PENETRATION TESTING

PROJECT PITCH

Black-Box Assessment — Metasploitable 3 (Linux VM)

Student	Feddis Kahindi
Date	June 6, 2026
Target IP	192.168.100.3
Attacker IP	192.168.100.4
Assessment Type	Black-Box Penetration Test
Classification	**CONFIDENTIAL — Training Environment**

This document presents the full project pitch for a black-box penetration test against Metasploitable 3, covering the business problem, problem-solving methodology, timeline, deliverable metrics, expected findings, and risk considerations.

1. Business Problem Scenario

The target organisation is a development team using Metasploitable 3 as an internal training and testing environment. The penetration test is commissioned to verify that this intentionally vulnerable system is properly isolated from production networks and that developers understand real-world attack paths.

Although this environment is purpose-built to be vulnerable, the risks it presents are direct mirrors of misconfigurations found in poorly secured production systems. The assessment must confirm the degree to which these risks are exploitable and document a full attack chain.

1.1 Core Risks Identified

Risk	Description	Business Impact
Default SSH Credentials(vagrant:vagrant)	Any network-connected host can authenticate without brute-force	Complete unauthorised remote access
Passwordless sudo(NOPASSWD: ALL)	All privilege separation is eliminated; authenticated users become root instantly	Immediate full system takeover
Dirty Cow — CVE-2016-5195(Kernel 3.13.0-24)	Critical kernel privilege-escalation bug; exploitable even if sudo is remediated	Independent root persistence surviving credential rotation
Exposed Sensitive Files(8_of_clubs.png)	Flag files stored in user home directories with weak permissions	Confidentiality breach of internal training materials

1.2 Primary Test Objectives

- Uncover vulnerabilities through active scanning and enumeration
- Test existing defences by safely exploiting findings to achieve root access
- Assess risk exposure by documenting the full attack chain from initial access to full compromise
- Deliver actionable remediation to guide the development team in hardening real-world systems

> The organisation needs a penetration test to ensure that even intentionally vulnerable training systems do not expose the broader network to unnecessary risk. The biggest risks are default credentials, privilege misconfigurations, and unpatched kernel vulnerabilities.

2. Problem-Solving Process

The engagement follows a four-stage process aligned with industry-standard penetration testing methodology (PTES). Each phase builds directly on the output of the previous one, ensuring a structured and repeatable approach.

2.1 Methodology Stages

#	Phase	Description	Key Output
1	**Reconnaissance**	Identify live hosts; confirm target 192.168.100.3 is active	Confirmed live target, active services
2	**Enumeration**	Scan open ports, detect service versions, identify attack vectors (SSH, HTTP, SMB, MySQL)	Vulnerability map, open port list
3	**Exploitation**	Gain initial access via default credentials, escalate to root via sudo and Dirty Cow	Root shell, exploitation evidence
4	**Post-Exploitation**	Enumerate compromised system, locate and extract 8_of_clubs.png flag, document impact	Flag captured, full attack chain documented

2.2 Toolset

Tool / Command	Purpose
netdiscover / ping	Host discovery — confirm target is live on network
nmap -sV -sC -p-	Comprehensive port and service version enumeration
nikto	Web server vulnerability scanning
enum4linux	SMB share and user enumeration
ssh vagrant@192.168.100.3	Default credential testing for initial access
sudo su -	Passwordless privilege escalation to root
find	Post-exploitation file discovery (flag hunting)
scp	Secure extraction of flag file to attacker machine
uname -r	Kernel version verification for Dirty Cow applicability

2.3 Custom Automation — auto_recon.sh

A custom bash automation script handles the initial enumeration phase, parallelising scan tasks and producing structured output files ready for manual review:

```
#!/bin/bash
# auto_recon.sh — Automated initial enumeration for Metasploitable 3

TARGET="192.168.100.3"

echo "[*] Starting reconnaissance on $TARGET"
ping -c 1 $TARGET

echo "[*] Running full port scan..."
nmap -sV -sC -p- $TARGET -oN port_scan.txt

echo "[*] Running web vulnerability scan..."
nikto -h http://$TARGET -o web_scan.txt

echo "[*] Running SMB enumeration..."
enum4linux $TARGET > smb_enum.txt

echo "[*] Reconnaissance complete. Review output files."
```

2.4 Workflow Integration

The automation script integrates into the overall workflow as shown below:

Stage	Steps
Phase 1: Recon	Manual: ping / netdiscover → Confirm 192.168.100.3 is live
Phase 2: Enum	auto_recon.sh → nmap full port scan → nikto web scan → enum4linux SMB → Manual review of output files
Phase 3: Exploit	Manual: SSH with vagrant:vagrant → sudo su - (passwordless root) → uname -r (verify Dirty Cow kernel)
Phase 4: Post-Exploit	Manual: find /home -name "*.png" → scp flag to attacker → Document full attack chain
Reporting	Draft report with findings, CVSS scores, evidence, and remediation → Peer critique → Final polish

> My process follows four stages: Reconnaissance, Enumeration, Exploitation, and Post-Exploitation. I will use Nmap, Nikto, enum4linux, SSH, and sudo, automating the initial enumeration phase with my custom bash script auto_recon.sh.

3. Timeline and Scope

While the problem-solving process defines the technical flow, this section maps those phases to project milestones and time allocations across the 8-day engagement window.

3.1 Three-Phase Pacing Model

Phase	Focus	Time Allocation	Deliverables
Phase 1: Plan & Propose	Define approach, prep tooling, plan automation strategy	~25% (Days 1–2)	Project Pitch (this document)
Phase 2: Build & Critique	Execute testing, build scripts, draft report, peer feedback	~50% (Days 3–6)	Draft Report + Scripts + Critique
Phase 3: Finalise & Showcase	Revise based on feedback, polish docs, prepare presentation	~25% (Days 7–8)	Final Report, Scripts, Video Showcase

3.2 Detailed Daily Timeline

Day	Phase	Technical Stage	Specific Tasks
Day 1	Phase 1	**Planning**	Define scope, research Metasploitable 3 vulnerabilities, and draft this pitch
Day 2	Phase 1	**Tooling Prep**	Write auto_recon.sh, test in an isolated lab environment
Day 3	Phase 2	**Recon + Enum**	Run auto_recon.sh against the target, analyse nmap / nikto output, and identify SSH vectors
Day 4	Phase 2	**Exploitation (Initial)**	Test default SSH credentials vagrant:vagrant, document successful access
Day 5	Phase 2	**Exploitation (PrivEsc)**	Execute sudo su - to root, verify Dirty Cow with uname -r
Day 6	Phase 2	**Post-Exploit + Draft**	Locate flag via find, extract via scp, draft full pentest report, submit for critique
Day 7	Phase 3	**Revision**	Integrate peer feedback, update findings, verify all evidence is properly documented
Day 8	Phase 3	**Final Polish**	Final formatting, script documentation, record video showcase, submit

3.3 Time Allocation Summary

Activity	Estimated Time
Initial testing and reconnaissance	**4 hours (Day 3)**
Script development and testing	**3 hours (Day 2)**
Exploitation — SSH + sudo + Dirty Cow	**3 hours (Days 4–5)**
Report writing	**5 hours (Day 6)**
Critique feedback integration	**2 hours (Day 7)**
Final submission polish	**2 hours (Day 8)**
TOTAL	**19 hours**

3.4 Buffer Time

Buffer Allocation	Reason
2 hours — Day 4	Troubleshooting script or network connectivity issues
2 hours — Day 7	Peer feedback integration and retest after remediation edits
1 hour — Day 5	Dirty Cow race condition — may require multiple exploit attempts

> Days 1–2 focus on planning and script development (Phase 1), followed by full testing and exploitation on Days 3–5. After the critique at the end of Day 6, the scope updates to error handling, missing exploitation steps, and report clarity before final submission on Day 8.

4. Deliverable Quality and Success Metrics

4.1 Script Success Metrics

Script: auto_recon.sh (Bash Automation)	
Success Criteria	Complete when it runs without errors and produces all three output files
Metric 1	Executes fully without crashing on the target IP (192.168.100.3)
Metric 2	Generates port_scan.txt, web_scan.txt, and smb_enum.txt
Metric 3	Completes all three scans within 10 minutes

Script: Port Scanner (Python) — Planned Phase 2	
Success Criteria	Complete when it scans ports 1–1024 and returns accurate open/closed status
Metric 1	Scans 1,024 ports in under 60 seconds
Metric 2	Does not crash on invalid IP or timeout
Metric 3	Matches at least 90% of nmap results for open ports

Script: Web Scanner (Python) — Planned Phase 2	
Success Criteria	Complete when it tests for reflected XSS on common endpoints
Metric 1	Tests at least 3 endpoints: /, /search, /test
Metric 2	Tests at least 3 XSS payloads per endpoint
Metric 3	Outputs clear Vulnerable / Secure status per test

4.2 Penetration Testing Report Metrics

Metric	Definition
Metric 1 — Completeness	Report includes all required sections: Executive Summary, Scope, Methodology (4 stages), Findings Table (with CVSS scores), Exploitation Steps (with commands), Remediation Plan, and Appendices
Metric 2 — Professional Quality	Zero spelling / grammatical errors; consistent formatting; clear command blocks with expected output; screenshots properly labelled and referenced
Metric 3 — Actionable Value	Each finding includes specific, step-by-step remediation commands (not general advice); critical findings include a remediation script

Metric 4 — Evidence Quality	At least 3 pieces of evidence: SSH login output, sudo escalation output, flag capture command + screenshot

4.3 Success Criteria Summary

Deliverable	Complete When…
auto_recon.sh	Runs without errors and creates all three output files
Port Scanner (Python)	Scans 1,024 ports and accurately identifies open ports
Web Scanner (Python)	Tests 3+ XSS payloads on 3+ endpoints without crashing
Penetration Test Report	Contains all 5 findings, CVSS scores, exploitation steps, and remediation plan
Flag Capture	8_of_clubs.png is located and extracted from the target via scp

My port scanner will be considered complete when it accurately identifies open TCP ports 1–1024 without crashing on invalid inputs. My web scanner will be complete when it tests for reflected XSS vulnerabilities on multiple endpoints and outputs clear pass/fail results. The report will meet expectations if it contains all required sections, is free of errors, and provides actionable remediation commands for each finding.

5. Expected Findings (Hypothesis)

Based on Metasploitable 3's known vulnerability profile, the following findings are anticipated. All critical findings form a contiguous attack chain from initial access through to full root compromise.

Priority	Finding	Exploitation Method	CVSS	Impact
Critical	Default SSH Credentials (vagrant:vagrant)	Direct SSH authentication without brute-force	**9.8**	Remote Access
Critical	Passwordless sudo (NOPASSWD: ALL)	sudo su - executed from vagrant session	**9.0**	Full Root
Critical	Dirty Cow — CVE-2016-5195Kernel 3.13.0-24	Local kernel privilege escalation exploit (race condition)	**7.8**	Root Persist.
High	Sensitive Flag Files in User Home Directories	Post-exploitation find enumeration + scp extraction	**7.5**	Data Breach
Low	Jetty Version Disclosure	Banner grabbing via HTTP response headers	**5.3**	Info Leak

5.1 Attack Chain Narrative

The three Critical findings combine to form a complete, layered attack chain:

Step	Action	Detail
1	**Initial Access**	SSH login with default credentials vagrant:vagrant — no brute-force required
2	**Privilege Escalation**	sudo su - executed from vagrant session → instant root shell via NOPASSWD: ALL
3	**Persistence (Alt.)**	Dirty Cow exploit confirms root is achievable even without sudo misconfiguration
4	**Data Exfiltration**	find /home -name "*.png" locates 8_of_clubs.png; scp transfers it to attacker at 192.168.100.4

6. Risks and Constraints

The following constraints have been identified for this engagement. Each carries a documented mitigation strategy to ensure testing stays on schedule and within scope.

Constraint	Mitigation Strategy
Time-limited (8-day project window)	Prioritised critical attack paths first; custom automation reduces manual enumeration time significantly
Lab environment only (no production access)	No production systems at risk; all exploitation is safely contained — risks mirror real-world misconfigurations
Scripting experience level	Custom scripts supplement (not replace) proven tools such as nmap and nikto; tested before live use
Dirty Cow race condition may be unstable	1-hour buffer allocated on Day 5; multiple exploit variants prepared in advance
Peer critique feedback turnaround	2-hour buffer on Day 7 reserved specifically for integrating critique feedback into the final report

7. Conclusion

This black-box penetration test against Metasploitable 3 will demonstrate how a combination of default credentials, privilege misconfigurations, and an unpatched kernel leads to complete and repeatable system compromise — mirroring risks present in real-world production environments.

Custom automation (auto_recon.sh) provides efficient, structured enumeration. Manual exploitation (SSH → sudo → root → flag capture) proves the full attack chain and quantifies business impact. All findings will be delivered with clear, step-by-step remediation guidance to help the development team harden their real-world systems against equivalent threats.

7.1 Summary of Key Commitments

Commitment	How It Will Be Met
Full attack chain documented	SSH login → sudo root → Dirty Cow → flag capture with command evidence
Custom automation script	auto_recon.sh covering nmap, nikto, and enum4linux with structured output files
Professional report	All required sections, CVSS scores, labelled screenshots, zero grammatical errors
Actionable remediation	Step-by-step remediation commands per finding; critical findings include a remediation script
On-time delivery	8-day timeline with explicit buffer allocations for troubleshooting and feedback

> This assessment will confirm that even intentionally vulnerable training environments can expose the broader network to unacceptable risk — and will provide the development team with the evidence and remediation steps needed to prevent the same vulnerabilities from appearing in production systems.