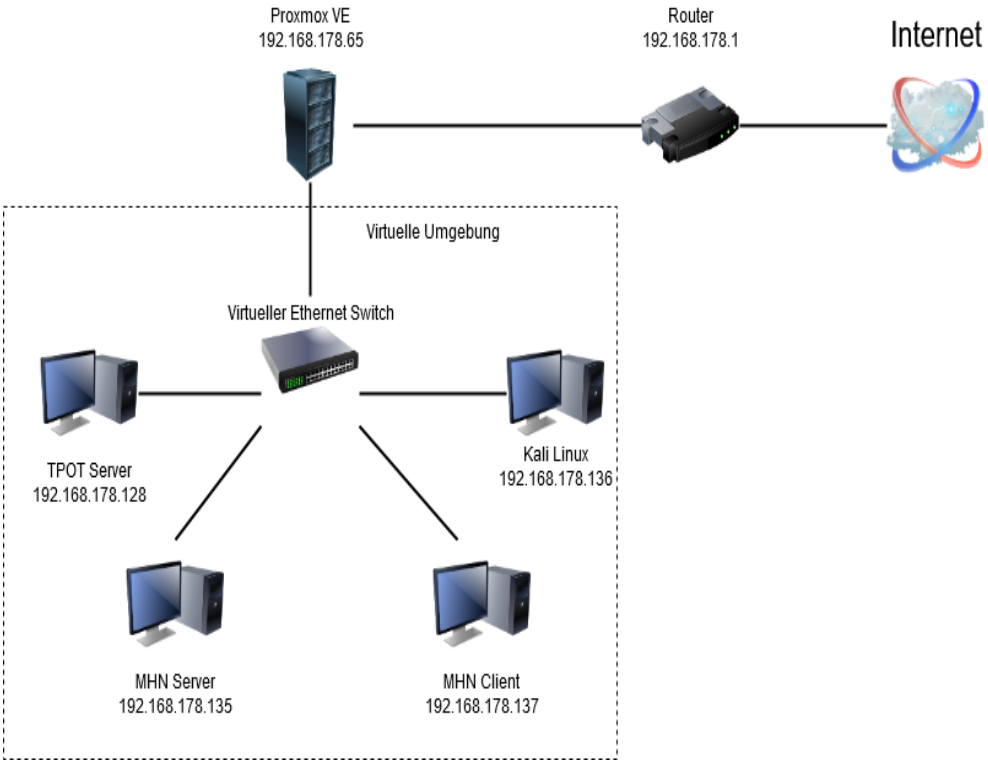


Abbildung 3.1: Netzwerkübersicht der Laborumgebung

3.2.4 T-Pot Honeypot Projekt

T-Pot ist eine Open-Source Softwarelösung der Deutschen Telekom AG, welche mehrere bereits vorhandene Honeypots in einem System vereint [18]. Das Ziel ist es, ein schnell aufzusetzendes Honeypotsystem mit minimalem Wartungsaufwand zu bieten. Das System kann nachträglich auf einem Debian-Server installiert werden, es ist allerdings auch möglich sich eine eigenes Installationsimage zu erzeugen. [14]

Jede Honeypotsoftware läuft in einem eigenem Docker-Container und ist somit vom restlichem System abgeschottet. Sollte ein Honeypot ausfallen, wird der Container neu gestartet. Diese Herangehensweise ermöglicht es auch, die verschiedenen Honeypots auf einem Interface zu bündeln. Die einzelnen Honeypots sind darauf ausgelegt, jegliche Interaktion zu protokollieren. Die Anbindung an Elasticsearch, Logstash und Kibana, zusammengefasst „ELK-Stack“ [19] genannt, ermöglicht eine gute Analyse der protokollierten Daten.

Die Grafik A.1 stellt die Architektur des T-Pot Honeypots in einer Übersicht dar.

3.2.5 Modern Honey Network

Modern Honey Network [15], abgekürzt MHN, ist zunächst ein zentralisierter Server um Datensammlungen zu verwalten, die von Honeypots gesammelt werden. MHN basiert auf einer in Python geschriebenen Flask Applikation. Diese ermöglicht es, Honeypotsoftware auf bereits bestehenden Clients im Netzwerk zu verteilen und diese als Sensoren im Netzwerk zu nutzen. In der für die vorliegende Arbeit genutzten Laborumgebung wird dieser Client als „MHN Client“ bezeichnet.

Anders als bei T-Pot, bei dem die Honeypotsoftware auf ein und demselben Server installiert ist, fungiert MHN als Client-Server System. Über die Weboberfläche des MHN Servers kann Honeypotsoftware an Clientcomputer übertragen und automatisiert installiert werden. Dies ermöglicht es, an beliebige Orte im Netzwerk Sensoren zu installieren. Mit dem Werkzeug hpfeeds [20] werden die Daten der Sensoren an den Server übertragen, der diese in eine Datenbank schreibt. Hierfür wird MongoDB [21] genutzt. Über die sogenannte „Honeymap“ werden die Ursprünge der Angriffe auf einer Weltkarte angezeigt. [15] Die Weboberfläche, über die der MHN Server gesteuert werden kann, ist in Abbildung A.3 abgebildet.

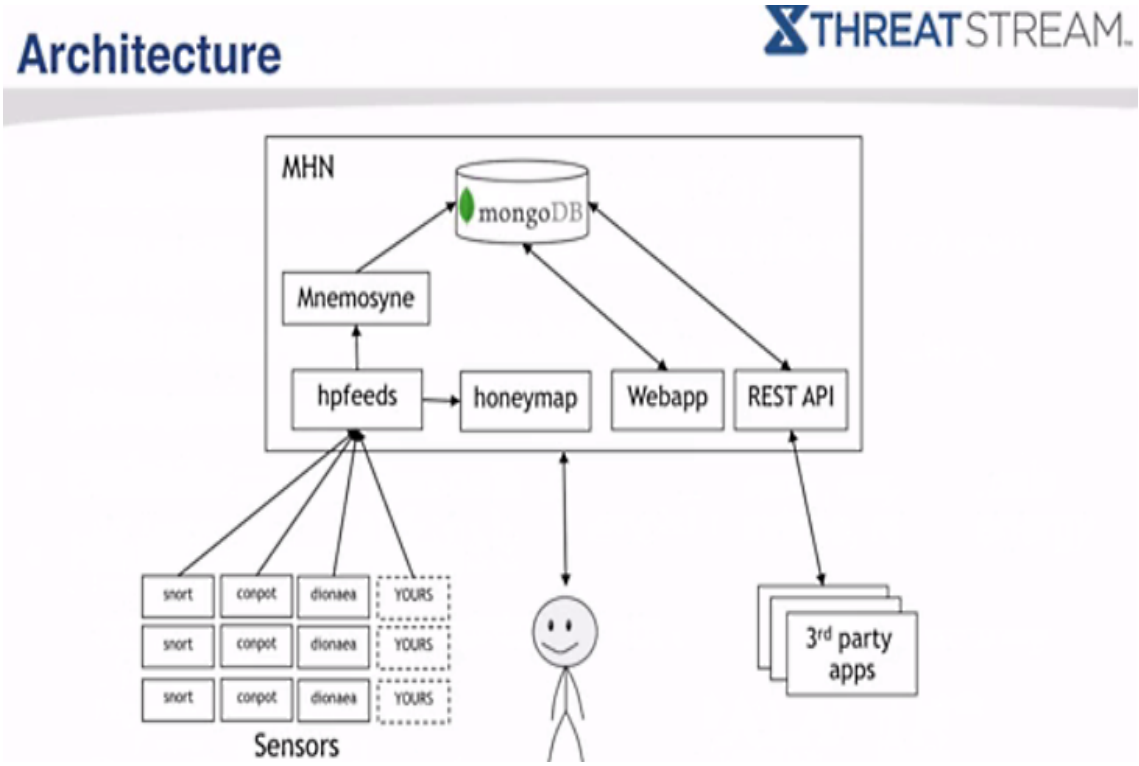


Abbildung 3.2: „MHN Architektur.“ Softpedia News Artikel: Easier Honeypot Deployment and Management with Modern Honey Network. URL: https://news-cdn.softpedia.com/images/fitted/340x180/Easier-Honeypot-Deployment-and-Management-with-Modern-Honey-Network.jpg, zuletzt Aufgerufen am 18.08.2020.

3.2.6 Kali Linux

Kali Linux ist eine Open-Source Distribution, welche speziell an Netzwerkanalysten und Penetrationstester gerichtet ist [16]. Dies macht Kali Linux ideal, um Angriffsszenarien zu entwerfen und die Honeypotsysteme damit zu testen.

Um die Angriffsszenarien so realistisch wie möglich zu halten, wurde Wert darauf gelegt, diese ähnlich zu strukturieren und durchzuführen wie es auch außerhalb der Laborumgebung der Fall ist.

4 Aufbau und Entwurf der Angriffsszenarien

In diesem Kapitel wird zunächst erläutert, wie die Angriffsszenarien entworfen werden und wie diese aufgebaut sind. Es folgt ein kurzer Einblick in die Vorgehensweise von Netzwerkanalysten und Angreifern und schließlich werden die Ergebnisse der Angriffsszenarien präsentiert.

Gezielte Angriffe auf IT-Systeme lassen sich in mehrere Teilschritte gliedern, die jeweils eigene Ziele verfolgen und aufeinander aufbauen. Das Buch „Mastering Kali Linux for Advanced Penetration Testing“ von Robert W. Beggs [22] beschreibt diese Teilschritte wie folgt:

- Identifizierung des Ziels - Passive Aufklärung
- Schwachstellenanalyse - Aktive Aufklärung
- Infiltration des Zielsystems
- Post-Infiltration: Zieldaten lokalisieren, Malware einschleusen
- Post-Infiltration: Persistente Verbindung zum kompromittiertes System Aufbauen

Normalerweise sind die Honeypotsysteme offen aus dem Internet zu erreichen, die Laborumgebung soll dies so weit wie möglich simulieren, weshalb der Client die Honeypotsysteme ansprechen und erreichen kann.

Im Zuge des vorliegenden Experiments wird der Schwerpunkt hauptsächlich auf die Schwachstellenanalyse, also der aktiven Aufklärung, gelegt. Dies hat zum Grunde, dass durch den Aufbau der Laborumgebung der Client, welcher den „Angreifer“ darstellt, bereits im selben Netzwerk agiert, in dem sich auch die Honeypots befinden, wie aus der Abbildung 3.1 ersichtlich. Im Experiment wird somit davon ausgegangen, das der Angreifer die Honeypotsysteme gefunden hat und nun mit den nächsten Schritten beginnt. Diese Schritte sollen die Angriffsszenarien darstellen.

4.1 Aktive und Passive Aufklärung

Aufklärungsmaßnahmen, auch häufig als „Recon“ bzw. „Reconnaissance“ bezeichnet, lassen sich im Bezug auf IT-Sicherheit in aktive und passive Aufklärung unterteilen. Um dies näher zu erläutern, wird sich der Abschnitt auf die Erklärungen und Ausführungen des Buches „Mastering Kali Linux for Advanced Penetration Testing“ [22] beziehen.

Generell ist die Aufklärung der erste Schritt, um zunächst Informationen über das Ziel zu sammeln.

4.1.1 Passive Aufklärung

Passive Aufklärung involviert keine direkte Interaktion mit dem eigentlichem Ziel. Vielmehr wird auf öffentlich zugängliche Informationen zurückgegriffen, um zunächst alles frei verfügbare zu sammeln. Solche Informationen werden als „Open-Source Intelligence“, kurz OSINT, bezeichnet. [22] Ist das Ziel ein Unternehmen, so wären Informationen, welche sich durch passive Aufklärung sammeln ließen, zum Beispiel:

- Frei Verfügbare Quellen aus dem Internet(Website, Social-Media, Unternehmensgröße)
- Kontaktdaten von verschiedenen Abteilungen, Mitarbeitern oder Einrichtungen
- Historie vergangener möglicher anderer Angriffe auf das Ziel

Solche Informationen können bereits die Grundlage von nachfolgenden Angriffsstrategien sein.

4.1.2 Aktive Aufklärung

Dieser deutlich direktere Weg der Informationsbeschaffung interagiert mit dem Ziel und hat ein deutlich höheres Risiko, von diesem als Angriff detektiert zu werden. Das Ziel ist es, die potentielle Angriffsfläche abzusuchen und Schwachstellen zu finden. In der Regel erzeugt die aktive Aufklärung nützlichere Informationen für den Angreifer als die passive Aufklärung, allerdings könnten diese Interaktionen mit dem Zielserver bereits protokolliert werden oder Alarmsignale von Firewall- oder Intrusion-Detection-Systemen auslösen. [22]

Aktive Aufklärungsmaßnahmen sind unter anderem:

- Gezielte Netzwerkscans
- Portscans
- Fingerprinting
- Schwachstellenanalyse

Diese Maßnahmen decken eine Vielzahl an Möglichkeiten ab. Netzwerkscans können Informationen über die Netzwerkstruktur aufzeigen, in der sich das Ziel befindet. Firewalls und weitere Sicherheitskonzepte können damit auch erkannt werden, und der Angreifer kann sich ein Bild von dem Sicherheitsgrad machen, den das Zielnetzwerk innehat. Offene Ports der Server, mögliche Dienste dahinter, sowie das installierte Betriebssystem können ebenfalls mit Netzwerkscans ermittelt werden. Das Betriebssystem zu ermitteln wird auch als „Fingerprinting“ bezeichnet [27]. Hierbei werden angepasste Datenpakete an das Zielsystem geschickt und das darauf antwortende Paket

wird mit einer Datenbank abgeglichen. Bis auf die sogenannte Schwachstellenanalyse, lassen sich diese Maßnahmen mit dem Tool nmap realisieren.

Die Schwachstellenanalyse beschreibt den Prozess aktiv nach offenen Schwachstellen, sogenannten „Exploits“ zu suchen. Solche Schwachstellenanalysen sind sehr auffällig und können schnell entdeckt werden, da viel mehr Netzwerkverkehr zum Ziel erzeugt wird. Weiterhin sind Schwachstellenanalysen signaturbasiert, folglich können nur bereits bekannte Sicherheitslücken entdeckt werden, und auch nur wenn diese Sicherheitslücke bereits eine ihr zugewiesene Signatur hat. [22]

Es gibt öffentlich zugängliche Datenbanken voller bekannter Schwachstellen, die sich herunterladen und für die Schwachstellenanalyse verwenden lassen. Ein Beispiel wäre hier die Website exploit-db.com, welche eine Datenbank für Schwachstellen bereitstellt [23]. Solche Schwachstellen sind nach einem Industriestandard genormt, dem CVE - „Common Vulnerabilities and Exposures“, und erhalten jeder seine eigene Nummer und Signatur.

Die Scanner für Schwachstellenanalysen produzieren eine sehr große Datenmenge, und darunter fallen häufig False-Positives, vor allem wenn das Zielnetzwerk mehrere unterschiedliche Betriebssysteme auf den Servern nutzt. Solche Scans erzeugen durch ihre Dauer und die Vielzahl an Paketen, die sie erzeugen, häufig eine hohe Latenz auf dem Zielserver und können das Netzwerk negativ beeinflussen [22]. Ein Open-Source Schwachstellenscanner ist zum Beispiel OpenVAS [25]. Ein weiterer, aber mittlerweile kommerzieller Schwachstellenscanner ist nessus. [26]

4.2 Entwurf der Angriffsszenarien

Die Angriffsszenarien werden die typischen Schritte der aktiven Reconaissance nachbilden und aufzeigen, wie ein Angreifer die Honeypots wahrnehmen würde.

- Portscanning und OS Fingerprinting mit Nmap
- Schwachstellenanalyse mit OpenVAS

4.3 Ziele der Angriffsszenarien

Folgende Fragen sollen durch die Angriffsszenarien beantwortet werden können:

- Welche Schwachstellen werden durch die Honeypots präsentiert?
- Inwieweit kann ein Angreifer mit den Honeypots interagieren?
- Ist es möglich, die Honeypotsoftware als Angreifer zu identifizieren?
- Wie werden die Honeypots die Angriffe protokollieren?

- Wie lassen sich die Daten der Honeypots auswerten?

Diese Fragen sollen als roter Faden dienen, und den Ergebnissen der Angriffsszenarien ein weiteres Ziel zuzuordnen. Damit dienen die Angriffsszenarien dem Zweck, die Honeypotsoftware grundlegend zu testen und um aufzuzeigen, welche Daten die Honeypots von solchen Angriffen protokollieren. Die Honeypotsoftware wird durch die Angriffsszenarien vorgestellt.

4.4 Eingesetzte Software

Das folgende Unterkapitel gibt eine kurze Einführung in die genutzte Software zur durchführung der Angriffsszenarien. Die Funktionalitäten der einzelnen Programme überschneiden sich etwas. Um False-Positives zu vermeiden, wird häufig auf mehrere Schwachstellenscanner zurückgegriffen, um die Ergebnisse miteinander abzugleichen.

4.4.1 Nmap

Nmap, auch „Network Mapper“ genannt, ist eine freie und Open-Source Software, welche zu den Standardpaketen von Kali Linux gehört. Die Software wird zum Großteil genutzt, um die Ports der verschiedenen Hosts eines Netzwerks zu scannen. Das Programm lässt sich über ein Terminal mittels Kommandozeilenparameter bedienen, es existiert auch eine Variante mit grafischer Oberfläche namens Zenmap. [28]

4.4.2 OpenVAS

OpenVAS (Open Vulnerability Assessment System) ist ein Framework zur Schwachstellenanalyse und Auswertung. Gesteuert wird OpenVAS über eine Weboberfläche. Neben dem eigentlichem Scanning steht OpenVAS ein sogenannter „Feed“ an Schwachstellentests, sogenannten NVTs (Network Vulnerability Tests) zur Seite. Diese werden täglich aktualisiert und umfassen mehr als 50.000 Schwachstellen-Prüfungen. [25]

5 Durchführung der Angriffsszenarien

In folgendem Kapitel werden die einzelnen Szenarien detaillierter beschrieben und die jeweiligen Ergebnisse präsentiert. Die Szenarien werden für beide Honeypotsysteme gleichermaßen durchgeführt. Die Präsentation der Ergebnisdaten wird uniform gehalten. Zunächst werden die Ergebnisse aus der Sicht des Angreifers präsentiert und welche Informationen dieser sammeln konnte. Im Anschluss wird dargestellt, wie die Honeypotsoftware die Angriffe protokolliert hat und wie sich diese Daten vom jeweiligen Honeypotsystem auswerten lassen.

5.1 Angriffe gegen das Honeypotsystem T-Pot

Da T-Pot eine große Menge an einzelnen Honeypots anbietet, würde es den Rahmen sprengen, auf jeden einzelnen Honeypot einzugehen. Stattdessen werden hauptsächlich die Honeypots betrachtet, welche eine signifikante Menge an Daten sammeln konnten, oder jene, welche ebenfalls von MHN eingesetzt werden.

5.1.1 Szenario 1: Portscan mit Nmap

Als erstes Szenario wurde ein Netzwerkscan mit Nmap durchgeführt. Dieser soll zunächst alle Ports des T-Pot Servers scannen und versuchen, den Service des Ports zu identifizieren. Folgender Befehl wurde hierzu ausgeführt:

```
1 nmap -A -T4 -oN /scans/basic_scan.txt 192.168.178.128
```

Die Tabelle 5.1 erläutert die einzelnen Parameter und protokolliert Start- und Endzeit.

Nmap Scan Parameter	
-A	Betriebssystem- und Versionserkennung, Script-Scanning, Traceroute
-T4	Schnelle Zeitvorgabe
-oN	Ausgabe in Textdatei „basic_scan.txt“
192.168.178.128	Zielhost, T-Pot IPv4-Adresse
Protokoll	
Startzeit	17:23 Uhr
Endzeit	17:57 Uhr
Dauer	0h 34m

Tabelle 5.1: Nmap Scan Protokoll

Ergebnisse des Nmap Scans

Der Scan beinhaltete alle Ports von Port 1 bis 65535. Dabei hat er 141 von 65535 „geschlossene “ Ports entdeckt, der Rest wurde als „offen“ gekennzeichnet. „Offen “ bedeutet, das dieser Port TCP oder UDP Verbindungen akzeptiert und bereit für eine Verbindung ist. „Geschlossen“ bedeutet, das der Server auf diesem Port nicht geantwortet hat. Sollte Nmap erfolgreich den Service dazu detektiert haben, so wird dies ebenfalls direkt mit ausgegeben.

Ein Ausschnitt der Ausgabedatei zeigt, dass Beispielsweise auf Port 21 ein ftp Dienst gefunden wurde, welcher einen anonymen Login erlaubt.

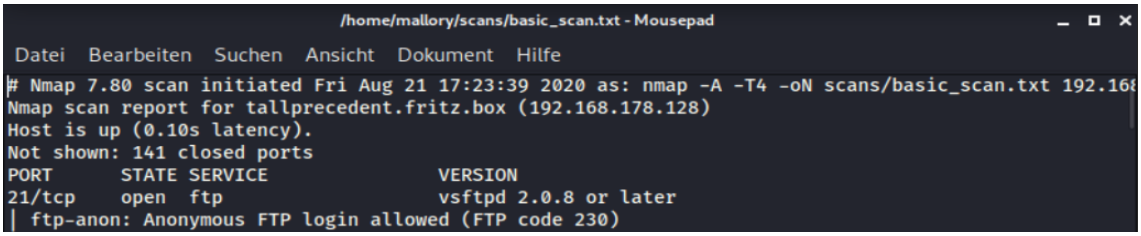


Abbildung 5.1: Die Ausgabe des ersten Nmap Scans. Quelle: Ausschnitt eines Screenshots des Kali Linux Terminals.

Zusammengefasst haben die folgenden Ports die meiste Information zurückgegeben:

Relevante erkannte Ports	
21/tcp	FTP Service, Anonymer Login erlaubt
22/tcp	OpenSSH 7.9
23/tcp	telnet, Login Shell
80/tcp	HTTP, AIOHTTP 3.4.4 Server
443/tcp	HTTPS, Apache Server, Citrix Login-Seite
445/tcp	SMB, Microsoft Dateifreigabe
1080/tcp	socks5, Login-Shell
1433/tcp	ms-sql, Microsoft SQL Server 2000 SP1
3306/tcp	mysql, MySQL 5.7.16
5060/tcp	SIP, SIP Endpoint
5900/tcp	VNC, VNC 3.7
8443/tcp	HTTPS-Alt, Get-Request lieferte Antwort

Tabelle 5.2: Relevante erkannte Ports des Nmap Scans.

Alle anderen Ports waren ebenfalls offen, haben aber keinen direkten Hinweis auf den Service gegeben, der dort installiert sein könnte. Diese Liste kann nun als weiteren Ausgangspunkt für den Angreifer dienen.

Protokollierte Daten des Honeypots

Direkt mit Beginn des Scans wurde die Interaktion mit dem Honeypots deutlich bemerkbar. Dank Kibana als grafisches Werkzeug zur anzeige der protokollierten Daten bietet T-Pot eine ausführliche Übersicht der einzelnen Angriffe. T-Pot bietet hierbei ein allgemeines Dashboard mit einer Übersicht, sowie ein eigenes Dashboard für jeden einzelnen Honeypot.

Jede Honeypot-Software hat unterschiedlich viel Angriffe gemeldet und protokolliert. Als Angriff wird dabei jede einzelne Interaktion mit einem der Honeypots gewertet.

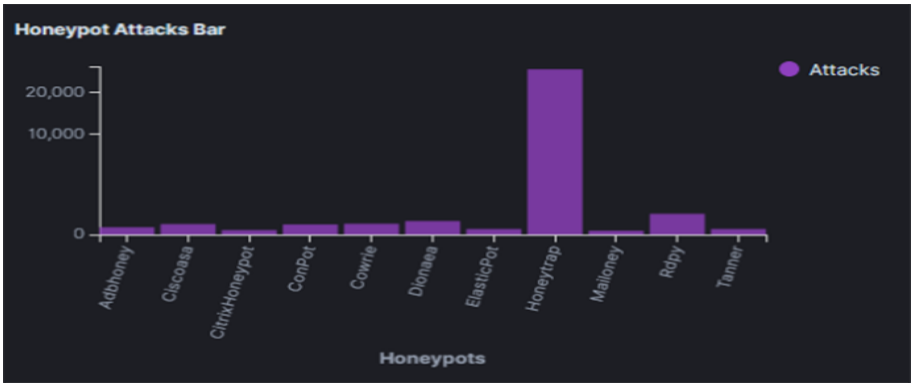


Abbildung 5.2: Säulendiagramm welche die Anzahl der protokollierten Angriffe auf die unterschiedlichen Honeypots vergleicht. Honeytrap hatte die meisten Angriffe zu verzeichnen. Quelle: Kibana Dashboard

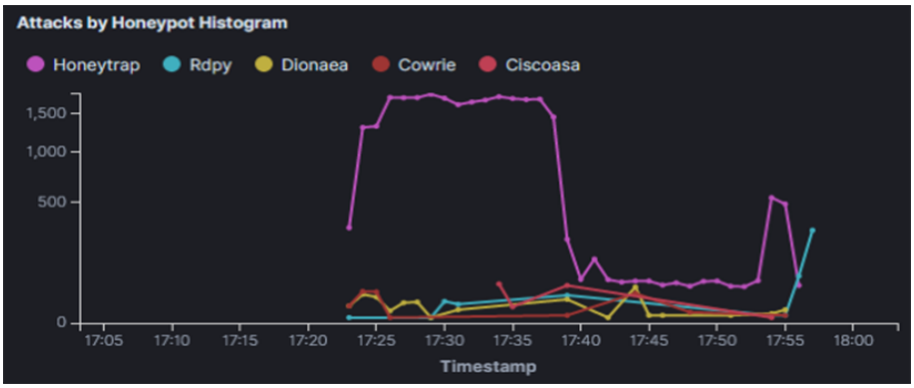


Abbildung 5.3: Zeitverlauf der Angriffe auf die Honeypots. Quelle: Kibana Dashboard

Am auffälligsten ist der Honeypot „Honeytrap“ mit über 27.058 Angriffen. Dies ist darin begründet, dass Honeytrap für einen Großteil der offenen TCP/UDP Ports zuständig ist und somit die meisten Attacken verzeichnet. Dies wird aus Abbildung 5.2 und der Abbildung 5.3 ersichtlich.

5.1.2 Szenario 2: Schwachstellenanalyse mit OpenVAS

Das zweite Szenario stellt eine Schwachstellenanalyse mit dem Tool OpenVAS dar. OpenVAS wird komplett über eine eigene Weboberfläche bedient und bietet verschiedene Scan-Vorlagen an.

Für dieses Angriffsszenario ist die Vorlage „Full and fast“ genutzt, welche zunächst einen eigenen Scan der Zielports durchführt und auf Basis dessen zu testende Schwachstellen wählt.

OpenVAS Scan	
Typ	Full and Fast Scan
Protokoll	
Startzeit	22.08., 17:38 Uhr
Endzeit	23.08., 4:06 Uhr
Dauer	10h 28m

Tabelle 5.3: OpenVAS Scan Protokoll

Überraschend war hierbei die lange Dauer des Scans. Dies lässt sich jedoch dadurch begründen, das der T-Pot Honeypot fast alle vorhandenen Ports offen gelegt hat, und OpenVAS jeden dieser Ports auf Schwachstellen überprüft. Insgesamt hat OpenVAS durch diese Scanvariante 64 NVT Gruppen durchlaufen. Darunter fällt auch das direkte Prüfen von Standardnutzernamen und Passwörtern, sofern es möglich war.

Ergebnisse des OpenVAS Scans

Insgesamt hat OpenVAS 35 Schwachstellen gefunden, davon sind 9 mit einer hohen Warnstufe von 7.5 bis 10 eingestuft worden, 24 mit einer mittleren Warnstufe von 4.0 bis 6.4, und zwei wurden als niedrig eingestuft. Die Abbildung 5.4 zeigt die ersten 16 Ergebnisse. So hat OpenVAS zum Beispiel herausgefunden, dass es möglich ist, sich mit dem Nutzeraccount „htr“ und dem Passwort „htr_pwd“ auf Port 3306 einzuloggen. Dies ist Teil des Honeypots Dionea, welcher auf Port 3306 eine mit Schwachstellen behaftete Datenbank präsentiert. Weiterhin wird jedem Ergebnis ein „QoD“ zugeordnet. Dies steht für „Quality of Detection“, gibt also den von OpenVAS eingeschätzten Qualitätswert des Fundes wieder.

Jeder Honeypot stellt falsche Banner zur Verfügung, um dahinterliegende Dienste vorzutäuschen. Im Beispiel der falschen Datenbank auf Port 3306 hat OpenVAS analysiert, dass es sich um eine MySQL Datenbank halten muss. Diese Information hat OpenVAS aus dem Banner extrahiert hat, wie in Abbildung 5.5 zu sehen ist. Solche und ähnliche Schwachstellen konnte OpenVAS erfolgreich feststellen und hat diese nach Risikostufe geordnet.

Vulnerability	Severity ▾	QoD
YaBB XSS and Administrator Command Execution	10.0 (High)	99 %
Elasticsearch End of Life Detection	10.0 (High)	80 %
Blackstratus LOGStorm default MySQL password for user `htr`	9.0 (High)	95 %
MySQL / MariaDB weak password	9.0 (High)	95 %
Dabber Worm Detection	8.3 (High)	70 %
Apache Tomcat AJP RCE Vulnerability (Ghostcat)	7.5 (High)	99 %
CactuShop XSS and SQL injection flaws	7.5 (High)	99 %
Android Debug Bridge (ADB) Accessible Without Authentication	7.5 (High)	80 %
GoSmart message board multiple flaws	7.5 (High)	99 %
Anonymous FTP Login Reporting	6.4 (Medium)	80 %
Missing `httpOnly` Cookie Attribute	5.0 (Medium)	80 %
Missing `httpOnly` Cookie Attribute	5.0 (Medium)	80 %
Telnet Unencrypted Cleartext Login	4.8 (Medium)	70 %
FTP Unencrypted Cleartext Login	4.8 (Medium)	70 %
IMAP Unencrypted Cleartext Login	4.8 (Medium)	70 %
SMTP Unencrypted Cleartext Login	4.8 (Medium)	70 %

Abbildung 5.4: Die ersten 16 Ergebnisse des OpenVAS Scans. Quelle: OpenVAS Dashboard

Summary

Detects the installed version of
MySQL/MariaDB.

Detect a running MySQL/MariaDB by getting the banner, extract the version
from the banner.

Detection Result

```
Detected MySQL

Version:        5.7.16
Location:       3306/tcp
CPE:            cpe:/a:oracle:mysql:5.7.16

Concluded from version/product identification result:
5.7.16
```

Abbildung 5.5: OpenVAS extrahierte aus dem Banner die Versionsinformation. Quelle: OpenVAS Ergebnisanzeige.

Protokollierte Daten des Honeypots

T-Pot hat den Schwachstellenscan komplett protokolliert. Insgesamt wurden für den Zeitraum des Schwachstellenscans 184.285 Angriffe festgestellt. Dementsprechend groß ist die Menge an Daten, die für diesen Zeitraum gesammelt wurden. Aus den Daten lässt sich auch ablesen, dass sich der Scan auf andere Dienste und Ports konzentriert hat, als dies der erste Nmap Scan in Unterabschnitt 5.1.1 tat.

Hier weist Honeytrap erneut viele Angriffe auf, doch die Mehrzahl wird diesmal von Tanner protokolliert. Tanner läuft auf Port 80 und emuliert verschiedene Web Applikationen, die typischerweise auf Port 80 zu finden sind. 17 der 35 gefundenen Schwachstellen

von OpenVAS lassen sich auf Port 80 zurückführen und erklären damit die hohe Anzahl der protokollierten Attacken seitens Tanner. Dionaea und CitrixHoneypot wiesen eine niedrigere Angriffsverteilung auf.

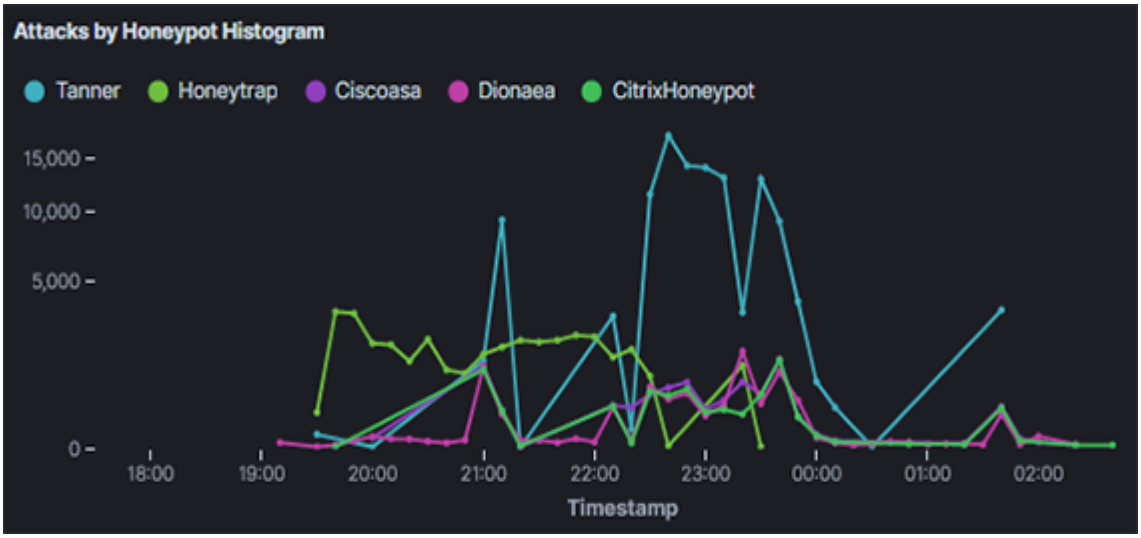


Abbildung 5.6: Der Graph verdeutlicht, mit welchen Honeypots der Scan am meisten Interagiert hat. Quelle: Kibana Dashboard

Die Login-Versuche von OpenVAS wurden ebenfalls protokolliert. Hier erstellt Kibana eine sogenannte Wordcloud, welche die verschiedenen Nutzernamen und Passwörter in Relation zu ihrer Häufigkeit größer oder kleiner darstellt. Größere hervorgehobene Wörter wurden häufiger genutzt als kleinere. Einsehbar bei Abbildung 5.7 und Abbildung 5.8.

Abbildung 5.7: Die Wordcloud zeigt, welche Nutzernamen am häufigsten von OpenVAS getestet wurden. Quelle: Kibana Dashboard

Abbildung 5.8: Die Wordcloud zeigt, welche Passwörter am häufigsten von OpenVAS getestet wurden. Quelle: Kibana Dashboard

5.1.3 Ergebnis der Szenarien

Abschließend werden die Ergebnisse der Angriffsszenarien mit den eigens aufgestellten Zielen aus Abschnitt 4.3 abgeglichen.

Das Honeypotsystem T-Pot wurde in den vorherigen Szenarien mittels aktiver Reconaissance untersucht, um zu zeigen, wie das Honeypotsystem solche Angriffe verwertet und speichert. Hierbei fiel vor allem die große Anzahl an Schwachstellen, sowie zu erreichender Ports auf, die dank der vielseitigen Honeypots, welche T-Pot unter sich vereint, erreicht wird. Dies könnte jedoch dazu führen, das der T-Pot Server unter dieser Standardinstallation von gezielten Hackerangriffen schnell als Honeypot entlarvt wird. Dies trifft allerdings nicht auf automatisierte Malware zu.

Mit einigen Ports der Honeypots ließ sich bis zu einem gewissen Grad interagieren, so konnte sich OpenVAS beispielsweise bei dem FTP Honeypot Dionaea als anonymer Nutzer anmelden. Da jedoch jeder einzelne Honeypot in einer Docker Container Umgebung arbeitet, ist eine Kompromittierung des Honeypotsystems ein geringes Risiko.

T-Pot nutzt Logstash und Elasticsearch zur Verwaltung der gesammelten Daten, und Kibana zeigt diese in einer großen Übersicht in verschiedenen Diagrammen und Grafiken an.

5.2 Angriffe gegen das Honeypotsystem Modern Honey Network

Der MHN-Server kann beliebige Honeypots an bereits bestehende Clients im Netzwerk verteilen. Als zu installierende Honeypotsoftware wurde der SSH-Honeypot Cowrie gewählt, sowie Dionaea. Beide Honeypots kommen ebenfalls bei dem Honeypotsystem

T-Pot zum Einsatz.

5.2.1 Szenario 1: Portscan mit Nmap

Der Portscan gegen den MHN-Client wird auf die gleiche Art und Weise durchgeführt, wie auch beim T-Pot Honeypot.

```
1 nmap -A -T4 -oN /scans/mhn_scan.txt 192.168.178.137
```

Nmap Scan Parameter	
-A	Betriebssystem- und Versionserkennung, Script-Scanning, Traceroute
-T4	Schnelle Zeitvorgabe
-oN	Ausgabe in Textdatei „mhn_scan.txt“
192.168.178.137	Zielhost, T-Pot IPv4-Adresse
Protokoll	
Startzeit	16:30 Uhr
Endzeit	16:34 Uhr
Dauer	0h 4m

Tabelle 5.4: Nmap Scan Protokoll gegen MHN

Ergebnisse des Nmap Scans

Der Scan konnte in deutlich schnellerer Zeit durchgeführt werden. Dies ist nicht überraschend, es sind deutlich weniger Ports offen, die gescannt werden können. Dies reduziert die Scandauer immens. Der Scan fand 14 offene Ports, davon gehören 13 zum Honeypot Dionaea, und einer zum Honeypot Cowrie.

Abbildung 5.9 zeigt die ersten Ergebnisse des Nmap-Scans. Interessant hierbei ist, dass bei Port 443, der üblicherweise für HTTPS-Verbindungen genutzt wird, ein eigenes Zertifikat gefunden wurde. Dieses enthält den Namen des Honeypots Dionaea, sowie von seinem Vorgänger Nepenthes.

Protokollierte Daten des Honeypots

Die MHN-Clients senden ihre gesammelten Daten an den MHN-Server, der diese bündelt und anschließend auf der Weboberfläche anzeigt. Hierbei weist der MHN-Server die eingegangenen Angriffe in einer Tabelle auf. Eine Weltkarte demonstriert hierbei sogar interaktiv, aus welchen Ländern die Angriffe durchgeführt worden sind. Da sich die Angriffe der vorliegenden Arbeit komplett in einer LAN-Umgebung befinden, fällt diese Auswertung weg.

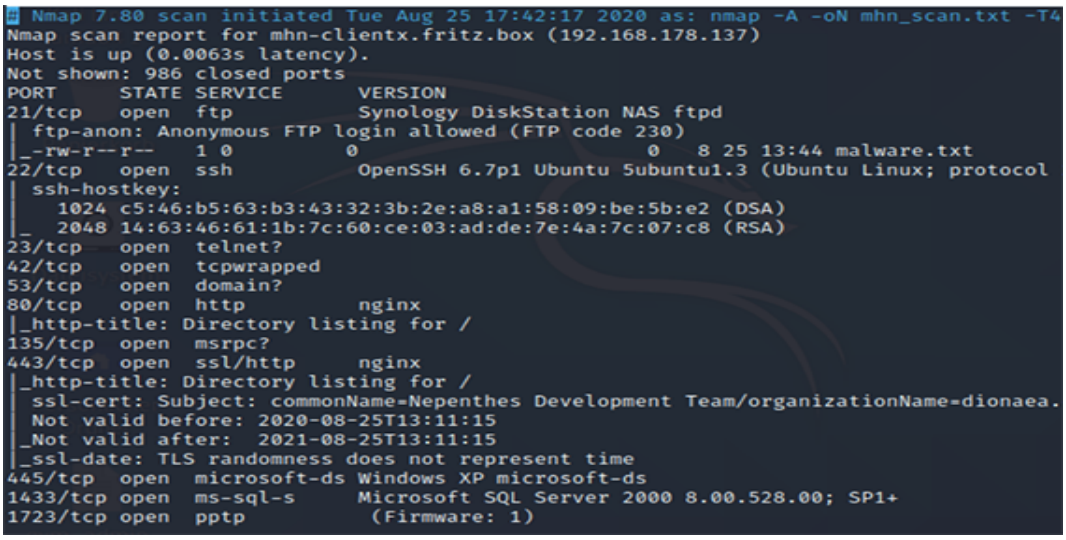


Abbildung 5.9: Die ersten Ergebnisse des Nmap Scans. Quelle: Kali Linux Terminal

Die Protokollierung der Angriffe ist Übersichtlich aufgebaut und gibt Auskunft über den Angreifer. In Abbildung 5.10 erkennt man die attackierten Honeypots Dionaea und Cowrie.

Die Angriffe werden ebenfalls gezählt und den Honeypots zugeordnet. Durch den Nmap Scan hat MHN 181 Angriffe aufgezeichnet, die Mehrheit davon bei dem Honeypot Dionaea.

	Date	Sensor	Src IP	Dst port	Protocol	Honeypot
1	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	443	httpd	dionaea
2	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	443	httpd	dionaea
3	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	443	httpd	dionaea
4	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	443	httpd	dionaea
5	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	443	httpd	dionaea
6	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	443	httpd	dionaea
7	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	443	httpd	dionaea
8	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	80	httpd	dionaea
9	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	22	ssh	cowrie
10	2020-08-25 16:31:40	mhn-clientx	192.168.178.136	443	httpd	dionaea

Abbildung 5.10: Die ersten protokollierten Angriffe des MHN-Servers. Quelle: Screenshot Webbrowser, MHN Server

5.2.2 Szenario 2: Schwachstellenanalyse mit OpenVAS

Wie bei der Schwachstellenanalyse des Honeypotsystems T-Pot, wird auch hier die Scanvorlage „Full and fast“ genutzt.

OpenVAS Scan	
Typ	Full and Fast Scan
Protokoll	
Startzeit	25.08., 16:35 Uhr
Endzeit	25.08., 16:58 Uhr
Dauer	0h 23m

Tabelle 5.5: OpenVAS Scan Protokoll

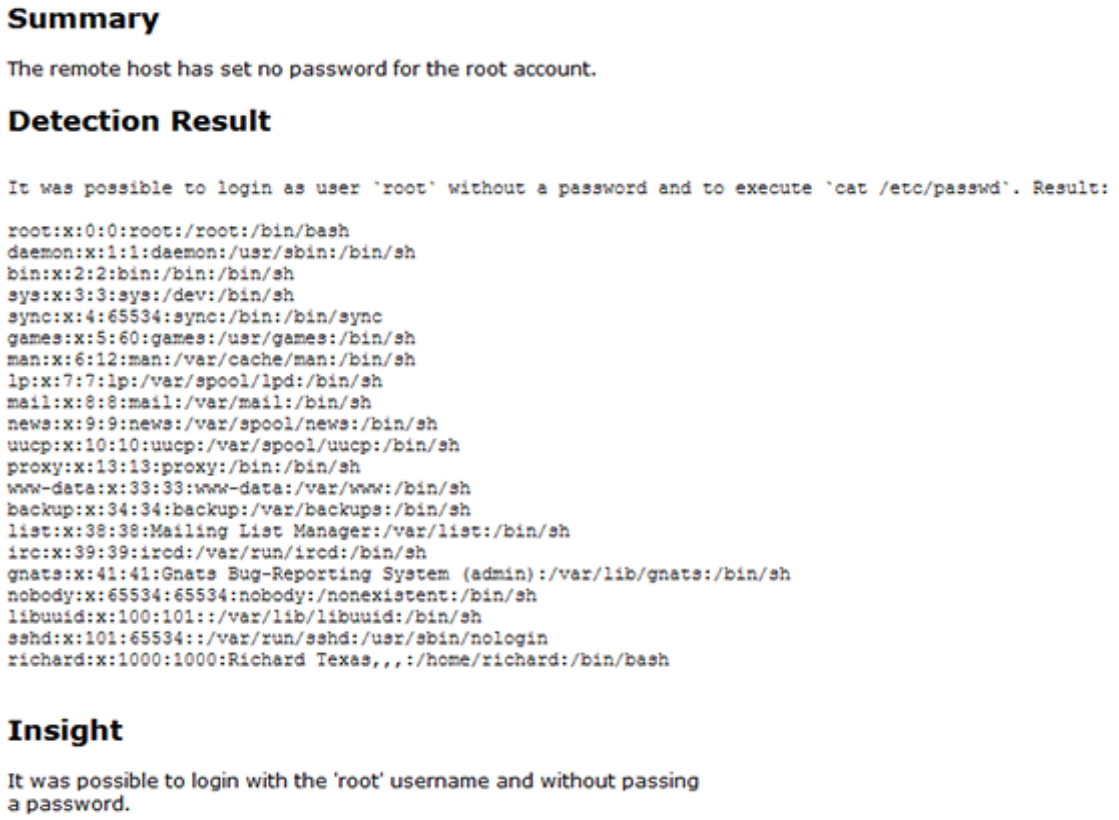
Ergebnisse des OpenVAS Scans

Die deutlich reduzierte Dauer aufgrund der geringeren Anzahl an Ports wurde auch bei diesem Scan sehr bemerkbar. OpenVAS hat insgesamt 21 Schwachstellen aufdecken können, davon wurden 13 ein hohes Risiko zugeschrieben, fünf ein mittleres Risko und zweien ein geringes Risiko. Die meisten Schwachstellen wurden hier auf Port 22 gefunden, dem SSH Dienst, der durch den Honeypot Cowrie bereitgestellt wird.

OpenVAS konnte sich mit verschiedenen Nutzer- und Passwortkombinationen einloggen und einen Befehl ausführen. Darüber hinaus konnte sich OpenVAS den Inhalt der Datei „/etc/passwd“ ausgeben lassen, wie in Abbildung 5.12 gezeigt wird. Diese ist von Cowrie vorgegeben und kann noch manuell angepasst werden. Abbildung 5.11 ist ein Ausschnitt des Ergebnisses der Schwachstellenanalyse.

Vulnerability		Severity ▾	QoD
Default password `3!acKStratuS` for root account		10.0 (High)	100 %
Default Password `avam@r` for root Account.		10.0 (High)	100 %
Unpassworded 'root' Account (SSH)		10.0 (High)	100 %
Rasplex Authentication Bypass Vulnerability		10.0 (High)	99 %
OpenVPN Access Server SSH Default Credentials		10.0 (High)	100 %
OpenELEC Authentication Bypass Vulnerability		10.0 (High)	99 %
Microsoft SQL Server End Of Life Detection		10.0 (High)	80 %
Default password `adminIWSS85` for root account		10.0 (High)	100 %
MySQL / MariaDB weak password		9.0 (High)	95 %
Blackstratus LOGStorm default MySQL password for user `htr`		9.0 (High)	95 %
SSH Brute Force Logins With Default Credentials Reporting		7.5 (High)	95 %
C.H.I.P. Device Default SSH Login		7.5 (High)	100 %
Riverbed SteelCentral SSH Default Credentials		7.5 (High)	100 %
SSH Brute Force Logins With Default Credentials Reporting		7.5 (High)	95 %
Anonymous FTP Login Reporting		6.4 (Medium)	80 %
MQTT Broker Does Not Require Authentication		6.4 (Medium)	95 %
FTP Unencrypted Cleartext Login		4.8 (Medium)	70 %
SSH Weak Encryption Algorithms Supported		4.3 (Medium)	95 %
SSL/TLS: Certificate Signed Using A Weak Signature Algorithm		4.0 (Medium)	80 %

Abbildung 5.11: Schwachstellenanalyse des MHN-Clients durch OpenVAS. Quelle: OpenVAS Dashboard.



Summary

The remote host has set no password for the root account.

Detection Result

```
It was possible to login as user `root` without a password and to execute `cat /etc/passwd`. Result:

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
sshd:x:101:65534::/var/run/sshd:/usr/sbin/nologin
richard:x:1000:1000:Richard Texas,,,:/home/richard:/bin/bash
```

Insight

It was possible to login with the 'root' username and without passing a password.

Abbildung 5.12: Zusammenfassung einer getesteten Schwachstelle. OpenVAS konnte sich als „root“-Nutzer Zugang verschaffen und einen Befehl ausführen. Quelle: OpenVAS Ergebnisanzeige.

Protokollierte Daten des Honeypots

Die durchgeführten Scans von OpenVAS wurden wie auch bereits beim Nmap Scan in einer Übersicht festgehalten. Darüber hinaus hat der Honeypot Cowrie die getesteten Nutzernamen und Passwörter abgespeichert, welche von OpenVAS getestet wurden, wie Abbildung 5.13 aufzeigt.

Die Testdateien, die von OpenVAS hochgeladen worden sind, hat Dionaea als Payload gekennzeichnet., wie in Abbildung 5.14 gezeigt wird. Von diesem Payload wurde ein Hashwert erzeugt. Dies ermöglicht es, nicht nur bekannte Malware einzufangen, sondern auch Hashwerte von neuer, noch unbekannter Malware zu erzeugen.

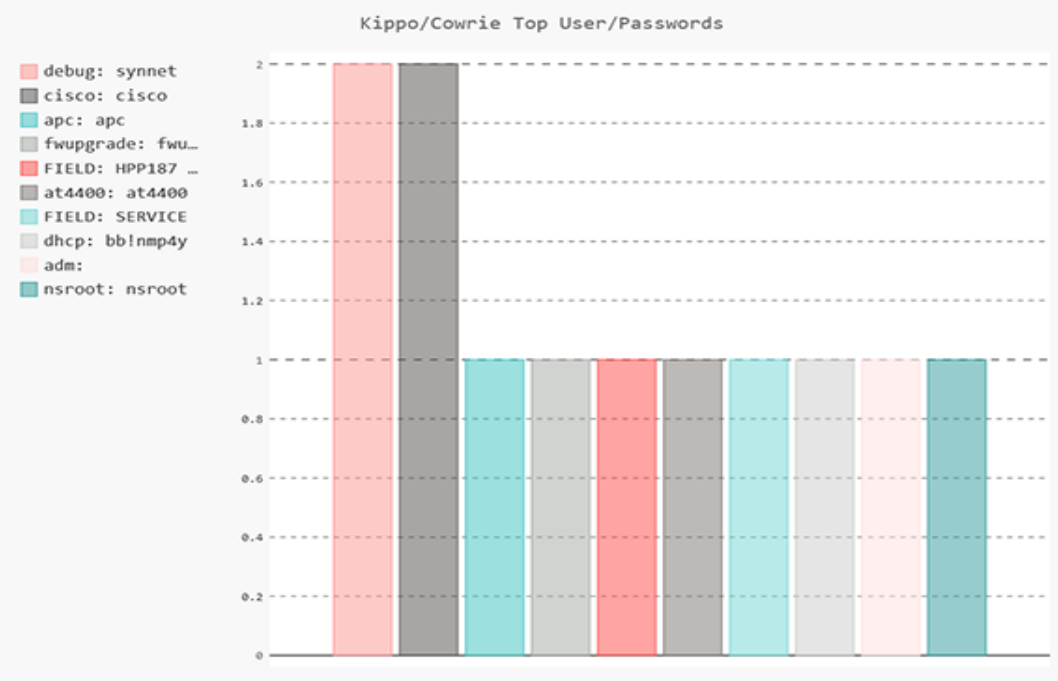


Abbildung 5.13: Die meistgenutzten Nutzer- und Passwortkombinationen des OpenVAS Scans wurden von Cowrie aufgezeichnet. Quelle: Weboberfläche MHN Server

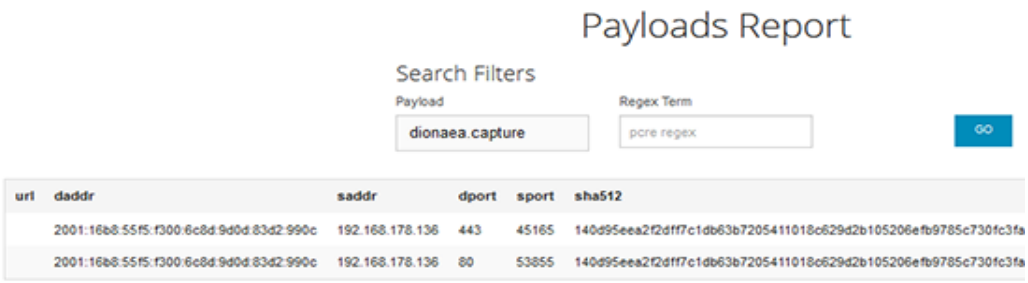


Abbildung 5.14: Zwei hochgeladene Testdateien von OpenVAS wurden von Dionaea als Payload möglicher Malware gekennzeichnet.

5.2.3 Ergebnis der Szenarien

Das Honeypotsystem MHN wurde durch die beiden Szenarien wie auch bei T-Pot getestet. Da die eingesetzten Honeypots manuell als Sensor auf ein bereits bestehendes System installiert werden, ist die Gesamtanzahl an zu testenden Honeypots geringer. Um einen ähnlichen Test wie bei T-Pot zu erzeugen, wurde daher Cowrie und Dionaea installiert, welche beide auch bei T-Pot zum Einsatz kommen.

Ein Angreifer könnte bereits bei dem Nmap-Scan erkennen, das es sich um ein Honeypot handelt, da das SSL-Zertifikat des HTTPS-Dienstes auf Port 443 den Namen des Honeypots, sowie den der Vorgängersoftware enthält. Dies könnte allerdings umgangen werden, wenn ein eigenes Zertifikat hinterlegt wird. Die Angriffe wurden auf

einer minimalistischen Tabelle dargestellt und sind nach Datum sortiert, neueste zuerst. Wahlweise kann dort zum Beispiel nach Honeypotsoftware oder Sensor gefiltert werden.

OpenVAS konnte bei beiden Honeypotsystemen einige Schwachstellen finden, über die schließlich Malware gefangen werden kann. Interessant ist hierbei vor allem die Möglichkeit von Dionaea, abgespeicherte Dateien, wie zum Beispiel den Payload von Malware, einzufangen und ein Hashwert davon zu bilden.

6 Vergleich der vorgestellten Honeypotsysteme

In den vorherigen Kapiteln wurden die beiden Honeypotsysteme T-Pot und Modern Honey Network vorgestellt, sowie deren Arbeitsweise durch eigene Angriffsszenarien vorgeführt. In diesem Kapitel findet nun die finale Gegenüberstellung statt.

Bei der Gegenüberstellung werden unterschiedliche Bewertungskriterien angesetzt. Zuerst wird auf die einzelnen Honeypotsysteme eingegangen und wie diese die Kriterien erfüllen. Abschließend werden die Ergebnisse des Vergleichs in einer Gegenüberstellung zusammengefasst. Die Bewertungskriterien für die beiden Honeypotsysteme sind wie folgt:

- Installations- und Wartungsaufwand
- Skalierbarkeit
- Anpassungsmöglichkeiten/Erweiterungsmöglichkeiten
- Gefahr der Komprimittierung
- Einsatzmöglichkeiten
- Langlebigkeit

6.1 T-Pot

Das T-Pot Honeypotsystem ist schnell einsetzbar und benötigt keine zusätzlichen Softwarebibliotheken oder ähnliches, stattdessen gibt es die Option, bereits fertige Installationsabbilder herunterzuladen oder ein eigenes zu erzeugen. Bei letzterem wird lediglich eine Vorinstallation des Betriebssystems Debian 10 empfohlen, anschließend wird der Anweder durch die Erzeugung des Installationsabbildes geführt und kann Änderungen vornehmen, bis das System installiert ist. Darüber hinaus kann T-Pot auch auf ein bereits bestehendes System installiert werden, beschränkt sich hierbei jedoch ebenfalls auf das Betriebssystem Debian 10.

Nach der Installation ist T-Pot direkt einsatzbereit und die installierten Honeypots können direkt anfangen, Daten zu sammeln. Das ganze System ist darauf ausgelegt, mit wenig Administrationsaufwand zu arbeiten. Dies wurde vor allem durch den Einsatz einer Umgebung in Docker erreicht. Jeder Honeypot hat seinen eigenen Docker-Container, welcher nicht nur für Sicherheit sorgt, sondern auch automatisch den Container neustartet, sollte der Honeypot ausfallen. Da normalerweise Daten in Docker-Containern als „flüchtig“, also als nicht persistent, bezeichnet werden, wird ein spezieller Ordner benutzt, in dem die gesammelten Daten der Honeypots für 30 Tage persistent abge-

speichert werden. Das Ausnutzen der Funktionen einer Docker-Umgebung sorgt auch für eine gute Absicherung im Falle einer Kompromittierung. Dieses Risiko ist in der Regel bereits recht gering durch die Tatsache, das es sich hierbei um Low-Interaction Honeypots handelt. Doch sollte es einem Angreifer dennoch gelingen, die Honeypotsoftware zu kompromittieren, so ist dieser immernoch auf die Umgebung des Docker Containers beschränkt und kann nicht mit dem restlichem Server interagieren. Damit wird der Risikofaktor als gering eingeschätzt.

T-Pot bringt eine Vielzahl an Honeypotsoftware mit, für unterschiedlichste Protokolle und Dienste. Darüber hinaus werden weitere Softwarewerkzeuge angeboten, welche als Unterstützung zu analytischen oder forensischen Arbeitsbereichen dienen können. Wie in den Angriffsszenarien aus Kapitel 5 verdeutlicht, hat dieser Ansatz unterschiedliche Vor- und Nachteile. Zunächst ermöglicht es, unterschiedlichste und auch sehr spezifische Malware einzufangen und zu protokollieren. Das Ziel dieser Strategie ist es, so viel Malware wie möglich einzufangen, um diese zu erforschen und um möglicherweise neue Trends oder Angriffsstrategien zu entdecken. Einzelne Angreifer werden davon wahrscheinlich eher abgeschreckt, aber das ist auch nicht die Zielgruppe von T-Pot.

Das Honeypotsystem basiert auf einem zentralisierten Ansatz, da die Daten der Honeypots auf demselben System abgespeichert und auch abgerufen werden können, auf dem diese auch gesammelt wurden. Damit wird das Problem umgangen, die protokollierten Daten über einen Service an einen entfernten Server zu senden. Dies stellt häufig für Honeypots ein Problem dar, da dies zum einen den Angreifern weitere Informationen über die Serverstruktur gibt, und zum anderen eine sensible Schwachstelle im Honeypotsystem darstellen könnte. Auf der anderen Seite ist der Honeypotserver dadurch in seiner Lokalität sehr gebunden. Je nach Einsatzfeld kann dies irrelevant sein oder einen Nachteil darstellen.

Hierbei stellt sich auch die Frage, in welche der Kategorien aus Kapitel 2 sich dieser einordnen ließe. Nach Betrachtung der bisher genannten Punkte ist das Honeypotsystem T-Pot mehr auf Forschung und Analyse ausgelegt, als auf die Nutzung in einem Produktionsnetzwerk. Die statische Lokalität erschwert es, das Honeypotsystem in einem eigenen Netzwerk zu verwenden. Häufig werden gleich mehrere solcher Sensoren installiert und mit T-Pot wäre dies nur durch mehrere Standalone-Installationen möglich. Diese wiederum müssen jedoch alle einzeln abgerufen und geprüft werden. Das erhöht den Wartungsaufwand mit jedem weiterem T-Pot System immens. Als forschungsorientierter Honeypot lässt sich T-Pot zum vollem Potential nutzen, die große Anzahl an unterschiedlicher und spezifischer Honeypotsoftware, sowie die gute Analysemöglichkeit durch Kibana begünstigen diesen Einsatz.

Die Mehrheit des Programmcodes von T-Pot wurde in der Programmiersprache C geschrieben und zum Standpunkt dieser Arbeit gab es immer wieder Änderungen oder Erweiterungen des Programmcodes im T-Pot Repository, was darauf hindeutet, dass

weiterhin eine aktive Community an diesem Projekt arbeitet. Dies begünstigt eventuelle zukünftige Bugfixes oder auch Erweiterungen und macht die Software langlebiger.

6.2 MHN

Als zweites Honeypotsystem wurde in dieser Arbeit „Modern Honey Network“ präsentiert und mit Angriffsszenarien getestet. Grundsätzlich handelt es sich bei MHN um eine Flask Applikation, welche in der Programmiersprache Python geschrieben wurde. Das Honeypotsystem lässt sich dementsprechend schnell installieren und ist direkt einsatzbereit. Im Gegensatz zu T-Pot setzt MHN auf eine Client-Server Architektur. Der MHN Server ist hauptsächlich für das Dateimanagement und das einsammeln der Daten der weiteren Honeypots zuständig. Die eigentlichen Honeypots werden auf anderen Systemen als Sensoren installiert und senden ihre gesammelten Daten an den Server.

Dies wird alles über die Weboberfläche des MHN-Servers gesteuert. Aufgrund der einfachen Bedienbarkeit ist es schnell möglich, eine größere Anzahl an Sensoren im Netzwerk zu verteilen. Das System skaliert dahingehend sehr gut. Für die Installation der Honeypots bringt MHN mehrere Scriptvorlagen mit, welche direkt genutzt werden können. Der Client, auf dem die Honeypotsoftware installiert werden soll, lädt sich das Script vom MHN Server herunter und führt es aus, um sich die entsprechenden Dateien der zu installierenden Honeypotsoftware herunterzuladen und stellt eine Verbindung zum MHN Server her. Es ist auch möglich, eigene Skripte zu schreiben, um auch andere Honeypotsoftware einsetzen zu können.

Die Weboberfläche des MHN-Servers ist minimalistisch gehalten und bietet neben einer Übersicht der Angriffe auch eine Weltkarte, um externe Angriffe einem Land zuordnen zu können. Darüber hinaus lassen sich die gesammelten Daten der einzelnen Sensoren in tabellarischer Ansicht anzeigen. Es möglich, den MHN Server mit Splunk oder ArcSight zu verknüpfen. Splunk und ArcSight bieten eine erweiterte Oberfläche ähnlich wie Kibana, um die gesammelten Daten in Grafiken und Diagrammen anzeigen zu können.

Das Risiko einer Kompromittierung ist abhängig von der eingesetzten Honeypotsoftware. Die meisten dieser verteilten Honeypots sind Low-Interaction Honeypots, weshalb eine Kompromittierung des Systems sehr gering ist. Allerdings werden die Honeypots nicht wie bei T-Pot in einer eigenen Docker-Umgebung installiert, weshalb das Sicherheitsrisiko im Vergleich dazu als etwas höher einzustufen ist.

Die Einsatzmöglichkeiten des MHN Honeypotsystems gestalten sich sehr variabel. Durch die schnelle Verteilung von mehreren Sensoren in einem Netzwerk eignet sich MHN hervorragend zum Einsatz in einem Produktionsumfeld, um zum Beispiel verschiedene Sensoren in unterschiedlichen Netzwerkbereichen zu platzieren. Damit lassen sich die

vorgestellten Verteilungsstrategien von Honeypots in dem Unterkapitel in Sektion 2.4 gut umsetzen.

Wie zu Beginn des Unterkapitels erwähnt, wurde der Großteil der MHN Software in der Programmiersprache Python verfasst. Hier ist wichtig zu erwähnen, dass zum Zeitpunkt dieser Arbeit Python der Version 2.7 genutzt wurde. Python 2.7 wurde im Januar 2020 von der Python Software Foundation als veraltet deklariert und seitdem werden keine neuen Updates oder Bugfixes mehr erfolgen [29]. Dies ist ein Nachteil, der beachtet werden sollte. Solange der Code nicht auf Python 3 migriert, stellt dies ein gewisses Sicherheitsrisiko dar.

6.3 Gegenüberstellung

In Sektion 6.1 und 6.2 wurden die beiden untersuchten Honeypotsysteme hinsichtlich der Erfüllung der genannten Bewertungskriterien analysiert. Die Ergebnisse werden in Tabelle 6.1 zusammengefasst. Die Bewertung bezieht sich dabei auf den in der vorliegenden Arbeit zu Grunde gelegten Aufbau der Systeme, mit dem sich diese gut vergleichen ließen.

Kriterium	**T-Pot**	**MHN**
Installations- und Wartungsaufwand	Sehr Niedrig	Niedrig
Skalierbarkeit	Mittel	Hoch
Erweiterungsmöglichkeiten	Mittel	Mittel
Maßnahmen gegen Kompromittierung	Hoch	Mittel
Einsatzmöglichkeiten	Mittel	Mittel
Langlebigkeit	Hoch	Niedrig

Tabelle 6.1: Bewertungstabelle

6.3.1 Erläuterung der Bewertung

Die Beweggründe der Bewertung in der Tabelle 6.1 werden wie nachfolgend erläutert.

Installations- und Wartungsaufwand

Der Installations- und Wartungsaufwand ist bei beiden Honeypotsystemen marginal. Beide Systeme sind darauf ausgelegt, schnell Einsatzbereit zu sein und durch die Natur von Low-Interaction Honeypots gestaltet sich auch der Wartungsaufwand als gering. T-Pot hat in dieser Hinsicht einen Vorteil, da durch die Integration von Docker und bereits implementierten Skripten ein automatischer Neustart einer der Docker-Umgebungen initiiert wird, wenn ein Honeypotsystem ausfallen sollte. Daher ist der Wartungsaufwand geringer als bei MHN.

Skalierbarkeit

Das Kriterium Skalierbarkeit betrachtet die Option, das Honeypotsystem in mehrfacher Anzahl einzusetzen, um weitere Bereiche abzudecken oder ähnliches. Hierbei glänzt MHN durch seine Client-Server Architektur und die Möglichkeit, beliebig viele Sensoren im Netzwerk zu verteilen, während die Daten weiterhin an einem Knoten gesammelt und ausgewertet werden. Daher erfüllt MHN dieses Kriterium sehr gut. Theoretisch ist es möglich, auch weitere T-Pot Systeme im Netzwerk zu installieren, aber wie bereits in der Zusammenfassung von Abschnitt 6.1 erwähnt wurde, würde dies mit jedem weiterem T-Pot System auch den Wartungsaufwand erhöhen, da die Daten jeweils auf dem T-Pot System gespeichert werden und bei jedem einzeln abgerufen werden müssten.

Erweiterungsmöglichkeiten

MHN und T-Pot bieten beide die Möglichkeit an, das bestehende System um weitere Softwarepakete zu erweitern. So bietet MHN die Option an, die Software Splunk und ArcSight mit MHN zu verknüpfen, um eine alternative Weboberfläche und Datenauswertung zu installieren. Weiterhin können die Script-Vorlagen zur Verteilung der Honeypotsensoren um eigens geschriebene Scripte erweitert werden. Mit diesen ist es möglich, beliebige Honeypotsoftware durch solche eigenen Scripte an Sensoren zu verteilen. T-Pot bietet im Vergleich die Möglichkeit einer Anbindung an das Softwaretool hpfeeds. Des weiteren wird die Option gegeben, T-Pot mit Ansible oder Terraform zu verknüpfen, um T-Pot auf einem Cloudserver wie zum Beispiel den Amazon Web Services (AWS) oder der Open Telekom Cloud (OTC) zu nutzen. Beide Systeme können somit in einem gewissen vorgegebenen Rahmen erweitert werden. Für beide wurde daher die Erweiterungsmöglichkeit als mittel eingestuft.

Maßnahmen gegen Kompromittierung

Beide Honeypotsysteme nutzen Low-Interaction Honeypots, weshalb das Risiko einer Kompromittierung des Systems schon von Anfang an als sehr gering einzustufen ist. Dennoch fängt T-Pot dieses Risiko noch weiter ab, in dem die installierten Honeypots in einer Docker Umgebung arbeiten, welches eine weitere Absicherung darstellt. Sollte eine der Honeypotsoftwares von einem Angreifer kompromittiert werden, so wären nachfolgende Handlungen des Angreifers weiterhin auf die Docker Umgebung beschränkt. Ohne weitere individuelle Anpassung ist dies bei MHN nicht der Fall, weshalb hier ein Unterschied der Sicherheitsmaßnahmen erwähnenswert ist.

Einsatzmöglichkeiten

Hierbei sind sich beide Honeypotsysteme ebenbürtig, jedoch konzentrieren diese sich in unterschiedlichen Ausprägungen. Betrachtet wird hier vor allem der Unterschied zwischen eines Einsatzes in einer Forschungsumgebung und einer Produktionsumgebung. T-Pot scheint sich hierbei eher auf den Forschungsaspekt zu konzentrieren. Die große Auswahl an teilweise sehr speziellen Honeypots, welche T-Pot nutzt, eignen sich hervorragend um im Rahmen der Forschung auch ebenso spezielle Malware zu finden und zu analysieren, während MHN sich durch seine Verteilung von Sensoren besser für Produktionsnetze eignet. Natürlich lassen sich beide Honeypotsysteme auch in der jeweils anderen Umgebung einsetzen, allerdings mit gewissen Nachteilen. In einem Produktionsumfeld ist der Umfang der Honeypots ein geringfügiger Nachteil, da selten alle Honeypots gleichermaßen beansprucht werden würden. Im Gegenzug wäre das Aufsetzen eines passenden Honeypotsensors mit MHN in einem Forschungsumfeld aufwändiger als der Einsatz von T-Pot, da die Honeypotsoftware für den Sensor einzeln nacheinander installiert werden müsste. Diese Unterschiede sind natürlich gering, aber prägend genug um diese zu erwähnen und somit einen Unterschied zwischen den beiden Honeypotsystem zu ziehen.

Langlebigkeit

Bei der Betrachtung der Langlebigkeit der Software schneidet MHN im Vergleich zu T-Pot deutlich schlechter ab. Die letzte Änderung an der MHN-Software auf dem zuständigen Repository in GitHub ist zum Zeitpunkt dieser Arbeit bereits 14 Monate her. Des weiteren wurde der Python Code in Python 2.7 verfasst, welche wie bereits in Abschnitt 6.2 erwähnt wurde, offiziell im Januar 2020 als veraltet deklariert wurde. Somit werden keine offenen Bugs der Programmiersprache dieser Version behoben. Dies kann ein Sicherheitsrisiko darstellen, solange die Software nicht auf Python 3 migriert wird. Im Kontrast dazu wird zum Zeitpunkt dieser Arbeit weiter an T-Pot gearbeitet, wie sich aus den aktualisierten Änderungen am Programmcode im Repository der Software ablesen lässt. Dieser Punkt, zusammen mit der Tatsache, dass der Programmcode von T-Pot in noch aktuellen Programmiersprachen geschrieben wurde, haben zu einer hohen Bewertung bewogen.

Hierbei muss individuell entschieden werden, ob dieses Risiko im Einzelfall einzugehen ist oder nicht. In Betrachtung dieses Vergleichs wurde dieser Aspekt dennoch mit einbezogen, da aus Sicht der IT-Sicherheit die Langlebigkeit von Software eine große Rolle spielt.

6.3.2 Fazit

In dieser Arbeit wurden zwei Honeypotsysteme vorgestellt und deren Arbeitsweise anhand von Angriffsszenarien vorgeführt. Die Honeypotsysteme wurden in einer lokalen Laborumgebung installiert, zusammen mit einem Client, mit dem die Angriffsszenarien durchgeführt wurden. Die Angriffsszenarien haben gezeigt, wie Hacker bei einem Cyberangriff vorgehen, wobei sich hier auf die aktive Reconnaissance beschränkt wurde. Dies konnte aufzeigen, welche unterschiedlichen Sicherheitslücken Honeypots präsentieren können um Angreifer in eine Falle zu locken und wie die beiden Honeypotsysteme die Daten protokollieren. Abschließend wurde ein Vergleich mittels eigener Bewertungskriterien durchgeführt. T-Pot erhielt in dieser Arbeit die bessere Bewertung, ausschlaggebend durch den Punkt der Langlebigkeit. Sollte für einen Einzelfall dieser Aspekt keine große Bedeutung spielen, so sind die beiden Honeypotsysteme als nahezu ebenbürtig einzustufen. Für ein beispielhaftes Unternehmen, welches lediglich weitere Sensoren innerhalb oder außerhalb des Netzwerkes platzieren möchte, würde sich MHN gut eignen. Forschungsbereiche der IT, welche sich beispielsweise mit Malwaretrends auseinandersetzen, träfen mit T-Pot die bessere Wahl, da hier einfacher spezielle Malware für unterschiedliche Dienste eingefangen und analysiert werden kann.

6.3.3 Ausblick auf weitere mögliche Forschungsthemen im Bereich der Honeypotsoftware

Zukünftige Arbeiten in diesem Bereich könnten anstelle einer Laborumgebung versuchen, Honeypots über einen langen Zeitraum echten Angriffen auszusetzen, um am Ende eine mögliche Trendanalyse oder ähnliches durchzuführen.

Die vorliegende Arbeit konzentriert sich auf Low-Interaction Honeypots. Eine ähnliche Untersuchung mit High-Interaction-Honeypots wäre ebenfalls interessant, hierbei könnte ein eigener High-Interaction Honeypot entworfen werden, um über einen Zeitraum einen tieferen Einblick in aktuelle Malware zu geben.

7 Reflexion und selbstkritische Einschätzung der Arbeit

Das Ziel der Bachelorarbeit, zwei Honeypotsysteme vorzuführen und zu vergleichen, konnte erreicht werden. Hierbei konnte gut gezeigt werden, wie die Honeypot-Software arbeitet. Da es sich bei den protokollierten Daten der Angriffsszenarien um Testdaten gehandelt hat, konnten über die eigentlichen Datensätze nur wenig Aussagen getroffen werden. Spannend wäre hier sicherlich auch, mit „echten“ Datensätzen zu arbeiten, welche die Honeypots über einen längeren Zeitraum gesammelt haben. Somit wäre es möglich gewesen, die Software vorzustellen, aber auch einige Aussagen über die gesammelten Daten zu treffen.

Des Weiteren war ursprünglich geplant, den Fokus auf High-Interaction Honeypots zu legen, da diese aufgrund des größeren Interaktionsgrades einen spannenden Einblick in die Vorgehensweise von Angreifern geben können. Dies hätte auch ermöglicht, die Angriffsszenarien selbst in einer Testumgebung weiter auszubauen und weitere Aspekte von Hackerangriffen zu beleuchten. Dies war leider nicht möglich, da die gängige Software zu High-Interaction Honeypots zum Zeitpunkt der Arbeit entweder veraltet oder mit Lizenzgebühren verbunden war. Aktuelle Open-Source Software konnte zum Zeitpunkt der Arbeit leider nicht gefunden werden, daher wurde der Fokus auf Low-Interaction Honeypots gelegt. Dies ermöglichte - aufgrund dessen, dass Low-Interaction Honeypots im einzelnen eine geringe Interaktionsmöglichkeit bieten - hingegen das Betrachten ganzer Honeypot-Systeme, bei denen der Einsatz vielfältiger Honeypots möglich ist. Hierbei konnte der Vergleich auf der Ebene eines ganzen Honeypot-Systems erfolgen, was mit T-Pot und MHN auch sehr gut funktioniert hat.

Ein alternativer Ansatz wäre es auch gewesen, ein eigenes High-Interaction Honeypot-System zu entwickeln, wie es beispielsweise in der Arbeit von „TrendMicro“ vorgeführt wurde [4]. Ein derartig grundlegender Wechsel wäre allerdings aufgrund der bereits fortgeschrittenen Zeit zu dem damaligen Zeitpunkt leider nicht mehr möglich gewesen, ohne den Zeitrahmen der Arbeit deutlich zu überreizen.

Persönlich konnte ich im Zuge dieser Arbeit mein Wissen nicht nur vertiefen, sondern auch viel Neues lernen, da ich mich bisher noch nicht derartig intensiv mit Honeypots auseinandergesetzt habe. Auch das Einarbeiten in das Betriebssystem Kali Linux zusammen mit einigen gängigen Tools anhand eines Fachbuchs hat meine Kenntnisse in diesem Bereich aufgefrischt und erweitert. Ein spannender Prozess war für mich zudem die anfängliche Recherche, sowie das Ausarbeiten der These.

Literaturverzeichnis

[1] Norton Security Center, „What is a honeypot? How it can lure cyberattackers“. Zuletzt aufgerufen am 30.08.2020 von https://us.norton.com/internetsecurity-iot-what-is-a-honeypot.html

[2] Lance Spitzner, Marty Roesch, „The value of honeypots, part one: Definitions and values of honeypots, october 10, 2001“. Zuletzt aufgerufen am 02.08.2020 von https://www.symantec.com/connect/articles/value-honeypots-part-one-definitions-and-values-honeypots

[3] B.Schneier, „Secrets & Lies: Digital Security in a Networked World “, Erste Auflage, New York, NY, USA: John Wiley & Sons, Inc., 2000

[4] Trend Micro Research, „Caught in the Act: Running a realistic factory honeypot to capture real threats.“,Zuletzt geprüft am 02.08.2020 von https://documents.trendmicro.com/assets/white_papers/wp-caught-in-the-act-running-a-realistic-factory-honeypot-to-capture-real-threats.pdf

[5] Infosec, „How Malware Detects Virtualized Environment(and its Countermeasures)“, Zuletzt geprüft am 02.08.2020 von https://resources.infosecinstitute.com/how-malware-detects-virtualized-environment-and-its-countermeasures-an-overview/

[6] Marcin Nawrocki, Matthias Wählisch, Thomas C. Schmidt, Christian Keil and Jochen Schönfelder, „A Survey on Honeypot Software and Data Analysis“. Zuletzt aufgerufen am 02.08.2020 von https://arxiv.org/pdf/1608.06249.pdf

[7] Vrije Universiteit Amsterdam, „Argos: An emulator for capturing zero-day attacks “. Zuletzt aufgerufen am 02.08.2020 von https://www.few.vu.nl/argos/?page=1

[8] The Honeynet Project, „Sebek Homepage“. Zuletzt aufgerufen am 02.08.2020 von http://honeynet.onofri.org/tools/sebek/

[9] The Honeynet Project, „Honeywall CDROM“. Zuletzt aufgerufen am 02.08.2020 von https://www.honeynet.org/projects/old/honeywall-cdrom/

[10] Müter, Michael & Freiling, Felix & Holz, Thorsten & Matthews, Jeanna. (2008). „A generic toolkit for converting web applications into high-interaction honeypots.“, Universität Mannheim

[11] Honeynet.org, „The Honeynet Project“. Zuletzt aufgerufen am 02.08.2020 von https://www.honeynet.org/

[12] Udacity(2016): Honeypot Deployment, veröffentlicht am 06.06.2016, Zuletzt aufgerufen am 15.08.2020 von https://www.youtube.com/watch?v=FBnTeryebzc

[13] Shodan.io, Shodan, Zuletzt aufgerufen am 15.08.2020 von https://www.shodan.io/

[14] DTAG Community Honeypot Project, T-Pot. Zuletzt aufgerufen am 15.08.2020 von http://dtag-dev-sec.github.io/

[15] Modern Honey Network, MHN. Zuletzt aufgerufen am 15.08.2020 von https://github.com/pwnlandia/mhn

[16] Kali Linux, Offensive Security. Zuletzt aufgerufen am 15.08.2020 von https://www.kali.org

[17] Proxmox VE,Proxmox Server Solutions GmbH. Zuletzt aufgerufen am 15.08.2020 von https://www.proxmox.com/de/ueber-uns

[18] T-Pot Übersicht, Deutsche Telekom AG. Zuletzt aufgerufen am 17.08.2020 von http://dtag-dev-sec.github.io/mediator/feature/2015/03/01/about.html

[19] ELK Stack, Elasticsearch, Logstash und Kibana. Apache Software Foundation. Zuletzt aufgerufen am 17.08.2020 von https://www.elastic.co/de/what-is/elk-stack

[20] hpfeeds, „Honeynet Project generic authenticated datafeed Protocol. “. Zuletzt aufgerufen am 18.08.2020 von https://github.com/hpfeeds/hpfeeds

[21] MongoDB, MongoDB, Inc. Zuletzt aufgerufen am 17.08.2020 von https://www.mongodb.com/

[22] Robert W. Beggs, „Mastering Kali Linux for Advanced Penetration Testing “, Birmingham B3 2PB, UK: Packt Publishing, Juni 2014.

[23] Exploit DB, Offensive Security. Zuletzt Aufgerufen am 19.08.2020 von https://www.exploit-db.com/

[24] Common Vulnerabilities and Exposures, Wikipedia.org. Zuletzt Aufgerufen am 19.08.2020 von https://de.wikipedia.org/wiki/Common_Vulnerabilities_and_Exposures

[25] OpenVAS - Open Vulnerability Assessment Scanner, Greenbone Networks GmbH. Zuletzt Aufgerufen am 20.08.2020 von https://www.openvas.org/index-de.html

[26] nessus - nessus Vulnerability Scanner, Tenable Inc., Zuletzt Aufgerufen am 19.08.2020 von https://de.tenable.com/products/nessus

[27] OS-Fingerprinting, Wikipedia.org, Zuletzt Aufgerufen am 19.08.2020 von https://de.wikipedia.org/wiki/OS-Fingerprinting

[28] Nmap, Introduction, nmap.org, Zuletzt Aufgerufen am 20.08.2020 von https://nmap.org/

[29] „Sunsetting Python 2“,python.org. Python Software Foundation. Zuletzt Aufgerufen am 28.08.2020 von https://www.python.org/doc/sunset-python-2/

Glossar

False-Positives Ein False-Positive liegt vor, wenn eine Übereinstimmung zwischen Ereignis und Kriterium gefunden wurde, obwohl diese eigentlich nicht gegeben sind.

Fingerprinting Identifizierung von Anwendungen und Diensten.

Flask Python Framework zur Webentwicklung.

GitHub Netzbasierter Dienst zur Versionsverwaltung mit dem Werkzeug Git.

Header Kopf; Kopfteil. Ein IPv4-Header enthält Steuer- und Protokollinformationen.

Payload Nutzdaten eines Datenpakets.

Repository Verzeichnisstruktur der Versionsverwaltungssoftware Git.

Webcrawler Programm, welches automatisiert das Internet durchsucht.

Zero-Day Exploit Ausnutzen einer Schwachstelle, welche bisher noch unbekannt war oder ist.

Anhang A: Anhang

A.1 T-Pot Architektur

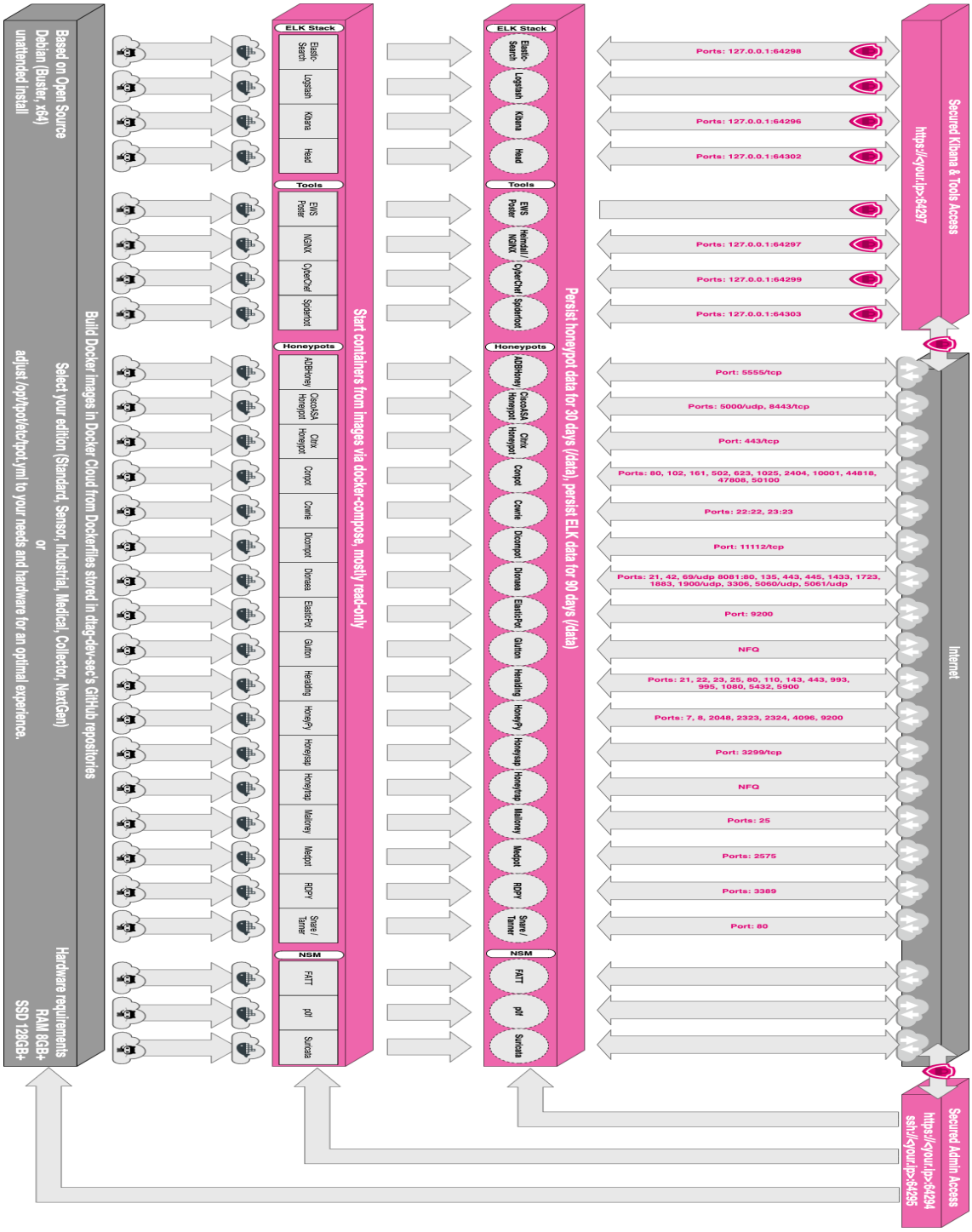


Abbildung A.1: „T-Pot Architektur.“ DTAG Community Honeypot Project, T-Pot. URL: https://github.com/dtag-dev-sec/tpotce/blob/master/doc/architecture.png, zuletzt Aufgerufen am 17.08.2020.

A.2 T-Pot Landing Page

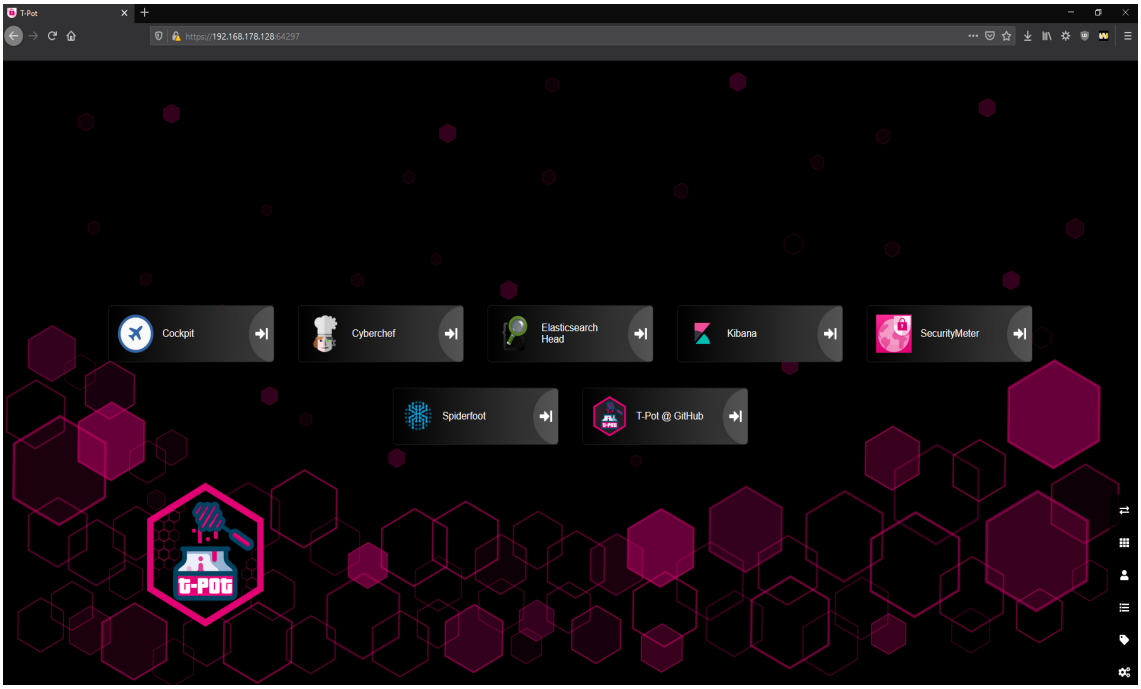


Abbildung A.2: „T-Pot Landing Page.“ Quelle: Screenshot aus dem Browserfenster.

A.3 MHN Dashboard

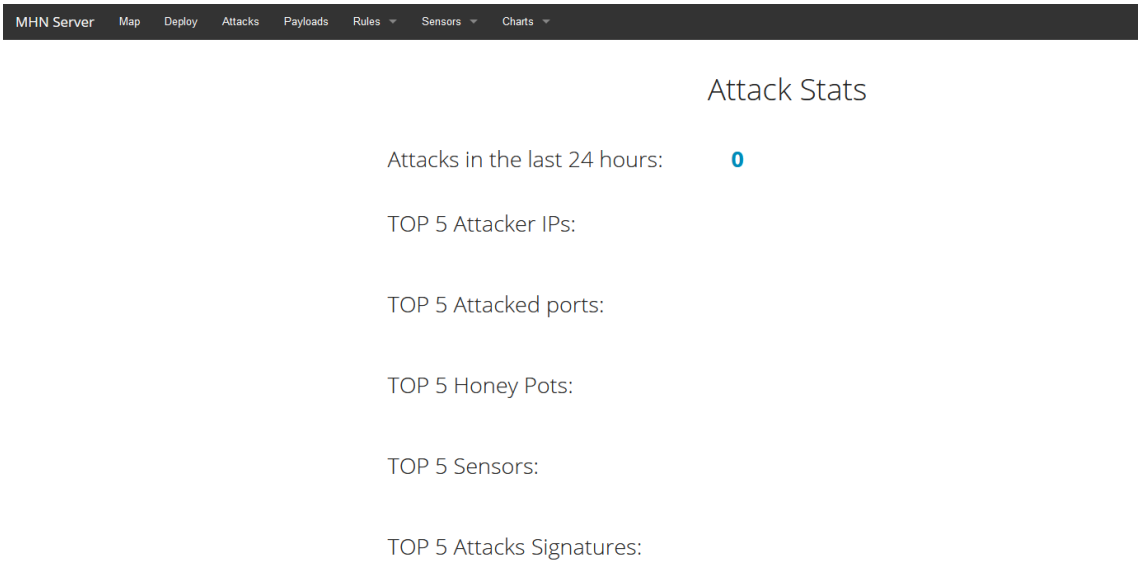


Abbildung A.3: „MHN Dashboard“ Quelle: Screenshot aus dem Browserfenster.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 31.08.2020