CERTIK

Security Assessment

Clore Wallet- Pentest

CertiK Assessed on Jul 11th, 2024

CertiK Assessed on Jul 11th, 2024

Clore Wallet- Pentest

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Wallet	Desktop Application	Dynamic Testing, Manual Review

LANGUAGE	TIMELINE	KEY COMPONENTS
Pentest	Delivered on 07/11/2024	N/A

Vulnerability Summary

10	9	0	0	1	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed immediately. Users should be cautious when interacting with any application with outstanding critical risks.
1	High	1 Resolved	High risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds, thief of user data, and/or loss control of the application.
5	Medium	5 Resolved	Medium risks may not pose a security risk at a large scale, but they can affect the overall functioning of a platform or be used to target a certain group of users.
3	Low	3 Resolved	Low risks can be any of the above, but on a smaller impact. They generally do not compromise the overall integrity of the project.
1	Informational	1 Acknowledged	Informational errors are often recommendations to improve the configuration or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the application.

TABLE OF CONTENTS | CLORE WALLET- PENTEST

SCOPE | CLORE WALLET- PENTEST

Source code	https://github.com/Corey-Code/clore-app/tree/8e05b62f74a72445dae42300aae6999d2bbe70d3
Electron app	Mac
Electron app	Windows
Retest Source Code	https://gitlab.com/cloreai-public/clore-wallet/-/commit/19073c55b505917769eb7c4709d3760192d60183

APPROACH & METHODS | CLORE WALLET- PENTEST

This report has been prepared for Clore Wallet to discover issues and vulnerabilities in the application of the Clore Wallet-Pentest project. Clore Wallet is a desktop application for managing Clore Coins. It allows users to store and manage their crypto assets.

The pentest was a manual assessment of the security of the application's functionality, business logic, and vulnerabilities, such as those cataloged in the OWASP Top 10. The assessment also included a review of security controls and requirements listed in the OWASP Application Security Verification Standard (ASVS). The pentesters leveraged tools to facilitate their work. However, the majority of the assessment involved manual analysis.

The main objective of the engagement is to test the overall resiliency of the application to various real-world attacks against the application's controls and functions and thereby be able to identify its weaknesses and provide recommendations to fix and improve its overall security posture.

Two members of the CertiK team were involved in completing the engagement, which took place over the course of 2 days in June 2024 and yielded 10 security-relevant findings. The most significant vulnerabilities are the devtools being enabled and the insecure management of the mnemonic.

Other weaknesses were also found and are detailed in the Findings section of the report. We recommend addressing these findings to ensure a high level of security standards and industry practices and to raise the security posture of the application.

REVIEW NOTES | CLORE WALLET- PENTEST

The reviewed desktop applications communicate with third-party domains that do not belong to Clore. These domains are out of scope for the security review.

- magic.vidulum.app
- blockbook.clore.zelcore.io

Additionally, the tests were conducted using the code hosted in the private GitHub repository at https://github.com/Corey-Code/clore-app/ (commit: `8e05b62f74a72445dae42300aae6999d2bbe70d3`). However, the retest was performed using the code published on GitLab https://gitlab.com/cloreai-public/clore-wallet (commit: `19073c55b505917769eb7c4709d3760192d60183`). The domains listed above were not present in the latest version of the code.

It is important to note that during the retest phase, only the fixes of the identified findings are validated. New functionalities or logic unrelated to the findings were not reviewed.

FINDINGS | CLORE WALLET- PENTEST

10	0	1	5	3	1
Total Findings	Critical	High	Medium	Low	Informational

This report has been prepared to discover issues and vulnerabilities for Clore Wallet- Pentest. Through this security assessment, we have uncovered 10 issues ranging from different severity levels. Utilizing the techniques of Dynamic Testing & Manual Review to complement rigorous testing process, we discovered the following findings:

ID	Title	Category	Severity	Status
GLOBAL-05	Enabled Developer Console In The Desktop Application	Security Misconfiguration	High	Resolved
GLOBAL-01	Insecure Key Derivation Mechanism	Insufficient Cryptography	Medium	Resolved
GLOBAL-02	Weak Password Policy	Account Policy	Medium	Resolved
GLOBAL-03	Insecure Mnemonic Storage	Insecure Data Storage	Medium	Resolved
GLOBAL-06	Unrestricted Navigation In The Desktop Electron Wallet Application	Security Misconfiguration	Medium	Resolved
GLOBAL-10	Logging Of Sensitive Information	Information Disclosure	Medium	Resolved
GLOBAL-04	Use Of Weak Random Number Generator	Insufficient Cryptography	Low	Resolved
GLOBAL-07	Password Not Required After Logout And Close Application	Logic Flaws	Low	Resolved
GLOBAL-08	Insecure Implementation Of Electron ContextBridge	Logic Flaws	Low	Resolved
GLOBAL-09	Potentially Insecure Electron Fuses Not Disabled	Security Misconfiguration	Informational	Acknowledged

GLOBAL-05 ENABLED DEVELOPER CONSOLE IN THE DESKTOP APPLICATION

Category	Severity	Location	Status
Security Misconfiguration	High		Resolved

Description

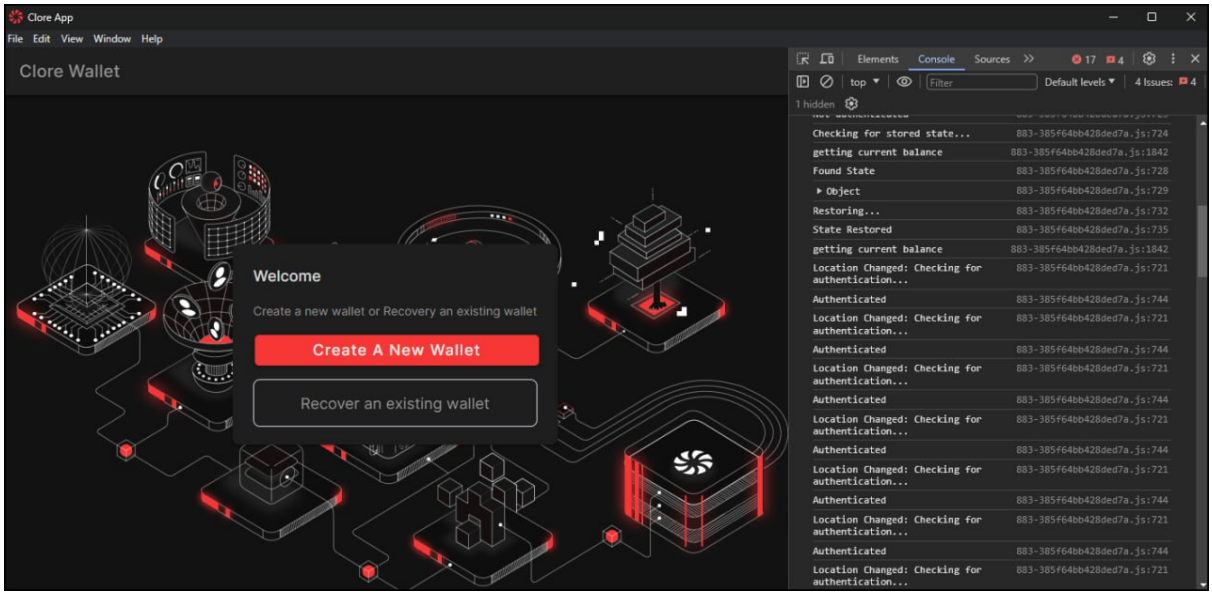
During the security assessment, it was discovered that the Electron desktop application enables access to the development console. This design flaw presents an increased attack surface, as malicious actors could exploit unwary users—that may copy and paste malicious scripts into the console—potentially compromising both the wallet application and the user's computer system.

Impact

The unintended exposure of the development console allows for the execution of JavaScript commands. If leveraged by an attacker, this could result in the injection of malicious code leading to financial loss from unauthorized transactions and overall compromise of the user's computer security.

Reproduce Steps

1. Open the Clore desktop wallet
2. Click on `View > Toggle Developer Tools`



Recommendation

It is recommended that the development console be disabled in the production build of the wallet application. Removing access to this function will significantly reduce the attack surface and protect end users from potential exploitation

Alleviation

[Clore Wallet Team, 07/08/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-01 | INSECURE KEY DERIVATION MECHANISM

Category	Severity	Location	Status
Insufficient Cryptography	Medium	components/hooks/use-crypto.ts:6	Resolved

Description

In the `use-crypto.ts` file, the `encrypt` function encrypts the mnemonic using a password hashed 18 times with SHA-256 to generate an AES CRT encryption key. This approach could be vulnerable to brute-force or dictionary attacks if the password is weak, predictable, or compromised. Typically, password-based key derivation should involve a key derivation function, such as bcrypt or scrypt, that includes a salt and multiple iterations to slow down such attacks.

```
export function encrypt(mnemonic: string, password: string) {
  const key = hash(password, 18).slice(32);
  const mnemonicBytes = aesjs.utils.utf8.toBytes(mnemonic);
  const aesCtr = new aesjs.ModeOfOperation.ctr(
    Buffer.from(key),
    new aesjs.Counter(5),
  );
  const encryptedBytes = aesCtr.encrypt(mnemonicBytes);
  const encryptedHex = aesjs.utils.hex.fromBytes(encryptedBytes);
  return encryptedHex;
}
```

Impact

The current hash method can be much faster to break down, compared to a proper key derivation function.

Recommendation

Use a password-based key derivation function such as Argon2d, Scrypt, Bcrypt with sufficient iteration and salt to generate an encryption key with the user password.

Alleviation

[Clore Wallet Team, 07/09/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-02 | WEAK PASSWORD POLICY

Category	Severity	Location	Status
Account Policy	Medium		Resolved

Description

An authentication mechanism is only as strong as its credentials. For this reason, it is important to require users to have strong passwords. Lack of password complexity significantly reduces the search space when trying to guess the user's passwords, making brute-force attacks easier.

The currently implemented password policy is permissive regarding the requirements related to the password length and complexity, as it allows any string value to be set as the wallet password.

Impact

Weak password policies restrict users from creating complex passwords and leaves them vulnerable to brute-force attacks by a malicious actor.

Reproduce Steps

The password check is performed in the function `handleCreateWallet` located in `components/routes/register/stepper/verify-phrase.tsx`, and there are no further checks.

```
if (getValues().password === getValues().confirmPassword) {
  const phrase = recover ? _m : mnemonic;
  const wallet = await createWallet(
    getValues().name,
    phrase,
    getValues().confirmPassword,
  );
```

Recommendation

It is recommended to implement a strong password policy for the user accounts. Strong password policies should include at least eight characters, containing uppercase and lowercase letters, numbers, and special characters.

Alleviation

[Clore Wallet Team, 07/08/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-03 | INSECURE MNEMONIC STORAGE

Category	Severity	Location	Status
Insecure Data Storage	Medium	components/hooks/use-secure-storage.ts	Resolved

Description

The application stores the mnemonic in an encrypted format using a random string (ent) as the encryption key. However, this random string (ent) is also stored in the same storage system (IndexedDB). This practice significantly weakens the security of the encrypted mnemonic, as an attacker who gains access to the storage system can retrieve both the encrypted mnemonic and the key used to encrypt it.

Impact

Storing the encryption key alongside the encrypted mnemonic nullifies the benefits of encryption. An attacker with access to the storage can decrypt the mnemonic easily, leading to unauthorized access to the cryptocurrency wallet and potential financial loss for the users.

Reproduce Steps

The `createWallet` function in `components/hooks/use-clore-state.ts` uses the `secureStorage` hook to store the mnemonic in the IndexedDB.

```
44 export async function createWallet(
45   name: string,
46   mnemonic: string[],
47   password: string,
48 ) {
49   try {
50     // FIXME: Perform validations here
51     const s = defaultState;
52     s.passwordHash = hash(password);
53     s.encryptedMnemonic = encrypt(mnemonic.join(' '), password);
54     await secureStorage.setItem('m', mnemonic.join(' '));
55     useCloreState.setState(s);
56     const success = await addWallet(name);
57     if (!success) return false;
58     return s;
59   } catch (error) {
60     console.error(error);
61     return false;
62   }
63 }
```

The `secureStorage` implementation is located in `components/hooks/use-secure-storage.ts`.

```
31 const setItem = async (key: string, value: any) => {
32   if (key === 'm') {
33     const ent = mEnt(randomIntFromInterval(30, 40));
34     console.log('ent', ent);
35     console.log('ent hash', sha256(ent));
36     console.log('value', value);
37
38     value = encrypt(value, sha256(ent));
39     console.log('encrypted value', value);
40     await storage.set('ent', ent);
41   }
42   await storage.set(key, value);
43 };
```

Recommendation

It is recommended not to store the encryption key within the same storage system as the encrypted mnemonic. Instead, the application should use a password entered by the user to derive the encryption key each time it is needed. This can be achieved using a key derivation function (KDF) such as PBKDF2, bcrypt, scrypt or Argon2, which generates a strong encryption key from the user's password. This approach ensures that even if the storage system is compromised, the mnemonic remains secure as the key is not stored and must be re-derived using the user's password.

Alleviation

[Clore Wallet Team, 07/08/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-06 UNRESTRICTED NAVIGATION IN THE DESKTOP ELECTRON WALLET APPLICATION

Category	Severity	Location	Status
Security Misconfiguration	Medium		Resolved

Description

The desktop application does not restrict or limit navigation, allowing the renderer process to navigate to any URL. This can lead to security vulnerabilities as it opens up the potential for attackers to exploit navigation to execute malicious code, steal sensitive data, or perform phishing attacks.

Impact

The unrestricted navigation capability in the Electron wallet application can lead to different security risks such as Cross-Site Scripting (XSS), Phishing Attacks, Code Injection, etc.

Recommendation

It is recommended to disable or limit navigation in the Electron wallet application. This can be achieved by using the `will-navigate` event to restrict navigation to trusted URLs only. For more information:

- https://www.electronjs.org/docs/latest/tutorial/security#13-disable-or-limit-navigation

Alleviation

[Clore Wallet Team, 07/08/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-10 | LOGGING OF SENSITIVE INFORMATION

Category	Severity	Location	Status
Information Disclosure	Medium	components/hooks/use-secure-storage.ts:36 components/hooks/use-secure-storage.ts:53	Resolved

Description

The desktop application logs the mnemonic phrase of the wallet. The mnemonic phrase is a critical piece of information used to restore access to the wallet and should be handled with the highest level of security.

Impact

An attacker with access to the log files can retrieve the mnemonic phrase and use it to restore and take control of the wallet.

Recommendation

It is recommended to remove the logging statements with sensible information such as mnemonic phrase.

Alleviation

[Clore Wallet Team, 07/08/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-04 | USE OF WEAK RANDOM NUMBER GENERATOR

Category	Severity	Location	Status
Insufficient Cryptography	Low	components/hooks/use-secure-storage.ts	Resolved

Description

The application uses `Math.random()` to generate both the length and characters of a random string (ent), which is subsequently used to encrypt the mnemonic. `Math.random()` is not a cryptographically secure random number generator, meaning the values it produces can be predictable and susceptible to attacks.

Impact

If an attacker can predict or reproduce the sequence of random numbers generated, they can decrypt the mnemonic, gaining full access to the cryptocurrency wallet and the funds within it.

Reproduce Steps

The affected code is located in `components/hooks/use-secure-storage.ts` in the functions `mEnt` and `randomIntFromInterval`.

```
 7  function mEnt(length: number) {
 8    let result = '';
 9    const characters =
10      'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
11    const charactersLength = characters.length;
12    let counter = 0;
13    while (counter < length) {
14      result += characters.charAt(Math.floor(Math.random() * charactersLength));
15      counter += 1;
16    }
17    return result;
18  }
19
20  function randomIntFromInterval(min: number, max: number) {
21    return Math.floor(Math.random() * (max - min + 1) + min);
22  }
```

Recommendation

It is recommended to use the Web Crypto API instead, and more precisely the `crypto.getRandomValues()` method for any functionality that requires cryptographically secure random numbers.

Alleviation

[Clore Wallet Team, 07/08/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-07 PASSWORD NOT REQUIRED AFTER LOGOUT AND CLOSE APPLICATION

Category	Severity	Location	Status
Logic Flaws	Low		Resolved

Description

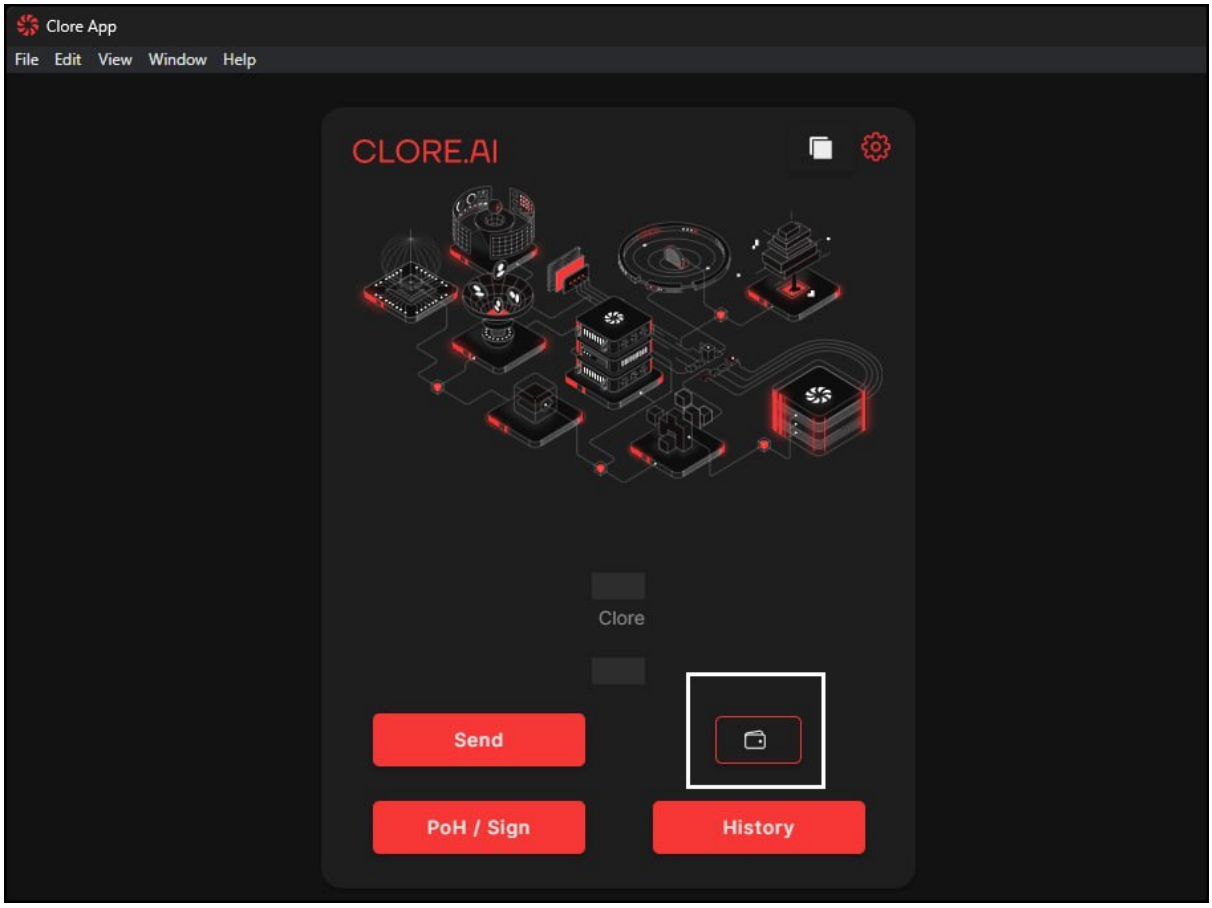
The desktop application does not prompt the user to enter their password after logging out and closing the application. Although the logout functionality is designed to lock the wallet, it fails to enforce password re-entry.

Impact

An attacker can easily access the wallet without needing the password, leading to potential unauthorized access. This can result in financial loss.

Reproduce Steps

1. While authenticated click on `Settings > Log out`.
2. Wait a few seconds and click on `Recover an existing wallet`.
3. Click on the wallet icon to select the wallet and load its information.



Recommendation

It is recommended to enforce password re-entry after logging out and closing the application.

Alleviation

[Clore Wallet Team, 07/08/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-08 INSECURE IMPLEMENTATION OF ELECTRON CONTEXTBRIDGE

Category	Severity	Location	Status
Logic Flaws	Low	electron/preload.js	Resolved

Description

The current implementation of contextBridge in the Electron application directly exposes IPC channels and arguments without proper validation. This allows any website loaded by the Electron app to send arbitrary IPC messages, which can lead to potential security vulnerabilities.

For more information:

- https://www.electronjs.org/docs/latest/tutorial/context-isolation#security-considerations

Although the implementation is considered insecure, the application does not handle IPC events. However, it is important to maintain a secure design.

Impact

By exposing IPC channels without validation, the application becomes susceptible to various attacks, including unauthorized access to privileged APIs, data leakage, and remote code execution. This can compromise the integrity and confidentiality of the application and its data.

Recommendation

It is recommended to provide specific methods for each IPC message to prevent arbitrary messages from being sent. Additionally, when using TypeScript, add types to the exposed APIs for better security.

Alleviation

[Clore Wallet Team, 07/08/2024]: The team heeded the advice and resolved the issue in commit: 19073c55b505917769eb7c4709d3760192d60183.

GLOBAL-09 POTENTIALLY INSECURE ELECTRON FUSES NOT DISABLED

Category	Severity	Location	Status
Security Misconfiguration	Informational		Acknowledged

Description

The application does not disable potentially insecure Electron fuses such as `runAsNode` and `nodeCliInspect` enabled. These can be exploited to run unauthorized commands through environment variables or CLI arguments.

Impact

With these fuses enabled external scripts can execute commands on the user's device, leading to unauthorized access and potential security breaches. The attacker should have command execution in the device to take advantage of this issue.

Recommendation

Review and disable unnecessary Electron fuses using the `@electron/fuses` module to minimize the attack surface and enhance application security.

For more information:

- https://www.electronjs.org/docs/latest/tutorial/fuses#how-do-i-flip-the-fuses

Alleviation

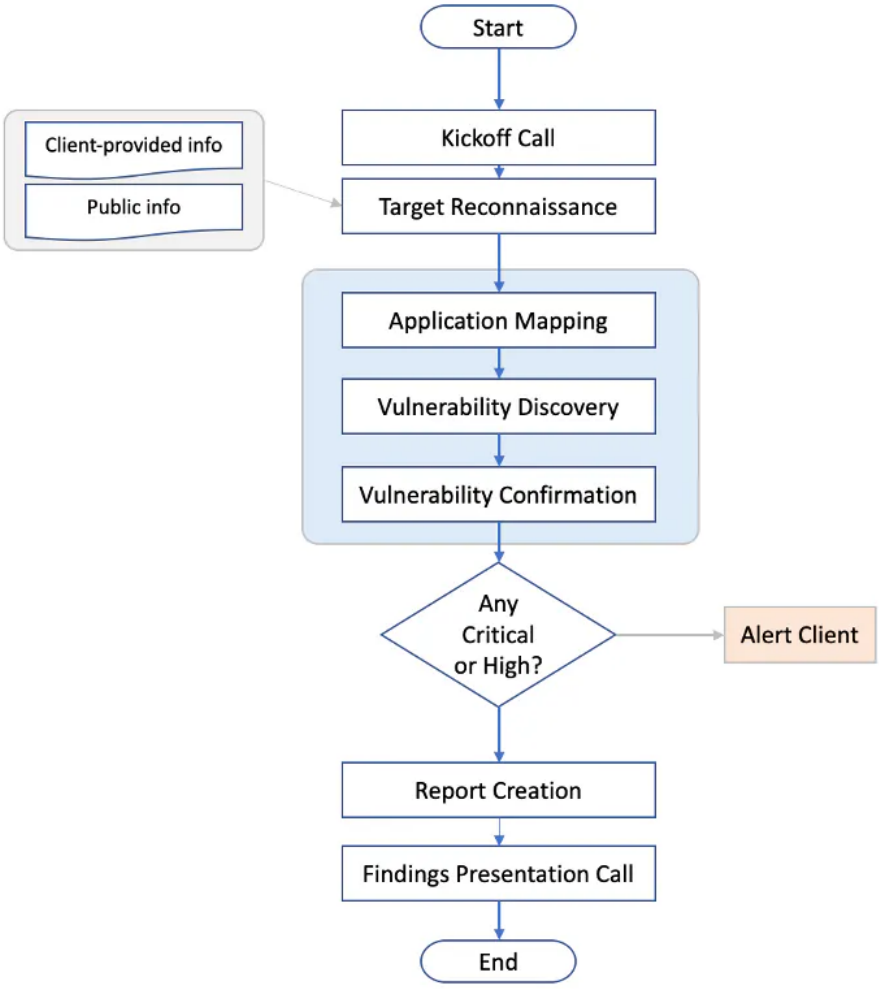
[Clore Wallet Team, 07/08/2024]: The team acknowledged the finding and decided not to change the current codebase.

APPENDIX | CLORE WALLET- PENTEST

Methodology

CertiK uses a comprehensive penetration testing methodology which adheres to industry best practices and standards in security assessments including from OWASP (Open Web Application Security Project), NIST, PTES (Penetration Testing Execution Standard).

Below is a flowchart of our assessment process:



Coverage and Prioritization

As many components as possible will be tested manually. Priority is generally based on three factors: critical security controls, sensitive data, and the likelihood of vulnerability.

Critical security controls will always receive the top priority in the test. If a vulnerability is discovered in the critical security control, the entire application is likely to be compromised, resulting in a critical-risk to the business. For most applications, critical controls will include the login page, but it could also include major workflows such as the checkout function in an online store.

The Second priority is given to application components that handle sensitive data. This is dependent on business priorities,

but common examples include payment card data, financial data, or authentication credentials.

Final priority includes areas of the application that are most likely to be vulnerable. This is based on CertiK' experience with similar applications developed using the same technology or with other applications that fit the same business role. For example, large applications will often have older sections that are less likely to utilize modern security techniques.

Reconnaissance

CertiK gathers information about the target application from various sources depending on the type of test being performed. CertiK obtains whatever information that is possible and appropriate from the client during scoping and supplements it with relevant information that can be gathered from public sources. This helps provide a better overall picture and understanding of the target.

Application Mapping

CertiK examines the application, reviewing its contents, and mapping out all its functionalities and components. CertiK makes use of different tools and techniques to traverse the entire application and document all input areas and processes. Automated tools are used to scan the application and it is then manually examined for all its parameters and functionalities. With this, CertiK creates and widens the overall attack surface of the target application.

Vulnerability Discovery

Using the information that is gathered, CertiK comes up with various attack vectors to test against the application. CertiK uses a combination of automated tools and manual techniques to identify vulnerabilities and weaknesses. Industry-recognized testing tools will be used, including Burp Suite, Nikto, Metasploit, and Kali. Furthermore, any controls in place that would inhibit the successful exploitation of a particular system will be noted.

Vulnerability Confirmation

After discovering vulnerabilities in the application, CertiK validates the vulnerabilities and assesses its overall impact. To validate, CertiK performs a Proof-of-Concept of an attack on the vulnerability, simulating real world scenarios to prove the risk and overall impact of the vulnerability.

Through CertiK's knowledge and experience on attacks and exploitation techniques, CertiK is able to process all weaknesses and examine how they can be combined to compromise the application. CertiK may use different attack chains, leveraging different weaknesses to escalate and gain a more significant compromise.

To minimize any potential negative impact, vulnerability exploitation was only attempted when it would not adversely affect production applications and systems, and then only to confirm the presence of a specific vulnerability. Any attack with the potential to cause system downtime or seriously impact business continuity was not performed. Vulnerabilities were never exploited to delete or modify data; only read-level access was attempted. If it appeared possible to modify data, this was noted in the list of vulnerabilities below.

Immediate Escalation of High or Critical Findings

If critical or high findings are found whereby application elements are compromised, client's key security contacts will be notified immediately.

Risk Assessment

Risk Level	CVSS Score	Impact	Exploitability
Critical	9.0-10.0	Root-level or full-system compromise, large-scale data breach	Trivial and straightforward
High	7.0-8.9	Elevated privilege access, significant data loss or downtime	Easy, vulnerability details or exploit code are publicly available, but may need additional attack vectors (e.g., social engineering)
Medium	4.0-6.9	Limited access but can still cause loss of tangible assets, which may violate, harm, or impede the org's mission, reputation, or interests.	Difficult, requires a skilled attacker, needs additional attack vectors, attacker must reside on the same network, requires user privileges
Low	0.1-3.9	Very little impact on an org's business	Extremely difficult, requires local or physical system access
Informational	0.0	Discloses information that may be of interest to an attacker.	Not exploitable but rather is a weakness that may be useful to an attacker should a higher risk issue be found that allows for a system exploit

DISCLAIMER | CERTIK

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

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 CertiK 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. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK'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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.