# Phantom Security Assessment

## Phantom

07 May 2021
Version: 1.0

Presented by:
Kudelski Security Research Team
Kudelski Security – Nagravision SA

Corporate Headquarters
Kudelski Security – Nagravision SA
Route de Genève, 22-24
1033 Cheseaux sur Lausanne
Switzerland

# DOCUMENT PROPERTIES

| | |
|---|---|
| Version: | 1.0 |
| File Name: | Phantom_Security_Review_v1.docx |
| Publication Date: | 07 May 2021 |
| Confidentiality Level: | For Public Release |
| Document Owner: | Scott Carlson |
| Document Recipient: | Phantom Project |
| Document Status: | Approved |

# TABLE OF CONTENTS

EXECUTIVE SUMMARY ..... 6
1.1 Engagement Limitations ..... 6
1.2 Engagement Analysis ..... 6
1.3 Observations ..... 7
1.4 General observations ..... 8
1.5 Issue Summary List ..... 9
2. METHODOLOGY ..... 10
2.1 Kickoff ..... 10
2.2 Ramp-up ..... 10
2.3 Review ..... 10
2.4 Reporting ..... 11
2.5 Verify ..... 12
2.6 Additional Note ..... 12
3. TECHNICAL DETAILS ..... 13
3.1 Public key saved in local storage ..... 13
3.2 Type any instead of string on validation ..... 15
3.3 Remove console log ..... 16
3.4 Potential functionality description in TODOs ..... 20
3.5 Code duplication ..... 22
3.6 Libraries with known vulnerabilities ..... 23
4. OTHER OBSERVATIONS ..... 26
4.1 Avoid eslint disable ..... 26
4.2 Unnecessary comment ..... 26
4.3 Create a constant ..... 27
4.4 Multiple initializations of the Sentry environment ..... 27
4.5 Deprecated ..... 28
4.6 Hardcoded public key ..... 28
4.7 Unnecessary declaration ..... 28
4.8 Validation of amount ..... 29
4.9 Extract global constant ..... 29
APPENDIX A: ABOUT KUDELSKI SECURITY ..... 31

APPENDIX B: DOCUMENT HISTORY ..... 32
APPENDIX C: SEVERITY RATING DEFINITIONS ..... 33

# TABLE OF FIGURES

Figure 1 Issue Severity Distribution ..... 7
Figure 2 Methodology Flow ..... 10

# EXECUTIVE SUMMARY

Kudelski Security ("Kudelski"), the cybersecurity division of the Kudelski Group, was engaged by Phantom to conduct an external security assessment in the form of a Security Assessment of the Phantom Wallet application on the Solana blockchain.

The assessment was conducted remotely by the Kudelski Security. The tests took place from March 15, 2021 to April 16, 2021, and focused on the following objectives:

1. To help the Client to better understand its security posture on the external perimeter and identify risks in its infrastructure, if any is included with the wallet
2. To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place both in the wallet itself and on the connected components
3. To identify potential issues and include improvement recommendations based on the result of our tests

This report summarizes the tests performed and findings in terms of strengths and weaknesses. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security Teams took to exploit each vulnerability, and recommendations for remediation.

## 1.1 Engagement Limitations

The architecture and code review are based on the documentation and code provided by Phantom. The code resides in a private repository at https://github.com/phantom-labs/phantom-wallet.

The reviews are based on the commit hash:

phantom-wallet: 008d3d47b9c4a88ff501f31854f2bd52165a2240

All third-party libraries were deemed out-of-scope for this review and are expected to work as designed. We have when needed based on the criticality of the dependency looked at the current state of the third-party libraries included.

## 1.2 Engagement Analysis

This engagement was comprised of a code review including reviewing how the architecture has been implemented as well as any security issues. The architecture implementation review was based on the documentation and the information retrieved through communication between the Epsilon team and the Kudelski Security team. The implementation review concluded that the team and code are mature, with no serious remaining issues.

The code review was conducted by the Kudelski Security team on the code provided by Epsilon, in the form of a Github repository. The code review focused on the handling of secure and private information handling in the code.

As a result of our work, we identified **0 High**, **1 Medium**, **13 Low**, and **23 Informational** findings.

The only issues found in the code were either LOW or INFORMATIONAL findings. This shows that the functional level of the application is good and that the risk profile of the application is low. The Medium finding is related to the risk on the local machine on which the wallet resides.

The findings referred to in the Findings section are such as they would improve the functionality and performance of the application and secure it further.

There is also a list of audited packages and dependencies at the end of the report that should be handled swiftly. As these are 3rd party libraries, we have not reviewed them apart from scanning them for known errors and/or vulnerabilities.

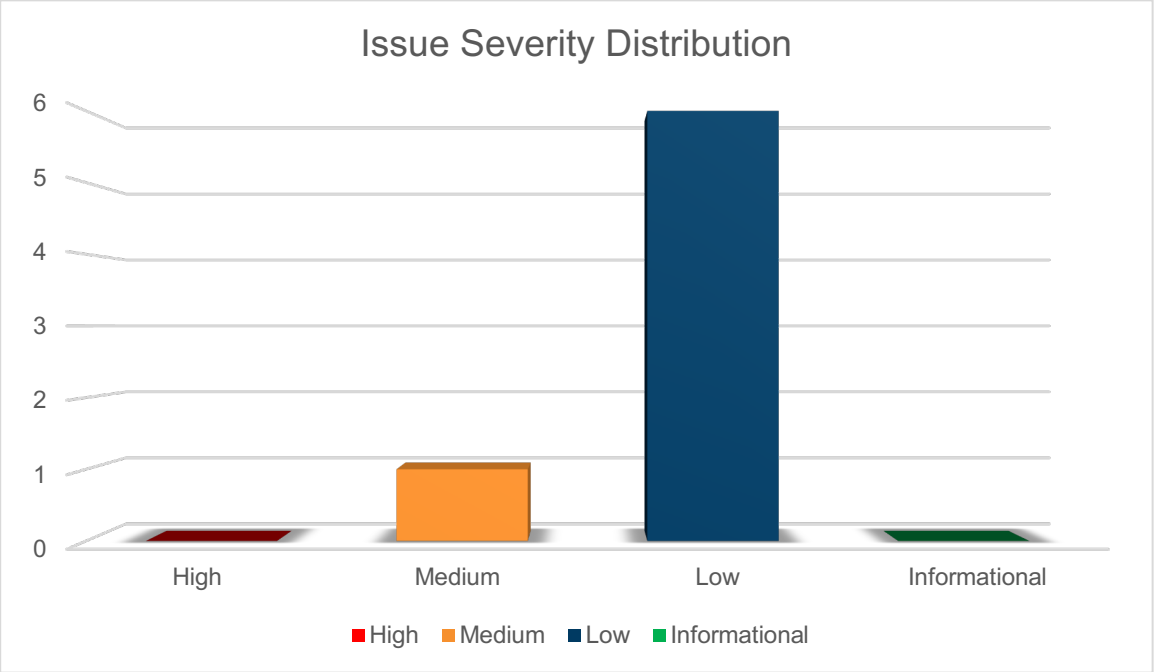


Figure 1 Issue Severity Distribution

## 1.3 Observations

## 1.4 General observations

The code base is written in relatively consistent style with some exceptions. The component structure should be improved as currently most of the components are in the same folder and the generally recommended practice is to separate different components in folders putting the related components in the same folder in individual files. Files with multiple component exports should be separated so that each file exports a single component.

Multiple files export resources that are not used anywhere. Reduce the export statements to the necessary amount of functionality and types to safeguard potential attack surface area.

When using environment variables, it is recommended to add declarations for NodeJS ProecessEnv. Consider adding the following to `src/globals.d.ts`

```
declare namespace NodeJS {
  export interface ProcessEnv {
    NODE_ENV: string;
    POSTHOG_KEY: string;
    GIT_SHA: string;
  }
}
```

Console logging of errors and other information should be avoided and replaced with information sent to the logging framework used in the solution (Sentry). Console log must not be used in production as it could display potentially sensitive information. All error handling should be done internally, and appropriate messages should be displayed to the users when necessary. A list of the log code snippets is provided in the findings section.

It is recommended to use type import/export when possible, to improve performance and security. *"`import type` only imports declarations to be used for type annotations and declarations.* ***It always gets fully erased, so there's no remnant of it at runtime.*** *Similarly, export type only provides an export that can be used for type contexts and is also erased from TypeScript's output."* Type-Only Imports and Export

There are multiple TODOs in the code with variable importance and impact. It is critical to evaluate and implement the appropriate functionality for the more impactful notes and properly describe the less critical issues as separate tasks to be planned as part of the future development of the code. TODOs in the production code could lead to potential exploits and vulnerabilities providing internal information for the workings of the solution to malicious parties. List of the individual TODOs will follow in the list of individual findings *<reference>*.

For readability it is recommend adding a new line between variable declarations, function declarations and functionality.

## 1.5 Issue Summary List

| ID | SEVERITY | FINDING |
|---|---|---|
| KS-PHANTOM-01 | Medium | Public key saved in local storage |
| KS-PHANTOM-02 | Low | Type any instead of string on validation |
| KS-PHANTOM-03 | Low | Remove console log |
| KS-PHANTOM-04 | Low | Potential functionality description in TODO |
| KS-PHANTOM-05 | Low | Code duplication |
| KS-PHANTOM-06 | Low | Libraries with known weaknesses |

# 2. METHODOLOGY

Kudelski Security uses the following high-level methodology when approaching engagements. They are broken up into the following phases.


Figure 2 Methodology Flow

## 2.1 Kickoff

The project is kicked off when the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It is an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

## 2.2 Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

## 2.3 Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review
2. Review of the code written for the project.

3. Compliance of the code with the provided technical documentation.

The review for this project was performed using manual methods and tools, utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

### Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general list and not comprehensive, meant only to give an understanding of the issues we are looking for.

### Cryptography

We analyzed the cryptographic primitives and components as well as their implementation. We checked in particular:

- Matching of the proper cryptographic primitives to the desired cryptographic functionality needed
- Security level of cryptographic primitives and their respective parameters (key lengths, etc.)
- Safety of the randomness generation in general as well as in the case of failure
- Safety of key management
- Assessment of proper security definitions and compliance to use cases
- Checking for known vulnerabilities in the primitives used

### Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

## 2.4 Reporting

Kudelski Security delivers a preliminary report in PDF format that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project as a whole.

We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We not only report security issues identified but also informational findings for improvement categorized into several buckets:

- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps, that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

## 2.5 Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes withing a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

## 2.6 Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessment the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. These is a solid baseline for severity determination. Information about the severity ratings can be found in **Appendix C** of this document.

# 3. TECHNICAL DETAILS

This section contains the technical details of our findings as well as recommendations for improvement.

## 3.1 Public key saved in local storage

Finding ID: KS-PHANTOM-01

Severity: **Medium**

Status: **Risk Accepted**

**Description**

Tampering risk while reading persistence.

**Proof of Issue**

**Filename:** src/app/contexts/accounts.tsx

Storing of account information and index in the local storage

```
Row | Code
45  | export const AccountsProvider: React.FC = ({ children }) => {
46  |   const [selectedAccountIndex, isLoadingSelectedAccountIndex,
47  | setSelectedAccountIndexStorage] = useStorage<number>(
48  |     StorageKeys.SelectedAccountIndex,
49  |     0,
50  |   );
51  |   const [accountMetas, isLoadingAccountMetas, setAccountMetas] =
52  | useStorage<AccountMeta[]>(
53  |     StorageKeys.AccountMetas,
54  |     [],
55  |   );
```

**Filename:** src/app/contexts/accounts.tsx

Initialization of the component with the stored account meta index and array of account metas

```
Row | Code
202 | return (
203 |   <AccountsContext.Provider
204 |     value={{
205 |       selectedAccountClient,
206 |       selectedAccountIndex,
207 |       accountMetas,
208 |       setSelectedAccountIndex,
209 |       addSeedAccount,
210 |       addPrivateKeyAccount,
211 |       setAccountMetas,
212 |       setSelectedAccountName,
```

```
213         removeSelectedAccount,
214       }}
215     >
216       {children}
217     </AccountsContext.Provider>
128   );
```

Loading of the selected account in the send funds popup modal

**Filename:** src/app/components/deposit_asset.tsx

```
Row  Code
79   const { accountMeta, accountClient } = useSelectedAccount();
…    …
131  <AddressContainer>
132    <AccountContainer>
133      <Account {...accountMeta} showParens={true} />
134    </AccountContainer>
135    <Copy copyText={accountMeta.publicKey.toString() ?? ""} />
136  </AddressContainer>
```

## Severity and Impact Summary

Wallet address/public key is persisted in local storage, and it's read while interacting with the application. The wallet address is available in storage even while the wallet is locked. This makes it possible for a malicious user to replace the wallet address. This is particularly dangerous while coping the address from the deposit UI. The user will have it difficult to notice that its address has been tampered with and will use that address for SOL transfers resulting in the loss of its SOL.

In general, this is a problem with many local wallet applications on local machines, making it very important to consider compensating controls within the wallet that expose tampering.

## Recommendation

The users public key should be validated every time is retrieved from local storage to make sure it hasn't been tampered with. This can be done by generating the public key from the user's private key and match it with the one in local storage. If there is a mismatch, displaying a message or failing so that the user must take action with the mismatch.

## 3.2 Type any instead of string on validation

Finding ID: KS-PHANTOM-02

Severity: **Low**

Status: **Open**

### Description

Validation of privateKey was `any` instead of `string` when passed to a `decodeSecretKey` which was expecting `string` parameter.

Directly displaying a technical error message to the user instead of user friednly information makes it difficult for the user to understand what is wrong with the validation.

### Proof of Issue

**Filename:** src/app/components/add_account/import_account.tsx

```
Row | Code
100 | validate: (privateKey: any) => {
101 |   try {
102 |     decodeSecretKey(privateKey);
103 |     return true;
104 |   } catch (err) {
105 |     return err.message;
106 |   }
107 | },
```

### Severity and Impact Summary

If the passed privateKey is made to be something else than the expected String, the called function could be made to not validate the key as a valid key.

If the error message contains any private data that has been passed on to the function, this could be used as a way of information gathering for a more sophisticated attack.

### Recommendation

Change the validation type to string and create a way to handle the error is such a way that it is useful for the application. An example could be to provide GUI guidance based on the error to the user.

## 3.3 Remove console log

Finding ID: KS-PHANTOM-03

Severity: **Low**

Status: **Open**

### Description

Possible private information disclosure when using the console logging facility.

### Proof of Issue

**Filename:** src/common/account_client/parse_instructions.ts

```
Row | Code
90  | } catch (err) {
91  |   console.error(err);
92  | }
…   | …
135 | } catch (e) {
136 |   console.error("Error loading market: " + e.message);
137 | }
```

**Filename:** src/app/onboarding.tsx

```
Row | Code
164 | } catch (err) {
165 |   console.error(err);
166 | }
…   | …
390 | } else {
391 |   console.error(err);
392 | }
```

**Filename:** src/background/content_script_connection_controller.ts

```
Row | Code
65  | if (err) {
66  |   console.error(err);
67  | }
```

**Filename:** src/app/components/change_lock_timer.tsx

```
Row | Code
58  | } catch (err) {
59  |   console.error(err);
60  | }
```

**Filename:** src/app/components/change_password.tsx

```
Row | Code
54  | } catch (err) {
55  |   console.error(err);
56  |   if (err.message && err.message.includes("Incorrect password")) {
```

**Filename:** src/app/components/deposit_asset.tsx

| Row | Code |
|---|---|
| 92 | `} catch (err) {` |
| 93 | `  analytics.capture("tokenAccountCreateFailure", { asset });` |
| 94 | `  console.error(err);` |
| 95 | `  if (accountMeta.type === AccountType.Ledger) {` |
| 96 | `  onClose();` |
| 97 | `  }` |
| 98 | `}` |
| … | … |
| 107 | `} catch (err) {` |
| 108 | `  analytics.capture("tokenAccountCreateFailure", { asset });` |
| 109 | `  console.error(err);` |
| 110 | `}` |

**Filename:** src/app/components/export_secret.tsx

| Row | Code |
|---|---|
| 98 | `} catch (err) {` |
| 99 | `  console.error(err);` |
| 100 | `  if (err.message && err.message.includes("Incorrect password")) {` |

**Filename:** src/app/components/unlock.tsx

| Row | Code |
|---|---|
| 66 | `} catch (err) {` |
| 67 | `  console.error(err);` |
| 68 | `  if (err.message && err.message.includes("Incorrect password")) {` |

**Filename:** src/content_script/content_script.ts

| Row | Code |
|---|---|
| 113 | `} catch (e) {` |
| 114 | `  console.error("PHANTOM: injection failed.", e);` |
| 115 | `}` |
| … | … |
| 124 | `function logStreamDisconnectWarning(remoteLabel: string, err?: Error) {` |
| 125 | `  let warningMsg = `PhantomContentscript - lost connection to ${remoteLabel}`;` |
| 126 | `  if (err) {` |
| 127 | `  warningMsg += `\n${err.stack}`;` |
| 128 | `  }` |
| 129 | `  console.warn(warningMsg);` |
| 130 | `}` |

**Filename:** src/content_script/rpc_inpage_provider.ts

| Row | Code |
|---|---|
| 77 | `private _handleStreamDisconnect = (err?: Error) => {` |
| 78 | `  let warningMsg = `PhantomInpage - Lost connection to contentscript.`;` |
| 79 | `  if (err && err.stack) {` |
| 80 | `  warningMsg += `\n${err.stack}`;` |
| 81 | `  }` |
| 82 | `  console.warn(warningMsg);` |
| 83 | `};` |

**Filename:** src/app/contexts/accounts.tsx

| Row | Code |
|---|---|
| 160 | } catch (err) { |
| 161 | console.error(err); |
| 162 | addSeedAccount(); |
| 163 | } |

**Filename:** src/app/contexts/assets.tsx

| Row | Code |
|---|---|
| 99 | } catch (err) { |
| 100 | console.error(err); |
| 101 | } |
| … | … |
| 126 | } catch (err) { |
| 127 | console.error(err); |
| 128 | } finally { |
| … | … |
| 144 | } catch (err) { |
| 145 | console.error(err); |
| 146 | } finally { |

**Filename:** src/app/contexts/blockchain.tsx

| Row | Code |
|---|---|
| 30 | } catch (err) { |
| 31 | console.error(err); |
| 32 | } |

**Filename:** src/app/hooks/useStorage.ts

| Row | Code |
|---|---|
| 24 | } catch (err) { |
| 25 | console.error(err); |
| 26 | } finally { |
| … | … |
| 37 | } catch (err) { |
| 38 | console.error(err); |
| 39 | } |

**Filename:** src/app/components/send_asset/send_confirmation.tsx

| Row | Code |
|---|---|
| 119 | } catch (err) { |
| 120 | console.error(err); |
| 121 | analytics.capture("sendAssetFailure", { asset }); |
| 122 | setSendTransferError(err); |
| 123 | setStep(SendConfirmationStep.Confirmed); |
| 124 | } finally { |
| … | … |
| 158 | } catch (err) { |
| 159 | console.error(err); |
| 160 | analytics.capture("sendAssetFailure", { asset }); |
| 161 | setSendTransferError(err); |
| 162 | setStep(SendConfirmationStep.Confirmed); |
| 163 | } |

**Filename:** src/common/utils/analytics.ts

```
Row | Code
128 | capture: (event: AnalyticsEvent, payload?: AnalyticsPayload) => {
129 |   // TODO(fragosti): Add debug levels.
130 |   console.log("ANALYTICS:", event, parsePayload(payload));
131 | },
```

**Filename:** src/common/utils/middleware.ts

```
Row | Code
16  | const { error } = res;
17  | if (!error) {
18  |   return done();
19  | }
20  | console.error(`Phantom - RPC Error: ${error.message}`, error);
21  | return done();
…   | …
37  | if (res.error) {
38  |   console.error("Error in RPC response:\n", res);
39  | }
40  | console.info(`RPC (${origin}):`, req, "->", res);
41  | cb();
```

**Filename:** src/common/utils/storage_utils.ts

```
Row | Code
65  | } catch (err) {
66  |   console.error(err);
67  |   // Better to unset than to have it potentially be set forever.
68  |   removeExtensionStorageValue(key);
69  |   reject(err);
70  |   return;
71  | }
```

## Severity and Impact Summary

By using the console logging facility, there is a real threat of private information leaks to parts of the browser and therefore also to unsecure parts. As the logging also only is visible locally there is no use when going in production as the developer have no visibility.

## Recommendation

Logging on the client side may expose information that should be kept secure. If any error information should be communicated to the user, it should be displayed as proper error/warning messages not in console log. Console log should be removed and replaced with Sentry logging for errors and messages that need to be looked over and there is no risk for information disclosure.

## 3.4 Potential functionality description in TODOs

Finding ID: KS-PHANTOM-04

Severity: **Low**

Status: **Open**

### Description

Multiple TODOs describing desired implementation logic and bug fixes should be transferred to tasks or tickets for the development team to plan and implement according to their priorities.

### Proof of Issue

**Filename:** src/common/account_client/index.ts

```
Row  Code
91   // TODO(bmillman): when source is public key, be "smart" and reroute
     instead of throwing
…    …
150  // TODO(bmillman): potentially move `additionalSignerAccounts` to be
     returned by `initializeTokenAccount`
…    …
186  // TODO(bmillman): open up a PR against @solana/web3.js to add `until` as
     an option for the getConfirmedSignaturesForAddress2 JSON RPC request
…    …
293  // TODO(bmillman): potentially move `additionalSignerAccounts` to be
     returned by `mintToken`
```

**Filename:** src/background/background.ts

```
Row  Code
38   // TODO(bmillman): figure out if we need to retain references to these
39   new ContentScriptConnectionController(remotePort, url, tabId);
```

**Filename:** src/app/components/asset_detail.tsx

```
Row  Code
47   // TODO(bmillman): fix a bug where SOL tx history is equivalent to all
     recent tx history
```

**Filename:** src/common/utils/solana_utils.ts

```
Row  Code
115  // TODO: fix
116  throw new Error();
```

**Filename:** src/app/components/asset_detail.tsx

| Row | Code |
| --- | --- |
| 47 | `// TODO(bmillman): fix a bug where SOL tx history is equivalent to all recent tx history` |

**Filename:** src/app/contexts/transaction_history.tsx

| Row | Code |
| --- | --- |
| 98 | `// TODO(bmillman): small edge case here, if there have been more than 100 txs in the last 10s, we may lose some history` |

**Filename:** src/app/components/ledger_action.tsx

| Row | Code |
| --- | --- |
| 65 | `if (ledgerTransportState === LedgerTransportState.Connected) {` |
| 66 | `  // TODO(bmillman): handle error here` |
| 67 | `  if (transport) {` |
| 68 | `  ledgerAction(transport);` |
| 69 | `  }` |
| 70 | `}` |

**Filename:** src/app/contexts/background_connection.tsx

| Row | Code |
| --- | --- |
| 30 | `// TODO(bmillman): add some hardening here to make sure we only get messages we can handle` |

**Filename:** src/app/contexts/hardware_wallet.tsx

| Row | Code |
| --- | --- |
| 63 | `// TODO(bmillman): do we need this delay?` |
| 64 | `await delayAsync(1500);` |
| 65 | `// TODO(bmillman): fix typing` |

## Severity and Impact Summary

Describing missing logic could be used as a part of a sophisticated attack where missing functionality could be used to crash or extract information from the application.

## Recommendation

Move the comments about missing logic to the task management system to be tracked and prioritized.

## 3.5 Code duplication

Finding ID: KS-PHANTOM-05

Severity: **Low**

Status: **Open**

Description

Duplication of the same code in the same file. Both methods are returning a promise which results in async execution regardless of the definition.

**Proof of Issue**

**Filename:** src/common/utils/promise_utils.ts

| Row | Code |
|---|---|
| 2 | `return new Promise(resolve => setTimeout(resolve, ms));` |
| … | … |
| 17 | `return new Promise(resolve => setTimeout(resolve, delayMs));` |

**Severity and Impact Summary**

Duplicated code is used as a code maturity metric in the industry to point out how maintainable the code base is. It is also a possible entrypoint for new bugs as code duplication leads to mistakes when updating/rewriting the codebase.

**Recommendation**

Replace the duplicated code with functions to provide the necessary functionality.

## 3.6 Libraries with known vulnerabilities

Finding ID: KS-PHANTOM-06

Severity: **Low**

**Description**

The result of a security audit of the 3rd party libraries used by the Phantom Wallet Application is the following. It could be good to note that there are some dependencies that needs to be updated to get the latest code with as many bug fixes as possible included.

**Proof of Issue**

| | |
|---|---|
| === npm audit security report === | |
| # Run<br>`npm update immer --depth 1`<br>to resolve 1 vulnerability | |
| **High** | Prototype Pollution |
| Package | immer |
| Dependency of | immer |
| Path | immer |
| More info | https://npmjs.com/advisories/1603 |
| # Run<br>`npm update elliptic --depth 6`<br>to resolve 7 vulnerabilities | |
| **Moderate** | Use of a Broken or Risky Cryptographic Algorithm |
| Package | elliptic |
| Dependency of | @project-serum/serum |
| Path | @project-serum/serum > @solana/web3.js > secp256k1 > elliptic |
| More info | https://npmjs.com/advisories/1648 |
| **Moderate** | Use of a Broken or Risky Cryptographic Algorithm |
| Package | elliptic |
| Dependency of | @solana/web3.js |
| Path | @solana/web3.js > secp256k1 > elliptic |
| More info | https://npmjs.com/advisories/1648 |
| **Moderate** | Use of a Broken or Risky Cryptographic Algorithm |

| | |
|---|---|
| Package | elliptic |
| Dependency of | bip32 |
| Path | bip32 > tiny-secp256k1 > elliptic |
| More info | https://npmjs.com/advisories/1648 |
| **Moderate** | Use of a Broken or Risky Cryptographic Algorithm |
| Package | elliptic |
| Dependency of | parcel-bundler [dev] |
| Path | parcel-bundler > node-libs-browser > crypto-browserify > browserify-sign > elliptic |
| More info | https://npmjs.com/advisories/1648 |
| **Moderate** | Use of a Broken or Risky Cryptographic Algorithm |
| Package | elliptic |
| Dependency of | parcel-plugin-git-sha [dev] |
| Path | parcel-plugin-git-sha > parcel-bundler > node-libs-browser > crypto-browserify > browserify-sign > elliptic |
| More info | https://npmjs.com/advisories/1648 |
| **Moderate** | Use of a Broken or Risky Cryptographic Algorithm |
| Package | elliptic |
| Dependency of | parcel-bundler [dev] |
| Path | parcel-bundler > node-libs-browser > crypto-browserify > create-ecdh > elliptic |
| More info | https://npmjs.com/advisories/1648 |
| **Moderate** | Use of a Broken or Risky Cryptographic Algorithm |
| Package | elliptic |
| Dependency of | parcel-plugin-git-sha [dev] |
| Path | parcel-plugin-git-sha > parcel-bundler > node-libs-browser > crypto-browserify > create-ecdh > elliptic |
| More info | https://npmjs.com/advisories/1648 |
| Manual Review<br>Some vulnerabilities require your attention to resolve<br>Visit https://go.npm.me/audit-guide for additional guidance | |
| **High** | Prototype Pollution in node-forge |

| | |
|---|---|
| Package | node-forge |
| Patched in | >= 0.10.0 |
| Dependency of | parcel-bundler [dev] |
| Path | parcel-bundler > node-forge |
| More info | https://npmjs.com/advisories/1561 |
| **High** | Prototype Pollution in node-forge |
| Package | node-forge |
| Patched in | >= 0.10.0 |
| Dependency of | parcel-plugin-git-sha [dev] |
| Path | parcel-plugin-git-sha > parcel-bundler > node-forge |
| More info | https://npmjs.com/advisories/1561 |
| found 10 vulnerabilities (7 **moderate**, 3 **high**) in 1291 scanned packages<br>run `npm audit fix` to fix 8 of them.<br>2 vulnerabilities require manual review. See the full report for details. | |

**Recommendation**

Go through the list of findings and update the outdated versions. After this has been done, this actions should be included in the CI/CD scripts for automated build management of the code.

# 4. OTHER OBSERVATIONS

## 4.1 Avoid eslint disable

Finding ID: KS-PHANTOM-26

Severity: **Informational**

**Description**

No need for eslint-disable-next-line.

**Proof of Issue**

**Filename:** src/common/utils/ledger_utils.ts

| Row | Code |
|---|---|
| 67 | `// eslint-disable-next-line` |

**Filename:** src/app/onboarding.tsx

| Row | Code |
|---|---|
| 256 | `// eslint-disable-next-line @typescript-eslint/no-unused-vars` |

**Recommendation**

Avoid disabling eslint/tslint.

## 4.2 Unnecessary comment

Finding ID: KS-PHANTOM-27

Severity: **Informational**

**Description**

The referenced issue is already fixed.

**Proof of Issue**

**Filename:** src/common/utils/wallet_provider_utils.ts

| Row | Code |
|---|---|
| 76 | `// @FIXME: https://github.com/project-serum/spl-token-wallet/issues/59` |

**Recommendation**

Remove the comment

## 4.3 Create a constant

Finding ID: KS-PHANTOM-29

Severity: **Informational**

**Description**

Export global constant in src/common/constants to be reused.

**Proof of Issue**

**Filename:** src/app/contexts/assets.tsx

```
Row | Code
49  | const SOLANA_ID = "solana";
```

## 4.4 Multiple initializations of the Sentry environment

Finding ID: KS-PHANTOM-30

Severity: **Informational**

**Description**

The initialization could be already done by onboarding.tsx, connect_hardware.tsx or popup.tsx

**Proof of Issue**

**Filename:** src/app/notification.tsx

```
Row | Code
22  | initSentry();
```

## 4.5 Deprecated

Finding ID: KS-PHANTOM-31

Severity: **Informational**

**Description**

The call is deprecated. Replace with the appropriate new version.

**Proof of Issue**

**Filename:** src/common/account_client/index.ts

| Row | Code |
| --- | --- |
| 234 | `transaction.setSigners(` |

## 4.6 Hardcoded public key

Finding ID: KS-PHANTOM-32

Severity: **Informational**

**Description**

Public key is hardcoded here. It appears to be invalid and should perhaps be removed. If this key is used, it seems like a weak occurrence.

**Proof of Issue**

**Filename:** src/common/account_client/instructions/createAssociatedTokenAccount.ts

| Row | Code |
| --- | --- |
| 10 | `const systemProgramId = new PublicKey("11111111111111111111111111111111");` |

## 4.7 Unnecessary declaration

Finding ID: KS-PHANTOM-33

Severity: **Informational**

**Description**

Unnecessary declaration and usage of type `any`. Use single => instead.

**Proof of Issue**

**Filename:** src/app/components/notification/application_approval.tsx

| Row | Code |
|---|---|
| 73 | `const handleCheckboxChange = (event: any) => {` |
| 74 | `  setAutoApprove(event.target.checked);` |
| 75 | `};` |
| … | `...` |
| 89 | `  <Checkbox checked={autoApprove} onChange={handleCheckboxChange} />` |

**Recommendation**

Replace with this snippet

| Row | Code |
|---|---|
| 89 | `<Checkbox checked={autoApprove} onChange={e => setAutoApprove(e.target.checked)} />` |

## 4.8 Validation of amount

Finding ID: KS-PHANTOM-34

Severity: **Informational**

**Description**

Replace `any` with `string`. Unclear behavior when the parse fails and when the amount is higher. This could lead to problems and unexpected results when validating the amount.

**Proof of Issue**

**Filename:** src/app/components/send_asset/send_form.tsx

| Row | Code |
|---|---|
| 149 | `validate: (amount: any) => {` |
| 150 | `  return parseFloat(amount) <= selectedAssetBalance;` |
| 151 | `},` |

## 4.9 Extract global constant

Finding ID: KS-PHANTOM-35

Severity: **Informational**

**Description**

Duplicate declaration of constant should be included in a global list of constants.

### Proof of Issue

**Filename:** src/app/components/notification/connect_request.tsx

| Row | Code |
|---|---|
| 14 | `const DEFAULT_TRUSTED_APPLICATIONS: TrustedApplications = {};` |

**Filename:** src/app/components/notification/sign_transaction_request.tsx

| Row | Code |
|---|---|
| 24 | `const DEFAULT_TRUSTED_APPLICATIONS: TrustedApplications = {};` |

# APPENDIX A: ABOUT KUDELSKI SECURITY

Kudelski Security is an innovative, independent Swiss provider of tailored cyber and media security solutions to enterprises and public sector institutions. Our team of security experts delivers end-to-end consulting, technology, managed services, and threat intelligence to help organizations build and run successful security programs. Our global reach and cyber solutions focus is reinforced by key international partnerships.

Kudelski Security is a division of Kudelski Group. For more information, please visit https://www.kudelskisecurity.com.

**Kudelski Security**

route de Genève, 22-24

1033 Cheseaux-sur-Lausanne

Switzerland

**Kudelski Security**

5090 North 40th Street

Suite 450

Phoenix, Arizona 85018

## APPENDIX B: DOCUMENT HISTORY

| VERSION | STATUS | DATE | AUTHOR | COMMENTS |
|---|---|---|---|---|
| 0.1 | Draft | 22 April 2021 | Krum Valkov | First Draft |
| 0.2 | Draft | 5 May 2021 | Ken Toler | Second Draft |
| 1.0 | Final | 7 May 2021 | Scott Carlson | Final Draft |

| REVIEWER | POSITION | DATE | VERSION | COMMENTS |
|---|---|---|---|---|
| Mikael Björn | Tech Lead | 23 April 2021 | 0.1 | Draft Internal QA |
| | | Select the Date | | |
| | | Select the Date | | |

| APPROVER | POSITION | DATE | VERSION | COMMENTS |
|---|---|---|---|---|
| | | Select the Date | | |
| | | Select the Date | | |
| | | Select the Date | | |

## APPENDIX C: SEVERITY RATING DEFINITIONS

Kudelski Security uses a custom approach when determining criticality of identified issues. This is meant to be simple and fast, providing customers with a quick at a glance view of the risk an issue poses to the system. As with anything risk related, these findings are situational. We consider multiple factors when assigning a severity level to an identified vulnerability. A few of these include:

- Impact of exploitation
- Ease of exploitation
- Likelihood of attack
- Exposure of attack surface
- Number of instances of identified vulnerability
- Availability of tools and exploits

| SEVERITY | DEFINITION |
|---|---|
| **High** | The identified issue may be directly exploitable causing an immediate negative impact on the users, data, and availability of the system for multiple users. |
| **Medium** | The identified issue is not directly exploitable but combined with other vulnerabilities may allow for exploitation of the system or exploitation may affect singular users. These findings may also increase in severity in the future as techniques evolve. |
| **Low** | The identified issue is not directly exploitable but raises the attack surface of the system. This may be through leaking information that an attacker can use to increase the accuracy of their attacks. |
| **Informational** | Informational findings are best practice steps that can be used to harden the application and improve processes. |