Scenario # 8 Incident Response Playbook

Table of Contents

1. Introduction *2*

1.1 Incident type **2**

1.2 Assumptions **2**

1.3 Audience **3**

2. Preparation *3*

2.1 Tools and Resources **3**

2.2 Communication Channels **4**

3. Identification *5*

3.1 Incident Detection Methods **5**

3.2 MITRE ATT&CK Techniques **5**

3.3 Incident Reporting Mechanism **5**

3.4 Initial Assessment Procedure **5**

4.Containment *6*

4.1 Short-Term Containment Strategies **6**

4.1.1 Activity Diagram- Short-Term Containment **7**

4.2 Long-Term Containment Measures **7**

5. Eradication *8*

5.1 Root Cause Analysis **8**

5.2 Eradication Procedures **8**

6. Recovery *9*

6.1 System Restoration Process **9**

6.2 Validation Checks **9**

6.3 Ongoing Monitoring **9**

7. Post-Incident Activity *10*

7.1 Lessons Learned Session **10**

7.2 Incident Report Writing **10**

References *11*

Scenario 8 – Rootkit Installed in Development Server

Organization: Viva la Vita Online
Playbook Version: 2.0
Last Updated: December 3, 2025

1. Introduction

Objective: The objective of this section is to define the scope, assumptions, and stakeholders involved in responding to malware and rootkit compromises affecting Linux development environments. This ensures that all team members understand when to use this playbook, what incident types it covers, and what operational expectations apply.

1.1 Incident type

This playbook is designed for Linux-based security incidents involving unauthorized installation of rootkits, worms, or backdoor components that compromise development infrastructure. It includes incidents where attackers exploit vulnerabilities in Linux services such as the OpenSSL vulnerability associated with the Slapper Worm to gain access, escalate privileges, or integrate malicious components into production-adjacent development systems.

This playbook covers incidents involving:

- Installation of rootkits or kernel-level persistence mechanisms
- Exploitation of Linux vulnerabilities leading to unauthorized access
- Source code manipulation or unauthorized access to code repositories
- Unauthorized lateral movement through development networks
- Compromise of CI/CD pipelines or build systems
- Unauthorized access to proprietary or regulated development data
- Data exfiltration through worm- or malware-initiated outbound channels

1.2 Assumptions

For the purposes of this playbook, the following assumptions are made:

- The last security scan date of the development server is unknown.
- The server contains critical development data, including internal code repositories.
- Logging (syslog, auditd, or SIEM ingestion) is enabled but completeness is uncertain.
- Backups exist but have not been recently validated for integrity.
- The Slapper Worm variant or similar malware may have established persistent components.
- SOC/IRT has authority to isolate, shut down, or forensically capture affected systems.
- The development system cannot be fully offline for extended periods due to business impact.

1.3 Audience

This Playbook is intended for the following audiences:

- Incident Response Team (IRT)
- SOC Analysts / Malware Response Analysts
- Linux System Administrators / DevOps
- Product Security / Software Engineering Leadership
- Legal, Compliance and Privacy stakeholders
- Senior Leadership (if required for breach notification or high-impact decisions)

According to NIST SP 800-61r2, organizations should clearly define incident types, roles, responsibilities, and assumptions to ensure coordinated and effective incident response (NIST SP 800-61r2, §2.3).

2. Preparation

Objective: The objective of this section is to ensure that necessary tools, monitoring systems, forensic capabilities, and communication structures are in place before a Linux compromise occurs. Adequate preparation ensures that the organization can respond quickly and effectively to rootkit-based intrusions.

2.1 Tools and Resources

The following tools, technologies, and resources must be available and routinely maintained to ensure rapid and effective response:

Linux Security & Forensic Tools

- Linux security tools (chkrootkit, rkhunter, lynis)
- Volatility Framework for memory analysis
- Disk imaging utilities (dd, dc3dd, FTK Imager)
- Sysinternals-equivalent Linux tooling (auditd, psacct, Sysmon for Linux)
- Integrity monitoring tools (AIDE, Tripwire)

Enterprise Security Tools

- SIEM Platform (e.g., Splunk, QRadar, ELK)
- Endpoint Detection and Response (EDR) with Linux support
- Network IDS/IPS (Suricata, Snort, Zeek)
- Vulnerability scanners (Tenable Nessus, OpenVAS)
- Fortinet / Palo Alto firewall logging and threat feeds

Forensic & Response Resources

- Chain of custody forms
- Secure evidence storage repositories (encrypted)
- Log archival and timestamp verification tools
- Dedicated forensic workstation or secure container environment

Periodic Requirements

- Quarterly server vulnerability scanning
- Regular Linux patching cycles and OpenSSL validation
- Annual review of development server hardening baselines
- Periodic integrity verification of critical binaries and code repositories

2.2 Communication Channels

Primary Channels

- Encrypted IR Slack/Teams channels
- Internal ticketing
- 24/7 SOC hotline

Secondary Channels

- Secure encrypted corporate email
- Emergency IR bridge calls

Tertiary Channels (as needed)

- Legal/compliance hotline
- Executive communication channels

Communication Requirements

- Strict need-to-know communication
- No transmission of source code or sensitive artifacts in plaintext
- All communication must be logged for audit and review
- Formal escalation paths must be followed

NIST highlights that preparation including tools, communication plans, and evidence handling procedures is essential to ensure timely and efficient incident response (NIST SP 800-61r2, §3.1).Proper chain-of-custody processes should be established to maintain the integrity of evidence during response activities (NIST SP 800-61r2, §3.2.4).

3. Identification

Objective: This section defines how Linux-based rootkit infections are detected, validated, escalated, and initially assessed. Early detection is critical to limiting unauthorized access and preventing further compromise.

3.1 Incident Detection Methods

Incident discovery may occur through any of the following mechanisms:

- SIEM alerts detecting suspicious processes or kernel modules
- Unexpected open ports associated with Slapper Worm variants
- Abnormal outbound connections or data exfiltration attempts
- Developer reports of unusual system behavior
- Integrity monitoring flags on altered binaries (e.g., /bin, /usr/bin, /lib/modules)
- IDS/IPS alerts related to OpenSSL vulnerabilities exploited by Slapper Worm
- Log anomalies indicating privilege escalation or unauthorized SSH activity
- EDR alerts for malware behavior or persistent components

3.2 MITRE ATT&CK Techniques

- T1068
- T1059.004
- T1543.002
- T1055
- T1176

3.3 Incident Reporting Mechanism

- All suspected or confirmed Linux server compromises must be reported through the internal IR process.
- SOC analysts must triage rootkit-related alerts within 15 minutes.
- High-risk incidents must be escalated to:
    - Legal and Compliance
    - Privacy/Data Protection Office
    - Senior Leadership
- All reports must be logged within the SIEM and ticketing platform.

3.4 Initial Assessment Procedure

1. Validate rootkit or Slapper Worm indicators using chkrootkit, rkhunter, SIEM alerts, or anomalous OpenSSL activity.
2. Identify affected systems, starting with the compromised development server.

3. Review system and authentication logs to confirm unauthorized access, privilege escalation, or abnormal network activity.
4. Determine the timeframe of compromise and whether worm propagation occurred.
5. Assess potential impact on development data, code repositories, credentials, or CI/CD pipelines.
6. Identify the account, process, or external entity responsible for initiating the compromise.
7. Notify legal, privacy, and leadership if sensitive data or code integrity is at risk.

NIST requires organizations to analyze alerts, logs, and indicators of compromise to verify incidents and establish initial scope (NIST SP 800-61r2, §3.2).NIST also recommends maintaining detailed documentation during initial detection and triage to support later stages of the response (NIST SP 800-61r2, §3.2.7).

4.Containment

Objective: The objective of containment is to stop the rootkit's activity, prevent lateral movement, secure the development environment, and preserve forensic data while minimizing business interruption.

4.1 Short-Term Containment Strategies

Action Steps:

1. Suspicious Activity Detected

- SOC identifies anomalous system behavior, alerts, or indicators consistent with Slapper Worm or rootkit activity.

2. Validate Indicators

- SOC correlates alerts, reviews logs, and performs initial checks (e.g., chkrootkit, rkhunter)to confirm a credible incident.
- If indicators are validated, proceed to formal escalation.

3. Alert the Incident Response Team

- SOC notifies the IRT and activates the Linux malware IR protocol.
- Assign IR Lead, Linux SME, and Forensics Lead.

4. Isolate the Affected Server

- Remove the server from the network (switch port shutdown or VLAN isolation).
- Block outbound traffic associated with Slapper Worm behavior.

5. Disable Compromised Accounts and Rotate Credentials

- Disable or lock any suspicious or compromised user/service accounts.
- Rotate SSH keys, API tokens, and any potentially affected credentials.
- Avoid rebooting the server to preserve volatile evidence.

6. Preserve Evidence

- Capture a memory (RAM) image.
- Snapshot the file system and repository state.
- Export system, authentication, and SIEM logs to secure storage.
- Evidence is preserved before stakeholder notification to ensure accuracy and chain-of custody integrity.

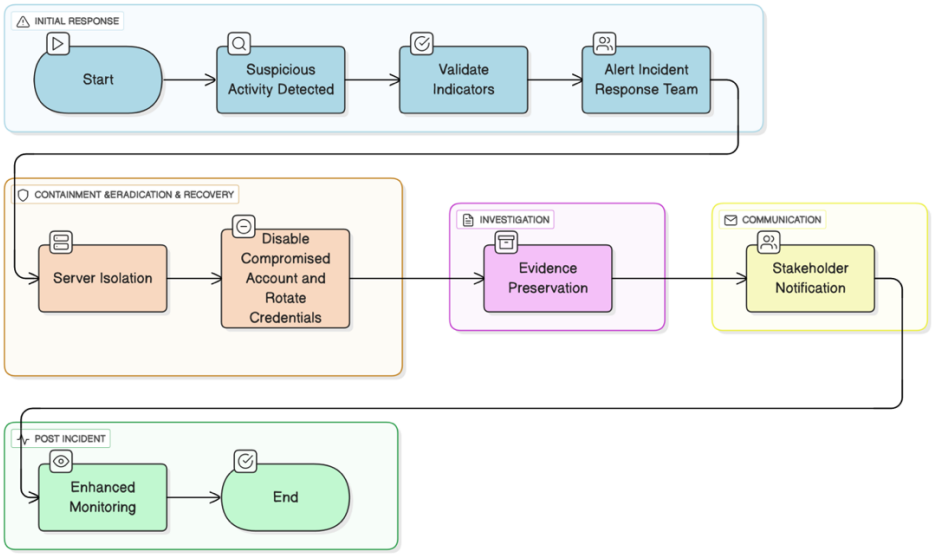
7. Notify Required Stakeholders

- SOC Manager, CISO, Legal, and Product Engineering.
- Notifications occur after evidence preservation to ensure reliable reporting.

8. Increase Monitoring

- Enable real-time SIEM alerts and enhanced log visibility.
- Monitor for repeated exploit attempts, worm propagation behavior, or lateral movement.

4.1.1 Activity Diagram- Short-Term Containment



4.2 Long-Term Containment Measures

- Patch all exploited vulnerabilities, including OpenSSL, Apache modules, and any outdated Linux packages.
- Rebuild or reimage the development server if rootkit removal cannot be fully validated.

- Implement hardened SSH configurations (disable password authentication, require MFA, restrict root login).
- Enforce network segmentation between development, staging, and production systems.
- Deploy file integrity monitoring (AIDE/Tripwire) on critical directories and binaries.
- Rotate all system, service, and repository credentials and reissue any potentially compromised SSH keys.
- Review and restrict access permissions for code repositories, CI/CD pipelines, and development tools.
- Enable continuous vulnerability scanning and routine Linux security audits.
- Establish automated alerts for unauthorized process creation, abnormal network traffic, or privilege escalation.
- Update Linux hardening baselines and ensure consistent application across all development servers.
- Conduct a full environment review to ensure no residual worm activity, backdoors, or lateral movement artifacts remain.

NIST states that containment strategies must limit the damage, prevent further unauthorized access, and preserve evidence for subsequent forensic investigation (NIST SP 800-61r2, §3.3.1). Containment strategies should be selected based on potential damage, evidence preservation requirements, and operational impact (NIST SP 800-61r2, §3.3.1).

5. Eradication

Objective: The objective of eradication is to remove malicious components, eliminate vulnerabilities exploited by the Slapper Worm, and restore the Linux environment to a secure baseline.

5.1 Root Cause Analysis

Identify the chain of vulnerabilities and process failures that enabled compromise.

Root Cause Factors:

- Unpatched OpenSSL vulnerability exploited by Slapper Worm.
- Missing or outdated security scans.
- Weak SSH hardening or credential exposure.
- Lack of integrity monitoring on development servers.
- Insufficient segmentation between dev resources and code repositories.
- Outdated Linux kernel or service packages.

5.2 Eradication Procedures

- Remove all malicious files, modules, and persistence mechanisms.
- Conduct full malware sweeps (chkrootkit, rkhunter, custom signatures).
- Patch OpenSSL, Linux kernel, Apache, and related services.

- Reinstall or replace altered binaries from verified packages.
- Rotate and revoke all affected SSH keys and credentials.
- Validate source code repositories for unauthorized modification.
- Conduct full vulnerability scan after cleaning.

NIST specifies that eradication activities should remove malicious artifacts, fix exploited vulnerabilities, and address underlying weaknesses that enabled the incident (NIST SP 800-61r2, §3.3.2).

6. Recovery

Objective: The objective of recovery is to restore secure development operations, validate system integrity, and ensure that no unauthorized access persists.

6.1 System Restoration Process

- Restore system from a clean, pre-compromise backup (after validation).
- If integrity cannot be assured, rebuild the server from a trusted image.
- Reinstall development tools from verified sources.
- Restore code repositories from validated, known-good versions.
- Re-enable network connectivity only after full system verification.

6.2 Validation Checks

- Review 48–72 hours of logs for anomalies.
- Conduct post-recovery vulnerability scanning.
- Validate no unauthorized SSH access or privilege anomalies.
- Confirm that repository integrity is intact.
- Verify hardening controls (SSH, firewall, SELinux/AppArmor).
- Complete regulatory or compliance notifications if required.

6.3 Ongoing Monitoring

- Daily log monitoring for first 2 weeks.
- Weekly integrity checks for the first month.
- Monthly malware scanning for 90 days.
- Quarterly development server hardening reviews.

NIST requires organizations to restore impacted systems to normal operations while thoroughly validating that vulnerabilities are eliminated and monitoring for signs of recurring issues (NIST SP 800-61r2, §3.3.3)

7. Post-Incident Activity

Objective: The objective of post-incident activities is to ensure lessons learned are documented, systemic weaknesses are addressed, and a full incident report is produced.

7.1 Lessons Learned Session

A formal Lessons Learned session must occur within five (5) business days after recovery. The Incident Response Team is responsible for coordinating the session, ensuring all relevant stakeholders contribute and corrective action are tracked. Key discussion and documentation points include:

- Validate timeline accuracy.
- Identify technical failures (patching gaps, vulnerability exposure).
- Identify operational failures (missing monitoring, incomplete scans).
- Document forensic findings and root cause.
- Define updates to hardening standards, patch cycles, and monitoring.
- Identify improvements to code repository controls and credential hygiene.

7.2 Incident Report Writing

A comprehensive incident report must be prepared by the Incident Response Team and approved by the CISO. The report serves as the authoritative record and may be used for compliance audits, regulatory inquiries, and vendor management follow-ups.The final report should include the following sections:

- Executive Summary
- Data/Code Impact Assessment
- Timeline of Events
- Identification & Analysis
- Root Cause Analysis
- Containment & Eradication Summary
- Recovery Summary
- Preventive Measures Implemented
- Regulatory Notifications (if any)
- CISO/IR Manager Sign-Off

NIST recommends that organizations conduct a Lessons Learned meeting shortly after the incident to review what happened, assess response performance, and identify improvements (NIST SP 800-61r2, §3.4). NIST also requires maintaining comprehensive incident documentation to support audits, investigations, and long-term process improvement (NIST SP 800-61r2, §3.4.2).

References

Bartock, M., Cichonski, J., Souppaya, M., Smith, M., Witte, G., & Scarfone, K. (2016). Guide for cybersecurity event recovery. *Guide for Cybersecurity Event Recovery*. https://doi.org/10.6028/nist.sp.800-184

CISA. (2021). *Cybersecurity Incident & Vulnerability Response Playbooks Operational Procedures for Planning and Conducting Cybersecurity Incident and Vulnerability Response Activities in FCEB Information Systems*. https://www.cisa.gov/sites/default/files/2024-08/Federal_Government_Cybersecurity_Incident_and_Vulnerability_Response_Playbooks_508C.pdf

National Institute of Standards and Technology. (2012). *Computer Security Incident Handling Guide (Special Publication 800-61 Revision 2)*. U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-61r2