Zellic

Pontem Aptos Wallet

Browser Extension Security Assessment

February 13, 2023

Prepared for:

Boris Povod

Pontem Technology Ltd.

Prepared by:

Corben Leo and Maik Robert

Zellic Inc.

Contents

About Zellic

Zellic was founded in 2020 by a team of blockchain specialists with more than a decade of combined industry experience. We are leading experts in smart contracts and Web3 development, cryptography, web security, and reverse engineering. Before Zellic, we founded perfect blue, the top competitive hacking team in the world. Since then, our team has won countless cybersecurity contests and blockchain security events.

Zellic aims to treat clients on a case-by-case basis and to consider their individual, unique concerns and business needs. Our goal is to see the long-term success of our partners rather than simply provide a list of present security issues. Similarly, we strive to adapt to our partners' timelines and to be as available as possible. To keep up with our latest endeavors and research, check out our website zellic.io or follow @zellic_io on Twitter. If you are interested in partnering with Zellic, please email us at hello@zellic.io or contact us on Telegram at https://t.me/zellic_io.

1 Executive Summary

Zellic conducted an audit for Pontem Technology Ltd. from October 10th to October 14th, 2022.

Our general overview of the code is that it was very well-organized and structured. The code coverage is high, and tests are included for the majority of the functions. The documentation was adequate, although it could be improved. The code was easy to comprehend, and in most cases, intuitive.

We applaud Pontem Technology Ltd. for their attention to detail and diligence in maintaining incredibly high code quality standards in the development of Pontem Aptos Wallet.

Zellic thoroughly reviewed the Pontem Aptos Wallet codebase to find application-breaking bugs as defined by the documentation and to find any technical issues outlined in the Methodology section (2.2) of this document.

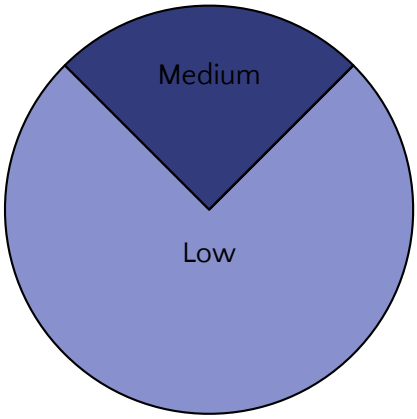
Specifically, taking into account Pontem Aptos Wallet's threat model, we focused heavily on issues that would break core invariants, such as insecure cryptographic functions, insecure seed-phrase storage, cross-site scripting, clickjacking, denial of service, and more.

During our assessment on the Pontem Aptos Wallet wallet, we discovered four findings. Fortunately, no critical issues were found. Of the four findings, one was medium severity, and the remaining findings were low severity.

Additionally, Zellic recorded its notes and observations from the audit for Pontem Technology Ltd.'s benefit in the Discussion section (4) at the end of the document.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	1
Low	3
Informational	0



2 Introduction

2.1 About Pontem Aptos Wallet

Pontem Aptos Wallet is a cryptocurrency wallet for Aptos.

2.2 Methodology

During a security assessment, Zellic works through standard security auditing phases, including automated testing and manual review. These processes can vary significantly per engagement, but most of our time is spent on thorough manual review.

Alongside a variety of open-source tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in web-based applications arise from oversights during development. Missing an authentication check on a single API endpoint can circumvent the entire authentication model of the application. Failure to properly sanitize and encode user input can lead to vulnerabilities like SQL injection or cross-site scripting. We use automated tools to identify unsafe code patterns and perform a thorough manual review for vulnerable code patterns.

Business logic errors. Business logic is the heart of all applications. We manually review logic to ensure that the code implements the expected functionality specified in the platform's design documents. We also thoroughly examine the specifications and designs for inconsistencies, flaws, and vulnerabilities. This involves use cases that open the opportunity for abuse.

Complex integration risks. Web projects contain third-party dependencies and interact with APIs and code that are not under the developer's control. Auditors will review the project's external interactions and identify potentially dangerous behavior, such as implicitly trusting API responses or assuming third-party code functionality.

Code maturity. We review for possible improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also suggest possible improvements to code clarity, documentation, and usability.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact; we assign it on a case-by-case basis based on our professional judgment and experience. As one would expect, both the severity and likelihood of an issue affect its impact; for in-

stance, a highly severe issue's impact may be attenuated by a very low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Similarly, Zellic organizes its reports such that the most important findings come first in the document rather than being ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their importance may differ. This varies based on numerous soft factors, such as our clients' threat models, their business needs, their project timelines, and so forth. We aim to provide useful and actionable advice to our partners that consider their long-term goals rather than simply provide a list of security issues at present.

2.3 Scope

The engagement involved a review of the following targets:

Pontem Aptos Wallet Browser Extension

Repository	https://github.com/pontem-network/pontem-pitaka
Versions	`5adff90870da7e72bcaad45ac52c31288a82c54f`
Programs	• src/*.js • scripts/*.js
Type	Javascript
Platform	Web

2.4 Project Overview

Zellic was contracted to perform a security assessment with two consultants for a total of two person-weeks. The assessment was conducted over the course of two calendar weeks.

Contact Information

The following project managers were associated with the engagement:

Jasraj Bedi, Co-founder
jazzy@zellic.io

Stephen Tong, Co-founder
stephen@zellic.io

The following consultants were engaged to conduct the assessment:

Corben Leo, Engineer
corben@zellic.io

Maik Robert, Engineer
maik@zellic.io

2.5 Project Timeline

The key dates of the engagement are detailed below.

October 10, 2022	Start of primary review period
October 17, 2022	End of primary review period

3 Detailed Findings

3.1 Malformed responses to the coinInfo API can soft lock the wallet

- **Target**: src/data/queries/coinInfo.ts
- **Category**: Coding Mistakes
- **Likelihood**: Low
- **Severity**: Medium
- **Impact**: Medium

Description

A request is automatically sent to the following endpoint `/v1/accounts/0x1/resource/0x1::coin::CoinInfo%3C0x1::aptos_coin::AptosCoin%3E` during startup. The handler fails to check for errors, leading to a permanent soft lock when malformed data is returned.

There are multiple scenarios where this could happen:

- RPC endpoint encounters an error
- RPC endpoint is malicious

The requests are repeated, so the extension stays bricked as long as the returned data is malformed.

```
async () => {
    return aptos.getAccountResource(extractAddressFromType(token
    as string), composeType(network.structs.CoinInfo, [token as string]))
    .then((value: AptosResource<AptosCoinInfoResource>) => {
        const type = token as string;
        const decimals = +value.data.decimals;
        const name = value.data.name;
        const symbol = value.data.symbol;
        const alias = network.tokenAlias[token as string]
    ?? value.data.symbol;
        addTokenInfo({ name, symbol, decimals });
        return { type, decimals, name, symbol, alias };
    })
},
{
    ...RefetchOptions.INFINITY,
```

```
    enabled:  !!token
}
```

Impact

It leads to a permanent soft lock of the whole extension. It can be fixed by directly visiting `chrome-extension://<extensionId>/index.html#/settings/` and switching the network or reinstalling the extension.

Recommendations

We recommend additional error handling when handling RPC responses.

Remediation

A fix was introduced in commit 9b4ad36e by incorporating error handling into the function, effectively preventing the wallet extension from experiencing a persistent, endless loop in the event of receiving malformed data.

3.2 Low password complexity threshold

- **Target**: src/extension/modules/SignUp/SetPasswordForm/index.tsx
- **Category**: Coding Mistakes
- **Severity**: Low
- **Likelihood**: Medium
- **Impact**: High

Description

The only requirement for the keyring password is that it needs to be at least six characters long.

```
const validate = (values: SubmitPasswordFormValues) => {
  const errors: SubmitPasswordFormErrors = {};

  if (!values.password) {
    errors.password = "Password required";
  } else if (values.password.length < MIN_PASSWORD_LENGTH) {
    errors.password = `Password length should contain minimum
    ${MIN_PASSWORD_LENGTH} characters`;
  }

  if (!values.confirm) {
    errors.confirm = "Password confirmation required";
  } else if (values.confirm.length < MIN_PASSWORD_LENGTH) {
    errors.confirm = `Password confirmation length should contain minimum
    ${MIN_PASSWORD_LENGTH} characters`;
  } else if (values.confirm !== values.password) {
    errors.confirm = "Password confirmation not similar";
  }

  if (!values.agreed) {
    errors.agreed = "You need to agree with terms and conditions";
  }

  return errors;
};
```

Impact

A six-character password can be bruteforced in a matter of seconds, leading to a compromise of the wallet.

Recommendations

We recommend Pontem Technology Ltd. increase the length requirements along with mandating special characters and lowercase and uppercase letters.

Remediation

A fix was introduced in commit e6ad1094 by adding multiple requirements on password entry such as minimum password length and special characters.

3.3 Cleartext password in the browser's session storage

- **Target**: src/auth/hooks/useKeyring.ts
- **Category**: Coding Mistakes
- **Likelihood**: Low
- **Severity**: Low
- **Impact**: High

Description

After a user creates or unlocks their wallet, their password is stored in plaintext in the session storage. This is a critical piece of information and should never be available in plaintext form.

```
const createWallet = async (password: string) => {
  const address = await controller.createNewKeychain(password);
  if (IS_EXTENSION_RUNTIME) {
    await extension.storage.session.set({ storedPassword: password });
  }

  return address;
};
```

```
const unlock = async (password: string) => {
  const keyrings = await controller.unlock(password);
  if (IS_EXTENSION_RUNTIME) {
    await extension.storage.session.set({ storedPassword: password });
  }

  return keyrings;
};
```

Impact

An attacker with physical access to the machine or a cross-domain exploit can leak the plaintext password and mnemonic phrase.

Recommendations

Handling of the plaintext password should be kept to the minimum and should be immediately deleted or encrypted after use.

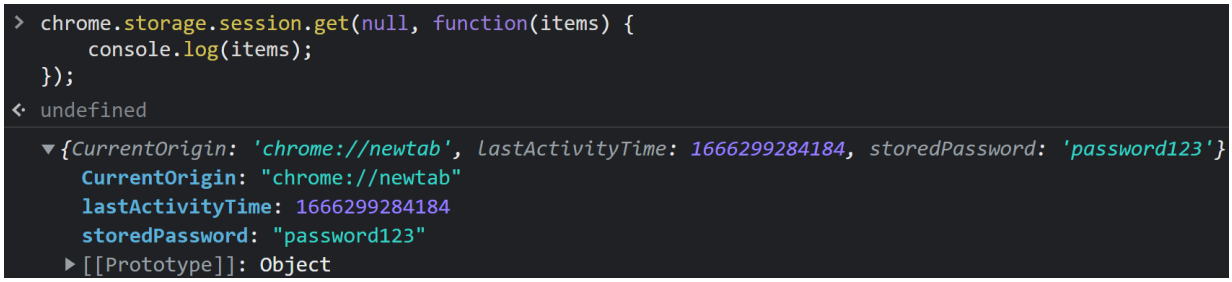


Figure 3.1: Example of cleartext password in session storage.

Remediation

A fix was introduced in commit 0b6c08fb by encrypting the password before setting it in the local storage. A refactor of the flow is planned, which will remove the password from storage entirely. It's worth noting that the password is not stored permanently and is automatically deleted after five minutes of inactivity.

3.4 RPC responses can overwrite local state

- **Target**: src/app/InitNetworks.tsx
- **Category**: Coding Mistakes
- **Severity**: Low
- **Likelihood**: Low
- **Impact**: Medium

Description

The extension implicity trusts all the information from the RPC API, even overriding local variables/state. We can see the local chainId has been overwritten with the chainId of the devnet via a malicious API response.

Figure 3.2: Devnet chainId in localstorage.

Figure 3.3: Local chainId set to the same value as the devnet chainId.

Impact

A malicious RPC can override locally stored variables like chainId, later prompting the users to sign messages with the overriden chainId. This may mislead the user as they would not expect a network to sign messages with a chainId they did not configure.

Recommendations

Users should be prompted for additional network information that should never deviate, such as the chainId. The RPC responses can then be compared with the local variables and rejected if they do not match.

Remediation

A fix was introduced in commit 4f29b735 to only allow the devnet to change the ChainId, since the devnet ChainId may change over time.

4 Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment.

4.1 New wallet creation workflow does not prompt a re-enter of seed phrase

During the creation of a new wallet, the user is not prompted to reconfirm their seed phrase after it is displayed once. This is not the case for most other wallets. It is a best practice that keeps the user safe if they accidentally make a typo when copying the seed phrase, making the wallet unrecoverable otherwise.

Remediation

An optional notification was introduced in commit e6ad1094 to verify the seed phrase.

4.2 Key derivation function may be potentially insecure

```
const iterations = 100

const encrypt = (msg: string, pass: string) => {
  try {
    const salt = CryptoJS.lib.WordArray.random(128 / 8)

    const key = CryptoJS.PBKDF2(pass, salt, {
      keySize: keySize / 32,
      iterations: iterations,
    })
```

OWASP recommends 720,000 iterations for `PBKDF2-HMAC-SHA1`, which is significantly higher than the 100 used. This may be unfeasible given the implementation in JavaScript. For more assurance, the iterations should be increased to at least 1000.

Remediation

Pontem states that this is currently not in use and will be removed in a future code cleanup.

4.3 Typographical error in result structure

```
await actions.message(
requestType,
{ id, origin },
{
    success: true,
    result: {
    address: message.address ? currentAccount?.address : false,
    application: message.application ? origin : false,
    chainId: message.chainId ? +chainId : false,
    fullMessage: messageToSign,
    message: message.message,
    none: message.nonce,
    prefix: 'APTOS',
    signature: signed,
    },
});
```

The nonce of the response message body is saved in a key called `none` instead of `nonce`.

Remediation

A fix was introduced in commit a9c748a3.

4.4 Unclear message signing flow

```
window.pontem.signMessage({
  address: true, // set true if you want include current address to
    message
  application: true, // // set true if you want include current
    application to message
  chainId: true, // set true if you want include current chain id to
    message
```

```
  message: “a message i trust”, // message like string or Uint8Array
  nonce: “random nonce”, // random nonce like string
});
```

The message signing function accepts multiple booleans for variables such as chainId, address, and application. This may mislead the users that values are uniquely positioned in the message.

At the backend, all the values are simply contacted with newlines as separators.

```
const getMessageToSign = () => {
validateMessage();
const messageBuffer = ['APTOS', `message: ${message.message}`, `nonce:
    ${message.nonce}`];
message.application && messageBuffer.push(`application: ${origin}`);
message.chainId && messageBuffer.push(`chain_id: ${+chainId}`);
message.address && messageBuffer.push(`address:
    ${currentAccount?.address || “”}`);
```

A malicious site can simply set the boolean values to false and construct a fake message by appending newlines to the `message` parameter. This can trick a user into signing a message for chainIds they were not expecting.

Remediation

The flow was overhauled in commit 9c7aa600 to be in line with the APTOS standard.

5 Audit Results

During our audit, we discovered four findings. Of these, one was medium severity, and the remaining findings were low severity. Pontem Technology Ltd. acknowledged all findings and implemented fixes.

5.1 Disclaimers

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any additional code added to the assessed project after the audit version of our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.