HYPERLEDGER

Linux Foundation
2022 Hyperledger Besu Web Application Penetration Test

Tevora Threat Research Group

Delivered August 17, 2022

CONFIDENTIAL

Table of Contents

Executive Summary

Purpose

The 2022 Hyperledger Besu Web Application Penetration Test for the Linux Foundation was conducted from June 24, 2022, to July 13, 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

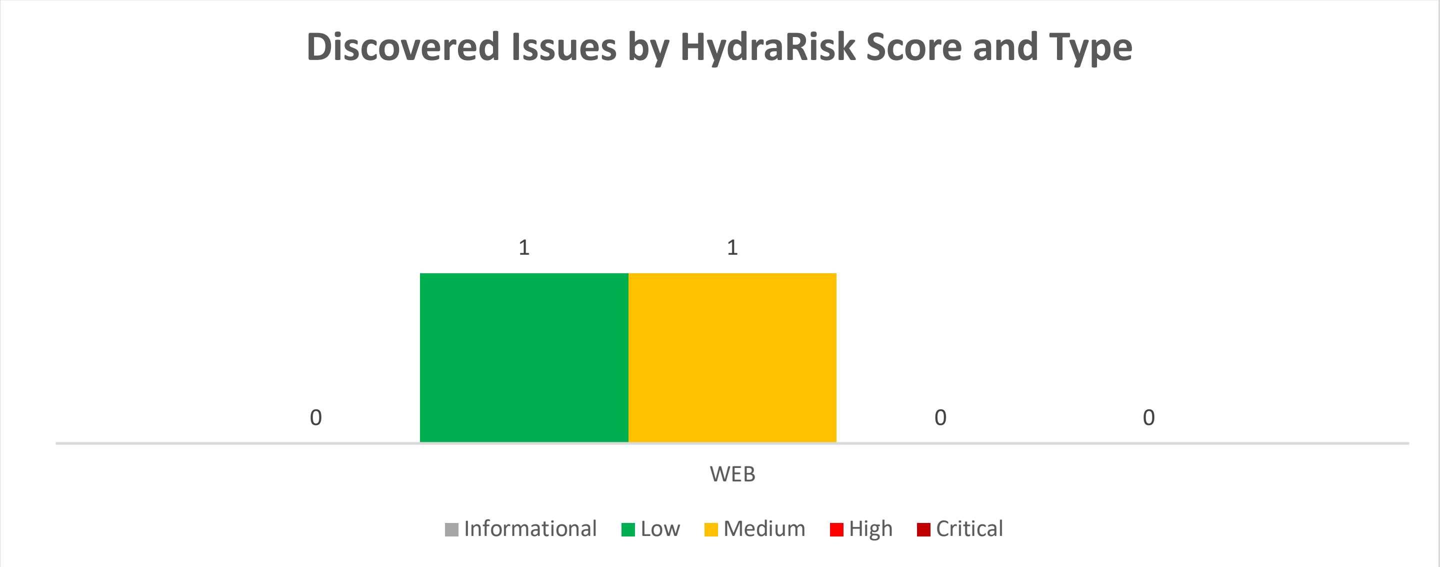
This report contains the summary of project scope, findings, and recommendations resulting from the Web Application Penetration Test conducted by Tevora against the Linux Foundation environment.

Web Application Penetration Test

The following items were considered in scope:

- Hyperledger Besu: https://github.com/hyperledger/besu

Findings Overview



Network Penetration Test Results

Tevora analyzed Hyperledger Besu (Besu) from an information security perspective to determine if the project presents a risk to end-users or developers.

The Besu project implements an open-source Ethereum client written in Java using a Gradle build environment. This build environment pulls a variety of third-party dependencies required for the functionality and implementation of Besu. This build process runs natively on the initiating machine, with no apparent protections against a malicious dependency performing direct actions at build time. However, the dependencies used by Besu seem to be well trusted, and do not appear to contain any currently known issues that would be exposed by Besu. Additionally, Tevora did not observe any apparently malicious activity because of the current build process. Because of the overall number of dependencies pulled in, not all of which are under Hyperledger's control, Hyperledger may wish to consider updating the documentation to recommend building within a docker container, or another method that avoids the direct risk of a malicious dependency executing code on a user's system when they compile Besu.

Besu implements interfaces required for interaction with a chosen Ethereum network, such as P2P communications. Interactions with Besu from the Ethereum side are subject to extensive validation within Besu's codebase and dependencies, and appear to be unlikely to result in a potentially malicious effect outside of Denial of Service. If a Denial-of-Service condition occurred, resulting in Besu failing to process new transactions, this is unlikely to have any overall security or privacy impact, and as Besu does not form the majority of peers on common networks, it is unlikely for the network as a whole to be affected.

To support "The Merge", where Ethereum is transitioning to proof-of-stake over proof-of-work, Besu has implemented new execution engine endpoints to provide the interface for a consensus client. These are accessible through a new execution API and web socket API, separate from existing Besu interfaces. These interfaces are designed to support a single consensus client and are protected by JWT authentication. These interfaces do not currently appear to be encrypted. While it may be common for the execution and consensus client on the same system, and the API currently defaults to only allowing localhost, it may become a potential issue if a user configures their setup so that execution and consensus are running on separate systems linked by a potentially insecure network. An attacker in an eavesdropping position would be able to capture the JWT authentication token, and then perform actions to interfere with or discredit the user’s node.

Outside of the interfaces required for Ethereum, Besu implements a variety of monitoring and administration APIs, such as GraphQL and a JSON RPC interface. These interfaces are disabled by default and present a variety of security options such as the ability to set up authentication, client allow-listing, and CORS configurations. This appears to be the best option from a “secure by default” perspective and provides users with the ability to implement security as needed for these interfaces, using mechanisms that appear to function as intended.

Tevora notes that Besu does not support user key management, and suggests users use software such as Eth-Signer for signing transactions. Therefore, the safety of a user’s private key is unaffected by Besu itself. Besu does store the node key locally, but Ethereum is designed to be resilient against node compromise, and better options for storage of the node key offer no practical benefit.

Overall, Tevora believes that the Besu project is in average condition compared to similar projects within the Java and crypto ecosystem. This is likely an acceptable condition for this type of project, but Hyperledger may wish to consider the strategic recommendations and findings within the report and address them on an as-needed basis.

Attack Surface

Exposed services were subjected to fuzzing, manual interaction and tampering, and analysis of the respective code base. Other activities were performed where applicable to build a picture of the project’s attack surface and risk areas. Tevora observed the following interfaces exposed, or available to be exposed based on application configuration:

Service	Description	Security
Peer to Peer(P2P)	Provides functionality for interfacing with the Ethereum network	Consists of Ethereum specific protocols. Node communications are not necessarily designed to be private by nature. The node communication surface is designed to be resilient against invalid or malicious actions by the nature of its design. If a node is overwhelmed such as through a denial-of-service attack, the network is designed to be resilient to the loss of nodes. If direct authorization and encryption of node communication is desired, TLS can be enabled through configuration as described in Besu’s documentation. This appears to use default options, as opposed to providing selectable ciphers and protocol versions such as the engine API. Besu may wish to expand these options, but in general this feature is marked as “early access” and has limited use cases.
GraphQL	Provides monitoring of Besu to the end user or integrations	Disabled by default. Defaults to only 127.0.0.1/localhost allowed. This interface only supports host-based authorization, and does not support encryption, but information served over this interface is not directly confidential. GraphQL may have denial-of-service concerns if left open an unauthenticated, even if the data served is not confidential. Besu may wish to offer industry standard options such as JWT authentication and TLS v1.2+ as implemented elsewhere within Besu. These provide protection for use cases where GraphQL is used directly over potentially insecure network links rather than from the local host or indirectly such as through a web proxy. While this situation may not be common, it may be desirable to update documentation to clarify this scenario.
Engine (JSON-RPC or Websockets)	Provides support for a Consensus client to connect to Besu	Disabled by default. Defaults to only 127.0.0.1/localhost allowed.

		Allows JWT authentication using the HS256 algorithm, which is industry standard. Currently does not support encryption, leaving this connection susceptible to eavesdropping or injection when used over an insecure network link, such as a configuration where the consensus client is running on a different machine. Besu may wish to offer TLS v1.2+ as offered for other services within the project and update documentation accordingly. Injection into this interface (such as by theft of the JWT token via eavesdropping may result in the node being discredited.
API (JSON-RPC or Websockets)	Provides a variety of monitoring, administration, and control endpoints to the end user or integrations, including realtime subscriptions to data	Disabled by default. Defaults to only 127.0.0.1/localhost allowed. Authentication is available through JWT which defaults to RS256 encryption, although this is user-selectable to other algorithms. Encryption is available through TLS which defaults to v1.2 and v1.3. These are currently considered industry standard options for authentication and encryption.
Metrics	Provides Prometheus or OpenTelemetry endpoints for collection of metrics	Disabled by default. Does not appear to support encryption or authentication, with a lack of relevant options provided by the Besu command line. Besu may wish to clarify that these listeners do not support this functionality and suggest alternatives for users which wish to protect metrics data being accessed across a potentially insecure connection.

Strategic Recommendations

- Consider recommending an isolated, containerized, or otherwise protected build environment that prevents non-specific attacks being delivered through a compromised or malicious dependency.
- Maintain an update schedule for patching existing systems. This procedure should include processes to discover, test, and install patches quickly to all systems. In general, ensure that critical security patches are applied within 30 days of release, and other patches within 90 days.
- Ensure dependencies are regularly scanned for announced issues, and that issue applicability is either fully triaged or that a fix or workaround is applied regardless of applicability.
- Consider maintaining an overall guide on securing a Besu implementation, referencing and discussing the available options
- Consider methods for directly securing auxiliary interfaces such as metrics collection, or ensure documentation includes alternatives

Summary of Findings

Total Penetration Test Findings 2

Web Application Findings	**Status**	**HydraRisk**	
WEB-01302 Gradle Repository Priority	Discovered	**12**	**Medium**
WEB-00954 Engine API lacks Encryption	Discovered	**9**	**Low**

Technical Findings

WEB-01302 Gradle Repository Priority

Description

Tevora discovered that Besu's "build.grade" file specifies Maven Central as the first repository, followed by Hyperledger- or Besu-specific repositories. Recent versions of Gradle resolve dependencies based first on repository order. As such, it is possible that a malicious actor may be able to get a dependency submitted with the same name as one used legitimately by the project, such as "org.hyperledger.besu/besu-datatypes". It appears that methods to get projects into the Maven Central repository require manual review, and as such, Tevora did not attempt to create such as project as a proof-of-concept, due to the potentially out-of-scope and disruptive nature. Tevora is ultimately unsure how likely it is for an attacker to conduct this approach, but notes that dependency confusion issues have been a high area of concern lately.

Status	Discovered	**CVSS Base Score**	7.1	**High**	**HydraRisk**	12	**Medium**
					Consequence	4	
					Probability	2	
					Velocity	3	
					Criticality	2	
					Responsiveness	1	

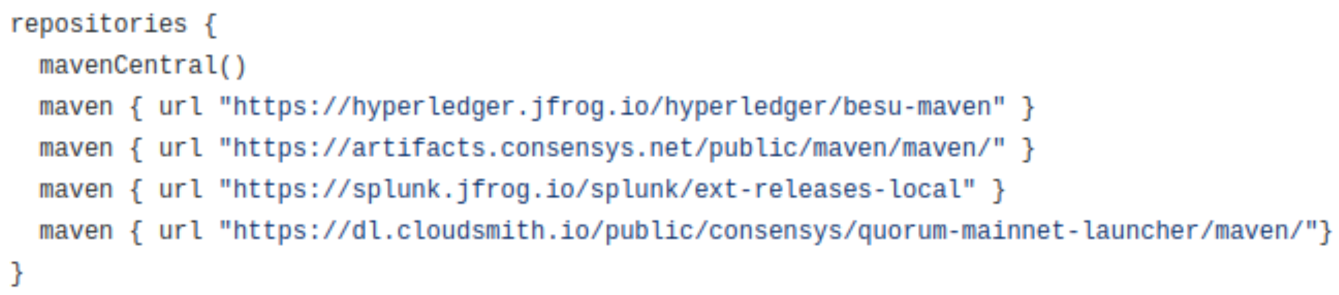
Affected Systems

The following system has been identified and is affected:

Affected file
besu/build.gradle

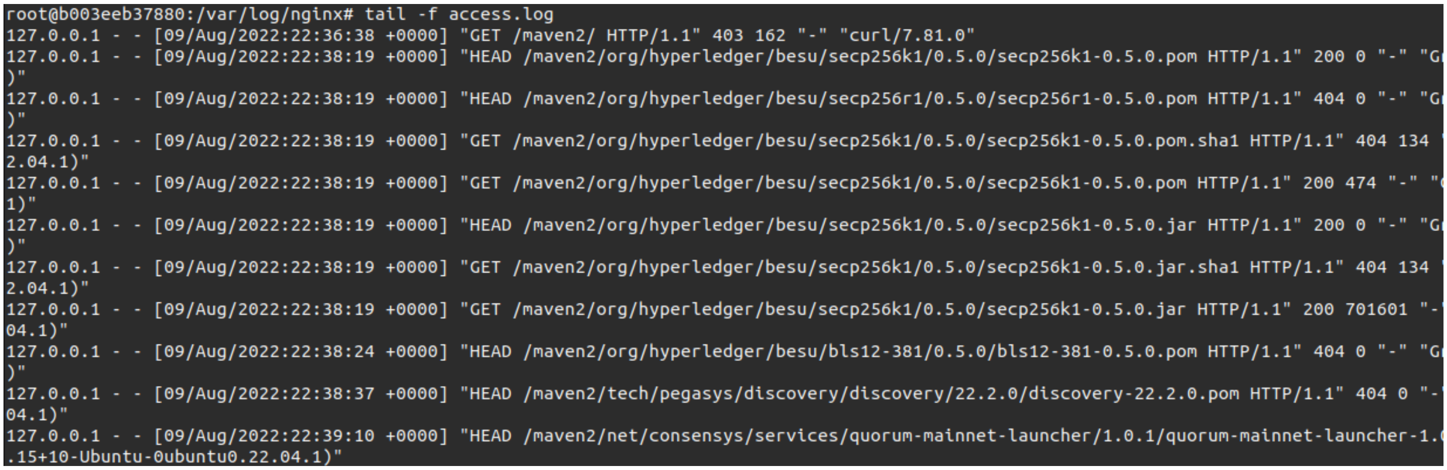
Details

Current Repository Order



```
repositories {
  mavenCentral()
  maven { url "https://hyperledger.jfrog.io/hyperledger/besu-maven" }
  maven { url "https://artifacts.consensys.net/public/maven/maven/" }
  maven { url "https://splunk.jfrog.io/splunk/ext-releases-local" }
  maven { url "https://dl.cloudsmith.io/public/consensys/quorum-mainnet-launcher/maven/"}
}
```

Theoretical scenario showing dependency fetch if besu secp256k1 was shadowed on mavenCentral (using a local repository placed above besu-maven as an example)



```
root@b003eeb37880:/var/log/nginx# tail -f access.log
127.0.0.1 - - [09/Aug/2022:22:36:38 +0000] "GET /maven2/ HTTP/1.1" 403 162 "-" "curl/7.81.0"
127.0.0.1 - - [09/Aug/2022:22:38:19 +0000] "HEAD /maven2/org/hyperledger/besu/secp256k1/0.5.0/secp256k1-0.5.0.pom HTTP/1.1" 200 0 "-" "G
)"
127.0.0.1 - - [09/Aug/2022:22:38:19 +0000] "HEAD /maven2/org/hyperledger/besu/secp256r1/0.5.0/secp256r1-0.5.0.pom HTTP/1.1" 404 0 "-" "G
)"
127.0.0.1 - - [09/Aug/2022:22:38:19 +0000] "GET /maven2/org/hyperledger/besu/secp256k1/0.5.0/secp256k1-0.5.0.pom.sha1 HTTP/1.1" 404 134 "
2.04.1)"
127.0.0.1 - - [09/Aug/2022:22:38:19 +0000] "GET /maven2/org/hyperledger/besu/secp256k1/0.5.0/secp256k1-0.5.0.pom HTTP/1.1" 200 474 "-" "
1)"
127.0.0.1 - - [09/Aug/2022:22:38:19 +0000] "HEAD /maven2/org/hyperledger/besu/secp256k1/0.5.0/secp256k1-0.5.0.jar HTTP/1.1" 200 0 "-" "G
)"
127.0.0.1 - - [09/Aug/2022:22:38:19 +0000] "GET /maven2/org/hyperledger/besu/secp256k1/0.5.0/secp256k1-0.5.0.jar.sha1 HTTP/1.1" 404 134 "
2.04.1)"
127.0.0.1 - - [09/Aug/2022:22:38:19 +0000] "GET /maven2/org/hyperledger/besu/secp256k1/0.5.0/secp256k1-0.5.0.jar HTTP/1.1" 200 701601 "-"
04.1)"
127.0.0.1 - - [09/Aug/2022:22:38:24 +0000] "HEAD /maven2/org/hyperledger/besu/bls12-381/0.5.0/bls12-381-0.5.0.pom HTTP/1.1" 404 0 "-" "G
)"
127.0.0.1 - - [09/Aug/2022:22:38:37 +0000] "HEAD /maven2/tech/pegasys/discovery/discovery/22.2.0/discovery-22.2.0.pom HTTP/1.1" 404 0 "-"
04.1)"
127.0.0.1 - - [09/Aug/2022:22:39:10 +0000] "HEAD /maven2/net/consensys/services/quorum-mainnet-launcher/1.0.1/quorum-mainnet-launcher-1.
.15+10-Ubuntu-0ubuntu0.22.04.1)"
```

Recommendations

Tevora generally recommends configuring package managers to prioritize private repositories over public. This prevents outside entities from shadowing internal libraries within the public repository in an effort to gain code execution on developers, users that are compiling the application, or within application build environments.

References

- Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies
  - https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610
- 3 Ways to Mitigate Risk When Using Private Package Feeds
  - https://azure.microsoft.com/mediahandler/files/resourcefiles/3-ways-to-mitigate-risk-using-private-package-feeds/3%20Ways%20to%20Mitigate%20Risk%20When%20Using%20Private%20Package%20Feeds%20-%20v1.0.pdf
- Gradle Documentation: Understanding dependency resolution
  - https://docs.gradle.org/current/userguide/dependency_resolution.html

WEB-00954 Engine API lacks Encryption but supports Authentication

Description

Tevora discovered that the "Engine API" implemented for consensus client connectivity supports authorization via a JWT token but does not support encryption at the protocol or transport level. While it may be common for users to run the consensus client on the same system, or other scenarios where the network connection is secure, this leaves users open to JWT key theft when using this API over a potentially insecure link such as a LAN or over the Internet. If this link is intercepted with authorization in use but no encryption or transport level validation, it is possible for an attacker to hijack the JWT key, and as a result, interface with this API and discredit the node.

Status	Discovered	**CVSS Base Score**	4.5	**Medium**	**HydraRisk**	9	**Low**
					Consequence	2	
					Probability	1	
					Velocity	1	
					Criticality	2	
					Responsiveness	3	

Affected Systems

The Engine JSON-RPC service (default port 7551) is affected.

Recommendations

Besu should consider implementing the option to carry this communication over TLS as implemented for other APIs within the project. This will add industry standard encryption on top of the existing authorization, making the connection suitable for use over potentially insecure connections.

References

- OWASP Top 10 2021: Cryptographic Failures
  - https://owasp.org/Top10/A02_2021-Cryptographic_Failures/

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

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.

CVSS v3.1 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

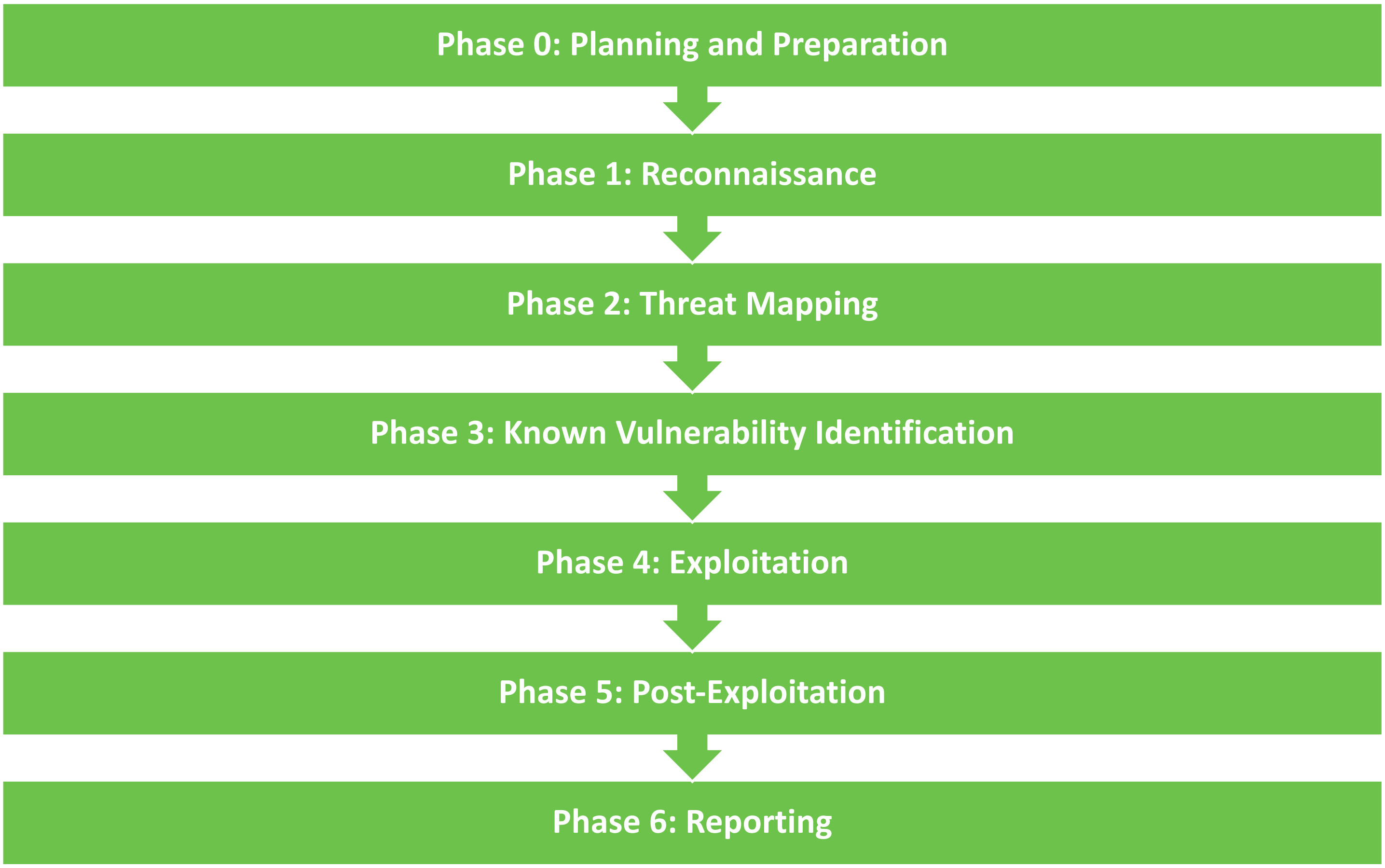
Penetration Testing Tools

Tevora uses 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
- BloodHound
- Cobalt Strike
- Covenant
- GhostPack
- Metasploit
- Responder
- Impacket
- Custom Malware

Appendix C: Penetration Testing Methodology

Tevora uses 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.



Phase 0: Planning and Preparation

Phase 1: Reconnaissance

Phase 2: Threat Mapping

Phase 3: Known Vulnerability Identification

Phase 4: Exploitation

Phase 5: Post-Exploitation

Phase 6: Reporting

[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: 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 5: 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 6: 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

HYDRARISK MODEL