IFA - Höhere Fachschule für IT und Digital Business

Boot-to-Root Penetrationstest (Pwned: 1)

Challenge der IFA - Höhere Fachschule für IT und Digital Business

Verfasser	Luca Knobel Mühlerain 14, 3210 Kerzers knobel.luca@gmail.com
Studiengang	Dipl. Informatiker HF, Schwerpunkt Application Security
Klasse	HFINFA.X.BA.5_6.25-ML-S2510-APST.TS1A
Examinator	Roman Camenzind
Abgabedatum	24.03.2026

1 Executive Summary

Im Rahmen dieser Sicherheitsanalyse wurde die bereitgestellte Boot2Root-Umgebung *Pwned: 1* systematisch untersucht. Ziel war es, Schwachstellen zu identifizieren, auszunutzen und einen vollständigen Zugriff auf das System bis hin zu administrativen Rechten (Root) zu erlangen.

Die Analyse zeigt, dass das System mehrere kritische Sicherheitsmängel aufweist, die in Kombination eine vollständige Kompromittierung ermöglichen. Ein Angreifer kann ohne grossen technischen Aufwand initialen Zugriff erlangen und diesen schrittweise bis zu vollständigen Systemrechten ausbauen.

Zentrale Ergebnisse Im Rahmen des Tests wurden insbesondere folgende kritische Schwachstellen identifiziert:

- **Im Quellcode hinterlegte Zugangsdaten:** Zugangsdaten sind direkt in der Webanwendung gespeichert und können ohne Authentifizierung ausgelesen werden.
- **Offen zugänglicher privater SSH-Schlüssel:** Ein privater Schlüssel ist ungeschützt abrufbar und ermöglicht direkten Systemzugriff.
- **Unsichere Verarbeitung von Benutzereingaben:** Eine Schwachstelle erlaubt die Ausführung beliebiger Systembefehle.
- **Fehlkonfiguration der Docker-Berechtigungen:** Durch unzureichende Zugriffsbeschränkungen kann ein Angreifer administrative Rechte (Root) erlangen.

Zusätzlich wurden weitere Schwachstellen identifiziert, die die Angriffsfläche erheblich vergrössern, insbesondere öffentlich zugängliche interne Dateien sowie ein unsicher konfigurierter FTP-Dienst.

Risikobewertung Die identifizierten Schwachstellen weisen insgesamt ein hohes Risikoniveau auf. Mehrere Findings ermöglichen bereits einzeln eine schwerwiegende Kompromittierung, in Kombination führen sie zu einer vollständigen Übernahme des Systems. Besonders kritisch ist dabei die einfache Ausnutzbarkeit ohne besondere Vorkenntnisse.

Empfohlene Sofortmassnahmen Zur Reduktion des Risikos sollten prioritär folgende Massnahmen umgesetzt werden:

- Entfernung von Zugangsdaten aus dem Quellcode und Einsatz sicherer Speichermechanismen
- Schutz und sichere Verwaltung von SSH-Schlüsseln
- Behebung der unsicheren Eingabeverarbeitung
- Einschränkung von privilegierten Systemberechtigungen, insbesondere im Zusammenhang mit Docker

Fazit Die Analyse verdeutlicht, dass grundlegende Sicherheitsprinzipien im System nicht konsequent umgesetzt wurden. Bereits einfache Schwachstellen ermöglichen weitreichende Angriffe. Eine zeitnahe Umsetzung der empfohlenen Massnahmen ist erforderlich, um die Sicherheit des Systems nachhaltig zu verbessern.

2 Inhaltsverzeichnis

3 Scope

Dieses Kapitel definiert den Untersuchungsrahmen der Sicherheitsanalyse. Es beschreibt das Zielsystem, die Testart, die Testumgebung sowie die geltenden Einschränkungen.

3.1 Zielsystem

Das Zielsystem ist die bereitgestellte Boot2Root-VM *Pwned: 1*. Es handelt sich um eine absichtlich verwundbare Linux-Maschine mit einer Webanwendung und mehreren Netzwerkdiensten.

Ziel der Analyse ist es, Schwachstellen zu identifizieren, auszunutzen und schrittweise Zugriff bis hin zu Root-Rechten zu erlangen sowie die vorgesehenen Flags zu finden.

Die IP-Adresse des Zielsystems ist zu Beginn nicht bekannt und wird im Rahmen der Netzwerkanalyse ermittelt.

3.2 Testart

Der Test wird als Black-Box-Penetrationstest durchgeführt. Zu Beginn liegen keine Informationen über die interne Struktur, Benutzer oder Konfiguration des Systems vor. Die Analyse erfolgt ausschliesslich über erreichbare Netzwerkdienste und die Webanwendung.

Die Aufgabenstellung gibt eine grobe Vorgehensweise sowie empfohlene Tools vor. Diese dienen der methodischen Orientierung und stellen kein Vorwissen über das Zielsystem dar.

Das Vorgehen orientiert sich an den Phasen des OWASP Testing Guides.

3.3 Testumgebung

Die Analyse erfolgt in einer isolierten lokalen Testumgebung. Als Host-System wird Fedora Linux 43 mit KVM und *virt-manager* eingesetzt. Das Angreifer-System ist eine Kali Linux VM.

Das Zielsystem wird aus dem bereitgestellten OVA-Image importiert und in ein kompatibles Format für KVM konvertiert. Aufgrund von Unterschieden zwischen VirtualBox und KVM hinsichtlich der virtuellen Hardware (insbesondere der Netzwerkschnittstellen) ist eine initiale Anpassung innerhalb des Zielsystems erforderlich.

Zur Sicherstellung der Netzwerkkommunikation werden beide virtuellen Maschinen in ein dediziertes, isoliertes internes Netzwerk (*libvirt virtual network*) eingebunden. Dieses Netzwerk stellt einen privaten Adressbereich mit DHCP bereit, sodass beide Systeme automatisch IP-Adressen aus demselben Subnetz beziehen können.

Da das Zielsystem die virtuelle Netzwerkschnittstelle nach der Migration nicht automatisch erkennt, wird diese manuell aktiviert und für DHCP konfiguriert. Dadurch ist das Zielsystem im selben Layer-2-Netzwerk erreichbar und kann mit dem Angreifer-System kommunizieren.

Die Ziel-IP wird dynamisch vergeben und im Rahmen der Reconnaissance-Phase identifiziert.

3.4 Einschränkungen

Die Analyse beschränkt sich auf das bereitgestellte Zielsystem. Es werden keine Angriffe auf externe Systeme durchgeführt.

Auf Denial-of-Service-Angriffe wird verzichtet, da diese primär die Verfügbarkeit des Systems beeinträchtigen und nicht zur Erlangung von Zugriff beitragen. Zudem besteht das Risiko, die Stabilität der Testumgebung zu gefährden und die weitere Analyse zu beeinträchtigen.

Das Zielsystem ist bewusst verwundbar und für Ausbildungszwecke konzipiert. Alle identifizierten Schwachstellen dürfen im Rahmen der Challenge ausgenutzt werden.

4 Methodik

Dieses Kapitel beschreibt die methodische Vorgehensweise der Sicherheitsanalyse. Es werden die verwendeten Standards, der Testansatz sowie die einzelnen Testphasen definiert.

4.1 Standards und Referenzen

Die Durchführung der Analyse orientiert sich am OWASP Web Security Testing Guide (WSTG). Dieser definiert strukturierte Testphasen und stellt einen anerkannten Standard für Web-Penetrationstests dar.

Die in der Aufgabenstellung vorgegebenen Phasen entsprechen weitgehend diesem Vorgehensmodell und werden als Grundlage für die strukturierte Durchführung des Penetrationstests verwendet.

4.2 Testansatz

Der Testansatz basiert auf einer Kombination aus automatisierten und manuellen Tests.

Automatisierte Tools dienen der effizienten Identifikation von offenen Diensten sowie potenziellen Schwachstellen. Die manuelle Analyse ermöglicht eine gezielte Verifikation der Ergebnisse sowie die tatsächliche Ausnutzung identifizierter Schwachstellen.

Dieses kombinierte Vorgehen entspricht gängigen Best Practices im Penetration Testing, da automatisierte Verfahren häufig Fehlalarme (False Positives) liefern, welche durch manuelle Prüfung validiert werden müssen.

4.3 Testphasen

Die Durchführung erfolgt in mehreren aufeinander aufbauenden Phasen. Diese folgen einem strukturierten Angriffsvorgehen gemäss OWASP Testing Guide und orientieren sich an einem realistischen Angriffsablauf.

4.3.1 Reconnaissance

In dieser Phase wird das Zielsystem identifiziert und grundlegende Informationen gesammelt. Dazu gehört insbesondere die Ermittlung der IP-Adresse sowie eine erste Analyse des Netzwerks.

Ziel ist es, eine Übersicht über die vorhandenen Systeme zu erhalten und das Angriffsziel eindeutig zu bestimmen. Diese Phase bildet die Grundlage für alle weiteren Schritte.

Die Reconnaissance entspricht der Informationsgewinnung im OWASP Testing Guide und ist notwendig, da zu Beginn keine Informationen über das Zielsystem vorliegen.

4.3.2 Enumeration

In der Enumeration-Phase werden die zuvor identifizierten Systeme detailliert analysiert. Dabei werden offene Ports, laufende Dienste sowie erreichbare Schnittstellen untersucht.

Ziel ist es, die Angriffsfläche des Systems zu verstehen und potenzielle Einstiegspunkte zu identifizieren.

Diese Phase baut direkt auf der Reconnaissance auf und vertieft die gewonnenen Informationen durch eine systematische Analyse.

4.3.3 Initial Access und Exploitation

Basierend auf den gesammelten Informationen werden mögliche Schwachstellen identifiziert und gezielt ausgenutzt.

Ziel ist es, einen ersten Zugriff auf das System zu erlangen. Dies kann beispielsweise durch unsichere Konfigurationen, exponierte Dateien oder fest codierte Zugangsdaten erfolgen.

Diese Phase stellt den Übergang von der passiven Analyse zur aktiven Kompromittierung des Systems dar und ist ein zentraler Bestandteil eines Penetrationstests.

4.3.4 Privilege Escalation

Nach erfolgreichem Erstzugriff wird versucht, die bestehenden Benutzerrechte zu erweitern. Dabei werden lokale Schwachstellen, Fehlkonfigurationen oder unsichere Berechtigungen ausgenutzt.

Ziel ist es, höhere Berechtigungen bis hin zu administrativen Rechten zu erlangen und somit eine vollständige Kontrolle über das System zu erreichen.

Im Rahmen dieser Phase erfolgt auch die Validierung der erfolgreichen Kompromittierung, beispielsweise durch den Zugriff auf geschützte Bereiche oder das Auslesen sensibler Daten.

Diese Phase umfasst somit sowohl die Rechteausweitung als auch die abschliessende Verifikation des Angriffs, wodurch eine vollständige Systemübernahme nachgewiesen wird.

5 Durchführung des Penetrationstests

Dieses Kapitel beschreibt den tatsächlichen Angriffspfad in chronologischer Reihenfolge entlang der Phasen Reconnaissance, Enumeration, Initial Access und Privilege Escalation. Ziel ist die nachvollziehbare Darstellung der identifizierten Schwachstellen sowie deren systematische Ausnutzung bis zur vollständigen Systemkompromittierung.

5.1 Reconnaissance

In dieser Phase wurde das Zielsystem im Netzwerk identifiziert sowie eine erste Analyse der erreichbaren Dienste durchgeführt. Die gewonnenen Informationen bilden die Grundlage für alle nachfolgenden Angriffsschritte.

Ziel Identifikation der Zielmaschine im Netzwerk sowie der erreichbaren Dienste als Grundlage für die weitere Analyse.

Vorgehen Zur Identifikation aktiver Hosts im lokalen Netzwerk wurde ein ARP-Scan durchgeführt:

```
1 sudo arp-scan -l
```

Quellcode 1: Ermittlung aktiver Hosts im lokalen Netzwerk

Anschliessend wurde ein vollständiger Port- und Dienstescan durchgeführt:

```
1 nmap -sV -sC -p- 192.168.100.167
```

Quellcode 2: Port- und Dienstescan des Zielsystems

Ergebnis Die Zielmaschine wurde unter der IP-Adresse `192.168.100.167` identifiziert. Der Scan ergab folgende erreichbare Dienste:

- **21/tcp** – FTP
- **22/tcp** – SSH
- **80/tcp** – HTTP

Die Ergebnisse des Scans sind in Abbildung 1 dargestellt.

Bewertung Der Webserver auf Port 80 stellt die primäre Angriffsfläche dar, da Webanwendungen häufig komplexe und potenziell verwundbare Logik enthalten.

Die zusätzlich erreichbaren Dienste FTP und SSH erweitern die Angriffsfläche und bieten potenzielle Einstiegspunkte für weiterführende Angriffe.

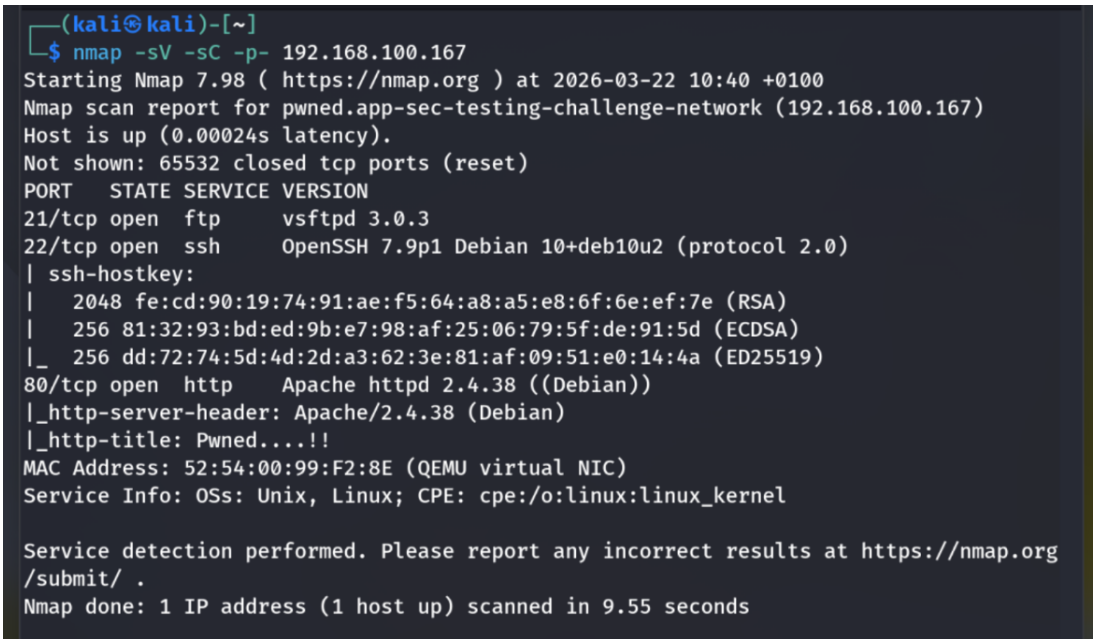

Abbildung 1: Ergebnis des Port- und Dienstescans
Eigene Darstellung

5.2 Enumeration

In der Enumeration-Phase wurden die identifizierten Dienste gezielt analysiert, um zusätzliche Informationen über potenzielle Angriffspunkte zu gewinnen. Der Fokus lag insbesondere auf dem Webserver.

Ziel Identifikation versteckter Ressourcen sowie Gewinnung zusätzlicher Informationen über potenzielle Angriffspunkte.

Vorgehen Zunächst wurde die Webanwendung manuell analysiert. Dabei wurde im HTML-Quellcode der Startseite (`http://192.168.100.167`) ein versteckter Hinweis identifiziert:

```
<!--
She sings very well. I loved it
-->
```

Quellcode 3: Relevanter Hinweis im HTML-Quellcode

Anschliessend wurde eine Directory Enumeration durchgeführt:

```
gobuster dir -u http://192.168.100.167 \
-w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt \
-x php,html,txt,old,bak
```

Quellcode 4: Directory Enumeration

Ergebnis Es wurden mehrere relevante Verzeichnisse identifiziert, insbesondere:

- `/hidden_text/`

- `/robots.txt`

Im Verzeichnis `/hidden_text/` wurde die Datei `secret.dic` gefunden, welche als Wortliste für weitere Enumeration-Schritte genutzt werden konnte.

Durch Verwendung dieser benutzerdefinierten Wortliste konnte der zuvor nicht auffindbare Pfad `/pwned.vuln` identifiziert werden. Dieser stellt eine Login-Oberfläche bereit (vgl. Abbildung 2).

Die Analyse des Quellcodes dieser Seite offenbarte fest codierte Zugangsdaten:

```
1 if ($un=='ftpuser' && $pw=='B0ss_B!TcH')
```

Quellcode 5: Hardcoded Credentials im Quellcode

Bewertung Die Speicherung von Zugangsdaten im Quellcode stellt eine kritische Schwachstelle dar, da diese ohne Authentifizierung extrahiert werden können.

Zusätzlich zeigt die Datei `secret.dic`, dass sensible Informationen ungeschützt im Webroot abgelegt wurden. Dies stellt eine Fehlkonfiguration dar und erleichtert gezielte Angriffe erheblich.

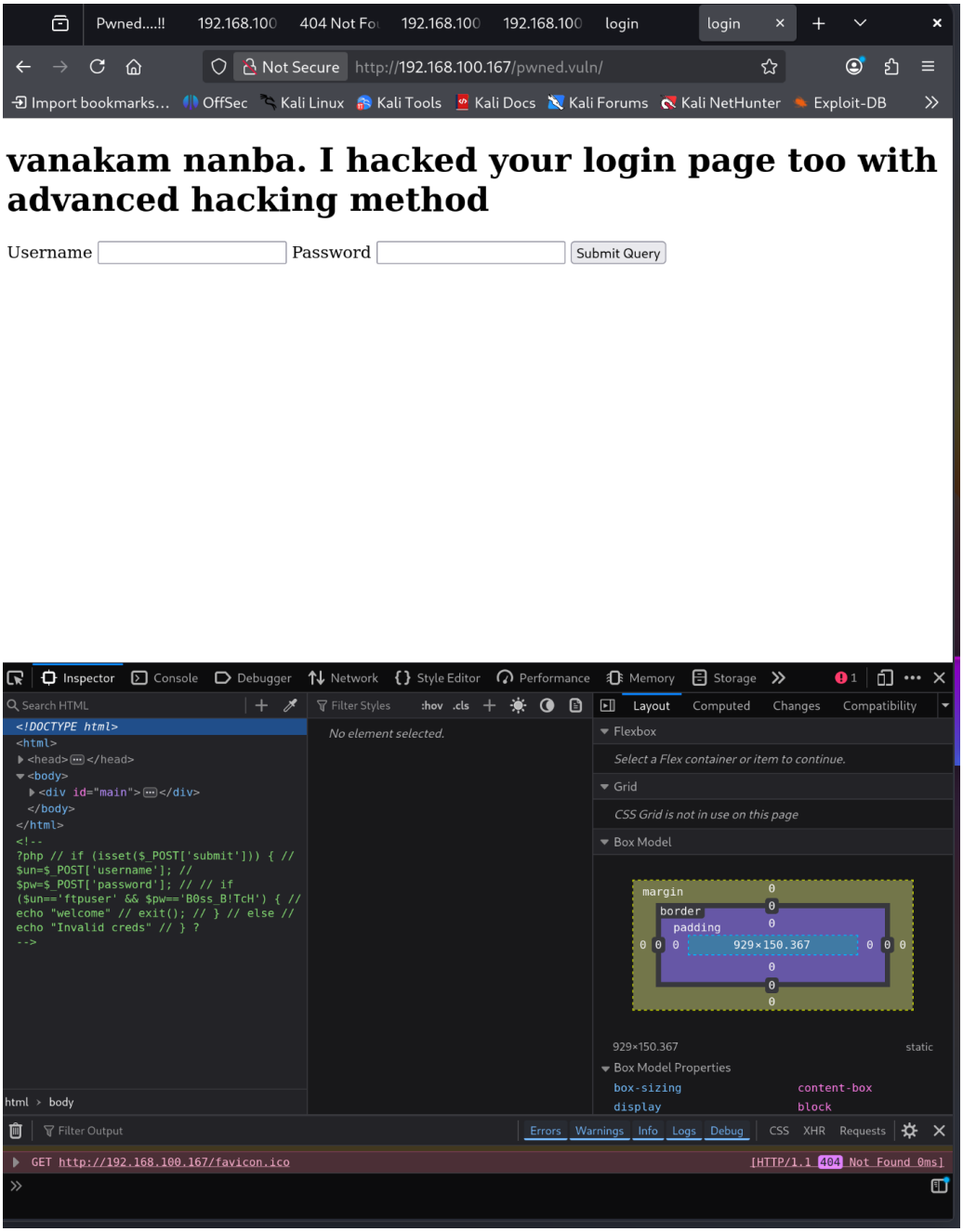

Abbildung 2: Identifizierte Login-Seite unter `/pwned.vuln`
Eigene Darstellung

5.3 Initial Access

Basierend auf den Ergebnissen der Enumeration wurde ein initialer Zugriff auf das Zielsystem angestrebt.

Ziel Erlangung eines initialen Zugriffs auf das Zielsystem unter Verwendung der identifizierten Zugangsdaten.

Vorgehen Die extrahierten Zugangsdaten wurden für einen Login über den FTP-Dienst verwendet:

```
ftp 192.168.100.167
```

Quellcode 6: Authentifizierung am FTP-Dienst

Nach erfolgreicher Anmeldung wurde im Dateisystem ein privater SSH-Schlüssel identifiziert und heruntergeladen:

```
get id_rsa
chmod 600 id_rsa
```

Quellcode 7: Download und Vorbereitung des SSH-Schlüssels

Anschliessend wurde der Zugriff über SSH durchgeführt:

```
ssh -i id_rsa ariana@192.168.100.167
```

Quellcode 8: SSH Zugriff mittels privatem Schlüssel

Ergebnis Der Zugriff als Benutzer `ariana` konnte erfolgreich hergestellt werden. Die erfolgreiche Authentifizierung ist in Abbildung 3 dargestellt.

Im Benutzerverzeichnis konnte die erste Flag ausgelesen werden, womit der initiale Zugriff bestätigt ist.

Bewertung Die Kombination aus wiederverwendeten Zugangsdaten und ungeschütztem privaten SSH-Schlüssel stellt eine schwerwiegende Sicherheitslücke dar.

Insbesondere die ungeschützte Ablage des privaten Schlüssels ermöglicht eine direkte Authentifizierung ohne zusätzliche Sicherheitsmechanismen und führt zu einer vollständigen Kompromittierung des Benutzerkontos.

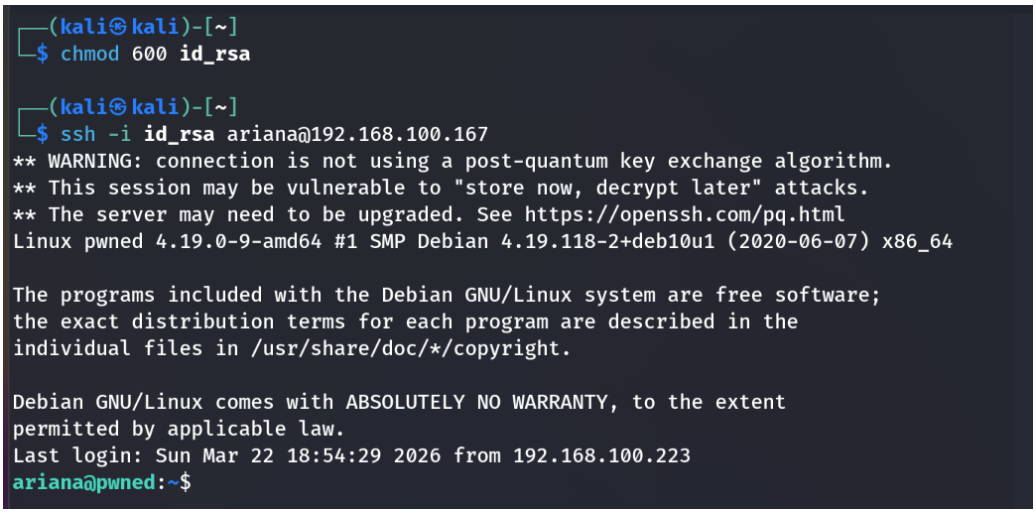

Abbildung 3: Erfolgreicher SSH-Zugriff als Benutzer `ariana`
Eigene Darstellung

5.4 Privilege Escalation

Nach erfolgreichem initialem Zugriff wurde eine Privilege Escalation durchgeführt, um administrative Rechte zu erlangen.

Ziel Eskalation der Benutzerrechte zur Erlangung administrativer (Root-)Rechte.

Vorgehen Zunächst wurden die verfügbaren sudo-Berechtigungen analysiert:

```
1 sudo -l
```

Quellcode 9: Analyse der sudo-Berechtigungen

Dabei wurde festgestellt, dass ein Script als Benutzer `selena` ausgeführt werden darf. Die Analyse des Scripts zeigte eine unsichere Verarbeitung von Benutzereingaben:

```
1 $msg 2> /dev/null
```

Quellcode 10: Unsichere Eingabeverarbeitung im Script

Dies ermöglichte eine Command Injection und somit die Ausführung beliebiger Befehle im Kontext von `selena`.

Anschliessend wurde festgestellt, dass dieser Benutzer Mitglied der Gruppe `docker` ist. Diese Fehlkonfiguration wurde zur Eskalation genutzt.

Ergebnis Durch die Ausnutzung der Docker-Gruppe konnte eine Root-Shell gestartet werden. Der erfolgreiche Zugriff ist in Abbildung 4 dargestellt.

Bewertung Die Kombination aus unsicherem Script (Command Injection) und der Mitgliedschaft in der Docker-Gruppe stellt eine kritische Fehlkonfiguration dar.

```
docker run -v /:/mnt --rm -it privesc chroot /mnt sh
```

Quellcode 11: Privilege Escalation über Docker

Die Docker-Gruppe ermöglicht einen indirekten Root-Zugriff, da Container mit Zugriff auf das Host-Dateisystem gestartet werden können. In Kombination mit der Command Injection führt dies zu einer vollständigen Kompromittierung des Systems mit administrativen Rechten.

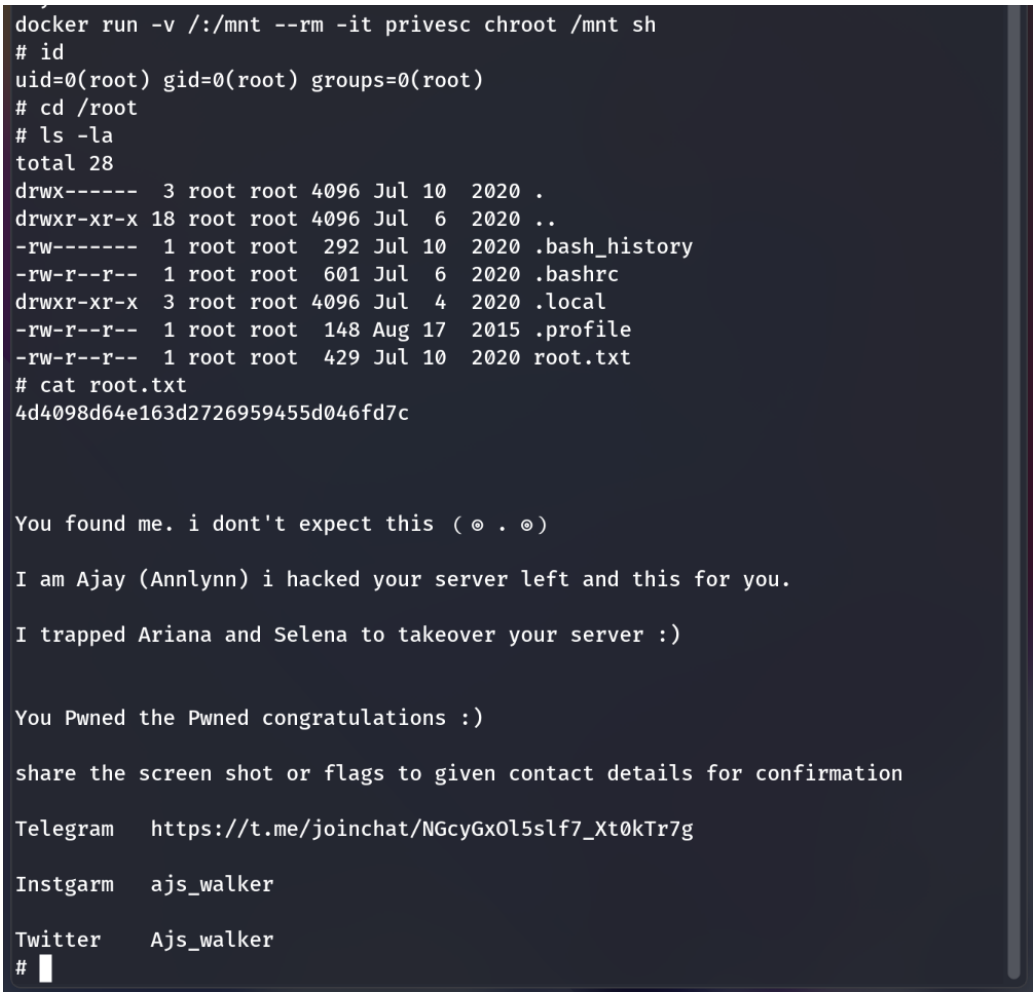

Abbildung 4: Erfolgreicher Zugriff auf Root-Ebene und Ausgabe des dritten Flags
Eigene Darstellung

6 Schwachstellenanalyse

Dieses Kapitel beschreibt die im Rahmen des Penetrationstests identifizierten Schwachstellen. Die Findings werden strukturiert dargestellt und hinsichtlich ihrer technischen Kritikalität (Severity) bewertet. Grundlage bildet der zuvor dokumentierte Angriffspfad (vgl. Kapitel 5).

6.1 Übersicht der identifizierten Schwachstellen

Die nachfolgende Tabelle gibt einen Überblick über alle identifizierten Schwachstellen. Die Sortierung erfolgt nach absteigender Kritikalität.

ID	Titel	Severity	Kurzbeschreibung
F-01	Hardcoded Credentials	Critical	Zugangsdaten sind direkt im Quellcode der Webanwendung hinterlegt und können ohne Authentifizierung extrahiert werden. Dies ermöglicht einen unmittelbaren Initial Access.
F-02	Offen zugänglicher SSH-Schlüssel	Critical	Ein privater SSH-Schlüssel ist ungeschützt zugänglich und kann direkt für eine Authentifizierung verwendet werden. Dies führt zu einer vollständigen Kompromittierung des Benutzerkontos.
F-03	Command Injection im Script	Critical	Unsichere Verarbeitung von Benutzereingaben ermöglicht die Ausführung beliebiger Systembefehle im Kontext eines privilegierten Benutzers.
F-04	Unsichere Docker-Konfiguration	Critical	Mitgliedschaft in der Docker-Gruppe erlaubt das Starten privilegierter Container mit Zugriff auf das Host-Dateisystem und ermöglicht Root-Zugriff.
F-05	Sensible Dateien im Webroot	High	Interne Dateien sind öffentlich zugänglich und ermöglichen eine gezielte Enumeration sowie das Auffinden versteckter Ressourcen.
F-06	Unsicherer FTP-Zugriff	High	Der FTP-Dienst erlaubt Zugriff mit bekannten Zugangsdaten und ermöglicht das Auslesen sensibler Dateien.

Tabelle 1: Übersicht der identifizierten Schwachstellen
Eigene Darstellung

Die Einstufung der Severity erfolgt auf Basis technischer Kriterien wie Ausnutzbarkeit, notwendige Voraussetzungen und potenzieller Systemimpact.

6.2 Detaillierte Schwachstellen

6.2.1 F-01 – Hardcoded Credentials

Schweregrad Critical

Beschreibung Im Quellcode der Webanwendung sind Zugangsdaten fest hinterlegt.

Auswirkung Ein Angreifer kann sich ohne weitere Authentifizierung direkt am System anmelden und Initial Access erlangen.

Reproduktion

- Zugriff auf Login-Seite
- Analyse des Quellcodes
- Extraktion der Credentials

Nachweis vgl. Abbildung 2

Massnahmen

- Keine Speicherung von Credentials im Code
- Einsatz von Secret Management

6.2.2 F-02 – Offen zugänglicher SSH-Schlüssel

Schweregrad Critical

Beschreibung Ein privater SSH-Schlüssel ist über den FTP-Dienst abrufbar.

Auswirkung Direkter Zugriff auf das System ohne Passwort.

Reproduktion

- Download des Schlüssels via FTP
- SSH-Login mit `id_rsa`

Nachweis vgl. Abbildung 3, 10 und 11

Massnahmen

- Schutz von SSH-Schlüsseln
- Verwendung von Passphrases

6.2.3 F-03 – Command Injection im Script

Schweregrad Critical

Beschreibung Ein Script verarbeitet Benutzereingaben unsicher und führt diese direkt aus.

Auswirkung Ausführung beliebiger Befehle im Kontext eines privilegierten Benutzers.

Reproduktion

- Ausführung via sudo
- Eingabe von `/bin/bash`

Nachweis vgl. Abbildung 15, 16 und 17

Massnahmen

- Input Validation
- Vermeidung von Shell-Ausführung

6.2.4 F-04 – Unsichere Docker-Konfiguration

Schweregrad Critical

Beschreibung Der Benutzer ist Mitglied der Docker-Gruppe und kann Container mit Host-Zugriff starten.

Auswirkung Vollständige Privilege Escalation zu Root.

Reproduktion

- Ausführen eines Containers mit `-v /:/mnt`
- Zugriff auf Root-Dateisystem

Nachweis vgl. Abbildung 4

Massnahmen

- Entfernen aus Docker-Gruppe
- Rootless Docker verwenden

6.2.5 F-05 – Sensible Dateien im Webroot

Schweregrad High

Beschreibung Interne Dateien sind öffentlich zugänglich.

Auswirkung Erleichtert Enumeration und Auffinden weiterer Angriffsvektoren.

Reproduktion

- Directory Enumeration
- Zugriff auf `/hidden_text/`

Nachweis vgl. Abbildung 6, 7 und 8

Massnahmen

- Entfernen sensibler Dateien
- Zugriffsbeschränkungen

6.2.6 F-06 – Unsicherer FTP-Zugriff

Schweregrad High

Beschreibung FTP erlaubt Zugriff mit bekannten Credentials.

Auswirkung Zugriff auf sensible Dateien möglich.

Reproduktion

- Login via FTP
- Dateidownload

Nachweis vgl. Abbildung 9 und 10

Massnahmen

- FTP deaktivieren oder ersetzen
- Zugriff einschränken

7 Nicht geprüfte Angriffsklassen

Im Rahmen der durchgeführten Analyse lag der Fokus auf der Identifikation und Ausnutzung konkreter Schwachstellen, welche zur Erlangung von Systemzugriffen führten.

Typische Webangriffe wie SQL Injection oder Cross-Site Scripting wurden nicht vertieft untersucht.

Dies ist darauf zurückzuführen, dass die analysierte Anwendung keine dynamische Eingabeverarbeitung oder erkennbare Datenbankanbindung aufweist.

Der Schwerpunkt der Untersuchung lag somit auf den tatsächlich ausnutzbaren Schwachstellen, welche zur vollständigen Systemkompromittierung führten.

8 Risikobewertung

Dieses Kapitel bewertet die identifizierten Schwachstellen hinsichtlich ihrer Eintrittswahrscheinlichkeit und ihrer potenziellen Auswirkungen. Ziel ist die Priorisierung der Risiken als Grundlage für geeignete Massnahmen.

8.1 Methodik

Die Risikobewertung erfolgt in Anlehnung an die OWASP Risk Rating Methodology. Ziel ist die strukturierte Bewertung der identifizierten Schwachstellen hinsichtlich ihrer Eintrittswahrscheinlichkeit (Likelihood) und ihrer potenziellen Auswirkungen (Impact).

Grundprinzip Das Risiko ergibt sich aus der Kombination von Wahrscheinlichkeit und Auswirkung:

$$Risk = Likelihood \times Impact$$

Likelihood (Wahrscheinlichkeit) Die Wahrscheinlichkeit beschreibt, wie einfach eine Schwachstelle ausgenutzt werden kann. Die Bewertung erfolgt qualitativ anhand folgender Kriterien:

- notwendiges Fachwissen des Angreifers
- Verfügbarkeit von Exploits oder Tools
- Komplexität der Ausnutzung
- erforderliche Voraussetzungen

Impact (Auswirkung) Die Auswirkung beschreibt die potenziellen Schäden bei erfolgreicher Ausnutzung:

- Vertraulichkeit (Confidentiality)
- Integrität (Integrity)
- Verfügbarkeit (Availability)

Bewertungsskala Die Bewertung erfolgt in drei Stufen:

- Low
- Medium
- High

8.2 Bewertung der identifizierten Risiken

ID	Likelihood	Impact	Risk	Begründung
F-01	High	High	High	Zugangsdaten sind direkt im Quellcode hinterlegt und ohne technische Hürden auslesbar. Ein Angreifer kann unmittelbar Zugriff erlangen.
F-02	High	High	High	Der private SSH-Schlüssel ist frei zugänglich und ermöglicht direkten Systemzugriff ohne Authentifizierung.
F-03	High	High	High	Die Command Injection erlaubt die Ausführung beliebiger Befehle im Kontext eines privilegierten Benutzers.
F-04	High	High	High	Durch die Docker-Gruppe kann ein Angreifer Root-Zugriff erlangen, was zu einer vollständigen Systemkompromittierung führt.
F-05	Medium	Medium	Medium	Öffentlich zugängliche Dateien erleichtern die Enumeration, führen jedoch nicht direkt zu einer Systemkompromittierung.
F-06	Medium	Medium	Medium	Der FTP-Zugriff ermöglicht das Auslesen von Dateien, stellt jedoch ohne weitere Schwachstellen keinen direkten Root-Zugriff dar.

Tabelle 2: Risikobewertung der identifizierten Schwachstellen
Eigene Darstellung

8.3 Risikoklassifizierung

Die Risikoklassifizierung erfolgt basierend auf der Kombination von Likelihood und Impact.

Risikostufe	Beschreibung
High	Schwachstellen mit hoher Wahrscheinlichkeit und schwerwiegenden Auswirkungen. Diese ermöglichen in der Regel eine vollständige Systemkompromittierung und erfordern sofortige Massnahmen.
Medium	Schwachstellen mit moderater Wahrscheinlichkeit oder begrenzten Auswirkungen. Diese sollten zeitnah behoben werden.
Low	Schwachstellen mit geringer Auswirkung oder schwer ausnutzbar. Behebung ist optional oder langfristig einzuplanen.

Tabelle 3: Klassifizierung der Risikostufen
Eigene Darstellung

9 Massnahmen

Dieses Kapitel beschreibt geeignete Massnahmen zur Behebung der identifizierten Schwachstellen. Die Empfehlungen sind nach Priorität gegliedert und orientieren sich an den zuvor bewerteten Risiken.

9.1 Sofortmassnahmen

Die folgenden Massnahmen sollten umgehend umgesetzt werden, da sie kritische Schwachstellen betreffen und eine unmittelbare Systemkompromittierung ermöglichen:

- **Entfernung von Hardcoded Credentials (F-01):** Zugangsdaten dürfen nicht im Quellcode gespeichert werden. Stattdessen sollen sichere Mechanismen wie Umgebungsvariablen oder Secret-Management-Lösungen eingesetzt werden.
- **Sicherung von SSH-Schlüsseln (F-02):** Private SSH-Schlüssel dürfen nicht öffentlich zugänglich sein. Schlüssel müssen geschützt gespeichert und zusätzlich mit einer Passphrase abgesichert werden.
- **Behebung der Command Injection (F-03):** Benutzereingaben müssen strikt validiert und bereinigt werden. Die direkte Ausführung von Eingaben in der Shell ist zu vermeiden.
- **Entzug von Docker-Rechten (F-04):** Benutzer dürfen nicht Mitglied der Docker-Gruppe sein, da dies Root-Zugriff ermöglicht. Alternativ ist der Einsatz von Rootless Docker zu prüfen.

9.2 Mittelfristige Massnahmen

Diese Massnahmen dienen der Verbesserung der Systemhärtung und der Reduktion der Angriffsfläche:

- **Absicherung des FTP-Dienstes (F-06):** Der FTP-Dienst sollte deaktiviert oder durch sichere Alternativen wie SFTP ersetzt werden. Zusätzlich sind starke Authentifizierungsmechanismen einzuführen.
- **Entfernung sensibler Dateien aus dem Webroot (F-05):** Interne Dateien dürfen nicht öffentlich zugänglich sein. Eine klare Trennung zwischen öffentlich erreichbaren und internen Ressourcen ist umzusetzen.
- **Minimierung von Benutzerrechten:** Das Prinzip der minimalen Rechtevergabe (Least Privilege) sollte konsequent angewendet werden, insbesondere für Systembenutzer und Dienste.
- **Einsatz von Logging und Monitoring:** Sicherheitsrelevante Aktionen sollten protokolliert und überwacht werden, um Angriffe frühzeitig erkennen zu können.

9.3 Langfristige Massnahmen

Zur nachhaltigen Verbesserung der Sicherheit sollten folgende organisatorische und technische Massnahmen etabliert werden:

- **Einführung sicherer Entwicklungsprozesse:** Sicherheitsaspekte sollten bereits in der Entwicklungsphase berücksichtigt werden (Secure Coding).

- **Integration von DevSecOps:** Sicherheitsprüfungen wie statische Codeanalyse oder Dependency-Checks sollten automatisiert in den Entwicklungsprozess integriert werden.
- **Regelmässige Penetrationstests und Audits:** Wiederkehrende Sicherheitsanalysen helfen, neue Schwachstellen frühzeitig zu identifizieren.
- **Verwendung von HTTPS:** Die Kommunikation sollte durch TLS verschlüsselt werden, um die Vertraulichkeit und Integrität der Daten zu gewährleisten.
- **Implementierung zusätzlicher Sicherheitsmechanismen:** Beispielsweise kann eine Content Security Policy (CSP) eingesetzt werden, um potenzielle Angriffsvektoren im Webkontext weiter zu reduzieren.

10 Fazit

Im Rahmen des Penetrationstests konnten mehrere kritische Schwachstellen identifiziert und erfolgreich ausgenutzt werden. Durch die Kombination von Hardcoded Credentials, ungeschützten SSH-Schlüsseln sowie unsicheren Systemkonfigurationen war eine vollständige Kompromittierung des Zielsystems bis hin zu Root-Rechten möglich.

Die Analyse zeigt, dass bereits grundlegende Sicherheitsmängel ausreichen, um weitreichende Angriffe zu ermöglichen. Insbesondere fehlende Zugriffskontrollen, unsichere Programmierpraktiken und unzureichende Systemhärtung stellen erhebliche Risiken dar.

Die durchgeführten Massnahmenempfehlungen adressieren diese Schwachstellen gezielt und zeigen auf, wie das Sicherheitsniveau des Systems nachhaltig verbessert werden kann.

Abschliessend wird deutlich, dass Sicherheit bereits in der Entwicklungs- und Betriebsphase berücksichtigt werden muss, um solche Schwachstellen frühzeitig zu vermeiden.

11 Abbildungsverzeichnis

12 Tabellenverzeichnis

13 Quellcodeverzeichnis

Anhang A Zusätzliche Evidenz

A.1 Reconnaissance

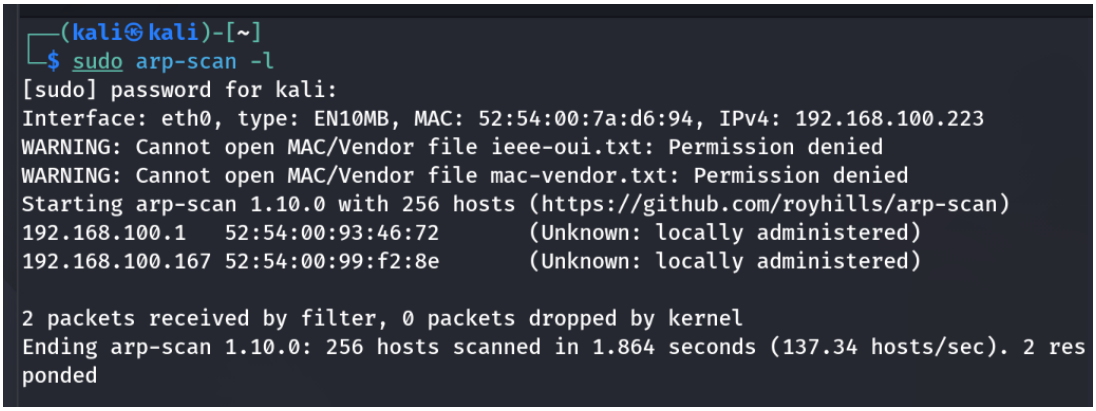

Abbildung 5: ARP-Scan zur Identifikation aktiver Hosts im lokalen Netzwerk als Grundlage zur Bestimmung der Ziel-IP-Adresse.
Eigene Darstellung

A.2 Enumeration

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.100.167 \
-w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt \
-x php,html,txt,old,bak
===============================================================
Gobuster v3.8.2
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.100.167
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirbuster/wordlists/directory-list-2.3-mediu
m.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.8.2
[+] Extensions:              php,html,txt,old,bak
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
index.html           (Status: 200) [Size: 3065]
robots.txt           (Status: 200) [Size: 41]
nothing              (Status: 301) [Size: 320] [--> http://192.168.100.167/nothing/]
server-status        (Status: 403) [Size: 280]
hidden_text          (Status: 301) [Size: 324] [--> http://192.168.100.167/hidden_te
xt/]
Progress: 1323348 / 1323348 (100.00%)
===============================================================
Finished
===============================================================
```

Abbildung 6: Ergebnis der Directory Enumeration mittels Gobuster zur Identifikation versteckter Verzeichnisse und Ressourcen.
Eigene Darstellung

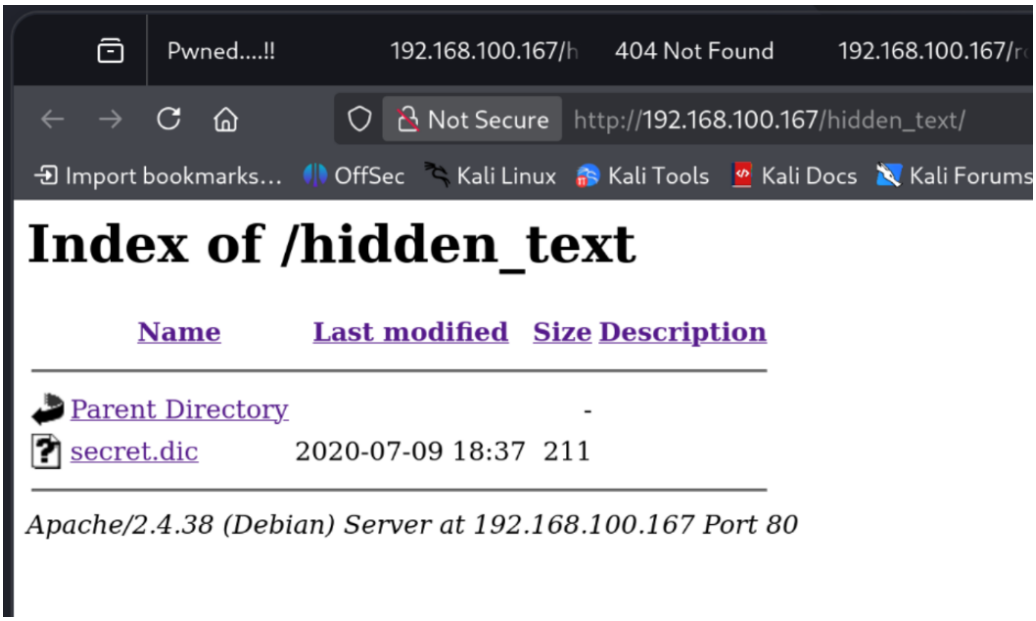

Abbildung 7: Öffentlich zugängliches Verzeichnis `/hidden_text/` mit internen Dateien als Hinweis auf eine Fehlkonfiguration.
Eigene Darstellung

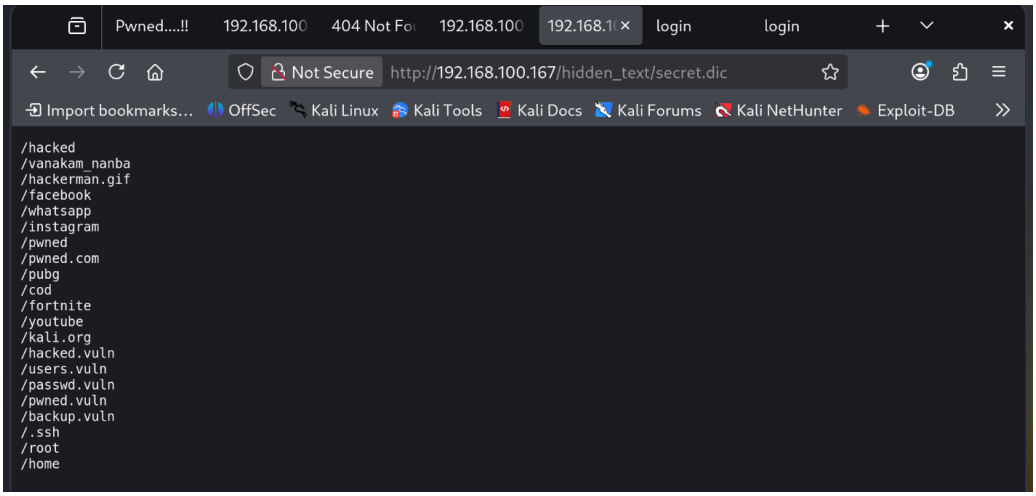

Abbildung 8: Inhalt der Datei `secret.dic`, genutzt als Wortliste für weiterführende Enumeration.
Eigene Darstellung

A.3 Initial Access

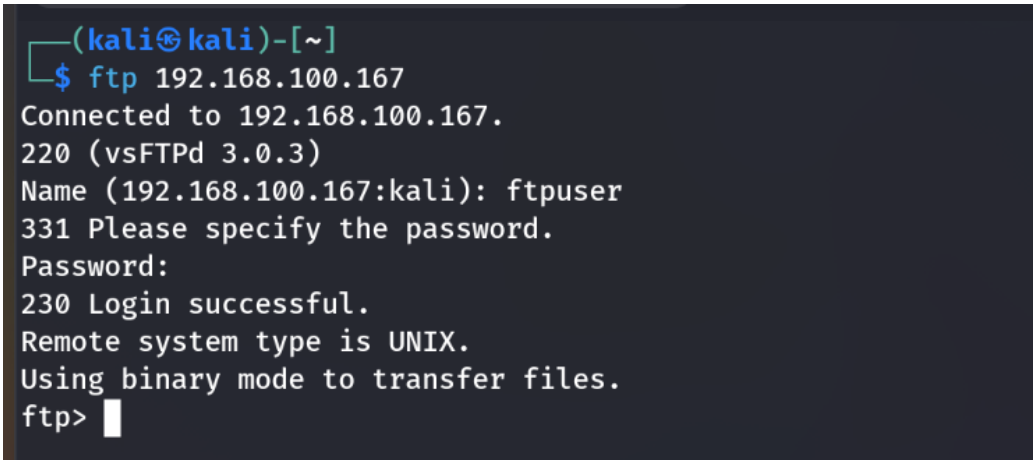

Abbildung 9: Erfolgreiche Authentifizierung am FTP-Dienst mit identifizierten Zugangsdaten.
Eigene Darstellung

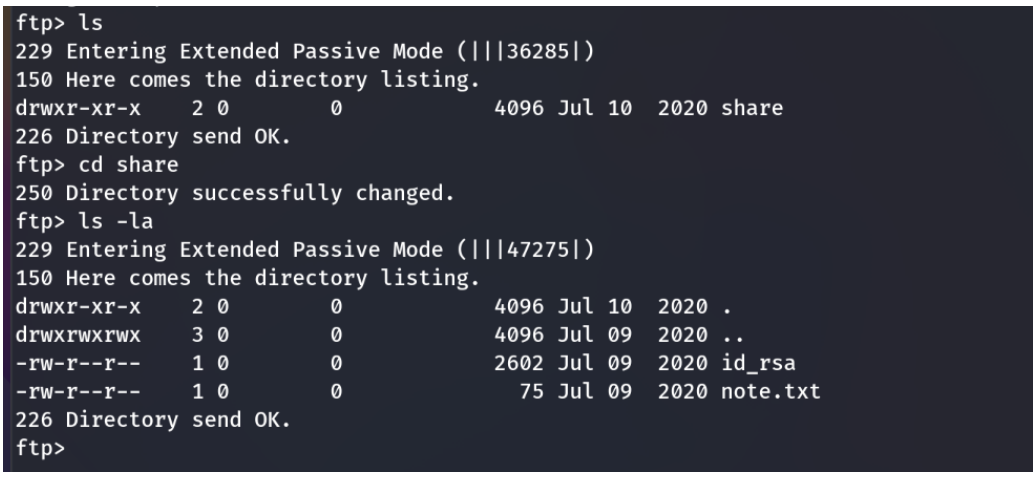

Abbildung 10: Verzeichnisstruktur des FTP-Servers mit relevanten Dateien, darunter ein privater SSH-Schlüssel.
Eigene Darstellung

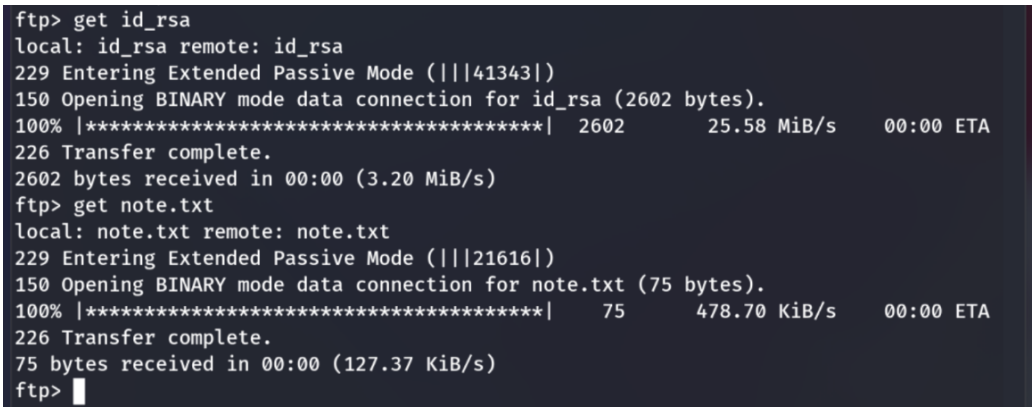

Abbildung 11: Download des privaten SSH-Schlüssels (`id_rsa`) vom FTP-Server.
Eigene Darstellung

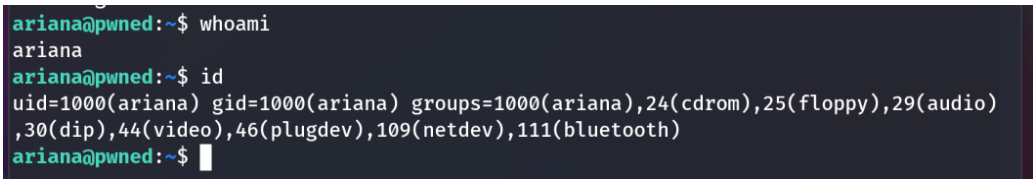

Abbildung 12: SSH-Sitzung als Benutzer `ariana`. Die Befehle `whoami` und `id` bestätigen den Zugriffskontext.
Eigene Darstellung

Abbildung 13: Auslesen der ersten Flag im Benutzerkontext `ariana` als Nachweis des Initial Access.
Eigene Darstellung

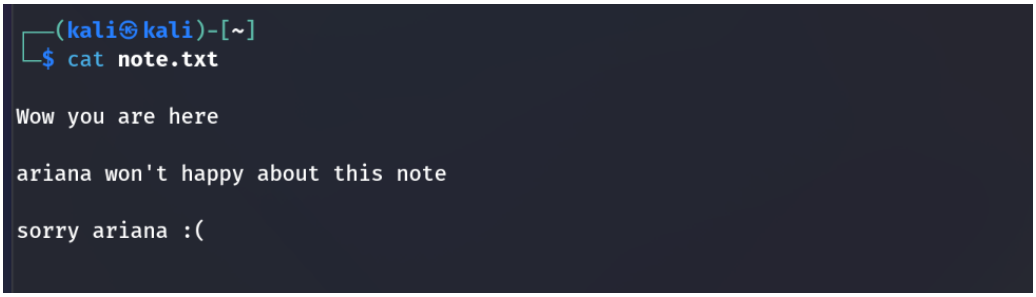

Abbildung 14: Ausgabe der Datei `note.txt` mit kontextuellen Hinweisen für die weitere Analyse.
Eigene Darstellung

A.4 Privilege Escalation

```
ariana@pwned:~$ sudo -l
Matching Defaults entries for ariana on pwned:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User ariana may run the following commands on pwned:
    (selena) NOPASSWD: /home/messenger.sh
```

Abbildung 15: Ausgabe von `sudo -l` zeigt erweiterte Berechtigungen für den Benutzer `ariana`.
Eigene Darstellung

```
ariana@pwned:~$ cat /home/messenger.sh
#!/bin/bash

clear
echo "Welcome to linux.messenger "
                echo ""
users=$(cat /etc/passwd | grep home |  cut -d/ -f 3)
                echo ""
echo "$users"
                echo ""
read -p "Enter username to send message : " name
                echo ""
read -p "Enter message for $name :" msg
                echo ""
echo "Sending message to $name "

$msg 2> /dev/null

                echo ""
echo "Message sent to $name :) "
                echo ""
```

Abbildung 16: Analyse des Scripts `messenger.sh` mit unsicherer Eingabeverarbeitung (Command Injection).
Eigene Darstellung

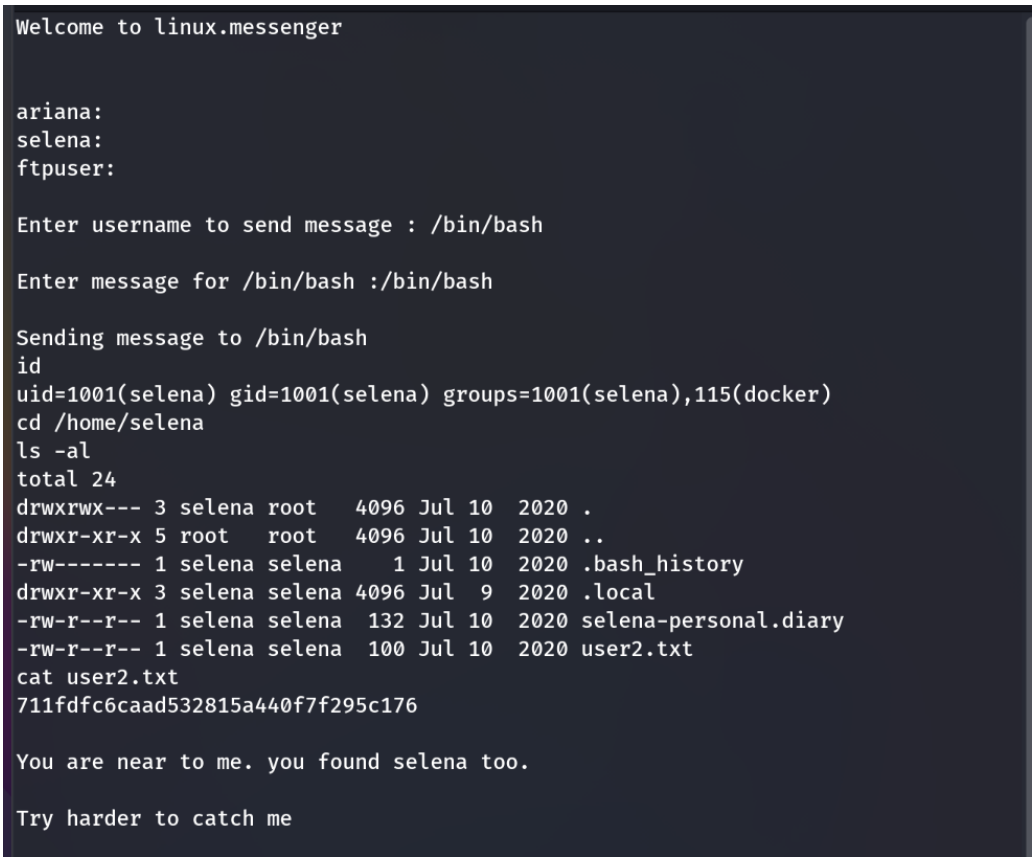

Abbildung 17: Auslesen der zweiten Flag im Kontext von `selena` als Nachweis der Privilege Escalation.
Eigene Darstellung