HYPERLEDGER

Linux Foundation

2021 Hyperledger Cactus Web Application Penetration Test

Tevora Threat Research Group

Delivered May 6, 2022

Table of Contents

Executive Summary

Purpose

The 2021 Hyperledger Cactus Web Application Penetration Test for Linux Foundation was conducted from April 04, 2022 to April 19, 2022 to help ensure Linux Foundation resources are secure from advanced threat actors.

Additional objectives for this penetration test were based on industry standard guidelines as follows:

- Identification of vulnerabilities so that they can be remediated prior to being exploited by an attacker
- Direct observation of restricted services or data in the absence of expected access controls
- Compromise of an intermediary device used by privileged users to access secure network zones
- Compromise of the domain used by privileged users
- Sensitive data leakage or exfiltration
- Verification of application logic, session handling, and API security for applications using supplied credentials
- Verification that only authorized services are exposed to the network perimeter
- Verification of network segmentation of non-privileged and privileged networks

Scope

This report contains the summary of project scope, findings, and recommendations resulting from the Web Application Penetration Test conducted by Tevora against the Linux Foundation environment.

Web Application Penetration Test

The following items were considered in scope:

- Hyperledger Cactus version 1.0.0

Findings Overview

Web Application Penetration Test Results

Tevora discovered that Hyperledger Cactus implements a framework for implementing cross-ledger functionality implemented primarily within the Node.js environment. A plugin-based system is used to provide generic interactions with multiple ledger systems. The security architecture of this type of environment appears to depend on two separate categories: The security and integrity of the Cactus code itself, and that of the supporting build and development environment.

The Hyperledger Cactus code appears to be developed to a high standard and continues many functional test cases for validating intended functionality. Implementation of project goals, such as asset transfer and atomic operations, look to be designed with security and reliability in mind. Tevora did not discover any new issues affecting the Cactus codebase, and notes that while reviewing past issues reported, these seem to primarily surround the build and development environment.

The Cactus build and development environment relies heavily on third-party libraries and build integration provided by Node.js. Tevora notes that historically, most security issues discussed appear to be related to the build environment or library dependencies. Tevora observed that Cactus has recently been changed to use specific package versions available in NPM rather than the latest or a regex-based version. This gives the project overall defense against direct supply chain attacks, as changes to the referenced libraries will not be automatically included into the codebase. As a downside, this requires that all security vulnerabilities discovered in attached libraries manually be evaluated and reacted to, as observed on the project's GitHub activity.

Cactus references both local copies (within the source directory) and remote copies (from Hyperledger on NPM) of libraries used within the environment. The methods in which this is implemented appear to be resistant to issues such as dependency confusion attacks. Similarly, this seems to apply to Docker and other tools that have package management capability used within the environment.

Cactus appears to implement strong authentication and authorization using modern JWT libraries and a freshly generated private key. Tevora is not currently aware of any method to attack this authorization scheme as implemented. Tevora notes that Cactus provides flexibility in areas such as secret storage, allowing the end-user to configure their environment as desired for their security purposes.

Tevora noted that the Cactus examples create the container with the "–privileged" flag. This configuration is used to establish a nested docker configuration for the purposes of keeping the example self-contained within a single top-level docker container. Tevora notes that, while the privileged flag enables the top-level container to perform actions on the host, there does not appear to be any current misuse of this, and that a solution to fully nest docker without the privileged flag may not be trivial to accomplish. As such, Tevora believes the current state may be the best configuration.

Tevora notes that the Cactus environment uses all-in-one docker packages for various ledgers to test interactions and integration. These appear to be configured with hardcoded keys and information, which Tevora has noted as a low risk finding. Overall, Tevora expects that it is unlikely, but there is a small chance that these example instances may be re-used without realization that the root key and state is known.

Overall, Tevora did not discover any directly exploitable vulnerabilities within Cactus itself or the use cases as described in documentation. Tevora expects that overall, the Cactus environment will maintain a reasonable level of security against a potential attacker, as it looks to be responsive to security issues and designed from a code perspective with security in mind. As with other applications within the Node.js ecosystem, the security of Cactus ultimately relies on the NPM build environment and the third-party libraries used by the application. As such, Tevora recommends that in production environments, additional hardening is applied and has corresponding strategic recommendations included. These hardening actions will reduce the potential that a third-party compromise can influence the Cactus project or its users, something that is not directly under the control of the Cactus project itself, outside of methods already applied such as specifying specific package version numbers.

Testing Summary

Tevora used a varied approach to testing Cactus, including the following:

- Execution of the provided examples and tests, including "Supply Chain, Discounted Car, and Electricity Trade"
    - Performing port scanning, service enumeration, and vulnerability scanning of created containers with the objective of
        - Determining if examples or development process expose services that may be abused by an attacker to gain access to a developer's environment
        - Obtaining familiarity with Cactus operation, functionality, and architecture
    - Performed active interaction and fuzzing of communications
        - Performed observation of how Cactus components react in practice to unexpected and design situations. Performed follow-throughs of flows of interest in combination with Cactus source code to better understand architecture and security context.
- Performed evaluation of Cactus dependencies (from both the top level and sub-component projects).
    - Comparing dependencies with known vulnerabilities or issues
        - Assessed applicability to Cactus, either in production or development environments. Noted no current and unresolved issues which would affect Cactus's security.
- Review of Cactus source code and methodologies
    - Looked for vulnerabilities, design issues, or areas of interest for other types of testing

Strategic Recommendations

- Ensure documentation recommends hardening steps surrounding ecosystem security, including:
    - Blocking outbound access to the public internet outside of required channels
    - Ensuring that only expected API connections can be made within the local network, either inbound or outbound from Cactus, such as through implementing explicit access controls to required APIs. Implementing these methods reduces the likelihood that any malicious package not designed to specifically target Cactus would be able to establish communications channels for performing attacker-controlled actions
    - Ensuring that any build or deployment process is run inside Docker or otherwise isolated from the host system. Choose build environments that do not preserve any persistent data between deployment runs. This reduces the likelihood that the integrity of future builds can be affected if a malicious package is installed at some point in time
- Continue to ensure that library changes are regularly reviewed against the inclusion of malicious code. Continue to ensure that node dependencies are regularly reviewed, are actively maintained, and have a history of good response and transparency with security issues. Continue to ensure that impactful vulnerabilities announced in third-party libraries are addressed through a library update or an appropriate specific remediation

Summary of Findings

Total Penetration Test Findings 1

Web Application Findings	Status	HydraRisk	
WEB-00953 Harcoded Keys	Discovered	**9**	**Low**

WEB-00953 Hardcoded Keys

Description

Tevora discovered that Cactus contains a variety of tools, including docker all-in-one packages for various Ledgers. These are used in the development and testing process to provide targets for interaction with various ledger systems from the development environment. Many of these appear to use hard-coded keys for initialization.

Tevora notes that while the average user of Cactus may understand that these are strictly to provide examples for development, that there isn't any specific warning against using these in a production system. If someone were to unwittingly use one of these all-in-one containers in a production system without realizing the consequences or otherwise changing the keys, they would be opening themselves to arbitrary actions against the target ledger.

Status	Discovered	**CVSS Base Score**	_	-	**HydraRisk**	9	**Low**
					Consequence	3	
					Probability	1	
					Velocity	1	
					Criticality	2	
					Responsiveness	2	

Affected Files

Tevora specifically noted the following instances of hardcoded encryption keys, credentials, or ledger state:

Path
/tools/docker/besu-all-in-one/nodeKey.pub
/tools/docker/geth-testnet/data-geth1
/tools/docker/iroha-all-in-one/genesis.block
/tools/docker/iroha-testnet/example
/tools/docker/quorum-all-in-one/key

Recommendations

Tevora recommends, in general, that development and testing flows use freshly generated cryptography keys and authentication credentials. This may have the added benefit of providing additional reliability testing, by providing dynamic rather than static input to the development and testing process. At a minimum, ensure that there is a sufficient warning either when using or in the corresponding documentation against re-use in a production environment.

References

- CWE-321: Use of Hard-coded Cryptographic Key
  - https://cwe.mitre.org/data/definitions/321.html

Appendix A: About Tevora

Tevora is a leading management consulting firm specializing in enterprise risk, compliance, information security solutions, and threat research. We offer a comprehensive portfolio of information security solutions and services to clients in virtually all industries and serve institutional and government clients.

Tevora's leaders are professionals with years of experience and records of accomplishments in technology as well as business. This dual background means that we understand the importance of growth and profitability and our solutions are designed to enhance both.

As a consulting firm that can fully implement whatever it recommends, Tevora works with all the industry's top vendors, yet is beholden to none. Our work and dedication have established us as a reliable partner CTOs CIOs, and CISOs can depend on to help protect against threats, both internal and external. With Tevora as a partner, business leaders can devote their energies to enhancing the overall value of information technology to their enterprise.

Tevora is a Qualified Security Assessor (QSA) and Payment Application Qualified Security Assessor (PA-QSA) in good standing with the PCI Security Standards Council. Tevora is also a DVBE (Disabled Veteran Business Enterprise) certified by the California General Services Department (Cert REF# 32786). For more information, please visit www.tevora.com.

Report Content

This report has been compiled for the exclusive use of Linux Foundation. Care has been taken to ensure that all report content and recommendations are of the highest quality and are based on sound analysis, research, and experience. Please direct any questions or concerns about the content of this report to Clayton Riness at criness@tevora.com.

Clayton Riness, Managing Director

Appendix B: Scoring of Findings

Penetration Test findings are qualified using the CVSS Version 3.1 Base Score and the Tevora proprietary HydraRisk model.

CVSSv3.1 Scoring

The CVSS version 3.1 vulnerability scoring system produces a base vulnerability score based on an Impact, and Exploitability metrics. This score is recorded for all applicable findings and is intended to provide an objective, industry-standard view of the vulnerabilities that have been found and potentially exploited.

Scoring guidelines:

- The CVSS version 3 Temporal and Environmental score metrics are not used in this report. Those factors are captured in the HydraRisk scoring model.
- In cases when multiple vulnerabilities with differing CVSS scores are summarized into a single finding, the highest contributing CVSS score is used for that finding.
- Some findings may not be given a CVSS since there is no known vulnerability but where an issue was found with the in-scope environment which differs from industry best practices or which may be used in combination with other findings to exploit a system.

HydraRisk Scoring

Enterprise risk management is an enterprise approach to addressing the culture, processes and structures that are directed towards effective management of potential opportunities and adverse effects as they relate to risk. Taking control of informed risks allows for risks to be identified, analyzed, evaluated, treated, and monitored.

Tevora's proprietary HydraRisk Model is founded on extensive experience in enterprise risk management which has been adapted for the scoring penetration testing results. The HydraRisk score is the sum of the score for all five factors defined as follows.

Consequence

The information security impact a threat and/or exploit has on the organization.

1	Trivial: Non-vital information disclosure: email addresses, WHOIS info, etc.
2	Reasonable: Disclosure of non-public but non-vital information
3	Significant: Non-privileged system access
4	Intolerable: Privileged system access through exploit, pivoting, or escalation
5	Major: Exfiltration of data: PCI, PII, intellectual property, etc.

Probability

The likelihood of the vulnerability/threat to be exploited.

1	Low: No known exploit, requires skilled attacker creating a new 0-day
2	Unlikely: Exploit only possible using specialized tools
3	Moderate: Exploit is possible using common attacks or attack chaining
4	High: Easy to exploit by low skilled penetration tester using common tools
5	Critical: Easy to exploit with simple tools that are readily available

Velocity

Assessment of how quickly a vulnerability could be exploited.

1	Protracted: Requires brute forcing crypto, application fuzzing, etc. over extended period
2	Slow: Requires extensive rainbow tables or other reference libraries to exploit
3	Moderate: Requires readily available reference libraries or casual observation to exploit
4	Quick: Requires casual observation to discover exploit
5	Immediate: Vulnerability can be discovered and exploited readily

Criticality

The depth and breadth of the impact including the types of systems compromised or affected by exploiting this vulnerability.

1	Trivial: vulnerability affects unimportant systems: ancillary support systems
2	Reasonable: exploitation affects access to DMZ or other highly segmented hosts
3	Significant: exploitation affects access to loosely segmented hosts or client environment
4	Intolerable: exploitation affects substantial portions of the environment and data
5	Major: exploitation affects access to critical data, data integrity, and availability

Responsiveness

The time required to treat and prevent the exploit from occurring.

1	Excellent: vulnerability patch or reconfiguration for exploit is readily available
2	Good: vulnerability patch is in development or a workaround is available
3	Moderate: patching, reconfiguration, and/or infrastructure re-architecting is required
4	Fair: infrastructure modification and/or downtime required to remediate
5	Poor: major infrastructure modification and/or downtime required to remediate

Scoring Key

The following scoring key is used throughout this report, with CVSS scores ranging from 0-10 while HydraRisk scores range from 5-25.

Risk Rating	HydraRisk Score
Critical	21-25
High	16-20
Medium	11-15
Low	5-10

Risk Rating	CVSS Score
High	7.0-10.0
Medium	4.0-6.9
Low	0.0-3.9

All findings are categorized as follows:

Status	Description
Informational	No security risk present
Discovered	Security risk discovered and verified, but not successfully exploited
Exploited	Security risk successfully exploited with proof of concept attack

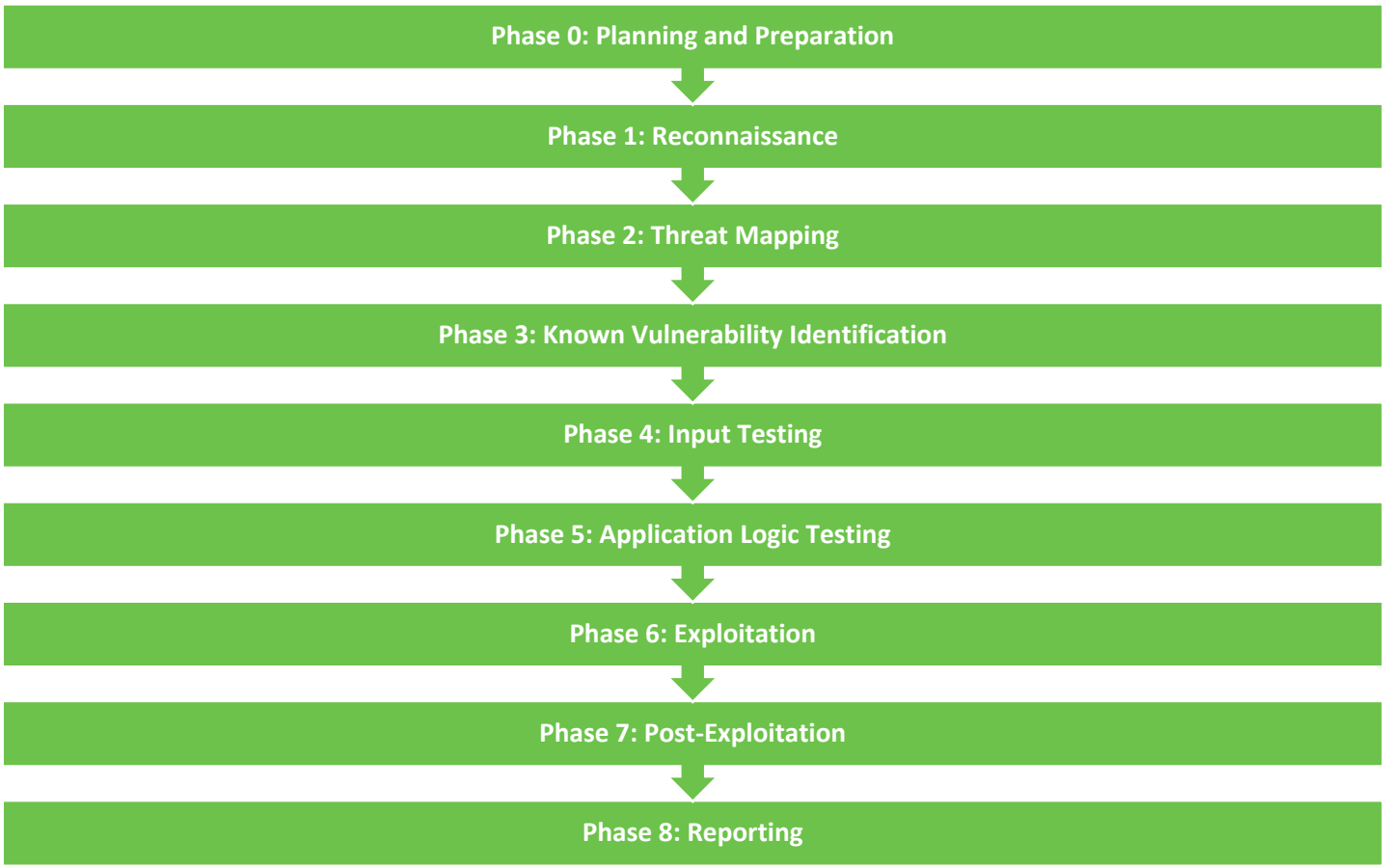
Penetration Testing Tools

Tevora employs many tools during penetration test to assist and complement manual testing including:

- Nessus Professional
- BurpSuite Pro
- ZAP (Zed Attack Proxy)
- SQLmap
- Acunetix
- NetSparker
- Custom Python scripts
- DirBuster

Appendix C: Penetration Testing Methodology

Tevora employs a standard methodology to ensure a repeatable level of quality in all assessments. Tevora's testing methodology is based on the Penetration Testing Execution Standard (PTES)[1], OWASP testing guide v4[2], and years of experience in network, web, and application penetration testing.



[1] http://www.pentest-standard.org/index.php/Main_Page
[2] https://www.owasp.org/images/5/52/OWASP_Testing_Guide_v4.pdf

Phase 0: Planning and Preparation

A successful penetration test begins with planning and preparation. During this phase, Tevora works with the Client to identify the scope and any prerequisites to project execution. Tevora performs the following pre-engagement activities to prepare for testing:

- **Scope Identification:** Tevora and the Client identify the in-scope targets to be tested.
- **Testing Window Identification:** The Client provides the range of acceptable testing windows and Tevora decides when the testing will be performed within that range.
- **Objective Identification:** Tevora and the Client discuss and agree on objectives for the test. These will be used to focus testing and ensure relevant results. Specifically, the expected security model of the target is discussed, and high impact compromises of the model are identified as objectives.
- **Gather Relevant Documentation:** Tevora works with the Client to acquire IT and business process documentation. Tevora can also take a zero-knowledge approach and attempt to acquire this information during the reconnaissance phase of the test.
- **Determine Level of Access:** Based on the objectives, Tevora and the Client determine if credentials are to be provided by the Client for testing. For the most thorough testing, Tevora will use low-level privileges.
- **Time Estimation:** Tevora determines the estimated time needed to cover the scope for the decided testing types.
- **Role Identification:** Tevora assigns a project lead, technical lead, and assistant technical lead to the test. Tevora's technical will have web services and web application specialists assigned to the project including at least one subject matter expert (SME) on the in-scope technologies.
- **Kickoff Meeting:** Tevora and the Client review the planned scope, discuss the project overview, and propose scheduling.
- **Testing Contact Identification:** Tevora and the Client identify their respective points of contact and determine testing status update intervals. Tevora provides an escalation list to the Client.
- **Incident Handling:** Tevora and the Client agree to a response plan for unexpected issues during testing.
- **Project Checklist:** Tevora ensures that every item for the project is checked prior to beginning the penetration test.

After preparation has been completed, the project checklist reviewed, and scheduling finalized, Tevora will begin the penetration test on the scheduled date.

Phase 1: Reconnaissance

The first phase of a penetration test is reconnaissance. This phase is conducted to gather information on the target and enumerate potential threat vectors. Tevora performs reconnaissance in a strategic manner that emulates the process of real-world adversaries. This process, called Open Source Intelligence Gathering (**OSINT**), is a multi-level approach that consists of several types of information gathering activities.

OSINT is done in three phases: **Passive**, **Semi-Passive**, and **Active**:

- **Passive:** Tevora searches the internet for information that is posted by the Client or their employees. Tevora reviews third-party databases that could contain archived Client or employee information including Google, Shodan, and social networking platforms. Traffic is never sent to the Client during this phase, making the testing difficult to detect.
- **Semi-Passive:** Tevora gathers information on the target using requests disguised as normal internet traffic, including DNS requests, service probes, and analysis of document metadata. Traffic may be sent to the Client but will be difficult to detect.
- **Active:** Tevora uses ping sweeps, port scans, banner grabbing, vulnerability scans, and forced browsing to actively enumerate the Client's attack surface. This is a more aggressive phase of reconnaissance that generates significant amounts of abnormal traffic. Tevora gathers a significant amount of reliable information on the Client's systems during this phase. This phase is most likely to be detected by the Client.

Phase 2: Threat Mapping

Tevora analyzes the information gathered during the reconnaissance phase to map targets to potential threat vectors. This map is used to enumerate threats to the business and prioritize testing on high-impact targets.

The threat mapping phase closely follows the PTES Standard's threat modeling phase. During threat mapping, Tevora performs the following steps:

- **Gather relevant documentation:** Tevora works with the Client to acquire IT and business process documentation. Tevora can also take a zero-knowledge approach and attempt to acquire this information during the reconnaissance phase.
- **Identify and categorize primary and secondary assets:** Tevora identifies the assets on the in-scope targets and divides them into primary and secondary categories. These are assets that can be reached directly, and assets that can be reached from pivoting, respectively.
- **Identify and categorize threats and threat communities:** Tevora enumerates the potential threats to the in-scope targets and categorizes them by the groups of people (e.g., threat communities) that may execute those threats.
- **Map threat communities against primary and secondary assets:** Tevora maps the categorized threat list to the categorized asset list to determine relevant threats and their potential impact on the business.
- **Cross-reference threat map to test objectives:** Tevora reviews the threat map to identify the impact of potential threats in the context of testing objectives defined during the planning phase.

Tevora uses the output of this phase to enumerate potential threat vectors and prioritize testing on high-impact attack scenarios. This also enables alignment of threat exposure to testing objectives.

Phase 3: Known Vulnerability Identification

Tevora reviews information gathered during the threat mapping and reconnaissance phases to identify known vulnerabilities. Tevora reviews banners, network and HTTP response signatures, and running services. These are then cross-referenced against vulnerability databases such as Exploit-DB, Rapid7, and CVE.

Tevora takes a multi-assessment approach by analyzing information gathered from both passive and active vulnerability identification:

- **Passive:** Tevora reviews metadata from public documents and archived content in search engines for vulnerability signatures. Additionally, Tevora performs traffic monitoring on the internal network and analyzes network protocols for signatures of vulnerable network services.
- **Active:** Tevora uses vulnerability scanners for automated vulnerability enumeration and augments this with output from port scanners, HTTP responses, SNMP enumeration, NetBIOS enumeration, and more.

After identifying vulnerabilities, Tevora attempts to validate vulnerabilities and prioritize them for exploitation. Tevora researches all discovered vulnerabilities and performs manual testing to check for false positives. Vulnerabilities are cross-referenced against the threat map to identify their impact and potential risk to the business.

Phase 4: Input Testing

Tevora tests for input validation and injection issues on web application forms. Tevora fuzzes input fields using a combination of manual and automated techniques.

Tests performed include:

- LDAP Injection
- ORM Injection
- Directory Traversal / File Inclusion
- XML Injection
- SSI Injection
- XPath Injection
- IMAP/SMTP Injection
- Code Injection
- OS Commanding
- Buffer Overflow
- Incubated Input Vulnerability Testing
- HTTP Splitting/Smuggling
- SQL /NoSQL Injection

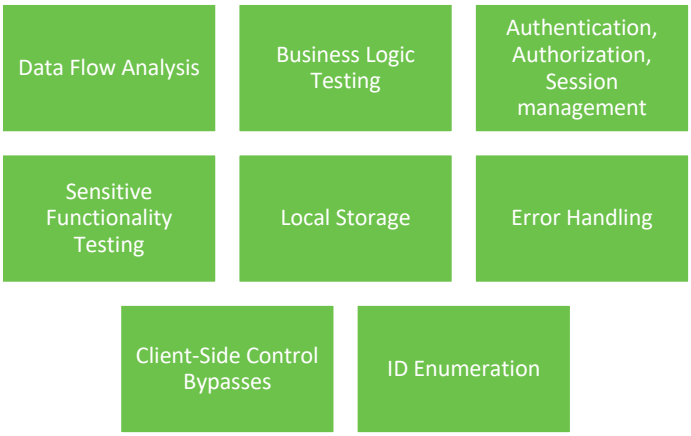
Any discovered input vulnerabilities are categorized, referenced against the threat map, and documented for use in the exploitation phase.

Phase 5: Application Logic Testing

Tevora uses information gathered during previous phases to analyze the web application's security model. Tevora reviews the application logic at all points in the platform to identify application logic weaknesses which may expose sensitive information or functionality.

Application logic tests make up the bulk of time spent on web application penetration tests. Application logic testing is a primarily manual process, with custom scripts and plugins used to automate testing for certain flaws, such as enumeration. Tevora focuses on high-impact functionality during this phase as well as complex multi-step processes, which are more likely to include dangerous bugs.

Testing performed includes:



The impact of any data leakage, or unauthorized activities discovered during application logic testing are categorized and referenced against the threat map. If relevant, identified issues may be used during the platform exploitation phase.

Phase 6: Exploitation

During the exploitation phase, Tevora attempts to access the targets enumerated during the threat mapping phase. Tevora reviews discovered vulnerabilities and potentially insecure services to develop an exploitation plan. Tevora then executes this plan in a precision strike against the Client.

Tevora uses publicly available exploits and pursues development of custom and/or "zero-day" exploits for high impact targets or when known vulnerabilities are not discovered.

- **Known Vulnerabilities:** Tevora modifies public exploits to target the Client environment. Public exploits are only acquired from trusted sources such as Exploit-DB and are reviewed before modification and use. Commercial exploitation frameworks are also used during this phase.
- **Unknown Vulnerabilities:** If known vulnerabilities are not found, Tevora takes a zero-day approach. A replica environment is created and Tevora tests the discovered services for previously unknown security issues.
- **Application Layer Vulnerabilities:** If any custom applications are discovered during testing, Tevora will perform application-level assessments as permitted by the timeframe. These tests will be performed according to Tevora's application testing methodologies.

Tevora delivers payloads during the exploit to gain access to the targets in accordance with testing objectives. Payloads are designed to bypass security measures used by the Client. These will include encoded, packed, encrypted, and custom payloads designed to bypass anti-virus, IPS/IDS systems, and firewalls. These payloads are also used in the post-exploitation phase to pivot the attack to other targets.

Phase 7: Post-Exploitation

During this phase, Tevora evaluates the impact of the exploitation, tests the Client's internal defenses, and uses the initial exploits to escalate access to additional targets. The following activities are performed during this phase:

- **Establish Persistence:** Tevora establishes secure, persistent access so Tevora may notify the Client of the exploit and the Client can remediate without interrupting post-exploitation activities.
- **Initial Enumeration:** Compromised resources are enumerated for relevant information. User accounts and passwords are extracted for use in pivoting.
- **Pivoting:** Tevora repeats the reconnaissance, threat mapping, vulnerability identification, and exploitation phases on newly accessible targets. Tevora begins the new reconnaissance phase with network analysis and shifts to an internal penetration test methodology. Tevora uses information acquired during previous phases to escalate access to the Client's systems.
- **Target Profiling:** Tevora enumerates data and information on exploited targets.
- **Data Exfiltration:** Based on the purpose of the penetration test, Tevora targets and attempts to extract (or simulate an extraction of) information that is vital to the organization.
- **Cleanup:** When the penetration test is complete, Tevora cleans up all the tools and payloads that were placed in the target's environment.

Post-exploitation is an iterative testing process to continually escalate the attack simulation. Previous steps of the methodology are repeated to assess potential threats from the newly acquired foothold. Additional information about the target may be discovered during this phase such as source code, undocumented endpoints, and additional credentials, which all warrant further testing.

Phase 8: Reporting

Tevora compiles the findings during the penetration test and organizes them into a final report which is sent to the Client. The report documents each discovered vulnerability, remediation recommendations, and provides an analysis of risk to the business.

Topics covered by the report include:

- Executive Summary
    - People involved
    - Project objective
    - Project scope
- Findings Overview
    - Test results
    - Strategic recommendations
- Technical Summary
    - Scoring of findings
    - Findings summary based on HydraRisk model
    - Detailed summary of each finding
        - CVSS score
        - HydraRisk score
        - Finding description
        - External references
        - Recommended remediation
- Penetration Testing Methodology

The report provides both a detailed technical breakdown and a high-level executive summary, allowing for review by both technical and non-technical staff. The report can be tailored to a Client's needs, including being split into multiple documents. The report is the final deliverable for testing and may go through review and editing phases prior to acceptance. Once the report has been accepted, the project is considered closed unless otherwise stated.

TEVORA™

Go forward. We've got your back.

Compliance – Enterprise Risk Management – Data Privacy – Security Solutions – Threat Management