HackerOne Pentest Methodology

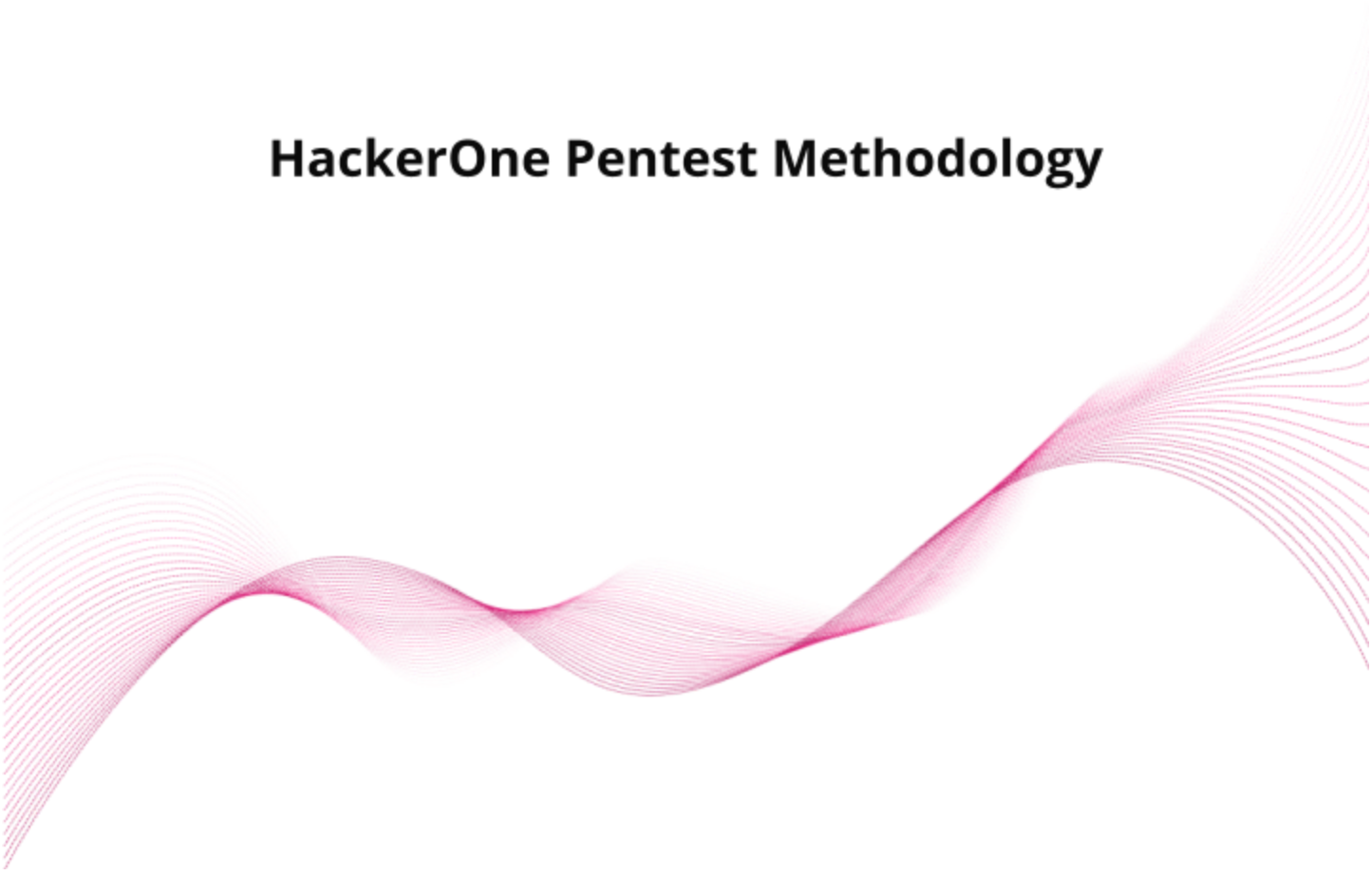


Table of Contents

Executive Summary

This document showcases HackerOne pentesters' methodologies throughout the pentest engagement to ensure comprehensive and deep coverage. It demonstrates how HackerOne assesses the effort required to perform thorough security engagements against customer technologies.

HackerOne developed this document by consulting internal and external industry experts, leveraging industry standards, evaluating thousands of time-bound and long-term global client programs, and reviewing millions of vulnerability reports. Through constant feedback and engagement, HackerOne continues to mature its methodologies in effort and approach to offer its customers maximum security assurance.

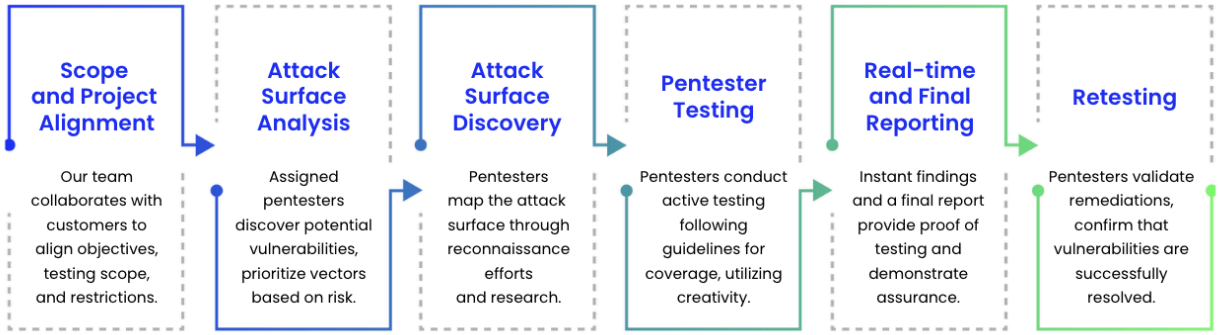
Scoping

HackerOne understands that no asset is the same; therefore, no pentest should be scoped using generic assumptions. Applying a one-size-fits-all model to pentests risks the quality and completion of the assessment. HackerOne has developed a scoping process that balances cumbersome traditional scoping processes with blanket figures.

In a customer-facing questionnaire, customers are asked to provide high-level details about the assets being engaged and relevant testing expectations. Once this information is gathered, HackerOne leverages a calculation process based on historical data to determine the proper product offering for its customers. HackerOne also uses this information to determine if and how pentesters can access the targets remotely. This process ensures pentesters can conduct quality work with reasonable expectations and customers are provided a thorough engagement bolstered by security assurance.

Phases

HackerOne's phased testing approach provides a series of guideposts for testers to ensure quality control and checkpoints throughout the engagement. This approach opens up opportunities to review progress and proactively communicate across parties to address questions as they come up. Customers gain increased awareness of testing activities to ensure these activities are completed in order. In this phased approach, testers understand the assets in-scope, assess the entire attack surface, and dive into discovered attack vectors using HackerOne-created guides to empower their creativity.



Project Alignment

HackerOne and the customer work closely to ensure they align on goals, Key Performance Indicators (KPI), and restrictions. Based on this understanding, HackerOne customizes testing guidance and assigns pentesters based on their skill sets and experience.

Attack Surface Discovery

Pentesters confirm access to all in-scope assets and gain an initial understanding of the attack surface they are asked to engage in or that could present avenues into it.

Attack Surface Analysis

Pentesters seek to identify potential attack vectors across the identified surface, prioritizing attack vectors based on their suspected ease of exploitation and potential impact.

Testing

Pentesters leverage the groundwork from previous phases to begin their assault against the attack surface, demonstrating the customer's security risk or exposure to a malicious actor. The pentest team uses a combination of automated and manual attacks to provide both breadth and depth of testing.

Reporting

Findings are reported in real-time through the platform, and customers can send them to issue trackers and vulnerability management software using built-in integrations.

Customers are provided with deliverables that empower them to communicate issues to internal teams, provide their own customers with proof of testing, review the state of security for scoped assets, and assure auditors of penetration testing.

Retesting

Pentesters review the remediations that customers implement to fix discovered vulnerabilities to confirm that the system has been successfully patched and is no longer vulnerable to the original findings. Customers have a window of time to request free retesting for the identified findings.

Project Alignment

Philosophy

It is always best to launch any project/engagement with alignment and a clear understanding of objectives rather than attempting to turn a moving train.

Outcome

There should be complete alignment on what assets are to be tested, how pentesters can access the targets remotely, how they are to be tested, and any restrictions regarding testing.

Methodologies

HackerOne uses an adaptive approach that meets customers' needs while maintaining a standard of excellence. It leverages team collaboration tools, media, meetings, training, and in-platform guidance and checklists. During testing, HackerOne pentesters follow asset-specific checklists for each asset in scope. The HackerOne checklists are based on industry-standard methodologies and security frameworks, such as OWASP Top 10[1] standards (web, API, and mobile), CWE guidance from MITRE[2], direct input from the AWS Security team, and the HackerOne Top 10[3] list.

[1] OWASP Top 10 Lists - https://owasp.org/Top10/
[2] MITRE - https://cwe.mitre.org/
[3] HackerOne Top 10 List - https://www.hackerone.com/top-ten-vulnerabilities

Attack Surface Discovery

Philosophy

Before any active attacks, pentesters must first understand and verify the entire surface through Open Source Intelligence (OSINT) Gathering and Reconnaissance.

Outcome

Pentesters are aware of the entire attack surface, including its perimeter, available access, and the size of the targeted assets when applicable.

Methodologies

Standard
Scope Review
Company Research

Web Application
Application mapping
Understand the business logic
Identify Core features to test
Identify input fields (injection points)
Brute Forcing directories
Google Dorking
Wayback machine/archive.org
Public Disclosures

Mobile Applications

Built-in Protections
Application mapping
Understand the business logic
Identify Core features to test
Identify input fields (injection points)
Looking for old versions of the same application
Google Dorking
Wayback machine/archive.org
Extract static strings and URLs from .apk or .ipa
Public Disclosures

Network
Quick TCP Port scanning to find live IPs
Full TCP port scanning
UDP port scanning
Nmap scripts
Shodan queries
Google Dorking
Wayback machine/archive.org (in relation to external network testing)

API
API mapping
Read API documentation
Understand the business logic

Identify Core endpoints to test
Public Disclosures
Versioning / Identify legacy API versions
Identify user interfaces (such as mobile apps) using this API

Desktop Applications / Executables
Built-in Protections
Application mapping
Understand the business logic
Identify Core features to test
Identify input fields (injection points)
Looking for old versions of the same application
Google Dorking
Wayback machine/archive.org
Extract static strings and URLs from the binary
Public Disclosures and CVEs

Attack Surface Analysis

Philosophy

Before engaging an attack vector, pentesters should discover possible attack vectors and entry points.

Outcome

Pentesters have an in-depth understanding of potential attack vectors and entry points within the attack surface and prioritize active testing toward the most promising observations.

Methodologies

Web Application
Automated Scanning
Focused Scanning
Findings Analysis

Mobile Application
Static Analysis
Integration Analysis
Automated Scanning
Focused Scanning
Findings Analysis

Network

Automated Scanning
Focused Scanning
Findings Analysis

Testing

Philosophy

Give pentesters the freedom to explore the attack surface utilizing their unique skills and expertise while providing coverage and consistency from each engagement.

Outcome

Testers complete testing against key objective areas developed for each asset technology. They are empowered with guidebooks for each objective that assist in ensuring the depth of their testing but do not confine it to prescriptive requirements.

Methodologies

Our methodology emphasizes structure and adaptability for meaningful assessments. Utilizing the insights gained from prior discovery and analysis of the attack surface, the testers probe weaknesses, follow potential attack paths, and adapt tactics based on system behaviour and findings. This adaptive approach successfully uncovers both existing vulnerabilities and novel attack vectors. By combining structured goals with expert analysis, our testing phase provides actionable insights that strengthen organizations' security against real-world threats. Supported methodologies include:

Web

A single web application is a contained codebase that typically interfaces with the end-user, passing commands over the internet and connecting UI with a web server. A web application is not a wildcard domain but a single fully-qualified domain name [FQDN]. Testing can be conducted internally or externally.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Broken Access Control
Cryptographic Failures
Injection
Insecure Design
Security Misconfiguration
Vulnerable and Outdated Components
Identification and Authentication Failures
Software and Data Integrity Failures
Security Logging and Monitoring Failures
Server-Side Request Forgery (SSRF)

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Broken Access Control
Task
Directory Traversal File Include
Bypassing Authorization Schema

Privilege Escalation
Insecure Direct Object References
OAuth Weaknesses
OAuth Authorization Server Weaknesses
OAuth Client Weaknesses
Business Logic Data Validation
Remote File Inclusion

Cryptographic Failures
Task
Weak Transport Layer Security
Padding Oracle
Sensitive Information Sent via Unencrypted Channels
Weak Encryption

Injection
Task
HTTP Parameter Pollution
SQL Injection
ORM Injection
Code Injection
Command Injection
Buffer Overflow
HTTP Splitting Smuggling
Server Side Template Injection
Reflected Cross-Site Scripting

Stored Cross-Site Scripting
DOM-based Cross-Site Scripting
HTML Injection
Cross-Site Flashing
Blind XSS

Insecure Design
Task
Privacy-related Controls in SDLC
Secure Design Patterns in Repeated Components
Adequate Threat Modeling: Critical Authentication, Access Control, Business Logic, Key Flows
Plausibility Checks
Adequate Handle of Misuse Cases
Adequate System and Network Layering
Adequate Segregation of Tenants
Adequate Resource Consumption Limits

Security Misconfiguration
Task
Network Infrastructure Configuration
Application Platform Configuration
File Extensions Handling for Sensitive Information
Review Old, Backup, and Unreferenced Files for Sensitive Information
Enumerate Infrastructure and Application Admin Interfaces
HTTP Methods
HTTP Strict Transport Security
RIA Cross Domain Policy

File Permission
Subdomain Takeover
Cloud Storage
Content Security Policy (CSP)
Path Confusion
Information Leakage (Search Engine Discovery and Recon)
Review Webpage Comments and Metadata for Information Leakage
Review Webserver Metafiles for Information Leakage
Clickjacking
XML Injection

Vulnerable and Outdated Components
Task
Enumerate Applications on Webserver
Map Application Architecture

Identification and Authentication Failures
Task
Credentials Transported over an Encrypted Channel
Default Credentials
Weak Lock-Out Mechanisms
Bypassing Authentication Schema
Vulnerable Remember Password
Browser Cache Weaknesses
Weak Password Policy
Weak Security Question Answer

Weak Password Change or Reset Functionalities
Weaker Authentication in Alternative Channel
Account Enumeration and Guessable User Account
Cookies Attributes
Session Fixation
Logout Functionality
Session Timeout
Session Puzzling

Software and Data Integrity Failures
Task
Altered Software or Data (Use of Digital Signatures and Similar Mechanisms)
Consumption of Trusted Dependencies
Dependency-Check SDLC Tooling
Code & Configuration Review Processes
CI/CD Pipeline Segregation, Configuration, and Access Control
Integrity of Serialized Data
Insecure Deserialization

Security Logging and Monitoring Failures
Task
Improper Error Handling
Error Code
Stack Traces

Server-Side Request Forgery (SSRF)
Task

Server-Side Request Forgery
Circumventing of Common SSRF Defenses
Blind SSRF
SSRF Vulnerabilities in Hidden Attack Surfaces

Android

A single software application designed to operate on the Android operating system. Testing could involve a development environment or a production phone/device.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Injection
Improper Platform Usage
Insecure Data Storage
Insecure Communication
Insecure Authentication
Insufficient Cryptography
Insecure Authorization
Client Code Quality
Code Tampering
Reverse Engineering
Extraneous Functionality

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Injections
Task
SQL Injection

XML Injection
Injection Attack Vectors
Memory Corruption
Cross-Site Scripting

Improper Platform Usage
Task
Insecure Intents
Insecure File Permissions

Insecure Data Storage
Task
Sensitive Data Is Sent to Third Parties
Local storage of Input Validation
Sensitive Data in Backups
Sensitive Data in Memory
Storage for Sensitive Data
Sensitive Data Exposure through Logs
Keyboard Cache being disabled for Text Input Fields
Sensitive Data Disclosure Through the User Interface
Sensitive Information in Auto-Generated Screenshots
Missing Device-Access-Security Policy

Insecure Communication
Task
Data Encryption on the Network
Critical Operations Using Secure Communication Channels

Custom Certificate Stores and Certificate Pinning
Network Security Configuration Settings
Security Provider
Endpoint Identification & Verification

Insecure Authentication
Task
Appropriate Authentication
Weak Stateless (Token-Based) Authentication
Weak OAuth 2.0 Flows
Missing Login Activity and Device Blocking
Confirm Credentials
Weak Biometric Authentication
Weak Stateful Session Management
Session Timeout

Insufficient Cryptography
Task
Insecure and/or Deprecated Cryptographic Algorithms
Common Configuration Issues
Weak Configuration of Cryptographic Standard Algorithms
Weak Key Management
Weak Random Number Generation

Insecure Authorization
Task
Insecure Direct Object References

Client Code Quality
Task
Weaknesses in Third-Party Libraries
Debug Mechanisms
Debugging Symbols
Debugging Code and Verbose Error Logging
Exception Handling
Properly Signed App

Code Tampering
Task
Root Detection
Anti-Debugging Detection
File Integrity Checks
Emulator Detection
Device Binding

Reverse Engineering
Task
Obfuscation
Missing Reverse Engineering Tools Detection

Extraneous Functionality
Task
Hidden Functions

iOS

A single software application designed to operate on the iOS operating system. Testing could involve a development environment or a production phone/device.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Injection
Improper Platform Usage
Insecure Data Storage
Insecure Communication
Insecure Authentication
Insufficient Cryptography
Insecure Authorization
Client Code Quality
Code Tampering
Reverse Engineering
Extraneous Functionality

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Injections
Task
SQL Injection
XML Injection

Injection Attack Vectors
Memory Corruption
Cross-Site Scripting

Improper Platform Usage
Task
Weak Permissions and Export of Sensitive Functionality.

Insecure Data Storage
Task
Sensitive Data Is Sent to Third Parties
Local Data Storage
Sensitive Data in Memory
Sensitive Data in Backups
Sensitive Data in Logs
Sensitive Data Is Exposed via IPC Mechanisms
Sensitive Data Disclosed Through the User Interface
Sensitive Data in the Keyboard Cache
Auto-generated screenshots for Sensitive Information

Insecure Communication
Task
Missing Data Encryption on the Network
Ensure that Critical Operations Use Secure Communication Channels
Custom Certificate Stores and Certificate Pinning
Missing Endpoint Identity Verification

Missing App Transport Security

Insecure Authentication
Task
Appropriate Authentication is in Place
Weak Stateless (Token-Based) Authentication
Weak OAuth 2.0 Flows
Local Authentication
Missing Login Activity and Device Blocking
Weak Stateful Session Management
Weak Session Timeout

Insufficient Cryptography
Task
Insecure and/or Deprecated Cryptographic Algorithms
Common Configuration Issues
Configuration of Cryptographic Standard Algorithms
Weak Key Management
Weak Random Number Generation

Insecure Authorization
Task
Insecure Direct Object References

Client Code Quality
Task

Weaknesses in Third-Party Libraries
Debuggable App
Debugging Symbols
Debugging Code and Verbose Error Logging
Exception Handling Weaknesses
Missing Properly Signed App

Code Tampering
Task
Missing Jailbreak Detection
Missing File Integrity Checks
Missing Device Binding

Reverse Engineering
Task
Missing Obfuscation

Extraneous Functionality
Task
Hidden Functionalities

API

Generally, an Application Programming Interface (API) is a set of routines, protocols, or methods used for integration and communication between two or more software applications.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Broken Object Level Authorization
Broken Authentication
Broken Object Property Level Authorization
Unrestricted Resource Consumption
Broken Function Level Authorization
Unrestricted Access to Sensitive Business Flows
Server-Side Request Forgery (SSRF)
Security Misconfiguration
Improper Inventory Management
Unsafe Consumption of APIs

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Broken Object Level Authorization
Task
Directory Traversal / File Include
Bypassing Authorization Schema
Privilege Escalation

Insecure Direct Object References (IDOR)

Broken Authentication
Task
Credential Stuffing
Lockout Mechanisms to Prevent Brute-force Attacks
Weak Passwords
Sensitive Authentication Details in URL Paths
Credential Updates Without Password Confirmation
Token Authenticity Validation
Accepting of Unsigned / Weakly Signed JSON Web Tokens (JWTs)
Unvalidated JWT Expiration Dates
Unsafe Password Transmission: Plaintext, Non-encrypted, Weakly Hashed
Weak Encryption Keys
[Microservices] Cannot Access Other Microservices Without Authentication
[Microservices] Weak or Predictable Tokens for Authentication

Broken Object Property Level Authorization
Task
Sensitive Object-Level Details Exposed via API Endpoint
Unauthorized Manipulation of Object Properties via API Endpoint

Unrestricted Resource Consumption
Task
Missing or Inappropriately Configured Execution Timeouts
Missing or Inappropriately Configured Maximum Allocable Memory
Missing or Inappropriately Configured Maximum Number of File Descriptors

Missing or Inappropriately Configured Maximum Number of Processes
Missing or Inappropriately Configured Maximum Upload File Size
Missing or Inappropriately Configured Number of Operations in a Single API Request
Missing or Inappropriately Configured Records Per Page Returned in a Single Response
Missing or Inappropriately Configured Third-party Service Providers' Spending Limit

Broken Function Level Authorization
Task
Privilege Escalation
Insecure Direct Object References (IDOR)
Forced Browsing / Predictable Resource Location

Unrestricted Access to Sensitive Business Flows
Task
Scalping
Scamming
Sniping
Expediting

Server Side Request Forgery (SSRF)
Task
Server-Side Request Forgery (SSRF)

Security Misconfiguration
Task
Missing Appropriate Security Hardening Across Application Stack
Missing Security Patches

Enabled Unnecessary Features
Missing or Weak Transport Layer Security (TLS) Configuration
Lack of Security Headers
Sensitive Information Exposed Through Stack Traces
Missing or Misconfigured Cross-Origin Resource Sharing (CORS) Policy
Missing or Misconfigured Cache-Control Directives

Improper Inventory Management
Task
Old or Unpatched API Versions
Documentation Existence and Accuracy
Lack of Retirement Plan for API Versions
Missing or Outdated Hosts Inventory
Outdated or Missing Integrated Services Inventory

Unsafe Consumption of APIs
Task
Interactions with Other APIs Over an Unencrypted Channel
Data from Other APIs processed without Validation / Sanitization
Blind Follow of Redirections
Third-party Services Responses Processed Without Limit
Interactions with Third-party Services Without Timeouts

External Network

In a general sense, a network is a group of cloud-hosted servers within a range of IPs or with individual IPs. These networks can be public-facing (external), require special access, or be in a cloud environment.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Network Discovery
Vulnerability Assessment
Domain Name System (DNS) Checks
Applicable Web Application Pentest Checks

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Network Discovery
Task
TCP port scan discovery
UDP port scan discovery
Service Identification
Operating System Fingerprinting
Google Dorking Discovery

Vulnerability Assessment
Task

Outdated or Vulnerable Software
Account Enumeration and Guessable User Account
Default/Weak Credentials
Internal Services Accessible from the Internet
Insecure TLS Versions and Weak Algorithms/Ciphers

DNS Checks
Task
DNS Cache Poisoning
DNS Cache Snooping
DNS Open Recursion
DNS Zone Transfer
Reverse DNS Name Resolution Discloses Private Network Addresses

Applicable Web Application Pentest Checks
Task
Critical Bugs on identified Web Applications (SQLi, RCE, SSRF, etc.)
Bypassing Authentication Schema
Weak authentication on alternative channels
Old, Backup, and Unreferenced Files for Sensitive Information
Network/Infrastructure Configuration
Weak Application Platform Configuration
File Extensions Handling for Sensitive Information
Sensitive information sent via unencrypted channels
Information Leakage using Search Engine Discovery
Information Leakage via Webpage Comments and Metadata

Enumerate Applications on Webserver

Internal Network

Generally, an internal network refers to the network infrastructure and assets within an organization's physical premises or private cloud environment. These networks are not public-facing and require special access privileges to be accessed.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Network Discovery
Vulnerability Assessment
Active Directory Assessment
Privilege Escalation Assessment
Applicable Web Application Pentest Checks

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Network Discovery
Task
TCP port scan discovery
UDP port scan discovery
Service Identification
Operating System Fingerprinting
Internal network topology mapping

Vulnerability Assessment

Task
Outdated or Vulnerable Software
Default/Weak Credentials
Insecure Configurations and Settings
Missing Patches and Updates
Internal Services Accessible from Unauthorized Users
Unrestricted File System Access
Sensitive Data Stored on Local Machines
Assessment of SSH, FTP, DNS, Email protocols, CI/CD, NFS, SMB, LDAP, RDP, SNMP, etc.

Active Directory Assessment
Task
Enumeration of Users, Groups and Systems
Assessment of Domain Policies and Permissions
Identification of Misconfigured Active Directory Services
Assessment of Domain Trusts and Relationships

Privilege Escalation Assessment
Task
Exploitation of Misconfigured Services and Applications
Assessment of Weak Credentials and Password Policies
Identification of Unauthorized Access Points and Channels
Assessment of Firewall and Access Control Rules
Lateral Movement and Escalation of Privileges

Applicable Web Application Pentest Checks
Task

Critical Bugs on identified Web Applications through BlackBox Testing (SQLi, RCE, SSRF, etc.)
Bypassing Authentication Schema
Weak authentication on alternative channels
Old, Backup, and Unreferenced Files for Sensitive Information
Network/Infrastructure Configuration
Weak Application Platform Configuration
File Extensions Handling for Sensitive Information
Information Leakage via Webpage Comments and Metadata
Enumerate Applications on Webserver

Network Segmentation Testing

Generally, a network segmentation test validates network segmentation controls with a focus on isolation between network security classifications.

This methodology is based on industry standards for network isolation and the controls defined in Payment Card Industry Data Security Standard (PCI DSS) segmentation requirements.

This includes:

- Host Discovery
- Port Scanning
- Testing for Network Broadcast Listeners
- Firewall and Access Controls
- Bi-Directional Testing
- Cloud / Container Specific Network Segmentation Tests (If Applicable)

Note: It is the customer's responsibility to define the scope and the placement of a Kali Linux Virtual Machine (VM) inside the correct network segment from which HackerOne will perform testing.

Some tools that may be utilized during the test include Nmap, Rustscan, Naabu, Masscan, etc.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Host Discovery
Port Scanning
Firewall and Access Controls
Cloud / Container Specific Network Segmentation Tests (If Applicable)

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Host Discovery
Task
Testing for live hosts in existing network segment
Scan IP ranges / subnets for live hosts

Port Scanning
Task
Full TCP port scans (1-65535)
Full UDP port scans (1-65535)
Service scans
IPv6 port scanning

Testing for Network Broadcast Listeners
Task
Monitor for Broadcast traffic
LLMNR/NetBIOS Poisoning Detection

Firewall and Access Controls
Task
Tests for commonly restricted protocols
Tests using source port manipulation
Firewall / IDS evasion tests

Nmap scripts for firewall bypass and web filtering

Bi-Directional Testing
Task
All of the above tests to confirm access from non-CDE networks to CDE segments and vice-versa

Cloud Environment Specific Network Segmentation Tests (If Applicable)
Task
All of the above tests from outside of the environment looking into your cloud environment

Note: Traditional methods of network segmentation tests are not possible in cloud environments. The network segmentation testing methodology includes tests related to host discovery, port scanning, and firewall and access controls from outside of the environment looking into the customer's cloud environment. HackerOne recommends performing a cloud configuration review of specific cloud configurations and container isolation if specific segmentation checks for those services are required.

AWS Security Configuration Review

Ensure comprehensive coverage of AWS security configurations and compliance with best practices.

The test primarily focuses on the AWS services and configurations critical to the organization's cloud infrastructure security.

This includes:

- Identity and Access Management (IAM)
- Amazon S3
- AWS Config
- CloudTrail
- EC2
- RDS
- VPC

Some tools that may be utilized during the test include Prowler, ScoutSuite, and Nessus.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Identity and Access Management
EC2
Logging
Monitoring
Networking
Storage
Database
Key Management

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Identity and Access Management
Task
IAM policies should not allow full "*" administrative privileges.
IAM users' access keys should be rotated every 90 days or less.
IAM root user access key should not exist.
MFA should be enabled for all IAM users that have a console password.
Hardware MFA should be enabled for the root user.
MFA should be enabled for the root user.
Ensure the IAM password policy requires a minimum password length of 14 or greater.
Ensure the IAM password policy prevents password reuse.
Ensure a support role has been created to manage incidents with AWS Support.
IAM user credentials unused for 45 days should be removed

EC2
Task
EC2 Amazon Machine Images (AMI) are inaccessible to all AWS accounts.

Logging
Task
CloudTrail should be enabled and configured with at least one multi-region trail that includes read and write management events.
CloudTrail should have encryption at rest enabled.

CloudTrail log file validation should be enabled.
CloudTrail trails should be integrated with Amazon CloudWatch Logs.
Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible.
Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket.
CloudTrail should be enabled and configured with at least one multi-region trail that includes read and write management events.

Monitoring
Task
A log metric filter and alarm should exist for usage of the "root" user.
Ensure a log metric filter and alarm exist for IAM policy changes.
Ensure a log metric filter and alarm exist for CloudTrail AWS Configuration changes.
Ensure a log metric filter and alarm exist for AWS Management Console authentication failures.
Ensure a log metric filter and alarm exist for disabling or scheduled deletion of customer-managed keys.
Ensure a log metric filter and alarm exist for S3 bucket policy changes.
Ensure a log metric filter and alarm exist for AWS Config configuration changes.
Ensure a log metric filter and alarm exist for security group changes.
Ensure a log metric filter and alarm exist for changes to Network Access Control Lists (NACL).
Ensure a log metric filter and alarm exist for changes to network gateways.
Ensure a log metric filter and alarm exist for route table changes.
Ensure a log metric filter and alarm exist for VPC changes.

Networking
Task
VPC default security groups should not allow inbound or outbound traffic.
VPC flow logging should be enabled in all VPCs.

Network ACLs should not allow ingress from 0.0.0.0/0 to port 22 or port 3389.

Storage
Task
S3 Block Public Access setting should be enabled.
S3 buckets should require requests to use Secure Socket Layer.
S3 Block Public Access setting should be enabled at the bucket level.
S3 general-purpose buckets should have MFA delete enabled.

Database
Task
RDS DB instances should have encryption at rest enabled.

Key Management
Task
AWS Key Management Should be enabled
KMS key rotation feature should be enabled for all Customer Master Keys (CMK).
KMS master keys should not be publicly accessible.

Azure Security Config Review

Ensure comprehensive coverage of Azure security configurations and compliance with best practices.

The methodology is based on the best security practices defined in the CIS Microsoft Azure Foundations and the security pillar of the Azure well-architected framework. The test primarily focuses on the Azure services and configurations critical to the organization's cloud infrastructure security.

This includes:

- Entra ID
- App Service
- Virtual Machines
- Storage Accounts
- Virtual Networks
- Database Services
- Logging and Monitoring
- Defender
- Azure Key Vault

Some tools that may be utilized during the test include Prowler, ScoutSuite, and Nessus.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Entra ID
App Service
Virtual Machines
Storage Accounts
Virtual Networks
Database Services
Logging and Monitoring
Defender

Azure Key Vault

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Entra ID
Task
MFA should be enabled for Windows Azure Service Management API
Security Defaults should be enabled on Microsoft Entra ID
MFA should be enabled for all privileged and non-privileged users
Security group creation should be restricted to administrators only
Guest user's access restrictions should be set to properties and memberships of their own directory objects
Non-admin users should be restricted from creating tenants and registering third-party applications
Microsoft 365 group creation should be restricted to administrators only

App Service
Task
App Service Authentication should be set up for apps in Azure App Service
FTP and FTPS deployments should be disabled for Azure Functions
Logging for Azure AppService 'HTTP logs' should be enabled
Web App should redirect All HTTP traffic to HTTPS in Azure App Service
Azure Functions should not be publicly accessible
Azure functions should not be configured with an identity with admin privileges
The latest version of runtime software should be used
Azure functions should be configured to use access keys

Virtual Machines
Task
Microsoft Azure virtual machines (VMs) should be configured to use managed disk volumes
VM-managed disk volumes (OS and data disk volumes) should be encrypted with a managed Key (CMK)
Unattached disks in the subscription should be encrypted with Customer Managed Key (CMK)

Storage
Task
Public access should be disabled for all blob containers
Shared Access Signature (SAS) tokens should be set to expire within an hour
Private endpoints should be configured for Microsoft Azure Storage accounts
Azure Storage accounts should use Customer Managed Keys (CMKs) instead of Microsoft Managed Keys
Storage account should be configured to deny access to traffic from all networks (including Internet traffic) to limit access

Virtual Networks
Task
Network watcher and virtual network flow logging should be enabled
Network Security Groups (NSGs) should not allow unrestricted access (0.0.0.0/0) on UDP ports
Network security groups (NSGs) should not allow ingress from 0.0.0.0/0 to port 22 or port 3389

Database Services
Task
Auditing should be enabled for SQL server
SSL connection should be enforced on PostgreSQL

Server logging parameters should be enabled for PostgreSQL Database Server
SQL server's TDE protector should be encrypted with Customer Managed Key (CMK)

Logging and Monitoring
Task
Activity Log Alert should exist for changes to security policies, network security groups, security solutions, and SQL firewall rules.
Logging for Azure Key Vault should be enabled

Defender
Task
Azure Defender should be enabled for servers, App Service, SQL databases, Storage Accounts, and other services in use
Latest OS patches for all virtual machines should be applied ('Apply system updates' status should be 'Completed')

Azure Key Vault
Task
Automatic key rotation should be enabled within Azure Key Vault
Azure Key Vault logging should be enabled
Role Based Access Control should be enabled for Azure Key Vault
Private Endpoints should be used for Azure Key Vault

Google Cloud Platform Security Config Review

Ensure comprehensive coverage of Google Cloud Platform security configurations and compliance with best practices.

The methodology is based on the best security practices defined in the CIS Google Cloud Platform Foundations and general Google Cloud security best practices. The test primarily focuses on the Google Cloud services and configurations critical to the organization's cloud infrastructure security.

This includes:

- Identity and Access Management
- Logging and Monitoring
- Networking
- Virtual Machines
- Storage
- Database
- BigQuery
- Key Management System (KMS)
- Google Container Registry (GCR)
- Google Kubernetes Engine (GKE)

Some tools that may be utilized during the test include Prowler, ScoutSuite, and Nessus.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Identity and Access Management
Logging and Monitoring
Networking
Virtual Machines
Storage
Database Services
BigQuery

Key Management System (KMS)
Google Container Registry (GCR)
Google Kubernetes Engine (GKE)

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Identity and Access Management
Task
Ensure Multi-Factor Authentication is Enabled for All Non-Service Accounts
Ensure Service Account External Keys are Rotated Within 90-Day Intervals
Ensure Service Accounts Operate Without Administrative Privileges
Ensure Service Accounts Utilize Only GCP-Managed Keys
Ensure User-Managed Service Account Keys Undergo 90-Day Rotation Cycles
Ensure Project-Level IAM Policies Exclude Service Account User and Token Creator Role Assignments
Ensure API Keys Exist Exclusively for Services in Active Use
Ensure API Keys Follow a 90-Day Rotation Schedule
Ensure API Keys Maintain Restrictions to Required APIs Only

Logging and Monitoring
Task
Ensure That Cloud Audit Logging Is Configured Properly
Ensure Log Metric Filters and Alerts Are Configured to Monitor Critical Configuration Changes Including Audit Settings, Storage IAM Permissions, Custom Roles, Project Ownership, SQL Instances, VPC Firewall Rules, VPC Networks, and VPC Routes
Ensure Log Entry Export Utilizes at Least One Configured Sink

Ensure That Google Cloud Audit Logs Feature Is Configured To Track All GCP Services And User Activities

Networking
Task
Ensure That SSH, and RDP Access Is Restricted From the Internet
Ensure that VPC Flow Logs is Enabled for Every Subnet in a VPC Network
Ensure That the Default Network Does Not Exist in a Project
Ensure That DNSSEC Is Enabled for Cloud DNS

Virtual Machines
Task
Ensure Compute Instances Are Not Configured To Use Default Service Account
Ensure App Engine Applications Enforce HTTPS-Only Connections
Ensure That Compute Instances Do Not Have Public IP Addresses
Ensure "Block Project-wide SSH Keys" Setting is Enabled for VM Instances

Storage
Task
Ensure That Cloud Storage Bucket Is Not Anonymously or Publicly Accessible
Ensure That Cloud Storage Buckets Have Uniform BucketLevel Access Enabled

Database Services
Task
Ensure That a MySQL Instance Does Not Allow Anyone To Connect With Administrative Privileges

Ensure That Cloud SQL Database Instances Do Not Have Public IP Addresses

BigQuery
Task
Ensure BigQuery tables and datasets are encrypted with Customer-Managed Keys (CMKs)
Ensure That BigQuery Datasets Are Not Anonymously or Publicly Accessible

Key Management System (KMS)
Task
Ensure Cloud Key Management Service (KMS) Keys Undergo Rotation Within 90-Day Periods
Ensure KMS Key IAM Policies Prevent Anonymous and Public Access

Google Container Registry (GCR)
Task
Ensure Image Vulnerability Scanning using GCR Container Scanning or a third-party provider

Google Kubernetes Engine (GKE)
Task
Ensure GKE Clusters Operate with Custom Service Accounts Instead of Compute Engine Default Service Account
Ensure network access to GKE clusters is restricted

Desktop Applications / Executables

In a general sense, a desktop application or executable is an application that runs locally on a computer device. This application runs independently of a web browser. Other examples include Command Line Interfaces (CLI), development tools, and utility programs. This application can perform routines and methods that communicate with the internet for various purposes.

HackerOne maintains a dedicated methodology for desktop applications built using the Electron framework. See the section Electron Desktop Applications.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Data Storage
Privileged Component
Network Communication
Other communication mechanisms with remote servers
Cryptography
Input sanitization
IPC/RPC
Denial of Service
Imports and exports
Code Quality
Authentication/Authorization Verification

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Data Storage

Unencrypted Sensitive Data Storage
Insertion of Sensitive Information into Externally-Accessible File or Directory
Insertion of Sensitive Information into Log File
UNIX Symbolic Link (Symlink) Following

Privileged Components
Creation of Temporary File with Insecure Permissions
(Windows) Unquoted Service Paths
Incorrect Permission Assignment for Critical Resource
(MacOS) Insecure Permissions on Binaries/Scripts
(Windows) Insecure Registry Permissions
(Linux) Permissive AppArmor / SELinux policies
Insecure Operations on Binaries/Scripts (local job scheduling)
Insecure Permissions on Startup Processes
Improper Authentication (New System User)
Abuse Elevation Control Mechanism (Setuid and Setgid)

Network Communication
Unauthenticated Services Listening on Localhost
Unnecessary Port Exposure
Public-facing Administrative Interfaces
Network Sniffing

Communication Mechanisms with Remote Servers
Remote APIs/Services

Cryptography

Secure Transport Layer Channel Verification
Secure Critical Operation Channel Verification
Custom Cryptographic Protocol
Weak Message Signature Verification and Hashing Collisions
Insecure and Deprecated Cryptographic Algorithms
Improper Certificate Validation
Weak Pseudo-Random Number Generator
Hardcoded Keys

Input Sanitization
Path Traversal Issues / ZIP Slip Vulnerabilities
XML External Entity Injection
Improper Input Injection

Inter-process Communication/Remote Procedure Call (IPC/RPC)
(Windows) Insecure Named Pipe Permissions
(MacOS/Linux) Insecure Permissions on Unix Domain Sockets
Custom Application Handle Schemas
Permissions for Custom Application Handle Schemas
Known Vulnerabilities in the RPC Implementation (e.g., RMI, AMF)

Denial of Service
External File Processing Operations
Externally Controlled File Operations
General Operations Impacting System Resources

Imports and Exports

(Windows) Testing for DLL Hijacking
(MacOS) Testing for Dylib Hijacking
Third-Party Libraries with Assigned CVEs

Code Quality
Binary Protection Mechanisms
Code Signing
Embedded Symbols/PDB Files
Race Conditions
Debugging Code and Verbose Error Logging
Obfuscation / Anti-debugging detection
File Integrity Checks on Critical Assets (certificates/keys)
General Exceptional Conditions

Authentication/Authorization Verification
Appropriate Authentication Verification
Password Best Practices
Session Timeout
Use of Single-factor Authentication
OAuth 2.0 Flows
Improper Restriction of Excessive Authentication Attempts
Insecure Direct Object References

Application Pentest for AWS (Add-on)

This is an “add-on” methodology if your web, mobile, or API-based application is hosted on AWS. You might have an array of services that support the platform, such as EC2, RDS, S3, Lambda, etc. This assessment will largely resemble a traditional application pentest but requires special consideration for specific AWS services used within your stack.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
API Gateway: HTTP Verb Tampering
API Gateway: Improper Access Control
DynamoDB: Injection
EC2: Local File Read / Local File Inclusion
EC2: Secrets Metadata
IAM Roles: Improper Access Control
Lambda: Injection & Pivoting
Lambda: Legacy Endpoint
Public Services or Resources
Route53/S3/EC2: DNS Misconfiguration & Subdomain Takeovers
S3 Bucket: Information Disclosure
S3 Bucket: Read Misconfiguration
S3 Bucket: Write Misconfiguration
Sensitive Data Exposure and Information Exposure Through Debug Information
Using Components with Known Vulnerabilities

Large Language Model (LLM) Applications (Add-on)

This is another “add-on” methodology for applications that leverage large language models (LLMs). It is tailored for testing AI/LLM functionalities integrated into larger, traditional applications. The focus is on secure integration, data flows, and the potential for the AI component to introduce new vulnerabilities into the existing application.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Prompt Injection
Sensitive Information Disclosure
Improper Output Handling
System Prompt Leakage
Misinformation
Unbounded Consumption

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Prompt Injection
Task
Basic Direct Injection Testing
Simple Indirect Injection via Content
Role Playing Restriction Bypass
Multi-prompt Restriction Bypass
Multi-language Restriction Bypass
Code Injection via Prompts

Unintentional Context Leakage
Output Manipulation

Sensitive Information Disclosure
Task
Advanced PII Exfiltration Techniques
Business Logic Disclosure
Internal System Architecture Revelation
API Key and Credential Extraction
User Session Data Leakage
Multi-tenant Data Isolation Testing
Differential Privacy Bypass

Improper Output Handling
Task
Arbitrary Code Execution via Unvalidated / Unsanitized Responses
Cross-Site Scripting (XSS) via Unvalidated / Unsanitized Responses
SQL Injection via Unvalidated / Unsanitized Responses
Output Sanitization Testing

System Prompt Leakage
Task
System Prompt Extraction
Internal Rule Discovery
Configuration Detail Exposure
Security Control Bypass via Prompt Leakage

Credential Discovery in System Prompts

Misinformation
Task
Hallucination
False Information
Biased Output Assessment
Misleading Content Generation
Fact Verification Bypass
Source Attribution Manipulation

Unbounded Consumption
Task
Rate Limiting Assessment
Denial of Service Testing
Cost Amplification Testing
Queue Overflow Testing

Large Language Model (LLM) Applications (Standalone)

This standard methodology is a full deep-dive for AI-native products and high-risk deployments. These applications utilize a large-scale artificial intelligence model specializing in understanding, generating, and working with human language. Applications and plugins powered by LLM technologies carry a unique set of risks. It encompasses a broader range of threats, including sophisticated adversarial attacks and in-depth model analysis. These standard methodology tasks extend and deepen those in the LLM Add-on methodology, adding advanced techniques and end-to-end coverage.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Prompt Injection
Sensitive Information Disclosure
Supply Chain Vulnerabilities
Data and Model Poisoning
Improper Output Handling
Excessive Agency
System Prompt Leakage
Vector and Embedding Weaknesses
Misinformation
Unbounded Consumption
Agentic AI Security Threats
AI Safety

*** Bold entries indicate objectives that differ from the Add-on variant.**

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Prompt Injection
Task
Advanced Direct Injection Testing
Complex Indirect Injection Scenarios
Multi-modal Injection (Image/Audio/Video)
Adversarial Suffix Attacks
Payload Splitting Techniques
Cross-modal Attack Vectors
Obfuscation Techniques (Base64, Unicode, Emoji)
Multi-language Obfuscation
Virtualization Attacks
Token Smuggling
Context Window Exploitation
Chain-of-Thought Manipulation
Role-based Injection Testing
System Prompt Override
Instruction Hierarchy Bypass

Sensitive Information Disclosure
Task
Advanced PII Extraction Techniques
Model Inversion Attacks
Membership Inference Attacks73
Proprietary Algorithm Exposure
Business Logic Disclosure

Internal System Architecture Revelation
API Key and Credential Extraction
User Session Data Leakage
Multi-tenant Data Isolation Testing
Differential Privacy Bypass

Supply Chain Vulnerabilities
Task
Advanced Model Provenance Analysis
LoRA Adapter Security Assessment
Model Hub Security Testing
AI Bill of Materials (AI-BOM) Analysis
Dataset Poisoning via Supply Chain
Backdoor Detection in Pre-trained Models
Model Registry Access Control Testing

Data and Model Poisoning
Task
Training Data Poisoning
Fine-tuning Data Manipulation
Embedding Data Poisoning
Label Poisoning Attacks
Backdoor Trigger Injection
Model Parameter Manipulation
Bias Amplification Testing
Model Behavior Alteration

Targeted Poisoning Attacks

Improper Output Handling
Task
Advanced XSS via LLM Output
Markdown Image injection
SQL Injection via Generated Queries
Command Injection Testing
Path Traversal via LLM
LDAP Injection
XML Injection Testing
Template Injection Attacks
Deserialization Attacks
API Abuse via Generated Content
Database Manipulation Testing
System Command Execution

Excessive Agency
Task
Autonomous Action Testing
Permission Escalation Assessment
Function Call Abuse Testing
Tool Misuse Detection
API Exploitation via Agents
Multi-agent Coordination Attacks
Workflow Manipulation

Decision Override Testing
Resource Access Abuse
External System Integration Exploitation
Privilege Boundary Testing
Human Approval Bypass
Agent-to-Agent Communication Exploitation

System Prompt Leakage
Task
Advanced Prompt Extraction Techniques
System Architecture Discovery
Internal Logic Revelation
Security Control Enumeration
Configuration Parameter Extraction
Business Rule Discovery
Filtering Mechanism Bypass
Administrative Instruction Exposure

Vector and Embedding Weaknesses
Task
RAG System Exploitation
Vector Database Injection
Embedding Manipulation
Similarity Search Poisoning
Context Injection via Embeddings
Multi-tenant Vector Leakage

Embedding Inversion Attacks
Knowledge Base Poisoning
Document Injection Testing
Semantic Search Manipulation
Vector Store Access Control Testing
Retrieval Ranking Manipulation
Cross-tenant Context Leakage
Embedding Model Backdoors

Misinformation
Task
Advanced Hallucination Generation
False Fact Injection
Source Attribution Manipulation
Evidence Fabrication
Credibility Score Manipulation
Citation Spoofing
Authoritative Source Impersonation
Statistical Data Manipulation
Historical Event Distortion
Scientific Misinformation Generation
News Article Fabrication
Expert Opinion Simulation
Research Citation Manipulation

Unbounded Consumption
Task
Advanced DoS Testing
Resource Exhaustion Attacks
Cost Amplification
Model Extraction Attacks
API Abuse
Computational Resource Hijacking
GPU Memory Exhaustion
Token Limit Exploitation

Agentic AI Security Threats
Task
Agent Goal Manipulation
MCP Protocol Security Tests
Memory Corruption Attacks
Tool Misuse Exploitation
Privilege Compromise Testing
Authentication Bypass via Agents
Asynchronous Workflow Exploitation
Identity Spoofing
AI-powered Social Engineering
Cascading Failure Induction
Data Exfiltration via Agent Actions
Lateral Movement via Agent Networks

AI Safety
Task
Safety policy jailbreak probes
Safety filter bypass checks
Harmful content generation

Electron Desktop Applications (Add-on)

This methodology is used in conjunction with the HackerOne Web Methodology to evaluate desktop applications built using Electron with an underlying Chromium architecture. Because these applications are files installed on users' machines, additional security checks are employed to ensure the safety of the executable.

Objectives

Objectives ensure that extensive scope coverage occurs.

Coverage Objectives
Injections
Broken Authentication and Session Management
Sensitive Data Exposure
Security Misconfiguration
Insecure Communication
Poor Code Quality

Guidebooks

Guidebooks provide non-restrictive test cases for testers to review per technology and scope.

Injections
Task
JavaScript, CSS Injection
User Host Compromise

Broken Authentication and Session Management

Task
Unhandled Session Permission Requests from Remote Content

Sensitive Data Exposure
Task
Sensitive Information Extraction

Security Misconfiguration
Task
Allowed Code Execution from Untrusted Content
Unsandboxed Process Execution
Improper Use of Preload Scripts
Chromium Web Security Override
Chromium Experimental Features
Allowed Navigation to Untrusted Origins
Clickjacking via Popups

Insecure Communication
Task
Missing HTTP, Mixed Content, TLS Validation

Poor Code Quality
Task
Unsafe Custom Protocol Handlers

Reporting

Philosophy

Customers must be equipped with deliverables that are relevant to their stakeholders.

Outcome

Empower customers with thorough documentation for internal and external customers that demonstrates relevant information about the engagement activities and the unbiased opinion of HackerOne.

All pentests come with a Letter of Attestation (LoA). This document asserts that a pentest was conducted on the customer's in-scope assets.

Report sections include:

- Executive Summary and Exec Analysis
- Most Severe/Prevalent Vulnerabilities
- Vulnerability breakdown (by severity, by asset, by type)
- Vulnerability reports, with remediation guidance
- Methodology and Scope