AI - DRIVEN PENTESTER

- BY DEEPTHINK

ASSIGNMENT COVER SHEET

UNIVERSITY OF WOLLONGONG IN DUBAI

Assignment Cover Sheet

Subject Code:	CSIT321
Subject Name:	Project
Submission Type:	Report
Assignment Title:	Capstone Project
Student Name:	Abbas Bhaiji, Ansaf Sabu, Derick Reni, Mohamed Rizvan, Mohammed Uddin
Student Number:	6345438, 7390440, 6728492, 7797965, 7842430
Student Email:	ab836@uowmail.edu.au, as3623@uowmail.edu.au, djr980@uowmail.edu.au, memrr999@uowmail.edu.au, msumk596@uowmail.edu.au
Lecturer Name:	Dr. Patrick Mukala
Due Date:	12/05/2025
Date Submitted:	12/05/2025

PLAGIARISM:
The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism.
The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation.

PLEASE NOTE: STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED

DECLARATION:
I/We certify that this is entirely my/our own work, except where I/we have given fully documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism.

Signature of Student:

Optional Marks:

Comments:

Lecturer Assignment Receipt (To be filled in by student and retained by Lecturer upon return of assignment)

Subject: **Assignment Title:**
Student Name: **Student Number:**
Due Date: **Date Submitted:**
Signature of Student:

Student Assignment Receipt (To be filled in and retained by Student upon submission of assignment)

Subject: **Assignment Title:**
Student Name: **Student Number:**
Due Date: **Date Submitted:**
Signature of Lecturer

Tables Of Contents

Background Information

Cyberattacks are increasingly frequent, complex, and damaging. Traditional defenses like periodic penetration testing (pentesting) are largely manual, labor-intensive, and unable to scale with fast-evolving infrastructures. They require skilled professionals and often result in static, disconnected reports that are difficult to act on quickly.

Recent advances in machine learning (ML), reinforcement learning (RL), and large language models (LLMs) enable new opportunities to automate and enhance pentesting workflows. Automation improves consistency, saves time, and enables adaptive testing that can evolve over time. Mapping technical vulnerabilities to business risks through interactive dashboards empowers both technical and non-technical stakeholders to make informed decisions.

Project Description and Objectives

What is our project?

Armour Eye is an AI-powered platform that fully automates the penetration testing lifecycle on Linux-based internal systems. It simulates a red team's workflow using a modular system that integrates traditional tools with intelligent AI agents. The system mimics the behaviors of ethical hackers, dynamically adjusting its strategies based on system configurations and past outcomes.

It goes beyond basic vulnerability scanning by simulating real exploit chains, generating custom attack payloads, and suggesting or applying mitigation strategies. Results are visualized through real-time, business-oriented dashboards.

Key Functional Modules

- Reconnaissance and scanning
- Exploit identification and chaining
- Generate automated payload delivery
- Risk reporting and business dashboards

Project Objectives

- Build an end-to-end autonomous pentesting agent focused on Linux systems.
- Minimize the need for human oversight in standard pentest operations.
- Generate automated payload delivery and dynamic exploits using AI.
- Provide remediation suggestions and business-focused dashboards.

Main problems

- Pentesting is typically manual and time-consuming, requiring expert input.
- Current tools are siloed and require integration by professionals.
- Traditional tools only identify vulnerabilities; they do not simulate full attacks or offer actionable patching advice.
- Most systems follow static rules, lacking adaptability or learning capabilities.
- Stakeholders often receive raw technical data that’s hard to interpret in a business context.

How do we address them?

- Full automation of the pentesting lifecycle using orchestrated security tools and AI agents.
- RL models to adapt and prioritize exploit paths based on service risk and historical outcomes.
- LLMs to generate tailored payloads, summaries, and remediation instructions.

- Use of patch databases and scraping systems to recommend or apply relevant fixes.
- Dashboards (React) convert findings into business risks, SLAs, and priorities.
- Modular design allows the platform to learn from previous test runs and improve over time.

Project scope

The Armour Eye project focuses specifically on automated penetration testing for Linux-based internal systems in this phase. The platform simulates real-world ethical hacking processes using an AI orchestration layer over traditional tools.

Key elements:

- Autonomous execution of all pentest stages: recon, scanning, exploitation, reporting.
- Use of LLMs for generating automated payload delivery and post-exploitation actions.
- Use of RL agents to select optimal exploit strategies based on system conditions.
- Integration with patch databases for remediation guidance.
- Business dashboards that translate technical findings into actionable risks.

This initial phase of the project focuses on proof-of-concept automation for Linux systems. Expansion to other OS environments and real-time integrations is reserved for future iterations.

Competitor Solution and their Limitation

Traditional Tools (e.g., Nessus, Metasploit, Burp Suite):

- Require human control at each step.
- Results are technical and disconnected from business impacts.
- Cannot self-adapt or learn from past engagements.

AI-Assisted Platforms (e.g., Horizon3.ai, Pentera, RidgeBot, AttackIQ):

- Automate some red-team tasks but often rely on predefined attack libraries.
- Limited use of LLMs or RL for contextual payload generation.
- Often closed-source, less customizable, and limited in integration flexibility.
- Focused more on compliance, BAS, or control validation — not full exploitation.

How Armour Eye is Different:

- Uses reinforcement learning to adapt and choose exploits dynamically.
- Leverages fine-tuned LLMs to generate automated payload delivery and mitigation steps.
- Provides structured business-oriented reporting and dashboards.
- Modular, open, and continuously improving from test feedback.

Target Niche

Armour Eye is designed for:

- **Startups and small companies** with limited security teams or budgets.
- **Educational institutions** and students learning ethical hacking techniques.
- **Security analysts** seeking automated, customizable red-team simulations.

It empowers non-experts by automating complex pentest tasks and offering actionable, understandable results.

Pre-requisites

Technical and platform knowledge required:

- Linux OS
- Networking basics
- CVEs and common vulnerability patterns
- Python programming
- Containerization (Docker), orchestration (Kubernetes)
- Machine learning foundations

Limitations

These limitations are expected at this stage.

- Not designed for live security monitoring or log ingestion (not a SIEM).
- Vulnerability-to-patch mapping may miss emerging (zero-day) exploits.
- LLM performance varies based on prompt structure and context.
- Requires GPUs for efficient ML inference and model execution.
- Complex integration may lead to initial orchestration overhead.

How we do it

Armour Eye is built as a modular AI-based framework that automates all red-team stages:

1. **Reconnaissance:** Nmap, OWASP ZAP, and ML classifiers identify high-risk services.
2. **Vulnerability Identification:** Services mapped to known CVEs using databases like NVD.
3. **Exploit Selection:** RL agents determine the best attack vector based on system context.

4. **Payload Generation:** LLMs craft targeted attack scripts based on vulnerability details.
5. **Execution:** Attacks executed using Metasploit and custom tools.
6. **Patching Guidance:** CVEs are checked against patch databases; updates recommended or simulated.
7. **Business Reporting:** LLMs generate natural-language summaries; dashboards visualize key risks and SLAs.

High Level System Architecture

- **Recon Module:** Nmap + ZAP + ML classifiers for host and service discovery.
- **Exploit Engine:** RL model maps CVEs to optimal exploits using Metasploit.
- **LLM Layer:** Generates automated payload delivery, post-exploitation steps, and summaries.
- **Visualization & Dashboard:** Shows actionable insights to stakeholders.

High Level System Operation

1. **User Input:** Define IP/scope of target Linux environment.
2. **Recon & Scanning:** Automated tools scan systems; ML prioritizes high-risk targets.
3. **Exploit Decisioning:** RL agent chooses best exploit based on previous outcomes and system traits.
4. **Execution:** Attack chain is deployed via Metasploit or LLM-generated scripts.
5. **Patch Suggestions:** CVEs matched to known fixes or workaround strategies scraped from trusted sources.
6. **Reporting:** LLM-generated summaries translated into dashboards for business users.

Machine Learning Data Generation

- **Training Environments:** We use vulnerable systems like Metasploitable, DVWA, and intentionally misconfigured containers.
- **Attack Datasets:** CVE databases (NVD), ExploitDB, public PoCs are used to simulate and learn effective attacks.
- **RL Training:** We use exploit outcomes from various system configurations to train agents in selecting optimal strategies.
- **LLM Fine-tuning:** Payload examples and mitigation patterns train the LLM to reason about vulnerabilities and generate context-aware scripts.

Software and Hardware Requirements

Software Requirements

Cybersecurity Tools:

- Nmap
- OWASP ZAP
- Metasploit
- Wireshark
- Burp Suite
- UnicornScan
- Tcpdump
- John the Ripper
- Hydra
- Aircrack-ng
- Kismet
- Fern
- Bettercap
- Arpwatch
- Sqlmap
- Social Engineer Toolkit
- Netcat

- CrackMapExec
- Nikto

AI/ML Tools (Python)

- PyTorch
- Keras
- Tensorflow
- Hugging Face Transformers
- LangChain
- Scikit
- Pandas

Orchestration & Automation

- Google Colab
- Docker
- Kubernetes

Frontend Visualization

- React
- TypeScript
- Flask

Project & Workflow Management

- GitHub
- Notion

Hardware Requirements

System Setup

- Linux system
- Minimum: 8-core CPU, 16 GB RAM, 256 GB Storage

Specialized Hardware

- NVIDIA GPU (RTX 4080 or better) for LLM inference and RL training
- Stable internet for patch database syncing and model updates

Requirement Analysis

Functional Requirements

These define what the system must do to fulfill its objectives.

Core Pentesting Workflow Automation

Automate all standard penetration testing stages:

- Reconnaissance
- Scanning and enumeration
- Vulnerability mapping
- Exploitation
- Post-exploitation
- Reporting and remediation guidance

Adaptive Testing with AI

Use **Reinforcement Learning (RL)** agents trained on historical test data to:

- Learn optimal exploit strategies from prior runs
- Identify and prioritize effective exploit strategies
- Dynamically adjust testing paths based on target behavior and system risk

Use **large language models (LLMs)** to:

- Generate automated payload delivery tailored to target configurations
- Summarize results into human-readable, business-relevant language
- Propose remediation or mitigation strategies

Realistic Red-Team Emulation

Simulate real-world attacker workflows, including:

- Dynamic decision-making based on target response
- Exploit chaining (multi-step attacks)
- Post-exploitation privilege escalation attempts

Patch Mapping and Recommendation

- Automatically map CVEs to known patches or mitigations
- Scrape trusted patch databases (e.g., NVD, vendor APIs)
- Provide suggestions or simulate patch applications

Business-Oriented Dashboarding

Translate technical findings into:

- Business risk scores
- SLA-driven priorities
- Actionable mitigation plans

Real-time dashboard using:

- **React** for UI
- **TypeScript** for data visualization
- **Flask** backend for API integration

Non-Functional Requirements

These cover system performance, scalability, usability, and technical constraints.

Scalability

- Modular architecture for plug-and-play tool integration
- Ability to add Windows, hybrid cloud, and external APIs in the future

Performance

- GPU-accelerated inference for AI models
- Optimized pipelines to minimize scan and exploit cycle time

Usability

- Simple UI for non-technical users
- CLI support for advanced users
- Clear separation of technical and business views

Security & Ethics

- Restricted to user-defined internal scopes
- Sandboxed testing environments to avoid unintended harm
- No support for real-time threat monitoring (not a SIEM)

Data Requirements

Training & Inference Data

- Simulated attack data generated from:
    - Vulnerable testbeds (e.g., Metasploitable, DVWA, custom Linux containers)
    - Public CVE/ExploitDB datasets
- RL agents trained on historical exploit success/failure outcomes
- LLM fine-tuned on exploit payloads, patch notes, and report examples

External Sources

- CVE data feeds (NVD, ExploitDB)
- Patch databases (vendor-specific and general-purpose)
- Open-source intelligence (OSINT) sources for vulnerability updates

Hardware & Software Requirements

Hardware

- Linux system
- Minimum: 8-core CPU, 16 GB RAM, 256 GB storage
- **NVIDIA RTX 4080 or higher GPU** for model inference

Software Stack

- **Cybersecurity tools:** Nmap, ZAP, Metasploit, SQLmap, Burp Suite, etc.
- **AI/ML:** PyTorch, TensorFlow, Hugging Face Transformers, Scikit-learn
- **Visualization:** React, Grafana, Flask
- **Orchestration:** Docker, Kubernetes
- **Collaboration & DevOps:** GitHub, Notion

Future Improvement Requirements

These are planned enhancements to extend system capability:

- Add Windows support and hybrid cloud environments
- Integrate real-time threat intelligence for smarter patching
- Enable AI-based patch impact prediction to avoid system breakage
- Add incident response generation via LLMs
- Include gamified red vs. blue team modes for training and education
- Implement log ingestion and alerting to link with external SOC/SIEM tools (optional module)
- Enable LLMs to generate automated incident response recommendations based on post-exploit findings.

Challenges

- **Model Accuracy:** RL agents require time to converge and can be sensitive to input variance.
- **Patch Availability:** Not all CVEs have public or current patches.
- **Ethical Boundaries:** Exploits are limited to test environments and user-defined scopes.
- **LLM Limitations:** Can hallucinate or generate unsafe code under incorrect prompts.
- **Integration Overhead:** Combining multiple tools demands robust orchestration and error handling.

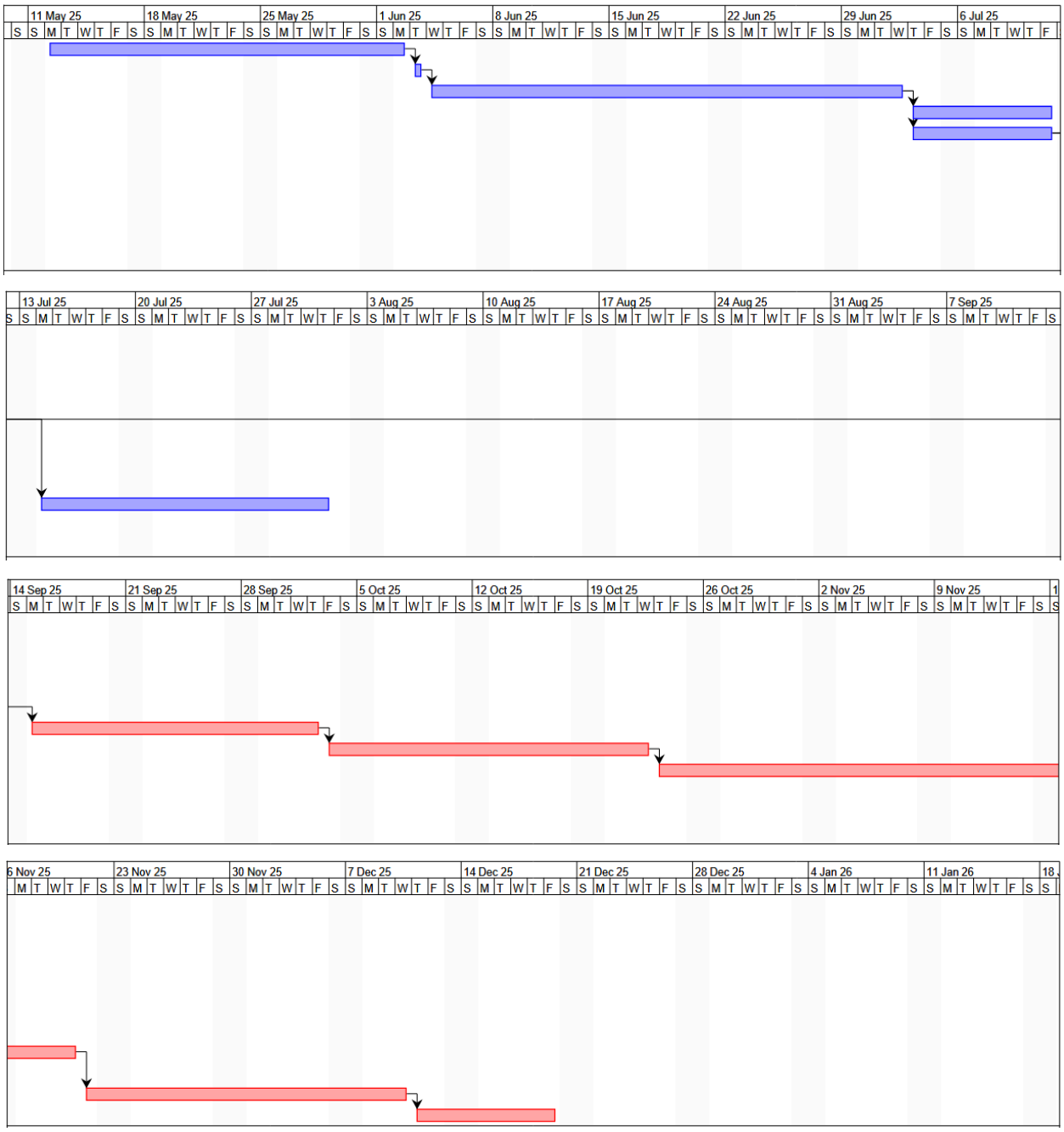
Future Improvements

- Add support for Windows and hybrid cloud environments
- Integrate real-time threat intelligence feeds
- Develop alerting and log ingestion for attack-response scenarios
- Enhance patching system with vendor-specific APIs and AI-based patch impact prediction
- Extend LLMs to generate incident response plans
- Include gamified attack-defense simulations for educational use

Timeline

Gantt Charts

		Name	Duration	Start	Finish	Predecessors
1		requirement and scope defini...	16 days?	5/12/25, 8:00AM	6/2/25, 5:00PM	
2		Report	1 day?	6/3/25, 8:00AM	6/3/25, 5:00PM	1
3		Design Planning	21 days	6/4/25, 8:00AM	7/2/25, 5:00PM	2
4		Web UI design	7 days	7/3/25, 8:00AM	7/11/25, 5:00PM	3
5		AI model design	7 days	7/3/25, 8:00AM	7/11/25, 5:00PM	3
6		Data gathering	14 days	9/15/25, 8:00AM	10/2/25, 5:00PM	5
7		DB development	14 days	10/3/25, 8:00AM	10/22/25, 5:00PM	6
8		AI model development	21 days	10/23/25, 8:00AM	11/20/25, 5:00PM	7
9		Front end	14 days	7/14/25, 8:00AM	7/31/25, 5:00PM	5
10		Integration	14 days	11/21/25, 8:00AM	12/10/25, 5:00PM	8
11		Testing	7 days	12/11/25, 8:00AM	12/19/25, 5:00PM	10



Project Management Tools to be used

- **GitHub:** Source code, version control
- **Google Meet:** Communication
- **Google Drive:** Collaboration on reports
- **Canva:** Collaboration on presentations

Team Work Distribution

Student ID	Student Name	Roles	Responsibilities
6345438	Abbas Bhaiji	Team Leader AI/ML Engineer	Model Development Model Training Data Processing Data Gathering Prompt Engineering Integrate LLM State Management Database Creation and Management
7390440	Ansaf Sabu	Scribe AI/ML Engineer	Model Development Model Training Data Gathering
7842430	Mohammed Uddin	Team Member Business Sys. Lead	Scope and Ethics Docker Orchestration Flask/React UI
7797965	Mohamed Rizvan	Team Member CyberSec Lead	Configure & maintain Nmap/OpenVAS/MSS Scope and Ethics CyberSec tools integration consultant
6728492	Derick Reni	Team Member CyberSec Lead	Flask/React UI CyberSec tools integration consultant

References

Techtarget.com. (2025). *What is Pen Testing and Why is it Important?* [online] Available at: https://www.techtarget.com/whatis/video/An-explanation-of-pen-testing [Accessed 11 May 2025].

Hammar, K. (2024). *Limmen/awesome-rl-for-cybersecurity*. [online] GitHub. Available at: https://github.com/Limmen/awesome-rl-for-cybersecurity.

Xu, H., Wang, S., Li, N., Wang, K., Zhao, Y., Chen, K., Yu, T., Liu, Y. and Wang, H. (2024). *Large Language Models for Cyber Security: A Systematic Literature Review*. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2405.04760.

Arxiv.org. (2017). *LLM-Assisted Proactive Threat Intelligence for Automated Reasoning*. [online] Available at: https://arxiv.org/html/2504.00428v1 [Accessed 11 May 2025].

Dinil Mon Divakaran and Sai Teja Peddinti (2024). *LLMs for Cyber Security: New Opportunities*. [online] arxiv.org. Available at: https://arxiv.org/pdf/2404.11338 [Accessed 12 May 2025].

GitLab. (n.d.). *Exploit-DB / Exploits + Shellcode + GHDB · GitLab*. [online] Available at: https://gitlab.com/exploit-database/exploitdb.

Ridge Security. (2025). *Automated Penetration Testing Tool | RidgeBot | Ridge Security*. [online] Available at: https://ridgesecurity.ai/ridgebot/ [Accessed 11 May 2025].

Garn, D. (2025). *Top Kali Linux tools and how to use them | TechTarget*. [online] Security. Available at: https://www.techtarget.com/searchsecurity/tip/Top-Kali-Linux-tools-and-how-to-use-them.