Metasploit Framework Demonstration (Controlled Environment)

2025-10-25

Prepared for Applied Information Assurance Projects
Date: October 2025

---

1. Executive Summary

This project demonstrates high-level concepts of exploit frameworks using the Metasploit Framework in a fully isolated, sandboxed lab. The focus is on the exploit lifecycle, payload handling, session management, and—critically—defensive countermeasures and detection. All tests were conducted on self-owned virtual machines under explicit rules of engagement and are presented in sanitized form for educational purposes.

> **Audience:** security practitioners, engineers, and managers interested in defensive controls and understanding attacker techniques at a conceptual level.

---

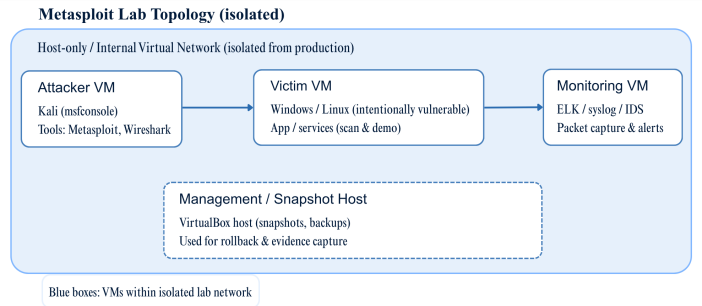
2. Scope & Rules of Engagement (ROE)

- **Scope (sanitized):** Attacker VM (`attacker-lab.local`), Victim VM (`victim-win-lab.local`), Logging/Monitoring VM (`siem-lab.local`) on an isolated host-only network.
- **Authorization:** All activity performed on lab assets owned by the researcher; no external systems were accessed.
- **Snapshots:** VM snapshots were taken prior to testing; a rollback plan was in place.
- **Safety constraints:** No destructive payloads, no live exploit chaining against production, and all artifacts were sanitized before publication.

---

3. Lab Topology

Diagram (brief): - Attacker (Kali / lab) — Metasploit Framework installed
- Victim (Windows/Linux intentionally vulnerable image)
- Monitoring (ELK / syslog / IDS) — captures network and host telemetry
- Isolated network: host-only / internal VirtualBox network



Metasploit topology

---

4. Tools & Versions

- Metasploit Framework — `msfconsole` (version: recorded during lab)
- Virtualization: VirtualBox / VMware (isolated host-only networks)
- Logging & Detection: syslog/ELK stack, Suricata/IDS (example)
- Supporting: Wireshark (capture), Windows Event Logs, and host OS tooling

> Note: Specific exploit modules or commands are not published. This documentation focuses on conceptual demonstration and defensive insight.

---

5. Demonstration Overview (Conceptual)

Each demo objective below was executed in an isolated lab and observed from a defensive perspective:

- **Exploit module lifecycle (conceptual):** discovery → select module → configure target & payload → launch → session management.
- **Payload handling:** types of payloads (stagers vs. stages), session reliability considerations, safe cleanup and session termination.
- **Post-exploitation artifacts:** what indicators appear in network captures, process listings, and host logs.
- **Privilege escalation (theory):** why it matters, containment strategies, and detection signals.
- **Cleanup & evidence preservation:** snapshot rollback, secure evidence export, and documentation for post-test review.

---

6. Observations & Defensive Insights

- **Session telemetry:** Metasploit sessions often produce distinct network and host artifacts (outbound C2 patterns, uncommon process spawn patterns). These can be detected by tuned IDS signatures and endpoint monitoring.
- **Persistence attempts:** Common persistence techniques leave identifiable traces (service creation, autoruns, scheduled tasks) — monitor these channels.
- **Payload trade-offs:** Meterpreter-like staged payloads reduce initial footprint but produce identifiable staged traffic; defenders can use flow correlation to detect sessions.
- **Privilege escalation indicators:** Unusual privilege elevation events, token manipulation, and unexpected process owners warrant immediate investigation.

---

7. Defensive Countermeasures (Practical)

Area	Recommended Controls
Initial access	Harden exposed services, enforce MFA, reduce attack surface
Detection	Centralize logs (SIEM), enable EDR with behavior analytics
Privilege control	Least privilege, restrict local admin, manage service accounts
Network controls	Egress filtering, denylist risky outbound destinations, microsegmentation
Recovery	Snapshot & rollback procedures, immutable backups, incident playbooks

---

8. Sample Sanitized Evidence (Safe)

Screenshot notes: Include sanitized screenshots that show UI (e.g., Metasploit workspace listing sessions) with hostnames/IPs redacted. Do **not** include exploit commands or payload code.

Sanitized log excerpt example

```
[2025-10-24 14:02:31] ALERT: Suspicious outbound connection detected:
  src: attacker-lab.local
  dst: 198.51.100.23:443
  proto: tcp
  signature: potential-c2-beacon
Action: Investigate process creation on victim and correlate with EDR
telemetry.
```

9. Remediation Checklist

- Ensure all internet-facing services are patched and minimized.
- Enforce multi-factor authentication on all privileged accounts.
- Deploy and tune host-based EDR rules to detect common payload behaviors.
- Implement strict egress filtering and outbound anomaly detection.
- Practice incident response with snapshots and post-incident validation.

10. Ethical & Compliance Note

This project strictly adheres to ethical testing standards. No instructions facilitating misuse are published. Any practical demonstrations were run in an isolated environment on systems controlled by the researcher, and artifacts released here are sanitized.

11. Reproducibility & References

Reproducibility steps (high-level, non-actionable): - Build three VMs on an isolated network: attacker, victim, monitoring. - Enable logging on the monitoring host (syslog/ELK or similar). - Use Metasploit to demonstrate session lifecycle in the attacker VM; observe and capture telemetry on the monitoring VM. - Roll back snapshots after each demonstration.

References

- Rapid7. *Metasploit Framework Documentation.*
- NIST SP 800-115. *Technical Guide to Information Security Testing and Assessment.*
- MITRE ATT&CK. *Tactics, Techniques, and Procedures.*
- OWASP. *Testing Guide* (for secure app testing principles).

Download Report (PDF)