CONFIDENTIAL DOCUMENT

# Penetration Testing Methodology v2.0

## Comprehensive Framework for Security Assessments

**Date:** February 4, 2026

**Classification:** Proprietary & Confidential

**Document ID:** SEC-PTM-2026-V2

# Section 1: Introduction

In an era where cyber threats evolve with unprecedented speed, traditional security measures often fall short of providing a complete defense. This document outlines the Enterprise Penetration Testing Methodology—a rigorous, repeatable, and standards-aligned framework designed to identify, analyze, and mitigate security vulnerabilities before they can be exploited by malicious actors. This methodology serves as the definitive guide for all security assessments performed by the Internal Security Team and Third-party Partners. It ensures consistency across different asset classes, provides clear expectations for stakeholders, and maintains a high bar for technical excellence. By shifting the focus from mere 'compliance-driven' testing to 'threat-informed' testing, we prioritize the protection of critical business assets and sensitive data.

# Section 2: Document Control

## 2.1 Version History

| Version | Date | Author | Description of Change |
|---|---|---|---|
| 1.0 | 2024-01-10 | Security Architecture Team | Initial Methodology Drafting |
| 1.5 | 2024-06-15 | Cyber Governance Group | Integration of NIST 800-115 standards and API testing modules. |
| 2.0 | 2026-02-04 | Lead Security Researcher | Complete overhaul; added Cloud and Mobile specific modules; updated risk scoring to CVSS v4.0. |

## 2.2 Change Log - Major Updates in v2.0

- **Cloud Native Assessment:** Added detailed procedures for AWS/Azure/GCP environment reviews.
- **DevSecOps Integration:** Methodology now includes hooks for CI/CD pipeline security validation.
- **Legal & Ethics:** Updated the Safe Harbor agreements and Rules of Engagement templates.

- **Post-Exploitation:** Expanded sections on lateral movement and privilege escalation in containerized environments.

## 2.3 Reviewers / Approvals

This document requires annual review and approval from the following departments:

- Chief Information Security Officer (CISO) - Final Approval
- Legal & Compliance Office - Regulatory Alignment
- VP of Engineering - Operational Impact Review
- IT Infrastructure Director - Technical Feasibility

## 2.4 Distribution List

This methodology is classified as **Internal - Read Only**. Distribution is managed via the Security Portal and is restricted to:

- Authorized Internal Penetration Testers
- External Security Auditing Partners
- Security Operations Center (SOC) Managers
- Compliance and Risk Officers

# Section 3: Purpose & Objectives

## 3.1 Why the Methodology Exists

Without a structured approach, penetration testing results are often inconsistent, leading to 'security theater' rather than genuine risk reduction. The purpose of this methodology is to provide a standardized roadmap that ensures every assessment is thorough, safe, and actionable.

## 3.2 Goals of Penetration Testing

The primary objectives of our security assessments are:

- **Risk Identification:** Discovering technical vulnerabilities, logic flaws, and configuration errors.
- **Defensive Validation:** Testing the effectiveness of existing security controls, such as WAFs, EDRs, and IDS/IPS systems.
- **Security Awareness:** Demonstrating the potential impact of vulnerabilities to business stakeholders to secure budget and resources for remediation.
- **Compliance:** Meeting regulatory requirements (PCI DSS, HIPAA, GDPR) that mandate regular security testing.

## 3.3 Intended Audience

This document is designed to be read by different stakeholders with various levels of depth:

- **Security Engineers:** To use as a step-by-step technical guide for execution.
- **Project Managers:** To understand timelines, dependencies, and logistics.
- **Executive Leaders:** To understand the value proposition and the rigorous standards applied to protect the organization.

# Section 4: Scope of Methodology

## 4.1 Types of Assessments Covered

This framework is modular and applies to the following assessment types:

- **Web Application Testing:** Deep dives into business logic, authentication, and data integrity.
- **Network Infrastructure (Internal/External):** Evaluation of servers, routers, firewalls, and protocols.
- **API Security:** Validation of RESTful, GraphQL, and SOAP endpoints for broken object-level authorization and injection.
- **Mobile Applications (iOS/Android):** Reverse engineering, local storage analysis, and traffic interception.
- **Cloud Environment Assessment:** Reviewing IAM policies, S3 bucket permissions, and serverless security.

## 4.2 In-Scope / Out-of-Scope Boundaries

Defining boundaries is critical for legal protection and operational stability. Specific IPs or URLs must be explicitly listed in the Statement of Work (SOW). General exclusions include:

- Third-party SaaS providers (unless explicitly authorized by the provider).
- Critical infrastructure components (UPS, HVAC) unless specifically requested.

- Physical security (tailgating, lockpicking) unless part of a "Red Team" engagement.

## 4.3 Engagement Assumptions

During an engagement, the following is assumed unless otherwise stated:

- Testers will have network visibility from the designated starting point.
- Uptime of the testing environment is managed by the client.
- Emergency contacts are available 24/7 for incident response.

# Section 5: Standards & Framework Alignment

Our methodology is not built in isolation; it is a synthesis of globally recognized security standards. This alignment ensures that our reports are recognized by regulators, auditors, and insurance providers.

## 5.1 OWASP Software Testing Guide (WSTG)

For web application and API assessments, we follow the OWASP Web Security Testing Guide. This provides a comprehensive framework for testing the full life cycle of an application including:

- Information Gathering
- Configuration and Deployment Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Input Validation Testing
- Error Handling
- Cryptography
- Business Logic Testing

- Client-side Testing

## 5.2 Penetration Testing Execution Standard (PTES)

PTES provides the overarching workflow of our methodology. It covers the seven main stages of a high-quality penetration test, from initial communication to the final report delivery and executive debrief.

## 5.3 NIST SP 800-115

We adopt the Technical Guide to Information Security Testing and Assessment from NIST. This is particularly relevant for federal-aligned organizations and provides the bedrock for our vulnerability scanning and network surveying techniques.

## 5.4 OSSTMM (The Open Source Security Testing Methodology Manual)

For network-level assessments, we use OSSTMM's operational security metrics. This focus on "Ravs" (Risk Assessment Values) helps quantify the attack surface of the target environment.

## 5.5 Compliance Mappings (ISO 27001, PCI DSS)

Our findings are categorized and mapped to common compliance frameworks:

- **PCI DSS 4.0:** Requirements 11.3 (Penetration Testing) and 6.x (Vulnerability Management).

- **ISO/IEC 27001:** Control A.12.6.1 (Management of technical vulnerabilities).
- **SOC 2:** Trust Services Criteria for Security and Availability.

# Section 6: Engagement Lifecycle Overview

A professional penetration test is a structured lifecycle. Each phase relies on the outputs of the previous phase to ensure depth and accuracy.

## 6.1 High-Level Flow Diagram (Conceptual)

The engagement follows a circular loop of discovery and validation:

1. **Pre-engagement:** Scoping, legal approvals, and ROE (Rules of Engagement).
2. **Discovery:** Passive and active reconnaissance to identify assets.
3. **Vulnerability Analysis:** Identifying weaknesses in the discovered assets.
4. **Exploitation:** Safely demonstrating the impact of found vulnerabilities.
5. **Post-Exploitation:** Understanding the depth of the breach (pivoting, data access).
6. **Reporting:** Documenting evidence and remediation steps.
7. **Retesting:** Verifying that the client has successfully mitigated the risks.

## 6.2 Phase Interdependencies

Each phase is not a silo. For example, Information Gathering (Phase 8) often feeds back into Scoping (Phase 4) if unexpected subdomains or business units

are discovered during DNS enumeration. Similarly, Post-Exploitation (Phase 12) may yield new credentials that restart the Vulnerability Analysis (Phase 10) on a different segment of the network.

## 6.3 Quality Gates

Between each major phase, a "Quality Gate" review is conducted by the Lead Pentester to ensure that:

- The assessment is still within the legal scope.
- No critical infrastructure is being negatively impacted.
- Data handling procedures are being strictly followed.

# Section 7: Pre-Engagement Phase

The pre-engagement phase is the most critical part of the penetration test from a legal and administrative perspective. Errors here can lead to legal liability or service outages.

## 7.1 Authorization & Legal Approval

No testing may commence without a signed 'Permission to Test' form. This document must include:

- The specific legal entity authorizing the test.
- Clear identification of the IP addresses, domains, and applications.
- Liability waivers for the testing team regarding accidental downtime if predefined rules were followed.

## 7.2 Rules of Engagement (RoE)

The RoE defines the 'how' of the assessment. Key components include:

- **Testing Windows:** e.g., Monday-Friday, 18:00 to 06:00 to minimize business impact.
- **Exclusion Lists:** Specific legacy systems known to be fragile.
- **IP Whitelisting:** The client must whitelist the tester's source IPs to prevent SOC lockout unless "blind testing" is the objective.

## 7.3 Communication Plan

A formal communication matrix is established:

- **Daily Status Updates:** High-level summary of activities performed and issues encountered.
- **Critical Findings:** Immediate notification (phone/encrypted chat) if a P1/Critical vulnerability is discovered.
- **Emergency Stop:** A "Kill Switch" protocol to halt all testing immediately if the client detects performance issues.

## 7.4 Success Criteria

Success is defined not just by finding bugs, but by answering specific questions, such as "Can an unauthenticated user access HR records?" or "Can a consultant on the guest Wi-Fi pivot to the production VLAN?"

## 7.5 Risk Acceptance

The client acknowledges that security testing inherently carries risk. By signing the SOW, they accept that automated scanning and manual exploitation may cause unexpected behavior in target systems.

# Section 8: Information Gathering & Reconnaissance

Reconnaissance is the art of discovering what an attacker can see before they even touch the target's infrastructure.

## 8.1 Passive Reconnaissance (OSINT)

Passive recon involves gathering information without directly interacting with the target organization's assets. Techniques include:

- **Search Engine Discovery:** Using Google Dorks to find leaked documents, indexing errors, and hidden directories.
- **Social Media Intelligence:** Reviewing LinkedIn and GitHub to identify employee names, technology stacks, and inadvertently leaked API keys in public repos.
- **WHOIS and DNS:** Investigating domain registration records and historical DNS entries.
- **Shodan/Censys:** Searching for the target's IP ranges in internet-wide scanners to see what was previously exposed.

## 8.2 Active Reconnaissance

Direct interaction with target systems to confirm existence and availability:

- **Port Scanning:** Using Nmap with optimized scripts to identify open services and versions.
- **DNS Enumeration:** Manual zone transfers (if allowed) or brute-forcing subdomains to identify development, staging, and UAT environments.
- **Service Fingerprinting:** Grabbing banners from SSH, FTP, and Web Servers to determine specific software builds.

## 8.3 Asset Identification & Attack Surface Mapping

All identified assets are cataloged into a Target List. This includes:

- Public-facing endpoints.
- Hidden administrative panels.
- Microservices and internal APIs.
- VPN gateways and remote access portals.

# Section 9: Threat Modeling

Threat modeling shifts the focus from "what is vulnerable" to "who could attack and how."

## 9.1 Application Architecture Review

Before deep testing, we attempt to recreate the application's architecture. This involves understanding how data flows from the UI through the API layer and into the Database/Storage tiers.

## 9.2 Trust Boundaries

We identify points where data moves between different levels of trust, such as:

- Internet -> Web Server (Unrestricted to Restricted).
- Web Server -> Database (Restricted to Highly Restricted).
- User Identity -> Administrative Identity.

## 9.3 Entry Points

Mapping all possible locations where a user can provide input, including:

- HTTP Request Parameters (GET/POST).
- Custom HTTP Headers.

- Cookies and JWT Tokens.
- File Upload Functionality.
- Webhooks and Third-party Integration callbacks.

## 9.4 Abuse Cases

We brainstorm "vulnerability scenarios" specific to the business logic. Example: "An attacker modifies their cart total during the checkout session to pay $0 for a high-value item."

# Section 10: Vulnerability Analysis

Once the attack surface is mapped, we begin the process of identifying specific flaws.

## 10.1 Automated Scanning Approach

While manual testing is superior for logic flaws, automated tools are essential for broad coverage. We utilize:

- **Dynamic Analysis (DAST):** Tools like Burp Suite Professional, OWASP ZAP, and Acunetix to crawl and test web applications.
- **Infrastructure Scanning:** Nessus or OpenVAS to identify unpatched software and weak TLS configurations.
- **Custom Scripts:** Python-based fuzzers for proprietary protocols or niche API endpoints.

## 10.2 Manual Testing Strategy

Manual testing accounts for 70% of the effort. We manually verify:

- Business logic bypasses (e.g., skipping payment steps).
- Authorization flaws (e.g., IDOR - Insecure Direct Object References).
- Complex injection vulnerabilities that require specific multi-stage steps to trigger.

## 10.3 False-Positive Handling

No vulnerability is reported without verification. If an automated tool flags a 'Critical' SQL Injection, the tester must manually confirm it with a non-destructive Proof of Concept (PoC) like a 'Sleep' command or version check before it enters the report.

## 10.4 Vulnerability Validation

Validation involves determining if a vulnerability is *exploitable* in the context of the environment. A vulnerability might exist but be unreachable due to network-level ACLs or Web Application Firewall (WAF) rules.

# Section 11: Exploitation

Exploitation is the bridge between a theoretical vulnerability and a demonstrated risk.

## 11.1 Exploit Selection Criteria

Testers must choose exploits that are:

- **Stable:** Minimizes the risk of crashing a service or "Blue Screening" a server.
- **Targeted:** Only executes the necessary code to prove the vulnerability.
- **Reliable:** Works across various versions of the target OS if possible.

## 11.2 Safe Exploitation Principles

We adhere to "First, Do No Harm." Principles include:

- Avoidance of 'Denial of Service' exploits unless specifically requested.
- Using 'benign' payloads (e.g., launching *calc.exe* or *id* command) rather than disruptive ones.
- Ensuring no persistent "backdoors" are left on the system without client awareness.

## 11.3 Proof-of-Concept (PoC) Standards

Every finding must have a PoC. This includes:

- Step-by-step instructions to reproduce the flaw.
- Screenshots with sensitive data (PII) redacted.
- Logged HTTP requests and responses.

## 11.4 Exploitation Depth Limits

We stop exploitation once the goal is reached. If the goal is to prove "Administrator" access, we do not need to dump the entire database once we have demonstrated we *could* have done so.

# Section 12: Post-Exploitation

What happens after the initial breach? This determines the true impact of a vulnerability.

## 12.1 Privilege Escalation

Attempting to move from a low-privileged account (guest/user) to a high-privileged account (Admin/Root/System). This involves searching for misconfigured services, insecure file permissions, and unpatched kernel vulnerabilities.

## 12.2 Lateral Movement

Testing the ability to move from one compromised system to others within the same network. Techniques include:

- **Pass-the-Hash (PtH):** Using stolen NTLM hashes to log into other Windows machines.
- **SSH Key Analysis:** Using found private keys to access other cloud instances.
- **Internal Scanning:** Scanning the internal network from a compromised 'pivot host'.

## 12.3 Data Access Validation

Demonstrating access to sensitive folders or databases to show real-world business impact. Testers will "list" filenames but generally will not "read" the contents of sensitive personal files.

## 12.4 Cleanup Procedures

Post-assessment, the tester must:

- Delete all uploaded shells, scripts, or temporary files.
- Revert any configuration changes made to facilitate the test.
- Restore original user accounts if any were created during testing.

# Section 13: Risk Rating Methodology

Findings are prioritized using a standardized risk scoring model to help stakeholders focus on the most impactful issues.

## 13.1 Severity Scoring Model (CVSS v4.0)

We utilize the Common Vulnerability Scoring System (CVSS) to provide an objective score based on:

- **Base Score:** Intrinsic qualities of the vulnerability (Attack Vector, Complexity, Privileges Required).
- **Temporal Score:** Factors that change over time (Exploit Code Maturity).
- **Environmental Score:** Factors specific to the target organization's implementation.

## 13.2 Likelihood vs Impact

Risk is also calculated as *Risk = Likelihood x Impact*.

- **Likelihood:** How easy is it for an attacker to find and exploit this? Factors include visibility and skill level required.
- **Impact:** What happens to the business if this is exploited? Factors include Confidentiality, Integrity, and Availability (CIA triad).

## 13.3 Business Context Integration

A "Critical" CVSS score on an isolated dev server might be downgraded to "Medium" in the final report, while a "Medium" score on a primary customer-facing payment gateway might be upgraded to "High" due to the business importance and potential for regulatory fines.

# Section 14: Reporting Methodology

The report is the only tangible deliverable of the penetration test. It must be professional, accurate, and secure.

## 14.1 Executive vs Technical Reporting

Our reports are split into two sections:

- **Executive Summary:** High-level overview for non-technical leadership. Includes a risk dashboard, key "wins" for the defense team, and global recommendations.
- **Technical Findings:** Detailed breakdown for engineers. Includes discovery evidence, replication steps, and specific remediation recipes (code snippets, config changes).

## 14.2 Evidence Standards

All evidence is stored securely during the test. Final reports include:

- Annotated screenshots.
- Redacted raw HTTP logs.
- Output from scanning tools.

## 14.3 Reproducibility Requirements

A finding is only valid if it can be reproduced by a third party. We provide the exact tool version and command syntax used to trigger every finding.

## 14.4 Data Handling

The report contains highly sensitive information. It must be delivered via encrypted channels (TLS-protected portal, PGP-encrypted email, or password-protected ZIP via secure file share).

# Section 15: Retesting & Verification

A penetration test is not complete until the identified risks are resolved.

## 15.1 Fix Validation Process

Once the development team marks a finding as "Fixed," the testing team performs a re-test. This involves running the same exploit or verification steps used originally to ensure the patch actually works and hasn't been bypassed.

## 15.2 Closure Criteria

An engagement is considered closed when:

- All Critical and High findings are mitigated or officially risk-accepted by management.
- A "Corrective Action Plan" is in place for Medium and Low findings.
- A final "Certificate of Completion" is issued for compliance purposes.

## 15.3 Regression Testing

Testers also perform a brief "sanity check" on related components to ensure that the security fix didn't break existing functionality or introduce new vulnerabilities (e.g., a fix for SQLi that introduces an XSS vulnerability).

# Section 16: Quality Assurance

Quality is guaranteed through multiple layers of review.

## 16.1 Peer Review

Every report is reviewed by a second penetration tester who was not involved in the original testing. This "fresh set of eyes" checks for logic errors, grammar issues, and ensure the remediation advice is practical.

## 16.2 Methodology Audits

On a quarterly basis, our methodology is audited against the latest releases from OWASP and NIST to ensure our internal procedures remain at the cutting edge of the industry.

## 16.3 Continuous Improvement

After chaque engagement, a "Post-Mortem" is held to discuss what went well and what could be improved in the workflow, toolset, or communication style.

# Section 17: Ethics & Legal Considerations

Testers act as "Ethical Hackers" and must maintain the highest integrity.

## 17.1 Responsible Disclosure

If a product-wide vulnerability is found in a commercial software used by the client (e.g., a 0-day in Windows or SAP), we follow a responsible disclosure process, notifying the vendor through the appropriate channels before making any information public.

## 17.2 Data Protection

Testers are custodians of client secrets. All data gathered during the test is stored on encrypted drives and is wiped 90 days after the final report is delivered, unless otherwise specified in the contract.

## 17.3 Tester Conduct Rules

Testers must never:

- Modify data for personal gain.
- Access or view PII/PHI unless necessary for the test.
- Share details of the engagement with external parties.

# Section 18: Limitations

## 18.1 Known Blind Spots

A penetration test is a "point-in-time" assessment. It cannot guarantee that a system is "unhackable" or that new vulnerabilities won't be discovered tomorrow.

## 18.2 Tool Constraints

Tools are only as good as their signatures. Obfuscated malware or custom-built zero-days may not be detected by automated scanners used during the vulnerability analysis phase.

## 18.3 Time-Boxing Impact

The biggest constraint is time. Real hackers have months; testers usually have weeks. This methodology prioritizes the "shortest paths to critical impact" to provide the most value within the allotted timeframe.

# Section 19: Conclusion

The Enterprise Penetration Testing Methodology provides a robust, ethical, and highly technical framework for evaluating organizational resilience. By following these 20 sections, we ensure that every assessment is conducted with precision and produces actionable intelligence that directly contributes to the security posture of the organization.

# Section 20: Appendices

## Appendix A: Daily Health Check Checklist

- [ ] Confirm target system availability.
- [ ] Verify logging is active on tester machine.
- [ ] Check for emergency stop emails from client.
- [ ] Sync findings to secure vault.

## Appendix B: Tool Mappings

| Category | Primary Tool |
| --- | --- |
| Proxy / Intercept | Burp Suite Pro |
| Reconnaissance | nmap, Amass, GoBuster |
| Cloud Testing | Pacu, ScoutSuite |
| Report Generation | Serpico / LaTeX |

## Appendix C: Sample Workflows

1. **Pre-req:** VPN Access -> 2. **Scan:** Nmap -> 3. **Enumerate:** Directory Brute-force -> 4. **Exploit:** SQL Injection -> 5. **Pivot:** Database privilege escalation.

## Appendix D: Glossary

- **RCE:** Remote Code Execution.
- **XSS:** Cross-Site Scripting.
- **SQLi:** SQL Injection.
- **PoC:** Proof of Concept.