



Evaluierung von KI-basierten Modellen zur automatisierten Schwachstellenanalyse im Rahmen von Penetrationstests

Agenda

- ① Problem & Motivation
- ② Ziel der Arbeit
- ③ Grundlagen (KI, LLMs, Pentest-Phasen)
- ④ Getestete Tools (RamiGPT, PentestGPT CLI/Web)
- ⑤ Szenarien & Demos
- ⑥ Ergebnisse & Vergleich
- ⑦ Bewertungsmatrix
- ⑧ Stärken & Schwächen
- ⑨ Fazit & Ausblick

Problem & Motivation & Ziel der Arbeit

Problem & Motivation

Problem:

- Pentests etabliert, aber teuer & zeitaufwendig
- Hoher Personalbedarf, Fachkräftemangel
- Wachsende Angriffsflächen (Cloud, Microservices, DevOps)

Motivation:

- Automatisierung & Beschleunigung
- Reduktion menschlicher Fehler
- Schnellere Reaktionen auf Bedrohungen
- Abfederung Fachkräftemangel

Ziel der Arbeit

Forschungsfrage: Inwiefern können KI-gestützte Modelle typische Aufgaben im Pentest unterstützen, beschleunigen oder automatisieren?

Vorgehen:

- Untersuchung von drei Tools:
 - ▶ RamiGPT (Privilege Escalation)
 - ▶ PentestGPT CLI (strukturierte Webtests)
 - ▶ PentestGPT Web (dialogorientierte Tests)

Ziele:

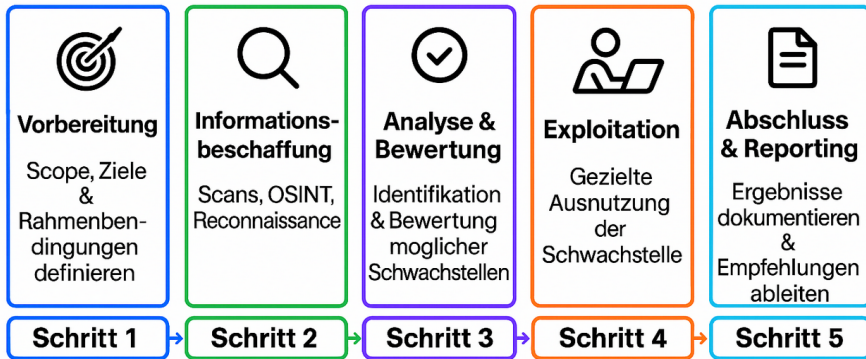
- Entwicklung eines praxisnahen Bewertungsrahmens
- Durchführung realitätsnahe Praxistests
- Vergleich: Stärken, Schwächen & Einsatzpotenziale

Problem & Motivation

Was sind Penetrationstests?

- **Definition:** Gezielte Simulation von Angriffen auf IT-Systeme
- **Zweck:** Schwachstellen frühzeitig finden & Sicherheitsniveau bewerten
- **Abgrenzung:** Unterschied zu reinen Scans → kreatives, manuelles Vorgehen
- **Arten:**
 - ▶ Black-Box (keine Vorkenntnisse)
 - ▶ Grey-Box (teilweise Infos)
 - ▶ White-Box (vollständige Infos)

Pentest-Phasen (BSI-Modell)



Grundlagen – Künstliche Intelligenz & LLMs

- **KI:** Mustererkennung, Automatisierung, Entscheidungsunterstützung

Large Language Models (LLMs):

- ▶ Zerlegen komplexer Aufgaben in Schritte
- ▶ Exploit-Vorschläge & Payload-Generierung
- ▶ Automatisierte Dokumentation

Maschinelles Lernen (ML):

- ▶ Überwachtes Lernen (Klassifikation)
- ▶ Unüberwachtes Lernen (Anomalien)
- ▶ Bestärkendes Lernen (adaptives Verhalten)

- **Schwächen:** Halluzinationen, begrenztes Kontextfenster, fehlende Security-Spezialisierung

Methodik

Welche Tools wurden getestet?

- **RamiGPT**
 - ▶ Speziell für Privilege Escalation (Linux/Windows)
 - ▶ Kombination von KI-Logik & Tools wie LinPEAS, BeRoot
- **PentestGPT (CLI)**
 - ▶ Open-Source, textbasiert, strukturierte Workflows
 - ▶ Unterstützt systematische Web-Pentest-Phasen
- **PentestGPT (Web)**
 - ▶ Kommerziell, dialogorientiert, direkte Nutzung im Browser
 - ▶ Eignet sich für schnelle Ad-hoc-Analysen

Rami-Privilege Escalation

Grundlagen – Privilege Escalation

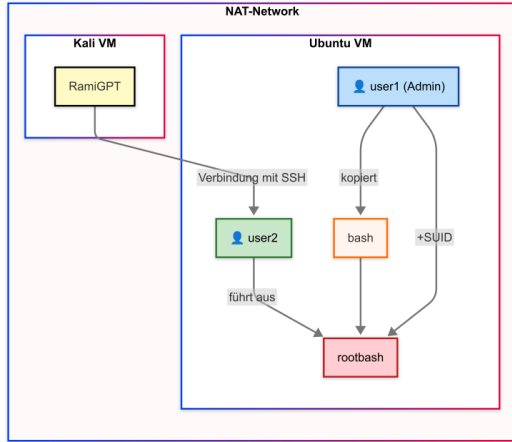
- **Ziel:** unberechtigter Zugriff auf höhere Rechte (Admin/Root)
- Häufig in Post-Exploitation-Phase
- **Typische Techniken (nach MITRE ATT&CK):**
 - ▶ Exploitation von Systemschwachstellen (T1068)
 - ▶ Missbrauch von Sudo/SetUID (T1548)
 - ▶ Manipulation von Zugriffstokens (T1134)
 - ▶ Nutzung gültiger, privilegierter Accounts (T1078)

RamiGPT – Szenario

- **Ziel:** Privilege Escalation unter Linux (Root-Rechte erlangen)
- **Setup:** Ubuntu-VM mit absichtlich fehlerhafter SetUID-Konfiguration („rootbash“)
- **Testmodus:**
 - ▶ Full-AI (komplett automatisch)
 - ▶ Halb-automatisch (mit manueller Unterstützung)

PentestGPT (CLI) – Testszenario

- Zielsystem: **OWASP Juice Shop** (verwundbare Web-App)
- Getestete Schwachstellen (OWASP Top 10):
 - ▶ A01 Broken Access Control
 - ▶ A02 Cryptographic Failures
 - ▶ A03 Injection
- Interaktion über CLI: strukturierte Workflows (next, more, discuss)



RamiGPT – Full-AI-Modus (Fehlschlag)

- Automatisierte Befehle gestartet
- Mehrere sudo-Varianten ausprobiert
 - ▶ z. B. `sudo -l`, `sudo su -l root`
- Alle Befehle blockieren bei Passwortabfrage
- Kein Fortschritt, rootbash wird nicht erkannt

Ergebnis: Full-AI-Modus scheitert in der Praxis

The screenshot displays the RamiGPT SSH Terminal interface. The main window shows the following text:

```
SSH Terminal powered by RamiGPT
* Connection established
Giving AI full freedom to send 20 comands
AI=====
s sudo -l
[Debug] Empty response from shell, possibly stuck at prompt
[Debug] Autonomous()-timeout occurred, possibly stuck at prompt
[sudo] password for user2:
Sorry, try again.
Start interacting with the shell again
Stopping..

user2@pehost:~$
```

On the right side, there is a sidebar with several buttons: "Facts", "Hints", "Avoid", "Full AI", "Guide Me", and "Import". At the bottom of the sidebar, there are "Logout" and "Go Ham" buttons.

Halb-automatischer Ablauf (1/2)

- Nutzer gibt manuell `ls` ein → Kontext für KI
- RamiGPT schlägt `find / -perm /4000 2>/dev/null` vor
- Ausgabe: mehrere SetUID-Dateien inkl. `/home/user2/rootbash`
- Aber: Tool führt rootbash nicht selbst aus

Ausgabe nach `find / -perm /4000`

\$ `ls`

`rootbash snap`

Giving AI full freedom to send 20 commands

AI

request#1=====

\$ `find / -perm /4000 2>/dev/null`

[Debug] Empty response from shell, possibly stuck at prompt

[Debug] Autonomous() - timeout occurred, possibly stuck at prompt

`/usr/bin/sudo`

`/usr/bin/passwd`

`/snap/core22/2045/usr/bin/passwd`

`/snap/core22/2045/usr/bin/su`

`/snap/core22/2045/usr/bin/sudo`

`/home/user2/rootbash`

\$ `/bin/sh: 5: !/bin/sh: not found`

\$ `uid=1001(user2) gid=1001(user2) groups=1001(user2),100(users)`

\$

Stopping...

Halb-automatischer Ablauf (2/2)

- Nach manuellem `ls -la` erkennt RamiGPT die Datei als auffällig
- Führt anschließend `./rootbash` automatisch aus
- Ergebnis: Root-Rechte erlangt (`uid=0/root`)
- Fazit: Erfolg nur mit Benutzerhilfe – User2 → Root

PentestGPT (CLI)

Grundlagen – Typische Schwachstellen (OWASP Top 10)

- **A01 – Broken Access Control**
 - ▶ Fehlende oder fehlerhafte Zugriffsbeschränkungen
 - ▶ z. B. Manipulation von JWT, IDOR
- **A02 – Cryptographic Failures**
 - ▶ Unsichere oder falsch eingesetzte Verschlüsselung
 - ▶ z. B. schwache Hashes, Klartextübertragung
- **A03 – Injection**
 - ▶ Unsichere Eingabevalidierung → Angriffe möglich
 - ▶ z. B. SQL Injection, Cross-Site Scripting (XSS)

OWASP Top 10 – 2021

A01:2021	Broken Access Control
A02:2021	Cryptographic Failures
A03:2021	Injection
A04:2021	Insecure Design (New)
A05:2021	Security Misconfiguration
A06:2021	Vulnerable and Outdated Components
A07:2021	Identification and Authentication Failures
A08:2021	Software and Data Integrity Failures
A09:2021	Security Logging and Monitoring Failures
A10:2021	Server-Side Request Forgery

PentestGPT (CLI) – Szenarien

- **Zielsystem:** OWASP Juice Shop (verwundbare Web-App)
 - ▶ A01 Broken Access Control → JWT-Manipulation, IDOR
 - ▶ A02 Cryptographic Failures → unsichere Hashes, „Weird Crypto“
 - ▶ A03 Injection → SQLi, XSS
- **Interaktion:**
 - ▶ Strukturierte Workflows über CLI-Befehle (next, more, discuss)
 - ▶ GPT-gestützte Vorschläge, manuelle Ausführung durch Nutzer

PentestGPT (CLI) – Exploit-Vorschlag

- Tool schlägt konkrete Exploit-Optionen vor (z. B. SQL Injection)
- Liefert Payloads wie ' OR '1'='1

```
**Possible Tasks:**
4. Exploit Identification -
  4.1 Test for Reflected XSS in Customer Feedback Feature -
  4.2 Test for Stored XSS in Customer Feedback Feature -
  4.3 Test for SQL Injection in Customer Feedback Feature -
  4.4 Evaluate Access Controls in Customer Feedback Feature -
5. Troubleshooting and Resolution -
  5.1 Diagnose and Resolve Tool Malfunction -
    5.1.1 Review Tool Configurations -
    5.1.2 Examine System and Tool Logs -
    5.1.3 Verify Connectivity and Dependencies -
    5.1.4 Seek Additional Resources or Support -

**Favored Sub-task**: 4.3 Test for SQL Injection in Customer Feedback Feature -

**Explanation**:
Manually input typical SQL injection payloads such as '' OR '1'='1' or '' OR 1=1 --''
into the input fields in the customer feedback form. Observe responses and any changes in
application behavior, such as retrieving more data than expected, to identify if SQL
injection vulnerabilities exist.
```

PentestGPT (CLI) – Ergebnisse

- **Stärken:**

- ▶ Liefern valider Payloads (z. B. SQLi-Bypass → Admin-Login)
- ▶ Systematische Struktur → didaktisch wertvoll (Ausbildung, Training)
- ▶ Gute Unterstützung bei IDOR & XSS durch Payload-Beispiele

- **Schwächen:**

- ▶ Keine echte Automatisierung → alles manuell auszuführen
- ▶ Teilweise generische Antworten, Detailtiefe nur mit Nachfragen
- ▶ Kein Reporting-Export, nur Logfiles

- **Fazit:** Hilfreiches Assistenztool, besonders für strukturierte Tests und Ausbildung

PentestGPT (WEB)

PentestGPT (Web) – Szenarien

- **Zielsystem:** OWASP Juice Shop (wie CLI-Version)
- **Testschwerpunkte:**
 - ▶ Broken Access Control (z. B. JWT-Manipulation, IDOR)
 - ▶ Cryptographic Failures („Weird Crypto“)
 - ▶ Injection (SQLi, XSS)
- **Interaktion:**
 - ▶ Dialogorientiert, ähnlich wie ChatGPT
 - ▶ Prompts in natürlicher Sprache (DE & EN)
 - ▶ Schnelle Ad-hoc-Analysen im Browser

PentestGPT (Web) – Ergebnisse

- **Stärken:**
 - ▶ Schnelle & präzise Antworten → sofort Exploit-Beispiele
 - ▶ Einfache Bedienung, keine Installation notwendig
 - ▶ Kontextsensitiv (Deutsch/Englisch kein Unterschied)
- **Schwächen:**
 - ▶ Keine Automatisierung → alles manuell auszuführen
 - ▶ Kein Reporting-Export, Ergebnisse nur im Chat
 - ▶ Volle Funktionen nur in kostenpflichtiger Version
- **Fazit:** Praktisch für schnelle Analysen & Proof-of-Concepts, weniger für strukturierte Tests

Vergleich der Tools

- **Bewertungskriterien:** Schwachstellenabdeckung, Exploit-Vorschläge, Automatisierung, Kontextverständnis, Reporting, Kosten-Nutzen
- **Gesamtpunkte (max. 12):**
 - ▶ RamiGPT: 5/12
 - ▶ PentestGPT (CLI): 8/12
 - ▶ PentestGPT (Web): 7/12
- **Stärken & Schwächen:**
 - ▶ RamiGPT: interessant für Privilege Escalation, aber unreif, wenig Automatisierung
 - ▶ CLI: methodisch klar, gute Payloads, aber langsamer & manuell
 - ▶ Web: schnell & flexibel, aber limitiert ohne Automatisierung/Reporting

Bewertungsmatrix

- 6 Kriterien:
 - ▶ Schwachstellenabdeckung
 - ▶ Exploit-Vorschläge
 - ▶ Automatisierungsgrad
 - ▶ Kontextverständnis
 - ▶ Reporting
 - ▶ Kosten-Nutzen

Stärken & Schwächen der KI-Tools

Stärken

- Unterstützung bei Routineaufgaben (z. B. Payload-Generierung)
- Nützliche Exploit-Vorschläge und Erklärungen
- Methodische Unterstützung (CLI) bzw. schnelle Ad-hoc-Analysen (Web)
- Niedrige Einstiegshürden für Einsteiger & Ausbildung

Schwächen

- Geringe Automatisierung → keine End-to-End-Pentests
- Schwaches Kontextverständnis (z. B. RamiGPT bei Passwortabfragen)
- Ergebnisse oft nicht reproduzierbar
- Fehlende Reporting-/Exportfunktionen

Fazit

- **KI = Unterstützung, kein Ersatz** ♦ menschliche Expertise bleibt unverzichtbar
- **Nutzen:**
 - ▶ Effizienzsteigerung bei Routineaufgaben
 - ▶ Hilfreich für Ausbildung & strukturierte Analysen
 - ▶ Schnelle Proof-of-Concepts möglich
- **Grenzen:**
 - ▶ Keine vollständige Automatisierung
 - ▶ Ergebnisse nicht immer reproduzierbar
 - ▶ Eingeschränktes Kontextverständnis

Ausblick

- **Integration in DevSecOps** ⚡ KI-gestützte Tools als Teil kontinuierlicher Sicherheitsprozesse
- **Technische Weiterentwicklung** ⚡ Größere Kontextfenster & verbesserte Modelle ⚡ Retrieval-Augmented Generation (RAG) für aktuelles Wissen
- **Anwendung in der Ausbildung** ⚡ KI als interaktiver Trainingspartner ⚡ Unterstützung beim Erlernen von Angriffstechniken & Abwehrmaßnahmen
- **Langfristige Perspektive** ⚡ KI erweitert klassische Pentests ⚡ Richtung: skalierbare & adaptive Sicherheitsprüfungen

Literatur I



Bitkom (2023):

Mangel an IT-Fachkräften droht sich zu verschärfen.

<https://www.bitkom.org/Presse/Presseinformation/>

Mangel-an-IT-Fachkraefte-droht-sich-zu-verschaerfen

Vielen Dank

XXX



**HOCHSCHULE
MITTWEIDA**
University of Applied Sciences

Hochschule Mittweida
University of Applied Sciences
Technikumplatz 17 | 09648 Mittweida
Angewandte Computer- und
Biowissenschaften

hs-mittweida.de