Suiet Wallet Extension

June 2023

WALLET EXTENTION AUDIT REPORT

NUMEN

www.numencyber.com

Table of Contents

1. EXECUTIVE SUMMARY

Numen Cyber Technology was engaged by Suiet to review wallet extension implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

There is a medium risk. After tampering with the password, you can directly use the fake password to obtain the mnemonic phrase. At present, this risk has been modified. There are 10 low-level risks, 7 of which have been modified, and the remaining risks have been temporarily ignored.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

1.1 METHODOLOGY

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood and impact are categorized into three ratings: High, Medium and Low. Severity is determined by likelihood and impact and can be classified into four categories accordingly, Critical, High, Medium, Low shown in table 1.1.

	Risk Matrix		
LIKELIHOOD	Medium	High	Critical
	Low	Medium	High
	Informational	Low	Medium
	IMPACT		

Table 1.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any issues found, we may test further in our private environment and run tests to confirm the results. The concrete list of check items is shown in Table 1.2.

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all the codes, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high-level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles.

While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation. We hypothesize what vulnerabilities may be present, creating Issue entries, and for each we follow the following Issue Investigation and Remediation process.

Documenting Results

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Category	Assessment Item
Transaction Process Security Audit	Transaction Signature Security Audit
	Transfer Security Audit
	Transaction Broadcast Audit
Private Key/Mnemonic Phrase Security Audit	Private Key/Mnemonic Generation Security Audit
	Private Key/Mnemonic Storage Security Audit
	Private Key/Mnemonic Use Process Security Audit
	Private Key/Mnemonic Backup Security Audit
	Private Key/Mnemonic Destroy Security Audit
	Random Security Audit
	Cryptographic Security Audit
Web Front-end Security Audit	Third-party JS Security Audit
	HTTP Response Header Security Audit
Communications Security Audit	Communication Encryption Security Audit
	Cross-domain Transmission Security audit
Architecture and Business Logic Security Audit	Access Control Security Audit
	DApp Communication Security Audit
	Business Design Security Audit
	Architecture Design Security Audit

Table 1.2: The Full List of Assessment Items

2. FINDINGS OVERVIEW

2.1 PROJECT INFO AND CONTRACT ADDRESS

Project Name: Suiet

Project URL: https://suiet.app/

Audit Time: May 8th, 2023 – June 13th, 2023

Source Code Link	Commit Hash
https://github.com/suiet	e9d6deea0ddc24f28fd32462c2a9ceb3cba5f29e

2.2 SUMMARY

Severity	Found	
Critical	0	
High	0	
Medium	1	🟧
Low	10	🟩🟩🟩🟩🟩🟩🟩🟩🟩🟩
Informational	1	🟦

2.3 KEY FINDINGS

ID	Severity	Findings Title	Status	Confirm
NVE- 001	Low	Missing Chrome Version Detection	Fixed	Confirmed
NVE- 002	Low	Insecure Plugin Configuration: Security Settings For Content_Scripts	Ignored	Confirmed
NVE- 003	Low	Unencrypted Persistence Of Wallet Private Key In Memory	Fixed	Confirmed
NVE- 004	Low	Unencrypted Persistence Of Wallet Mnemonic In Memory	Fixed	Confirmed
NVE- 005	Low	Unencrypted Persistence Of Wallet Password In Memory	Fixed	Confirmed
NVE- 006	Low	Unencrypted Persistence Of Wallet Private Key In Clipboard	Ignored	Confirmed
NVE- 007	Low	Potential Display Of Incorrect Data In The UI	Ignored	Confirmed

ID	Severity	Findings Title	Status	Confirm
NVE- 008	Medium	Mnemonic Retrieval Risk via Fake Password after Tampering	Fixed	Confirmed
NVE- 009	Low	Failure To Reset Sensitive Information After Closing Pop-Up Window	Fixed	Confirmed
NVE- 010	Low	Malicious User Login Risk with Altered Passwords	Fixed	Confirmed
NVE- 011	Low	Unfiltered Plugin Wallet Name String	Fixed	Confirmed
NVE- 012	Informational	Proper Handling Of Forgotten Password	Fixed	Confirmed

Table 2.1: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

3.1 MISSING CHROME VERSION DETECTION

ID:	NVE-001		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

The Suiet wallet, relying on Chrome's security measures, is built on top of the Chrome browser. However, the lack of Chrome version testing means that the wallet might be operating in an environment with known security vulnerabilities. These vulnerabilities could be exploited by malicious attackers, potentially compromising the security of users.

```
SQL
// Resolve agent head
function getChromeVersion() {
  const userAgent = navigator.userAgent;
  const chromeVersion = userAgent.match(/Chrome\/(\d+)/);
  if (chromeVersion && chromeVersion.length > 1) {
    return chromeVersion[1];
  } else {
    return ' Unknown version';
  }
}
```

Recommendations:

To mitigate this risk, it is strongly recommended to implement Chrome browser version checking within the Suiet wallet. By incorporating version detection, the wallet can ensure that it operates in a secure environment and take appropriate measures if an outdated or vulnerable version of Chrome is detected.

Result: CONFIRMED

Fix Result: Fixed

3.2 INSECURE PLUGIN CONFIGURATION: SECURITY SETTINGS FOR CONTENT_SCRIPTS

ID:	NVE-002		
Severity:	Low	**Category:**	Business Issues

Likelihood:	Low	**Impact:**	Low

Description:

The Suiet 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.

Recommendations:

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 Suiet 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.

Result: CONFIRMED

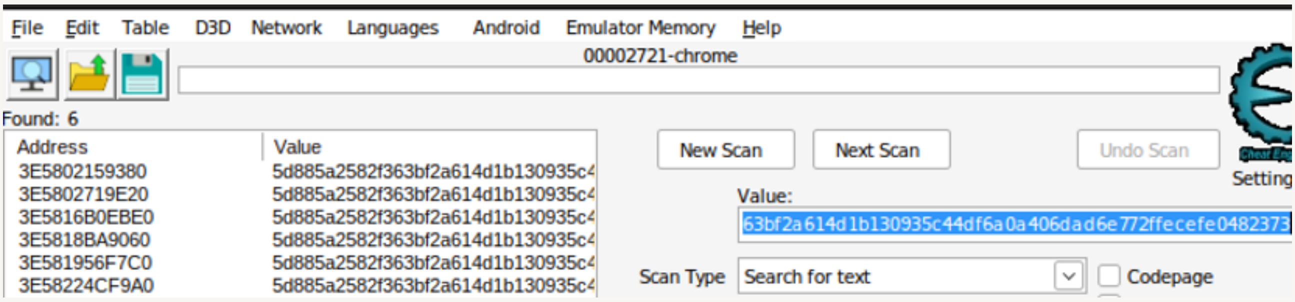
Fix Result: Ignored

3.3 UNENCRYPTED PERSISTENCE OF WALLET PRIVATE KEY IN MEMORY

ID:	NVE-003		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

During the processing of wallet data streams, the Suiet wallet fails to encrypt the private key, leading to its persistence in memory. Consequently, the unencrypted private key remains accessible not only within the child process but also in the main Chrome process for an extended duration.



00002721-chrome

Found: 6

Address	Value
3E5802159380	5d885a2582f363bf2a614d1b130935c4
3E5802719E20	5d885a2582f363bf2a614d1b130935c4
3E5816B0EBE0	5d885a2582f363bf2a614d1b130935c4
3E5818BA9060	5d885a2582f363bf2a614d1b130935c4
3E581956F7C0	5d885a2582f363bf2a614d1b130935c4
3E58224CF9A0	5d885a2582f363bf2a614d1b130935c4

Value: 63bf2a614d1b130935c44df6a0a406dad6e772ffecefe0482373

Scan Type: Search for text

□ Codepage

Recommendations:

To enhance the security of the wallet, it is crucial to encrypt the private key during the processing of wallet data streams. By implementing encryption, the Suiet wallet can protect the confidentiality of the private key and prevent unauthorized access to sensitive information. This encryption should be applied consistently throughout the transaction processing, including both the child and main Chrome processes.

Result: **CONFIRMED**

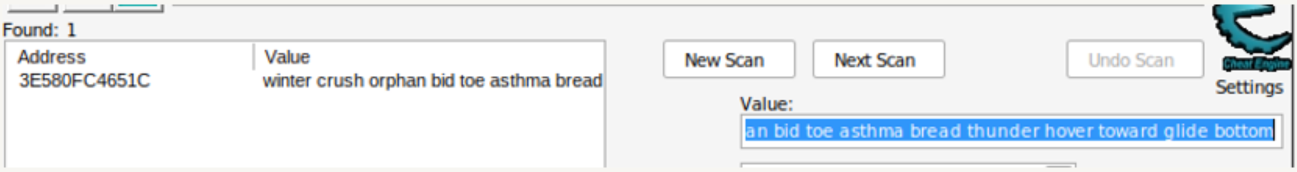
Fix Result: Fixed

3.4 UNENCRYPTED PERSISTENCE OF WALLET MNEMONIC IN MEMORY

ID:	NVE-004		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

The Suiet wallet does not apply encryption to the wallet's mnemonic, resulting in its persistence in memory during the processing of wallet data streams. This poses a security risk as the unencrypted mnemonic remains accessible within the system's memory, potentially exposing sensitive information.



Found: 1

Address	Value
3E580FC4651C	winter crush orphan bid toe asthma bread

Value: an bid toe asthma bread thunder hover toward glide bottom

Recommendations:

To address this vulnerability, it is highly recommended to encrypt the wallet's mnemonic during the processing of wallet data streams. Encryption ensures the confidentiality of the mnemonic and adds an additional layer of protection against unauthorized access or exposure of sensitive data.

Result: **CONFIRMED**

Fix Result: Fixed

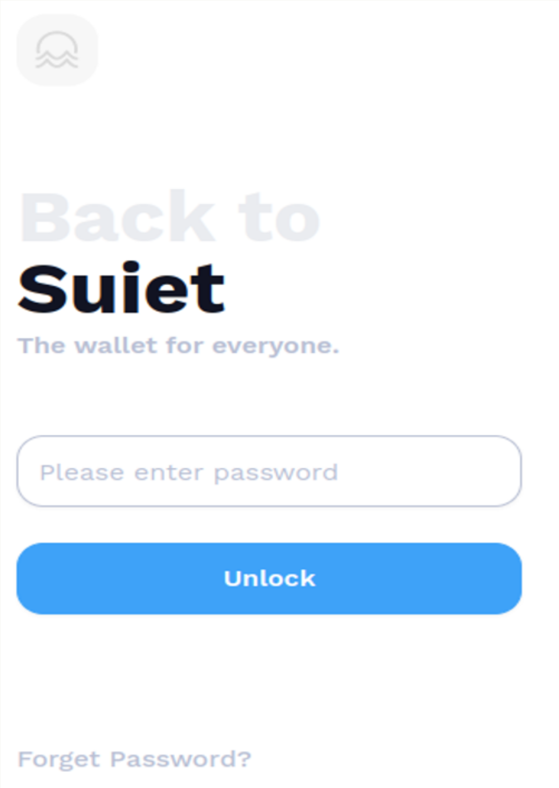
3.5 UNENCRYPTED PERSISTENCE OF WALLET PASSWORD IN MEMORY

ID:	NVE-005		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

During the processing of the wallet data stream, the Suiet wallet does not encrypt the user's password, allowing it to persist in memory. It is important to note that the wallet password is not leaked to the main process, but its retention in memory during transaction processing poses a security concern.

Reproduction Process:
1. Open the wallet and enter the password.



Back to
Suiet
The wallet for everyone.

2. After entering the password, the page transitions to the Wallet. However, at this stage, the password is not erased from memory. A lingering password record remains visible in memory. Even if the Chrome plugin is later suspended, the password continues to persist in memory for an extended duration.

Recommendations:

To mitigate information leakage, it has been observed that the current approach of using the gc policy and arrayBuffer does not effectively and consistently resolve the issue. Although utilizing arrayBuffer helps to some extent, the password still remains in memory briefly.

a-Attack cost mitigation

Temporary solution to generate more strings in memory with key/key matches, increasing the cost of the attack

b-use chrome.runtime.reload()

After the user enters the password, consider prompting the user that the wallet must be reloaded for security, then local marker or remote state marker, and finally call chrome.runtime.reload() to flush the memory.

The leakage of the main process helper and private key occurs when these elements are transferred to the main process during communication and subsequently retained in the cache. Given that the Suiet wallet password does not leak to the main process, it is advisable to prioritize addressing the leakage of the wallet password first. Subsequently, further experimentation and verification through multiple iterations are required to explore complete reuse of the private key and helper for password handling.

Result: CONFIRMED

Fix Result: Fixed

3.6 UNENCRYPTED PERSISTENCE OF WALLET PRIVATE KEY IN CLIPBOARD

ID:	NVE-006		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

The Suiet wallet shares a clipboard with other Chrome plugins, meaning that when exporting keys and clicking the copy button, other plugins can concurrently access the keys stored in the clipboard. This creates a potential security vulnerability.

Recommendations:

Unfortunately, the issue of shared clipboard access cannot be fully mitigated due to the nature of the feature supported by Chrome. However, it is recommended to prompt users effectively about the risks associated with shared clipboard access. Users should be advised to disable the plugin support for clipboard access feature to prevent other plugins from accessing sensitive key information stored in the clipboard. This can be done by navigating to the following *URL: chrome://settings/content/clipboard*. By following these recommendations, the Suiet wallet can strengthen its security measures, protect user data, and mitigate potential risks associated with the mentioned findings.

Result: CONFIRMED

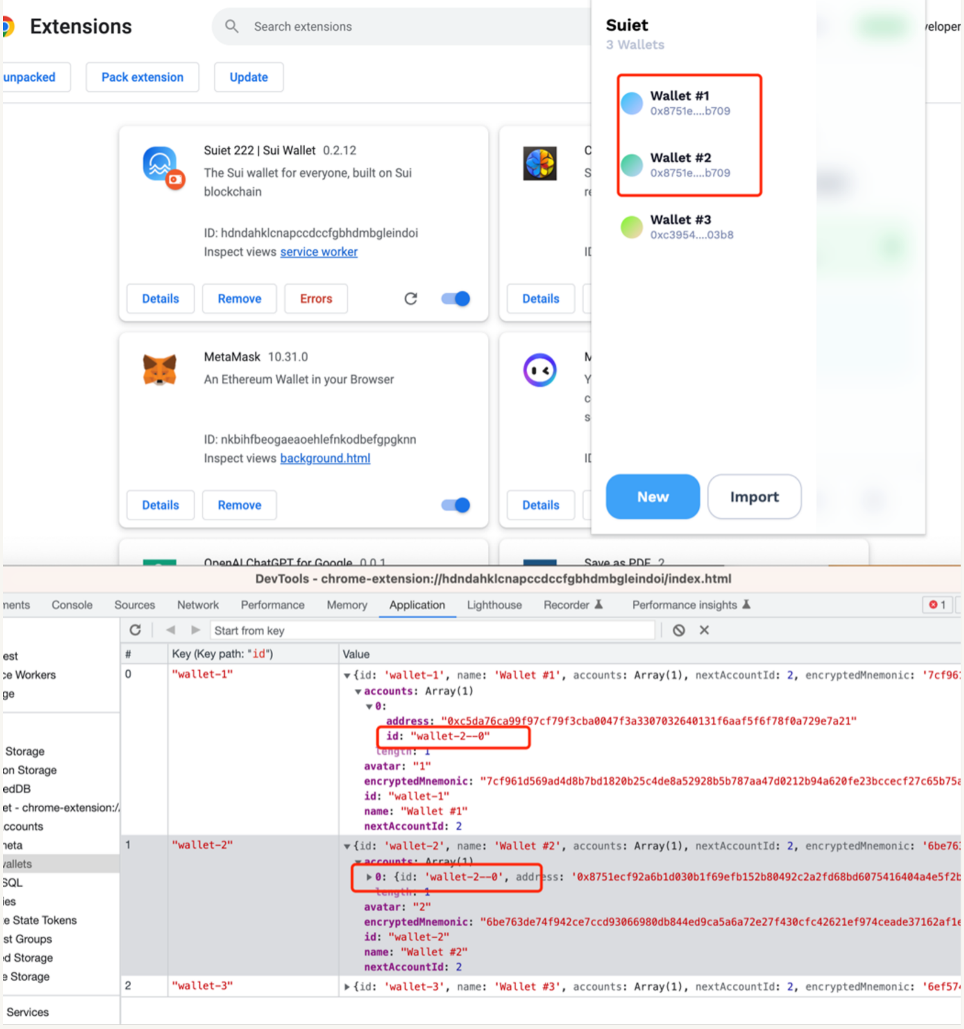
Fix Result: Ignored

3.7 POTENTIAL DISPLAY OF INCORRECT DATA IN THE UI

ID:	NVE-007		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

The UI may inadvertently show incorrect or "dirty" data due to the generation of repetitive wallet IDs. This can lead to inconsistencies in the displayed information.



```
#   Key (Key path: "id")   Value
0   "wallet-1"             ▾{id: 'wallet-1', name: 'Wallet #1', accounts: Array(1), nextAccountId: 2, encryptedMnemonic: '7cf96
                             ▾accounts: Array(1)
                               ▾0:
                                   address: "0xc5da76ca99f97cf79f3cba0047f3a3307032640131f6aaf5f6f78f0a729e7a21"
                                   id: "wallet-2--0"
                                 length: 1
                             avatar: "1"
                             encryptedMnemonic: "7cf961d569ad4d8b7bd1820b25c4de8a52928b5b787aa47d0212b94a620fe23bccecf27c65b75a
                             id: "wallet-1"
                             name: "Wallet #1"
                             nextAccountId: 2
1   "wallet-2"             ▾{id: 'wallet-2', name: 'Wallet #2', accounts: Array(1), nextAccountId: 2, encryptedMnemonic: '6be76
                             ▾accounts: Array(1)
                               ▸0: {id: 'wallet-2--0', address: '0x8751ecf92a6b1d030b1f69efb152b80492c2a2fd68bd6075416404a4e5f2b
                                 length: 1
                             avatar: "2"
                             encryptedMnemonic: "6be763de74f942ce7ccd93066980db844ed9ca5a6a72e27f430cfc42621ef974ceade37162af1e
                             id: "wallet-2"
                             name: "Wallet #2"
                             nextAccountId: 2
2   "wallet-3"             ▸{id: 'wallet-3', name: 'Wallet #3', accounts: Array(1), nextAccountId: 2, encryptedMnemonic: '6ef57
```

Recommendations:

1. Instead of using a self-increasing order for keys, it is advisable to implement UUIDs in indexedDB. This prevents easy guessing of key values, especially when default names correspond to the ID and name serial number.
2. During the unlocking or updating process, apart from the password, it is recommended to check other fields, such as "accounts.id," for duplicates. If duplicates are detected, users should be alerted about the potential risk of an attack.
3. To mitigate the mentioned risk, it is crucial to configure the content_security_policy appropriately. This ensures that malicious scripts cannot be injected into indexedDB, modifying data and potentially causing security vulnerabilities.

Result: CONFIRMED

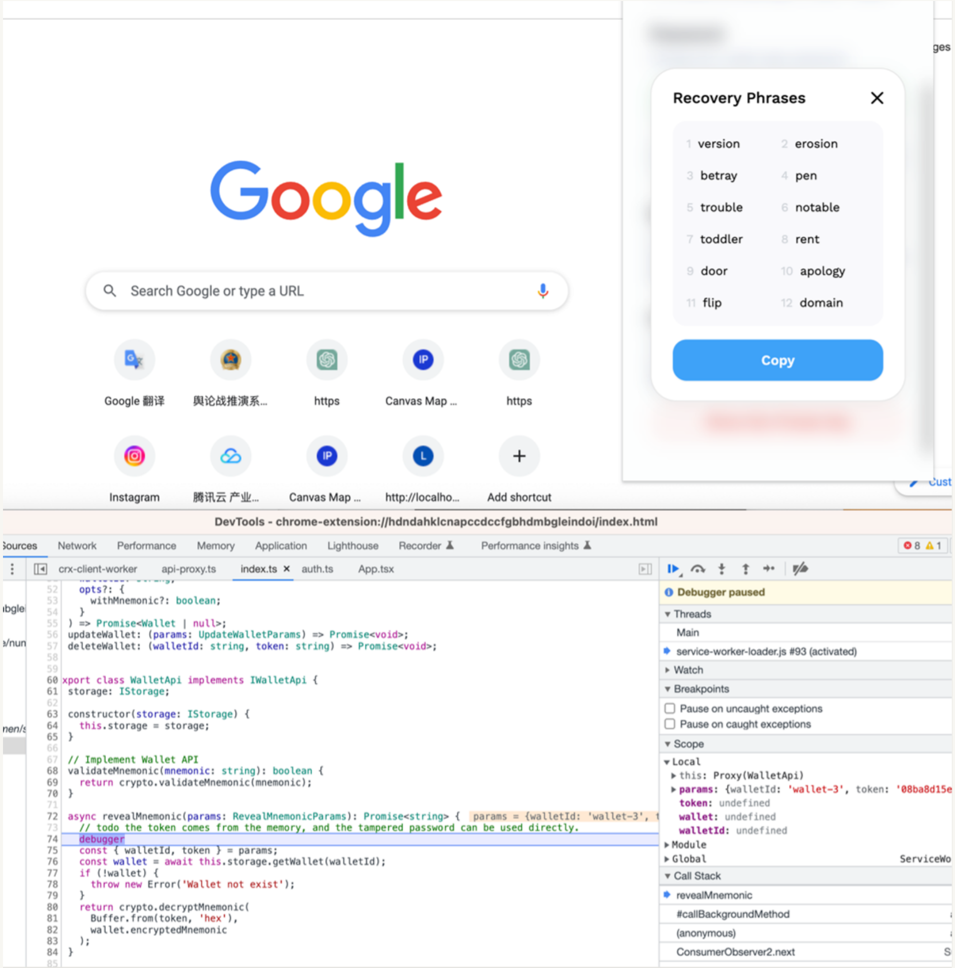
Fix Result: Ignored

3.8 MNEMONIC RETRIEVAL RISK VIA FAKE PASSWORD AFTER TAMPERING

ID:	NVE-008		
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Medium

Description:

When tampering with the cipher, a vulnerability arises in the system. Exploiting this vulnerability allows an attacker to use a fake password to retrieve the mnemonic through the "wallet.revealMnemonic" function. This function relies solely on the token stored in the session, bypassing the need for the actual password. To enhance security, it is recommended to enforce password confirmation through the "verifypassword" method to verify the user's password.



Recovery Phrases

1 version	2 erosion
3 betray	4 pen
5 trouble	6 notable
7 toddler	8 rent
9 door	10 apology
11 flip	12 domain

Copy

DevTools - chrome-extension://hdndahklcnapccdccfgbhdmbgleindoi/index.html

```
  opts?: {
    withMnemonic?: boolean;
  }
) => Promise<Wallet | null>;
updateWallet: (params: UpdateWalletParams) => Promise<void>;
deleteWallet: (walletId: string, token: string) => Promise<void>;


xport class WalletApi implements IWalletApi {
storage: IStorage;

constructor(storage: IStorage) {
  this.storage = storage;
}

// Implement Wallet API
validateMnemonic(mnemonic: string): boolean {
  return crypto.validateMnemonic(mnemonic);
}

async revealMnemonic(params: RevealMnemonicParams): Promise<string> {  params = {walletId: 'wallet-3', t
  // todo the token comes from the memory, and the tampered password can be used directly.
  debugger
  const { walletId, token } = params;
  const wallet = await this.storage.getWallet(walletId);
  if (!wallet) {
    throw new Error('Wallet not exist');
  }
  return crypto.decryptMnemonic(
    Buffer.from(token, 'hex'),
    wallet.encryptedMnemonic
  );
}
```

Debugger paused

▾ Threads
Main
service-worker-loader.js #93 (activated)

▸ Watch

▾ Breakpoints
□ Pause on uncaught exceptions
□ Pause on caught exceptions

▾ Scope

```
▾Local
 ▸this: Proxy(WalletApi)
 ▸params: {walletId: 'wallet-3', token: '08ba8d15e
  token: undefined
  wallet: undefined
  walletId: undefined
▸Module
▸Global                              ServiceWo
```

▾ Call Stack
revealMnemonic
#callBackgroundMethod
(anonymous)
ConsumerObserver2.next

Recommendations:

To prevent the exploitation of the token stored in the session after tampering with the cipher, any interface involving sensitive information, such as the helper or private key, should be decrypted using the password as an additional safeguard.

Result: CONFIRMED

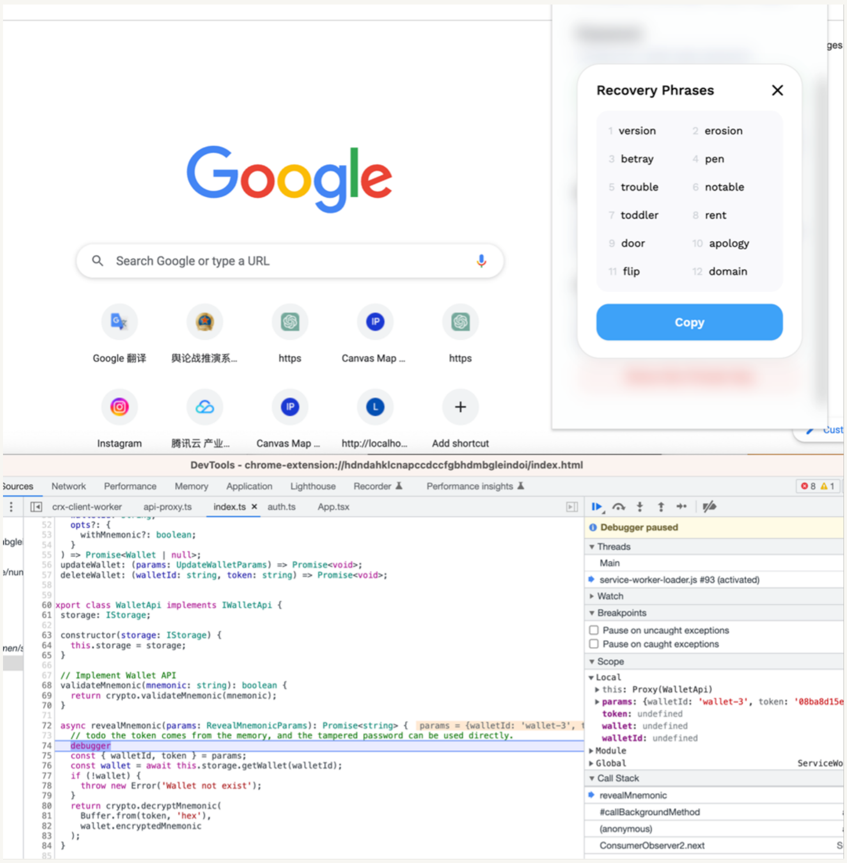
Fix Result: Fixed

3.9 FAILURE TO RESET SENSITIVE INFORMATION AFTER CLOSING POP-UP WINDOW

ID:	NVE-009		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

The UI displays the "revealMnemonic" interface without properly parsing the helper words. This occurs because sensitive information is not reset after closing the previous pop-up window. Even after addressing other vulnerabilities, there remains a possibility of exposing the helper words when using a tampered password.



Recovery Phrases

1 version	2 erosion
3 betray	4 pen
5 trouble	6 notable
7 toddler	8 rent
9 door	10 apology
11 flip	12 domain

DevTools - chrome-extension://hndndahklcnapccdccfgbhdmbgleindoi/index.html

```
    opts?: {
      withMnemonic?: boolean;
    }
  ) => Promise<Wallet | null>;
  updateWallet: (params: UpdateWalletParams) => Promise<void>;
  deleteWallet: (walletId: string, token: string) => Promise<void>;


xport class WalletApi implements IWalletApi {
  storage: IStorage;

  constructor(storage: IStorage) {
    this.storage = storage;
  }

  // Implement Wallet API
  validateMnemonic(mnemonic: string): boolean {
    return crypto.validateMnemonic(mnemonic);
  }

  async revealMnemonic(params: RevealMnemonicParams): Promise<string> {  params = {walletId: 'wallet-3', t
    // todo the token comes from the memory, and the tampered password can be used directly.
    debugger
    const { walletId, token } = params;
    const wallet = await this.storage.getWallet(walletId);
    if (!wallet) {
      throw new Error('Wallet not exist');
    }
    return crypto.decryptMnemonic(
      Buffer.from(token, 'hex'),
      wallet.encryptedMnemonic
    );
  }
```

Debugger paused

Threads
Main
service-worker-loader.js #93 (activated)

Watch

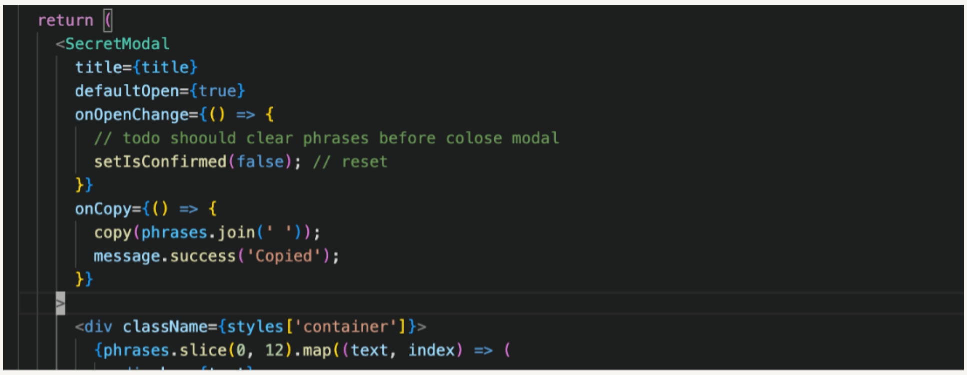
Breakpoints
□ Pause on uncaught exceptions
□ Pause on caught exceptions

Scope
```
Local
  this: Proxy(WalletApi)
  params: {walletId: 'wallet-3', token: '08ba8d15e
  token: undefined
  wallet: undefined
  walletId: undefined
Module
Global                                    ServiceWo
```

Call Stack
revealMnemonic
#callBackgroundMethod
(anonymous)
ConsumerObserver2.next

Recommendations:

When uninstalling any component that contains sensitive information, it is crucial to ensure that the component state is properly reset. Refer to the provided figure for an example:



```
return (
  <SecretModal
    title={title}
    defaultOpen={true}
    onOpenChange={() => {
      // todo shoould clear phrases before colose modal
      setIsConfirmed(false); // reset
    }}
    onCopy={() => {
      copy(phrases.join(' '));
      message.success('Copied');
    }}
  >
    <div className={styles['container']}>
      {phrases.slice(0, 12).map((text, index) => (
```

Result: CONFIRMED

Fix Result: Fixed

3.10 MALICIOUS USER LOGIN RISK WITH ALTERED PASSWORDS

ID:	NVE-010		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

The "maybeFixDataConsistency" function, responsible for verifying data consistency, currently does not throw an exception when the account is not found. This creates a potential security risk, allowing malicious users to log into the wallet using tampered passwords. If the account ID is not found, the validation of the helper will not be performed to update the wallets, enabling the use of a malicious password to bypass this function. Once logged in, the accounts table and cipher can be restored, erasing any traces of password tampering.

Risk Level: Medium to low. It is important to note that while funds cannot be stolen, other tested operations such as disabling Touch ID or modifying the network can still be carried out.

Recommendations:

It is recommended to throw an exception in the condition (!accountData) or utilize the wallets table to recover the accounts data.

Result: CONFIRMED

Fix Result: Fixed

3.11 UNFILTERED PLUGIN WALLET NAME STRING

ID:	NVE-011		
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

Potential malicious attacks: If the plugin wallet name string is not properly filtered, hackers can exploit this vulnerability to steal sensitive user information, including passwords, private keys, and more, by sending malicious requests.

Risk of data leakage: In the absence of proper filtering of the plugin wallet name string, there is a risk of sensitive user data being exposed during communication with others. For instance, if a user's plugin wallet name is "My Personal Wallet," it could be stolen by a hacker, leading to a severe data breach.

Misconfiguration risk: Failure to filter the plugin wallet name string can result in misconfiguration, leading to erroneous transfers of the user's wallet assets to unintended addresses.

Recommendations:

To ensure security, it is crucial to implement proper filtering and encryption for the plugin wallet name string during its development. This will prevent the disclosure of sensitive user information. It is also advisable to avoid using sensitive terms such as "password" or "private key" in the plugin wallet name string to minimize the risk of exploitation.

Result: CONFIRMED

Fix Result: Fixed

3.12 PROPER HANDLING OF FORGOTTEN PASSWORD

ID:	NVE-012		
Severity:	Informational	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Informational

Description:

Proper handling of a forgotten password in a plugin wallet involves importing the helper word, rather than resetting the password directly. Direct password resets can pose security implications.

Risk of asset vulnerability: Resetting the password directly results in the user losing control over their wallet assets, rendering them unable to secure them. If a user forgets their wallet password, there is a possibility of theft or unauthorized transfer of assets to an unsecured address.

Loss of asset recovery capability: Direct password resets may hinder the user's ability to recover previously stored assets in the wallet. The helper word is a crucial piece of information used to authenticate the wallet's identity, and without it, previous assets cannot be retrieved.

Inability to perform security upgrades: Resetting the password directly can prevent the wallet from implementing necessary security upgrades, leaving the assets vulnerable. For example, if the wallet introduces a new encryption algorithm, but the user fails to perform the security upgrade, the wallet may be unable to handle the new encryption algorithm effectively, potentially resulting in asset theft.

Recommendations:

To ensure proper handling of a forgotten password, it is recommended to import the helper word and reset the password using that information. This approach ensures that the user maintains control over their wallet assets and enables the implementation of necessary security upgrades. The process of importing the helper word should incorporate security measures such as encryption and signatures to maintain the confidentiality and accuracy of the helper word.

Result: **CONFIRMED**

Fix Result: Fixed

4. CONCLUSION

In this audit, we thoroughly analyzed Suiet wallet extension implementation. The problems found are described and explained in detail in Section 3. The issues identified in the audit have been raised with project leaders, and high- and medium-risk vulnerabilities need to be fixed as soon as possible. We, therefore, consider the audit result to be **PASSED**. To improve this report, we greatly appreciate any constructive feedback or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without Numen's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Numen to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Numen's position is that each company and individual are responsible for their own due diligence and continuous security. Numen's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

www.numencyber.com

contact@numencyber.com

@numencyber

github.com/NumenCyber

NUMEN