HackerOne Pentest Methodology

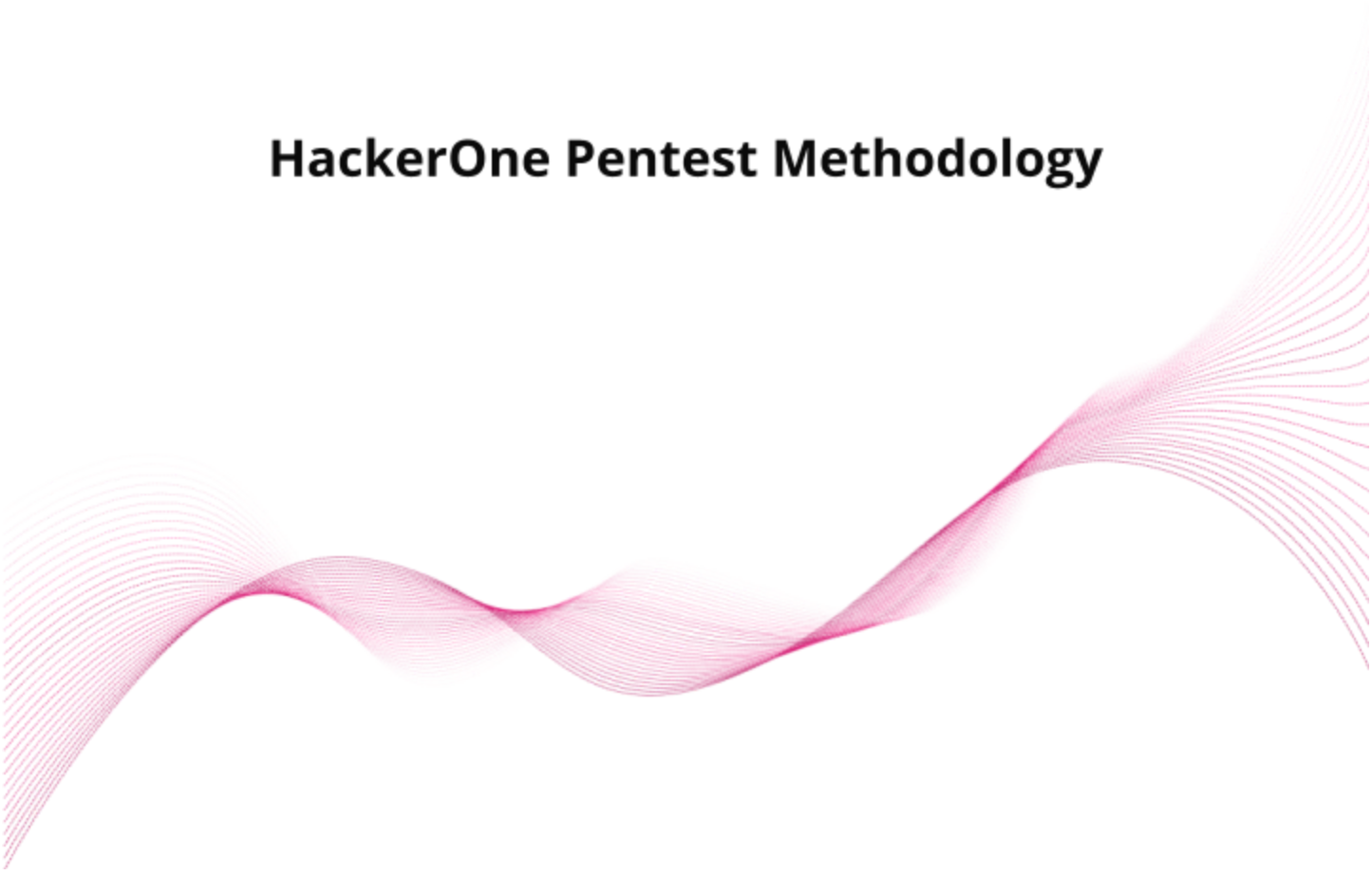

Table of Contents

Executive Summary **4**
Scoping **5**
Phases **6**
Project Alignment 6
Attack Surface Discovery 6
Attack Surface Analysis 6
Pentester Testing 7
Reporting 7
Retesting 7
Project Alignment **8**
Philosophy 8
Outcome 8
Methodologies 8
Attack Surface Discovery **9**
Philosophy 9
Outcome 9
Methodologies 9
Attack Surface Analysis **12**
Philosophy 12
Outcome 12
Methodologies 12
Pentester Testing **14**
Philosophy 14
Outcome 14
Web 15
Objectives 15
Guidebooks 16
Android 21
Objectives 21
Guidebooks 22
iOS 26
Objectives 26
Guidebooks 27
API 31
Objectives 31
Guidebooks 31

External Network 35
Objectives 35
Guidebooks 35
Internal Network 38
Objectives 38
Guidebooks 38
Application Pentest for AWS 41
Objectives 41
Reporting **42**

Executive Summary

The purpose of this document is to showcase the methodologies that HackerOne pentesters follow throughout the pentest engagement to ensure wide and deep coverage. The document demonstrates how HackerOne assesses the effort required to perform thorough security engagements against customer technologies. HackerOne has 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 both 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 are able to 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 own creativity.

Project Alignment

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

Attack Surface Discovery

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

Attack Surface Analysis

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

Pentester 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.

Reporting

Findings are reported in real-time through the platform, and customers can send these findings 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 provide assurance of penetration testing to auditors.

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.

Project Alignment

Philosophy

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

Outcome

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 towards testing.

Methodologies

HackerOne uses an adaptive approach that meets customers' needs while maintaining a standard of excellence, leveraging 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 / 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

Pentesters must first understand and verify the entire attack surface before any active attacks.

Outcome

Pentesters are aware of the entire attack surface, including the perimeter of the attack surface, 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

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
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

Prior to engaging an attack vector, pentesters should first 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

Pentester 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. The testers are empowered with guidebooks for each objective to ensure the depth of their testing but do not confine their testing to prescriptive requirements.

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 single web application is not a wildcard domain but rather a single fully-qualified domain name [FQDN]. Testing can be conducted internally or externally.

Objectives

Objectives ensure that extensive coverage of scope occurs.

Coverage Objectives
Injections
Broken Authentication
Sensitive Data Exposure
XML External Entities (XXE)
Broken Access Control
Security Misconfiguration
Cross-Site Scripting (XSS)
Insecure Deserialization
Using Components with Known Vulnerabilities
Server-Side Request Forgery
Cross-Site Request Forgery
Open Redirect

Guidebooks

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

Injections
Task
Code Injection
Command Injection
Buffer Overflow
HTTP Splitting/Smuggling
SQL Injection
ORM Injection
IMAP/SMTP Injection
HTTP Parameter Pollution
Server Side Template Injection (SSTI)
Unrestricted File Upload

Broken Authentication
Task
Default Credentials
Bypassing Authentication Schema
Privilege escalation
Account Enumeration and Guessable User Account
Weak password policy
Session Fixation
Session Puzzling
Weaker authentication in alternative channel
Weak security question/answer
Weak password change or reset functionalities
Cookies attributes
Vulnerable Remember Password
Logout Functionality
Session Timeout
Weak lock out mechanism

Sensitive Data Exposure
Task
Sensitive information sent via unencrypted channels
Credentials Transported over an Encrypted Channel
Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection
Padding Oracle
HTTP Strict Transport Security

XML External Entities (XXE)
Task
XML External Entities (XXE)

Broken Access Control
Task
Directory traversal/file include
Insecure Direct Object References
Business Logic Data Validation and Bypass

Security Misconfiguration
Task
Information Leakage using Search Engine Discovery
Information Leakage via Webpage Comments and Metadata
Network/Infrastructure Configuration
Application Platform Configuration
File Extensions Handling for Sensitive Information
Old, Backup and Unreferenced Files for Sensitive Information
Information Leakage via Webserver Metafiles
Stack Traces
Clickjacking
HTTP Methods
RIA cross-domain policy
Error Codes

Cross-Site Scripting (XSS)
Task
Stored Cross-Site scripting
Blind XSS
Reflected Cross-Site scripting
Cross-Site flashing
DOM-based Cross-Site scripting
HTML Injection

Insecure Deserialization
Task
Insecure Deserialization

Using Components with Known Vulnerabilities
Task
Enumerating Applications on Webserver
Mapping out Application Architecture
Known Vulnerabilities on Dependencies and Services

Server Side Request Forgery
Task
Server-Side Request Forgery

Cross-Site Request Forgery
Task
Cross-Site Request Forgery

Unvalidated Redirects and Forwards
Task
Open Redirect

Android

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

Objectives

Objectives ensure that extensive coverage of scope 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 that is designed to operate on the iOS operating system. Testing could involve a development environment or a production phone/device.

Objectives

Objectives ensure that extensive coverage of scope 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

In a general sense, 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 coverage of scope occurs.

Coverage Objectives
Injection
Broken Object Level Authorization
Broken User Authentication
Excessive Data Exposure
Lack of Resources & Rate Limiting
Broken Function Level Authorization
Mass Assignment
Security Misconfiguration
Improper Assets Management

Guidebooks

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

Injections
Task
Client-Supplied Data is not Directly Used or Concatenated to SQL/NoSQL/LDAP queries, OS commands, XML parsers, and Object Relational Mapping (ORM)/Object Document Mapper (ODM).
Data Coming from External systems (e.g., integrated systems) is Validated, Filtered, or Sanitized by the API.
Client-Supplied Data is Validated, Filtered, or Sanitized

Broken Object Level Authorization
Task
Insecure Direct Object References

Broken User Authentication
Task
Credential Stuffing
Sensitive information passed in URL
Lack of Validation of the Authenticity of Tokens
Usage of Plain Text, Non-Encrypted, or Weakly Hashed Passwords
Unsigned/Weakly signed JWT tokens
Weak Encryption Keys
Weak Passwords

Excessive Data Exposure
Task
Sensitive Data Returned by API
Reliance on the Client-Side to Filter Sensitive Data

Lack of Resources & Rate Limiting
Task
Lack of Throttling Implemented when Retrieving Resources from the Server and Executing asks on the Server

Broken Function Level Authorization
Task
Unrestricted Access to Administrative Endpoints
Sensitive Action by Changing HTTP Method
Access to Resources or Unauthorized Information

Mass Assignment
Task
Client-Side Variables Impact on Sensitive Permission-Related Properties
Client-Side Variables Impact on Sensitive Process-Dependent Properties
Client-Side Variables Impact on Sensitive Internal Properties

Security Misconfiguration
Task
Missing Appropriate Security Hardening Across Application Stack
Missing Security Patches
Enabled Unnecessary Features
Missing or Weak TLS Configuration
Lack of Security Headers
Sensitive Information Exposed Through Stack Traces

Improper Assets 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.

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 in a cloud environment.

Objectives

Objectives ensure that extensive coverage of scope occurs.

Coverage Objectives
Network Discovery
Vulnerability Assessment
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

In a general sense, 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 coverage of scope 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

Application Pentest for AWS

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 like 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 coverage of scope 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

Reporting

Philosophy

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

Outcome

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