MUFEX
Web Application Penetration Test Report v0.2

Table of Content

Document Control

PREPARATION

Jia Liu
Email: jialiu@numencyber.com
Security Consultant

Document History

Version No.	Date	Author	Description
0.1	29/5/2023	Jia Liu	Initial Draft
0.2	30/5/2023	Jia Liu	Modify CVSS severities

Executive Summary

Numen Cyber Labs was engaged by MUFEX to conduct a Vulnerability Assessment and Penetration Testing for MUFEX Web and Mobile Applications which allows users to sign up.

The purpose of the engagement was to utilize active exploitation techniques in order to evaluate the security of the application against best practice criteria, validate its security mechanisms and identify possible threats and vulnerabilities.

The assessment provides insight into the resilience of the application to withstand attacks from unauthorized users and the potential for valid users to abuse their privileges and access.

Penetration testing was conducted from the perspective of external and internal users in the production environment using a Grey Box approach for Web Application and Black Box approach for Mobile Application. Numen observed that the Web and Mobile Application has a weak security posture against attackers of high skilled level.

This report provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and remedial advice that will resolve the underlying technical issue.

A total of **4** security observations identified fall into the following CVSS severities:

CVSS Severity	Count
Critical	0
High	0
Medium	1
Low	2
Informational	1

Throughout the course of testing, an attacker with appropriate knowledge and access to the environment would be able to withdraw uncontrolled Cryptocurrency.

This could have reputational, legal and ultimately financial consequences for MUFEX in the event that these vulnerabilities were exploited by an attacker. Numen strongly recommends for all critical and high risks to be addressed immediately and all medium and low risks to be addressed in a timely manner.

Moving forward, Numen recommends for MUFEX to review its secure software development life cycle to focus on user's input amount.

Overview of Technical Findings

The following table is a summary listing of notable issues identified by Numen over the course of the Web Application Penetration Test.

Ref#	Description	Risk	CVSS
R.1	Withdrawal amount tampering	Low	CVSS: 8.1 (/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H)
R.2	Information leakage	Low	CVSS: 3.7 (/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N)
R.3	Long cookie expiration	Informational	-

The following table is a summary listing of notable issues identified by Numen over the course of the Mobile application Penetration Test.

Ref	Description	Risk	CVSS
R.1	Sensitive Information Disclosure	Medium	CVSS: 4.8 (/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N)

Web Application Technical Findings

Web Application Penetration Test		Risk	CVSS
R.1	Withdrawal amount tampering	Low	CVSS: 8.1 (AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H)

Description

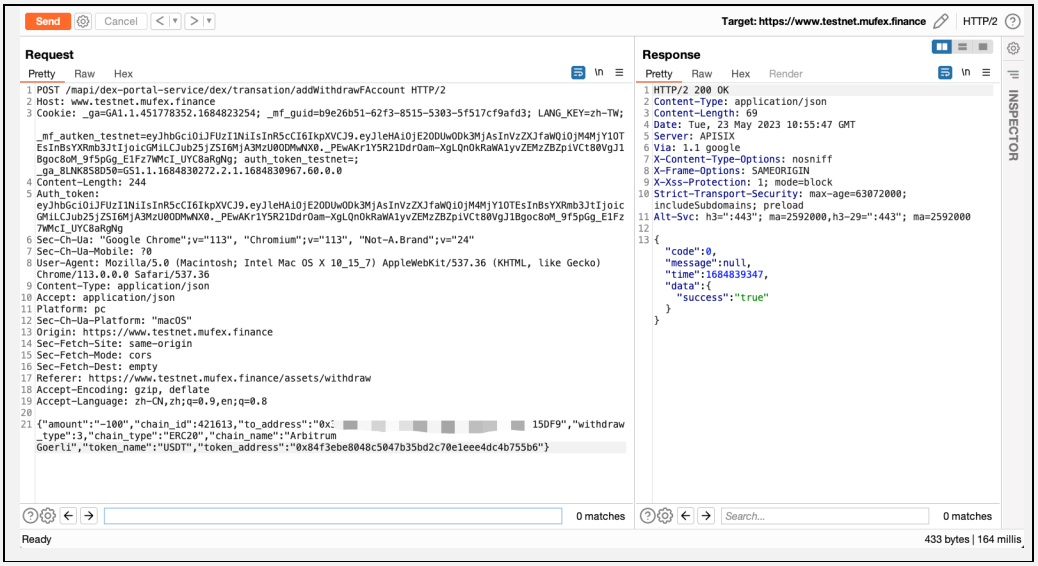
Many applications implement payment functionality, including e-commerce sites, subscriptions, charities, donation sites and currency exchanges. The security of this functionality is critical, as vulnerabilities could allow attackers to steal from the organization, make fraudulent purchases, or even to steal payment card details from other users. These issue could result in not only reputational damage to the organization, but also significant financial losses, both from direct losses and fines from industry regulators.

Impact

An attacker can withdraw unexpected amount.

Proof Of Concept

in the withdrawal function, numen can change the amount to -100 and get success.



Recommendation: Encrypted Transaction Details
In order to prevent the transaction being tampered with, some payment gateways will encrypt the details of the request that is made to them. For example, PayPal does this using public key cryptography.

The first thing to try is making an unencrypted request, as some payment gateways allow insecure transactions unless they have been specifically configured to reject them.

References

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/10-Business_Logic_Testing/10-Test-Payment-Functionality

Affected System(s)

`https://www.testnet.mufex.finance/assets/withdraw`

Web Application Penetration Test		Risk	CVSS
R.2	Information leakage	Low	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

Description

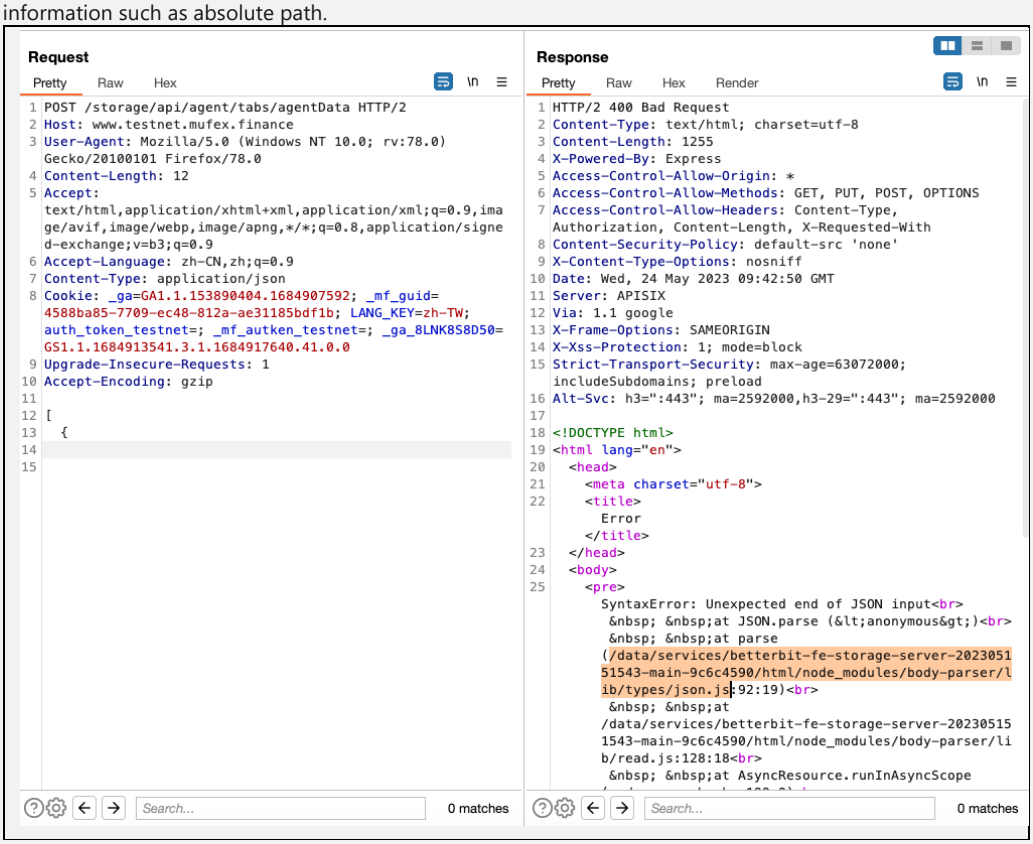
Full Path Disclosure (FPD) vulnerabilities enable the attacker to see the path to the webroot/file. e.g.: /home/omg/htdocs/file/. Certain vulnerabilities, such as using the load_file() (within a SQL Injection) query to view the page source, require the attacker to have the full path to the file they wish to view..

Impact

An attacker can abuse the knowledge and use it in combination with file inclusion vulnerabilities (see PHP File Inclusion) to steal configuration files regarding the web application or the rest of the operating system.

Proof Of Concept

Make a request to https://www.testnet.mufex.finance/storage/api/agent/tabs/agentData, numen can get sensitive information such as absolute path.



Recommendation: Error Handling

Error message handling: When handling error messages, do not return the error information directly to the user. The alternative is to provide a uniform error message, categorized by the type of error, in order to provide useful error information to the user without exposing sensitive information.

Input validation and filtering.

References

https://owasp.org/www-community/attacks/Full_Path_Disclosure

Affected System(s)

`https://www.testnet.mufex.finance/storage/api/agent/tabs/agentData`

Web Application Penetration Test		Risk	CVSS
R.3	Long cookie expiration	Informational	-

Description

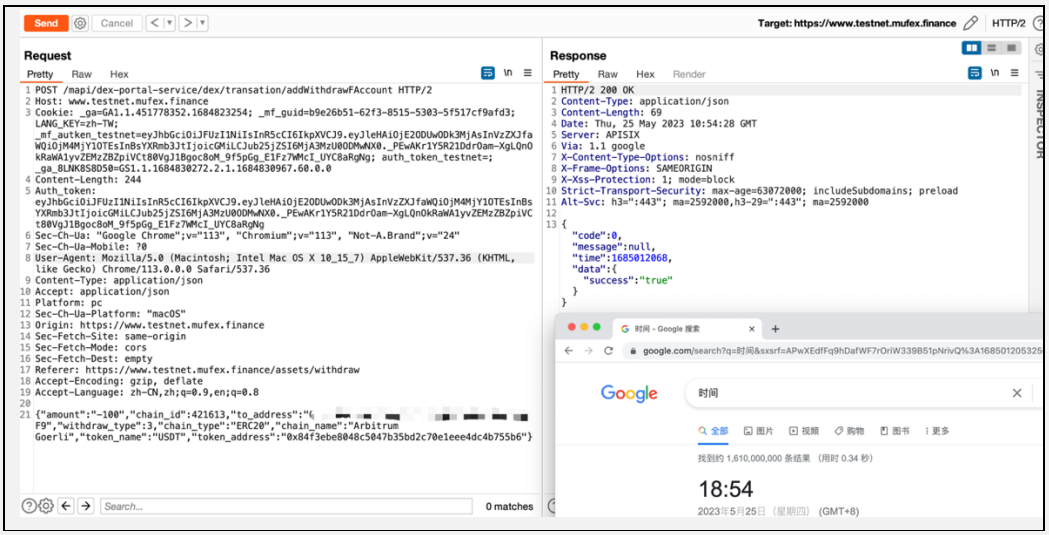
Setting the expiration date of a cookie is an important security consideration. If the cookie expiration date is too long, Setting the expiration date of a cookie is an important security consideration. If the cookie expiration date is too long, it may lead to security risks and privacy issues.

Impact

Cookies that are valid for a long period of time can be a potential target for attackers. If an attacker is able to obtain a user's cookie, they can gain unauthorized access to the user's account and personal information, resulting in data leakage, identity theft, and other security issues.

Proof Of Concept

Get a valid cookie, but the cookie has not expired after 4 days.



Recommendation: Set the expiration date appropriately

Set the appropriate expiration date for cookies based on specific needs and security requirements. The validity period of a cookie should not be set too long, especially for cookies containing sensitive information such as authentication credentials, session tokens, etc. Usually, set them as session cookies, i.e., they expire only when the user closes the browser, to limit the lifetime of the cookie.

References

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes

Affected System(s)

`https://www.testnet.mufex.finance/*`

Mobile Application Technical Findings

Web Application Penetration Test		Risk	CVSS
R.1	Sensitive Information Disclosure	Medium	CVSS: 4.8 /AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N

Description

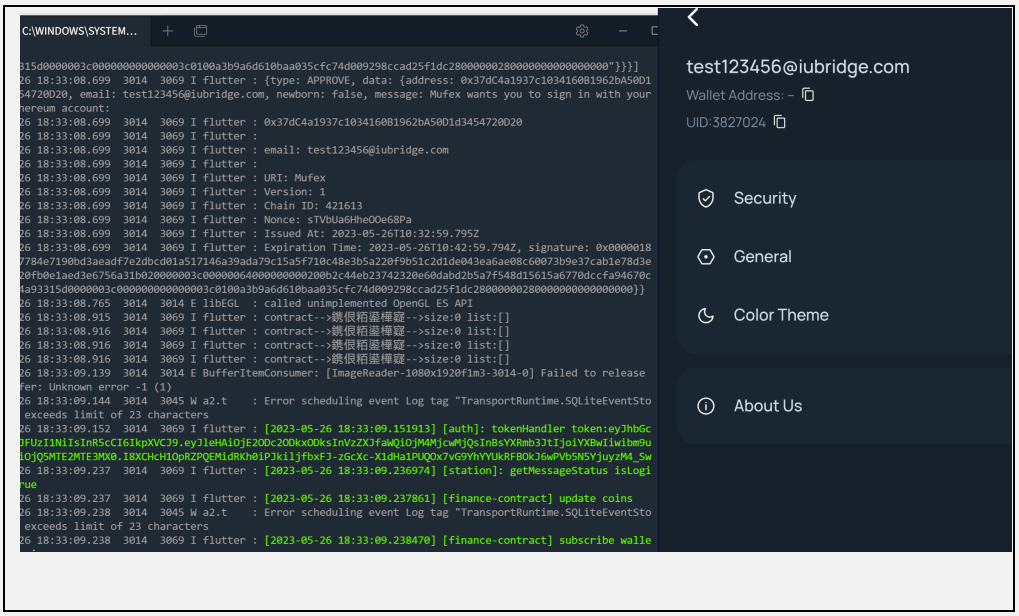
Every application use logs .Mobile application developers use Log class to log debugging information in to the device logs that interacting . These logs are accessible to any other application with READ_LOGS On Vulnerable Application Due to developer Mistake.

Impact

This an attacker can take advantage of if attacker has physical access attacker easily use ***LOGCAT*** and capture all logs details and steal user data such as bank details, user name, password, APi , CSRF Token etc.:

Proof Of Concept

By viewing application log, numen can find token and signature.



Recommendation: Do not log sensitive information

Applications should make sure that they do not send sensitive information to log output. If the app includes a third party library, the developer should make sure that the library does not send sensitive information to log output. One common solution is for an application to declare and use a custom log class, so that log output is automatically turned on/off based on Debug/Release. Developers can use ProGuard to delete specific method calls. This assumes that the method contains no side effects.

References

https://securiumsolutions.com/blogs/sensitive-information-using-logs-cause-leak-of-users-personal-details-password-token/

Affected System(s)

`https://www.testnet.mufex.finance/*`

Appendix A - Penetration Testing Methodologies

A.1 Web Application Assessment

Web Application Assessments can be performed either remotely or on site, depending on the exposure of the application. The purpose of the assessment is to identify any vulnerabilities that can be exploited in order to attack the system or other users, bypass controls, escalate privileges, or extract sensitive data.

During the assessment, the consultants will use proven non-invasive testing techniques to quickly identify any weaknesses. The application is viewed and manipulated from several perspectives, including with no credentials, user credentials, and privileged user credentials.

The primary areas of concern in Web Application Security are authentication bypass, injection, account traversal, privilege escalation, and data extraction.

Our methodology covers all of the OWASP Top 10 web application security risks and more.

Ref	Risk	Description
W1	Broken Access Control	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
W2	Cryptographic Failures	Cryptography is a crucial component of modern security systems, as it provides a way to protect sensitive data by transforming it into an unreadable format. Cryptographic failures can occur due to weak key generation, insecure data storage, broken authentication and session management, insufficient encryption strength, insecure transmission, and insecure random number generation. These vulnerabilities can lead to unauthorized access to sensitive data, or to tamper with or modify data in transit.
W3	Injection	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
W4	Insecure Design	Insecure design refers to the vulnerabilities that arise when security is not properly integrated into the design of the software application. This includes issues such as improper access control, poor data validation, insufficient error handling, and inadequate security testing. Insecure design can lead to various security risks, such as unauthorized access to sensitive data, injection attacks, and denial of service attacks

W5	Security Misconfiguration	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.
W6	Vulnerable and Outdated Components	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover.
W7	Identification and Authentication Failures	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
W8	Software and Data Integrity Failures	Software and data integrity failures refer to vulnerabilities that can arise when there is a lack of proper controls in place to ensure that software and data are not modified or corrupted without authorization. This can occur due to a range of issues, such as improper input validation, insufficient logging and monitoring, and insecure storage of data. These vulnerabilities can lead to modification, delete, or steal sensitive data, or to execute unauthorized commands within the system.
W9	Security Logging and Monitoring Failures	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to other systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.
W10	Server-Side Request Forgery	This vulnerability allows an attacker to send crafted requests from a vulnerable web application to other internal or external systems that the application can communicate with, potentially allowing the attacker to bypass firewalls or access sensitive data. SSRF attacks can be used to perform actions such as port scanning, accessing internal services, and reading or writing files on the affected system. This vulnerability can be exploited if the web application fails to properly validate or sanitize user-supplied input, such as URLs, that are used to construct requests to other systems.

A.2 API Application Assessment

API Application Assessments can be performed either remotely or on site, depending on the exposure of the application. The purpose of the assessment is to identify any vulnerabilities that can be exploited in order to attack the system or other users, bypass controls, escalate privileges, or extract sensitive data.

During the assessment, the consultants will use proven non-invasive testing techniques to quickly identify any weaknesses. The application is viewed and manipulated from several perspectives, including with no credentials, user credentials, and privileged user credentials.

The primary areas of concern in API Application Security are authentication bypass, injection, account traversal, privilege escalation, and data extraction.

Our methodology covers all of the OWASP API Top 10 security risks and more.

Ref	Risk	Description
A1	Broken Object Level Authorization	Broken Object Level Authorization is a security vulnerability that occurs when an API fails to properly enforce access controls at the object level. This means that attackers may be able to access or modify sensitive data without proper authorization. This vulnerability arises due to improper or incomplete implementation of access controls in the API.
A2	Broken Authentication and Session Management	This vulnerability occurs when an API fails to properly implement user authentication and session management mechanisms, making it vulnerable to attacks such as account takeover and session hijacking. Attackers may exploit this vulnerability by stealing user credentials or session tokens and using them to impersonate legitimate users or gain access to sensitive data or functionality
A3	Excessive Data Exposure	This vulnerability occurs when an API exposes too much data or sensitive information, or fails to properly restrict access to sensitive data. Attackers may exploit this vulnerability to steal sensitive data or cause other security issues
A4	Lack of Resources and Rate Limiting	This vulnerability occurs when an API does not have adequate resources or rate limiting mechanisms to handle requests from clients. Attackers may exploit this vulnerability by overwhelming the API with requests, causing it to crash or become unavailable. This can lead to denial-of-service (DoS) attacks and other security issues.
A5	Broken Function Level Authorization	Broken Function Level Authorization occurs when an API fails to properly enforce access controls at the function level, allowing attackers to access or modify sensitive functions or features without proper authorization. Attackers may exploit this vulnerability to gain unauthorized access to sensitive functions or features,

		which can lead to data breaches, privilege escalation, and other security issues.
A6	Mass Assignment	This vulnerability occurs when an API allows clients to modify or create multiple object properties in a single request, without properly enforcing validation or authorization checks. Attackers may exploit this vulnerability to modify or create objects with unauthorized or malicious data, which can lead to data breaches, privilege escalation, and other security issues.
A7	Security Misconfiguration	This vulnerability occurs when an API is not properly configured or secured, leading to potential security issues. Attackers may exploit this vulnerability to gain unauthorized access to the API or its underlying systems, which can lead to data breaches, system compromise, and other security issues.
A8	Injection	This vulnerability occurs when an API accepts untrusted data as input and fails to properly sanitize or validate it, allowing attackers to inject malicious code or commands. For example, an API that accepts SQL queries as input may be vulnerable to SQL injection attacks if it does not properly sanitize or validate the input data. Attackers may exploit this vulnerability to steal or modify data, execute arbitrary code, or gain unauthorized access to systems or data.
A9	Improper Assets Management	This vulnerability occurs when an API fails to properly manage its assets, such as credentials, certificates, keys, or tokens. For example, an API may store sensitive information in plaintext, use weak or default passwords, or fail to properly revoke access to assets when necessary. Attackers may exploit this vulnerability to steal or misuse sensitive assets, which can lead to data breaches, identity theft, and other security issues.
A10	Insufficient Logging and Monitoring	This vulnerability occurs when an API does not provide sufficient logging and monitoring capabilities, making it difficult to detect and respond to security incidents. For example, an API may not log access attempts or errors, or fail to alert administrators of suspicious activity. Attackers may exploit this vulnerability to carry out undetected attacks or maintain unauthorized access for extended periods of time.

A.2 Mobile Application Assessment

Mobile Application Security assessments are unique in that they must take into account the mobile platform and its specific security features, such as permissions, sandboxing, and encryption. Additionally, mobile applications often have access to sensitive data and features such as GPS, cameras, and microphone, which need to be protected.

During the assessment, the consultants will use a combination of automated tools and manual testing to identify vulnerabilities in the application, including those specific to the mobile platform. The application will be tested on both the client-side and server-side, as well as in a dynamic environment.

The primary areas of concern in Mobile Application Security are insecure data storage, insufficient transport layer protection, unintended data leakage, poor authorization and authentication, broken cryptography, client-side injection, security decisions via untrusted inputs, improper session handling, lack of binary protections and insecure communication with backend servers and other external resources.

Our methodology covers all of the OWASP MSTG Top 10 Mobile Application Security risks and more and also includes testing of the mobile-specific features such as permissions, encryption and network communication.

Ref	Risk	Description
M1	Insecure Data Storage	This risk relates to the storage of sensitive data on the mobile device in an insecure manner, such as storing data in clear text or using weak encryption. This can lead to sensitive data being compromised if the device is lost or stolen. Developers should use secure data storage mechanisms such as encrypted databases or secure key-value stores to store sensitive data.
M2	Weak Server Side Controls	This risk relates to the security controls on the server-side that protect against attacks on the mobile application. Weak server-side controls can lead to data leakage, unauthorized access, and other security issues. Developers should implement robust server-side controls such as input validation, authentication, and access controls to protect the mobile application.
M3	Insufficient Transport Layer Protection	This risk relates to the lack of secure communication between the mobile device and the server. Insufficient transport layer protection can lead to man-in-the-middle attacks and other security issues. Developers should use secure communication protocols such as HTTPS and SSL/TLS to protect the data in transit.
M4	Unintended Data Leakage	This risk relates to sensitive data being leaked from the mobile application, either intentionally or unintentionally. This can include data such as login credentials, personal information, and other sensitive data. Developers should

		implement proper data leakage prevention mechanisms such as data masking, encryption, and sanitization to protect sensitive data from being leaked.
M5	Poor Authorization and Authentication	This risk relates to the lack of proper authorization and authentication controls in the mobile application. Poor authorization and authentication can lead to unauthorized access to sensitive data and other security issues. Developers should implement robust and secure authentication and authorization mechanisms such as OAuth, OpenID Connect and Two-factor Authentication to protect the mobile application.
M6	Broken Cryptography	This risk relates to the use of weak or broken encryption in the mobile application. This can lead to sensitive data being compromised if the encryption is broken. Developers should use strong encryption algorithms such as AES, RSA, and ECC and ensure that the encryption keys are securely stored and managed.
M7	Client-Side Injection	This risk relates to the injection of malicious code on the client-side, such as through a SQL injection attack. This can lead to sensitive data being compromised and other security issues. Developers should implement input validation, sanitization and encoding techniques to prevent client-side injection.
M8	Security Decisions via Untrusted Inputs	This risk relates to the use of untrusted inputs in security decisions, such as in access controls. This can lead to security bypasses and other security issues. Developers should validate and sanitize user inputs before using them in security decisions.
M9	Improper Session Handling	This risk relates to the improper handling of user sessions in the mobile application. Improper session handling can lead to session hijacking and other security issues. Developers should implement secure session management
M10	Lack of Binary Protections	This risk relates to the lack of protections on the binary code of the mobile application, such as the lack of code signing or anti-tampering protections. This can lead to the mobile application being easily reverse-engineered and exploited. Developers should implement binary protections such as code signing, anti-debugging and anti-tampering to protect the mobile application.

A.2 External Infrastructure Assessment

An External Infrastructure Assessment checks for the vulnerabilities on which a majority of attacks are based. Infrastructure layer vulnerabilities are usually introduced via misconfiguration or an insufficient patching process.

The assessment is divided into four distinct phases: profiling, discovery, assessment, and exploitation.

Profiling of the corporate Internet-facing infrastructure using non-invasive techniques including OSINT frameworks, but not limited to.

Numen Cyber Labs engineers use a variety of scanning tools and techniques to locate live hosts and services within the target IP range and perform a comprehensive assessment against all IP addresses in scope:

- UDP and TCP port scanning – commonly done using standard port-scanning tools.
- Operating system fingerprinting.
- Service identification – service identification tools are used to analyse all live systems.
- User enumeration – dependent on what services are offered.
- Network mapping – Hping, traceroute, IP fingerprinting.

Once the automated discovery is completed, the results will be investigated in a manual test to identify possible attack vectors. Manual assessment of all live hosts and exposed services focuses on:

- Host and service configuration – misconfigurations and poor build processes can leave insecure services available. These often allow a trivial route to achieve system compromise.
- Patching vulnerabilities – lack of a stringent patching strategy can leave hosts vulnerable; efforts will be made to locate out-of-date services and operating- system-wide missing patches.
- Use of insecure credentials or protocols such as Telnet and FTP may increase the risk of compromise. Any use of these protocols will be highlighted. Default and easy-to-guess passwords will be attempted.

  All exploitation is done in strict accordance with agreed rules of engagement. It should be noted that exploitation is highly dependent on circumstances. Once exploits have succeeded, we use any access and privileges gained to attempt to escalate access rights to the highest level possible. Detailed records are kept of all data recovered and copies are taken before changes are made to any files. All exploits are risk- assessed to minimize disruption to live systems.

Appendix B – CVSS Score Breakdown

Numen uses CVSS v3.1 for Risk Ratings. The Common Vulnerability Scoring System (CVSS) is an open framework that exists to assist in converting vulnerabilities into actionable information and to prioritize vulnerabilities and remediation efforts. CVSS consists of two sets of metrics which are Exploitability and Impact metrics.

Exploitability Metrics consist of Attack Vector, Attack Complexity, Privileges Required, User Interaction, and Scope.

Impact Metrics consist of Confidentiality, Integrity, and availability.

B.1 Exploitability Metrics

Attack Vector

Metric Value	Description
Network (N)	The vulnerable component is bound to the network stack and the set of possible attackers extends beyond the other options listed below, up to and including the entire Internet. Such a vulnerability is often termed "remotely exploitable" and can be thought of as an attack being exploitable at the protocol level one or more network hops away (e.g., across one or more routers). An example of a network attack is an attacker causing a denial of service (DoS) by sending a specially crafted TCP packet across a wide area network (e.g., CVE-2004-0230).
Adjacent (A)	The vulnerable component is bound to the network stack, but the attack is limited at the protocol level to a logically adjacent topology. This can mean an attack must be launched from the same shared physical (e.g., Bluetooth or IEEE 802.11) or logical (e.g., local IP subnet) network, or from within a secure or otherwise limited administrative domain (e.g., MPLS, secure VPN to an administrative network zone). One example of an Adjacent attack would be an ARP (IPv4) or neighbour discovery (IPv6) flood leading to a denial of service on the local LAN segment (e.g., CVE-2013-6014).
Local (L)	The vulnerable component is not bound to the network stack and the attacker's path is via read/write/execute capabilities. Either: • the attacker exploits the vulnerability by accessing the target system locally (e.g., keyboard, console), or remotely (e.g., SSH); or • the attacker relies on User Interaction by another person to perform actions required to exploit the vulnerability (e.g., using social engineering techniques to trick a legitimate user into opening a malicious document).
Physical (P)	The attack requires the attacker to physically touch or manipulate the vulnerable component. Physical interaction may be brief (e.g., evil maid attack1) or persistent. An example of such an attack is a cold boot attack in which an attacker gains access to disk encryption keys after physically accessing the target system. Other examples include peripheral attacks via FireWire/USB Direct Memory Access (DMA).

Attack Complexity

Metric Value	Description
Low (L)	Specialized access conditions or extenuating circumstances do not exist. An attacker can expect repeatable success when attacking the vulnerable component.
High (H)	A successful attack depends on conditions beyond the attacker's control. That is, a successful attack cannot be accomplished at will, but requires the attacker to invest in some measurable amount of effort in preparation or execution against the vulnerable component before a successful attack can be expected. For example, a successful attack may depend on an attacker overcoming any of the following conditions: • The attacker must gather knowledge about the environment in which the vulnerable target/component exists. For example, a requirement to collect details on target configuration settings, sequence numbers, or shared secrets. • The attacker must prepare the target environment to improve exploit reliability. For example, repeated exploitation to win a race condition, or overcoming advanced exploit mitigation techniques. • The attacker must inject themselves into the logical network path between the target and the resource requested by the victim in order to read and/or modify network communications (e.g., a man in the middle attack).

Privileges Required

Metric Value	Description
None (N)	The attacker is unauthorized prior to attack, and therefore does not require any access to settings or files of the vulnerable system to carry out an attack.
Low (L)	The attacker requires privileges that provide basic user capabilities that could normally affect only settings and files owned by a user. Alternatively, an attacker with Low privileges has the ability to access only non-sensitive resources.
High (H)	The attacker requires privileges that provide significant (e.g., administrative) control over the vulnerable component allowing access to component-wide settings and files.

User Interaction

Metric Value	Description
None (N)	The vulnerable system can be exploited without interaction from any user.
Required (R)	Successful exploitation of this vulnerability requires a user to take some action before the vulnerability can be exploited. For example, a successful exploit may only be possible during the installation of an application by a system administrator.

Scope

Metric Value	Description
Unchanged (U)	An exploited vulnerability can only affect resources managed by the same security authority. In this case, the vulnerable component and the impacted component are either the same, or both are managed by the same security authority.
Changed (C)	An exploited vulnerability can affect resources beyond the security scope managed by the security authority of the vulnerable component. In this case, the vulnerable component and the impacted component are different and managed by different security authorities.

B.2 Impacts Metrics

Confidentiality (C)

Metric Value	Description
High (H)	There is a total loss of confidentiality, resulting in all resources within the impacted component being divulged to the attacker. Alternatively, access to only some restricted information is obtained, but the disclosed information presents a direct, serious impact. For example, an attacker steals the administrator's password, or private encryption keys of a web server.
Low (L)	There is some loss of confidentiality. Access to some restricted information is obtained, but the attacker does not have control over what information is obtained, or the amount or kind of loss is limited. The information disclosure does not cause a direct, serious loss to the impacted component.
None (N)	There is no loss of confidentiality within the impacted component.

Integrity (I)

Metric Value	Description
High (H)	There is a total loss of integrity, or a complete loss of protection. For example, the attacker is able to modify any/all files protected by the impacted component. Alternatively, only some files can be modified, but malicious modification would present a direct, serious consequence to the impacted component.
Low (L)	Modification of data is possible, but the attacker does not have control over the consequence of a modification, or the amount of modification is limited. The data modification does not have a direct, serious impact on the impacted component.
None (N)	There is no loss of integrity within the impacted component.

Availability (A)

Metric Value	Description
High (H)	There is a total loss of availability, resulting in the attacker being able to fully deny access to resources in the impacted component; this loss is either sustained (while the attacker continues to deliver the attack) or persistent (the condition persists even after the attack has completed). Alternatively, the attacker has the ability to deny some availability, but the loss of availability presents a direct, serious consequence to the impacted component (e.g., the attacker cannot disrupt existing connections, but can prevent new connections; the attacker can repeatedly exploit a vulnerability that, in each instance of a successful attack, leaks a only small amount of memory, but after repeated exploitation causes a service to become completely unavailable).
Low (L)	Performance is reduced or there are interruptions in resource availability. Even if repeated exploitation of the vulnerability is possible, the attacker does not have the ability to completely deny service to legitimate users. The resources in the impacted component are either partially available all of the time, or fully available only some of the time, but overall there is no direct, serious consequence to the impacted component.
None (N)	There is no loss of availability within the impacted component.

B.3 CVSS Score Breakdown

Severity	Base Score Range
Informational	0.0
Low	0.1 – 3.9
Medium	4.0 – 6.9
High	7.0 – 8.9
Critical	9.0 – 10.0

Appendix C: Scope, Disclaimer and Limitations

C.1 Scope of Work

URL(s) In Scope

Binary(s) In Scope

C.2 Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without Numen's prior written consent.
This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Numen to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

C.3 Testing Limitations

Testing reflects a point-in-time assessment of in-scope systems and services. Future changes to the environment and the availability of new public vulnerabilities could introduce new security vulnerabilities or alter the risk of security findings outlined within this report.