



Wigwam – Browser Extension Wallet

WebApp Pentest

Prepared by: Halborn

Date of Engagement: December 6th, 2023 – December 18th, 2023

Visit: Halborn.com

DOCUMENT REVISION HISTORY	5
CONTACTS	6
1 EXECUTIVE OVERVIEW	7
1.1 INTRODUCTION	8
1.2 ASSESSMENT SUMMARY	9
1.3 SCOPE	10
1.4 TEST APPROACH & METHODOLOGY	11
RISK METHODOLOGY	11
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	13
3 FINDINGS & TECH DETAILS	14
3.1 (HAL-01) UNENCRYPTED MNEMONIC PHRASE IN-MEMORY - HIGH	16
Description	16
Proof of Concept	17
CVSS Vector	18
Risk Level	18
Recommendation	18
References	19
Remediation Plan	19
3.2 (HAL-02) UNENCRYPTED USER PASSWORD IN-MEMORY - HIGH	20
Description	20
Proof of concept	20
CVSS Vector	24
Risk Level	24
Recommendation	24

Remediation Plan	25
3.3 (HAL-03) PLAIN TEXT CONNECTIONS OVER HTTP - MEDIUM	26
Description	26
Code Location	26
CVSS Vector	28
Risk Level	28
Recommendation	28
Remediation Plan	28
3.4 (HAL-04) LACK OF USER INPUT SANITATION - MEDIUM	29
Description	29
Proof of Concept	29
CVSS Vector	32
Risk Level	32
Recommendation	32
Remediation Plan	33
3.5 (HAL-05) PACKAGES WITH KNOWN VULNERABILITIES - MEDIUM	34
Description	34
CVSS Vector	35
Risk Level	35
Recommendation	35
Remediation Plan	35
3.6 (HAL-06) EXCESSIVE TIME FOR WALLET AUTO-LOCK - MEDIUM	36
Description	36
Code Location	36
CVSS Vector	36
Risk Level	37

Recommendation	37
Remediation Plan	37
3.7 (HAL-07) INSECURE AUTHENTICATION METHODS - LOW	38
Description	38
Proof of Concept	38
CVSS Vector	38
Risk Level	38
Recommendation	39
Remediation Plan	39
3.8 (HAL-08) LACK OF DEFAULT CLAUSE ON SWITCH STATEMENT - LOW	40
Description	40
Source Code Snippets	40
CVSS Vector	55
Risk Level	56
Recommendation	56
Remediation Plan	56
3.9 (HAL-09) OLD PASSWORD RE-USAGE - LOW	57
Description	57
Proof of Concept	57
CVSS Vector	58
Risk Level	58
Recommendation	58
Remediation Plan	59
3.10 (HAL-10) BROKEN LINKS - INFORMATIONAL	60
Description	60
Proof of Concept	60

	CVSS Vector	61
	Risk Level	61
	Recommendation	61
	Remediation Plan	61
4	PERFORMED TESTS	62
4.1	STATIC ANALYSIS	63
	Description	63
	NodeJSScan	63
	SonarQube	65

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	12/15/2023
0.2	Draft Review	12/22/2023
0.3	Draft Review	12/22/2023
1.0	Remediation Plan	01/10/2024
1.1	Remediation Plan Review	01/10/2024
1.2	Remediation Plan Review	01/11/2024
1.3	Remediation Plan Modification	01/12/2024
1.4	Remediation Plan Review	01/12/2024
1.5	Remediation Plan Review	01/13/2024
1.6	Remediation Plan Modification	01/15/2024

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
David Manzano	Halborn	David.Manzano@halborn.com
Erlantz Saenz	Halborn	Erlantz.Saenz@halborn.com



EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Wigwam engaged Halborn to conduct a security assessment on their web application, beginning on December 6th, 2023 and ending on December 18th, 2023 . The security assessment was scoped to the [Wigwam Wallet](#) browser extension. Halborn was provided access to `dev` branch of the [GitHub repository of the Wigwam Wallet](#) to conduct a security testing in the application and reporting the findings at the end of the engagement.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full-time security engineer to verify the security of the **Wigwam Wallet** application. The security engineer is a penetration testing expert with advanced knowledge in web, recon, discovery & infrastructure penetration testing and blockchain and smart-contracts security.

The purpose of this assessment is to:

- Improve the security of the application by testing it both as white and black-box approaches
- Identify potential security issues that could be affecting the web application

In summary Halborn did not identify any critical issues but found some security risks, including **two HIGH**, **four MEDIUM**, and **three LOW** issues. It was possible to leak the mnemonic phrase from the memory dump, as well as the users' password under different scenarios.

Moreover, it was detected that Wigwam wallet was using plaintext connections over HTTP in several snippets of the source code. Other than that, the auto-lock period of time for automatically locking the wallet was set up to a very high value of time.

Some vulnerable dependencies were being used by Wigwam wallet.

Finally, it was possible to set up the same old password as the new one, allowing the users to re-use the same password.

Finally, the Wigwam team successfully addressed all the above-mentioned issues.

1.3 SCOPE

Wigwam Wallet browser extension from Chrome Store:

- Chrome Store URL (version 1.7.2 and the earlier)

Wigwam Wallet source code:

- URL: GitHub repository of the WigWam Wallet
 - Environment: dev branch.
 - Commit: 31618e9a64ab8d584f2997743cca3f17b745cbe5
 - GitHub commit files: <https://github.com/wigwamapp/local-wigwam/tree/31618e9a64ab8d584f2997743cca3f17b745cbe5>

1.4 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the assessment:

- Mapping Application Content and Functionality
- Technology stack-specific vulnerabilities and Code Assessment
- Known vulnerabilities in 3rd party / OS dependencies
- Application Logic Flaws
- Authentication / Authorization flaws
- Input Handling
- Fuzzing of all input parameters
- Testing for different types of sensitive information leakages: memory, clipboard, . . .
- Test for Injection (SQL/JSON/HTML/JS/Command/Directories. . .)
- Brute Force Attempts
- API testing and rate-limiting testing
- Perform static analysis on code
- Ensure that coding best practices are being followed by Wigwam team
- Technology stack-specific vulnerabilities and code assessment
- Identify other potential vulnerabilities that may pose a risk to Wigwam

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities.

The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

- 5 - Almost certain an incident will occur.
- 4 - High probability of an incident occurring.
- 3 - Potential of a security incident in the long term.
- 2 - Low probability of an incident occurring.
- 1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

- 5 - May cause devastating and unrecoverable impact or loss.
- 4 - May cause a significant level of impact or loss.
- 3 - May cause a partial impact or loss to many.
- 2 - May cause temporary impact or loss.
- 1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
----------	------	--------	-----	---------------

- 10 - CRITICAL
- 9 - 8 - HIGH
- 7 - 6 - MEDIUM
- 5 - 4 - LOW
- 3 - 1 - VERY LOW AND INFORMATIONAL

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	2	4	3	1

LIKELIHOOD

IMPACT

			(HAL-01)	
	(HAL-03)		(HAL-02)	
(HAL-07)		(HAL-04) (HAL-05) (HAL-06)		
(HAL-10)	(HAL-08) (HAL-09)			

SECURITY ANALYSIS	RISK LEVEL	REMEDATION DATE
(HAL-01) UNENCRYPTED MNEMONIC PHRASE IN-MEMORY	High	SOLVED - 01/09/2024
(HAL-02) UNENCRYPTED USER PASSWORD IN-MEMORY	High	SOLVED - 01/09/2024
(HAL-03) PLAIN TEXT CONNECTIONS OVER HTTP	Medium	SOLVED - 01/09/2024
(HAL-04) LACK OF USER INPUT SANITATION	Medium	SOLVED - 01/09/2024
(HAL-05) PACKAGES WITH KNOWN VULNERABILITIES	Medium	SOLVED - 01/10/2024
(HAL-06) EXCESSIVE TIME FOR WALLET AUTO-LOCK	Medium	SOLVED - 01/09/2024
(HAL-07) INSECURE AUTHENTICATION METHODS	Low	SOLVED - 01/09/2024
(HAL-08) LACK OF DEFAULT CLAUSE ON SWITCH STATEMENT	Low	SOLVED - 01/09/2024
(HAL-09) OLD PASSWORD RE-USAGE	Low	SOLVED - 01/09/2024
(HAL-10) BROKEN LINKS	Informational	ACKNOWLEDGED



FINDINGS & TECH DETAILS

3.1 (HAL-01) UNENCRYPTED MNEMONIC PHRASE IN-MEMORY - HIGH

Description:

The **Mnemonic Phrase** of the wallet remained unencrypted in memory. As a result, an attacker who compromised the user's machine could exfiltrate and steal the Mnemonic Phrase.

It was possible to retrieve the Mnemonic Phrase from memory in the following scenario:

- When creating the wallet, it was possible to dump the mnemonic from memory.

It was NOT possible to retrieve the Mnemonic Phrase from memory in the rest of the scenarios, including but not being limited to:

- Revealing the mnemonic after having logged in.
- With a locked wallet.
- Downloading the mnemonic from the download button as file.
- When recovering a wallet by copying the mnemonic and pasting it directly to the browser extension.

It is important to note that the mnemonic could be leaked into memory not only by the application state, but by the browser displaying the mnemonic in clear text.

The severity of this vulnerability has been lowered from "Critical", since the Mnemonic Phrase was not present in the memory from the start-up of the application.


```

636970 "m.hola",
636971 "m.pizza",
636972 "m.resist",
636973 "m.skull",
636974 "(constant elements)",
636975 "(constant pool)",
636976 "system / BytecodeArray",
636977 "system / TransitionArray",
636978 "radix-r6:",
636979 "position: relative !important;\n padding-left: 0px;\n padding-top: 0px;\n padding-right: 0px;\n
margin-top:0;\n margin-right: 0px !important;\n ",
636980 "w-full mx-auto max-w-4xl flex items-stretch",
636981 "system / Foreign",
636982 "extensions::SafeBuiltins",
636983 "mr-2 transition-transform group-hover:-translate-x-1.5 group-focus:-translate-x-1.5",
636984 "absolute inset-0 z-[-5] rounded-[2.5rem] overflow-hidden bg-brand-dark/10 backdrop-blur-[30px]",
636985 "system / StoreHandler",
636986 "w-full text-center select-none mb-12",
636987 "0px",
636988 "artist.vote.give.twist.brick.volcano.liquid.private.dynamic.when.seat.accuse",
636989 "push.13836",
636990 "n.d.body",
636991 ":r8:",
636992 "n.d.hair",
636993 "o.open",
636994 "o.sleep",
636995 "o.wink",
636996 "o.glasses",
636997 "o.happy",
636998 "o.sunglasses",
636999 "a.long",
637000 "a.sideShave",
637001 "a.bobCut",
637002 "a.curly",
637003 "a.pigtails",
637004 "a.curlyBun"

```

Figure 2: Unencrypted Mnemonic Phrase in-memory, after creating the wallet

CVSS Vector:

- CVSS:3.1/AV:P/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H

Risk Level:

Likelihood - 4

Impact - 5

Recommendation:

This vulnerability was caused by the application processing sensitive data as plain text. For that reason, it is recommended to save the entropy on disk instead of the mnemonic. In cases where the mnemonic needs to be used in the code, it is recommended to break it up into several variables, or even better, obfuscate the original phrase and then deference the variable which used to hold the original phrase. In

the cases where handling the Mnemonic Phrase is needed, it is better to use the obfuscated variable along with a function that would reconstruct the original Mnemonic Phrase at the exact point where it is needed. Other than that, when the wallet is in a locked state, the mnemonic phrase should be cleared out from memory.

During wallet creation and the revealing of the mnemonic once the user has logged in, it is recommended to display the mnemonic phrase in an HTML5 canvas. This would be difficult to copy it, which allows the mnemonic to be leaked into memory through the clipboard. In any case, it is generally recommended not to allow users to copy the whole mnemonic from the extension, as that may cause a leakage from the clipboard.

References:

- [CWE-316: Cleartext Storage of Sensitive Information in Memory](#)
- [CVE-2022-32969](#)
- [Halborn Demonic](#)

Remediation Plan:

SOLVED: Wigwam Team solved this issue in the following commit ID: [af2f322c83f3d14d27f956c99f6b1bacd85674c9](#)

3.2 (HAL-02) UNENCRYPTED USER PASSWORD IN-MEMORY - HIGH

Description:

The user password in the wallet was not encrypted in memory for **Wigwam Wallet**. As a result, an attacker who had compromised the user's machine could exfiltrate and steal the Mnemonic Phrase.

Proof of concept:

The plain text user password was available in memory during various scenarios. Memory dumps were taken throughout the testing process. These memory dumps contained an exact replica of what was in memory while the application was open.

Searching among all the memory dump strings, the plain text user password appeared in the following scenarios:

- **Case I:** New wallet created, wallet unlocked:

```

1014131 "chrome-extension://lccbohngfkdikahanoclbmaolidjdf1/scripts/main.js",
1014132 "about:blank",
1014133 "!function(){\"use strict\";const t=new CSSStyleSheet;t.replaceSync('/*\n * Copyright 2019 T
governed by a BSD-style license that can be\n * found in the LICENSE file.\n */\n\nbody {
#222;\n}\n\nbody.platform-linux {\n font-family: Roboto, Ubuntu, Arial, sans-serif;\n}\n
\\.SFNSDisplay-Regular\", \"Helvetica Neue\", \"Lucida Grande\", sans-serif;\n}\n\nbody.pla
sans-serif;\n}\n\n.fill {\n position: absolute;\n top: 0;\n right: 0;\n bottom: 0;
display: none !important; /* stylelint-disable-line declaration-no-important */\n}\n');clas
n{viewportSize={width:800,height:600};viewportSizeForMediaQueries;deviceScaleFactor=1;emulatio
1014134 "offer",
1014135 "library",
1014136 "junk",
1014137 "decide",
1014138 "mix",
1014139 "[a-z0-9!#$%&'*/+=?^_{}|~.-]+@[a-z0-9](?:[a-z0-9-]{0,61}[a-z0-9])?(?:\\.\\.[a-z0-9]([a-z0-9-]{0,
1014140 "123456Aa!",
1014141 "tNgCS1DZQVGmain",
1014142 "<svg xmlns=\"http://www.w3.org/2000/svg\" viewBox=\"0 0 64 64\" fill=\"none\" shape-rendering
licensed under \"CC BY 4.0\". / Remix of the original. - Created with dicebear.com</desc><meta
xmlns:cc=\"http://creativecommons.org/ns\" xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax
Draftbit</dc:title><dc:creator><cc:Agent rdf:about=\"https://draftbit.com/\"><dc:title>Draftb
https://personas.draftbit.com/</dc:source><cc:license rdf:resource=\"https://creativecommons.o
id=\"viewboxMask\"><rect width=\"64\" height=\"64\" rx=\"0\" ry=\"0\" x=\"0\" y=\"0\" fill=\"f
0v-5.92A14.04 14.04 0 0 1 18.58 37h-.08a4.5 4.5 0 0 1-.5-8.97V27a14 14 0 1 1 28 0v1.03a4.5 4.
1014143 "0xC8d9AaD730987d29276f24D1b12AFDB6eDa91f91",

```

Figure 3: User wallet password leaked from Chrome memory dump – Case I

- **Case II:** Wallet just unlocked with by introducing the password (no copy-paste):

```

1031685 "mix",
1031686 "[a-z0-9!#$%&*+/?^_`{|}~.-]+@[a-z0-9](?:[a-z0-9-]{0,61}[a-z0-9])?(?:\.[a-z0-9-]{0,61}[a-z0-9])?)*",
1031687 "tNgCS1DZQVGmain",
1031688 "<svg xmlns=\\"http://www.w3.org/2000/svg\\" viewBox=\\"0 0 64 64\\" fill=\\"none\\" shape-rendering=\\"auto\\"><desc>\\"Pers
licensed under \\"CC BY 4.0\\". / Remix of the original. - Created with dicebear.com</desc><metadata xmlns:dc=\\"http://
xmlns:cc=\\"http://creativecommons.org/ns#\\" xmlns:rdf=\\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\\"><rdf:RDF><cc:W
Draftbit</dc:title><dc:creator><cc:Agent rdf:about=\\"https://draftbit.com/\\"><dc:title>Draftbit - draftbit.com</dc:t
https://personas.draftbit.com/</dc:source><cc:license rdf:resource=\\"https://creativecommons.org/licenses/by/4.0/\\"
id=\\"viewboxMask\\"><rect width=\\"64\\" height=\\"64\\" rx=\\"0\\" ry=\\"0\\" x=\\"0\\" y=\\"0\\" fill=\\"#fff\\" /></mask><g mask
0v-5.92A14.04 14.04 0 0 1 18.58 37h-.08a4.5 4.5 0 0 1-.5-8.97V27a14 14 0 1 1 28 0v1.03a4.5 4.5 0 0 1-.58 8.97A14.04
"0xC8d9AaD730987d29276f24D1b12AFDB6eDa91f91",
1031689 "absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2",
1031690 "chrome-extension://lccbohghfkdikahanoclbmaolidjdf1/icons/network/ethereum.png",
1031691 "NATIVE_0_0",
1031692 "NATIVE_0_0",
1031693 "123456Aa!",
1031694 "chrome-extension://lccbohghfkdikahanoclbmaolidjdf1/icons/nativeToken/ethereum.png",
1031695 "0,00",
1031696 "1_0xC8d9AaD730987d29276f24D1b12AFDB6eDa91f91_NATIVE_0_0",
1031697 "system / Foreign",
1031698 "native_bind",
1031699 "system / JSProxy",
1031700 "2158,79",
1031701 "2.158,79\u00A0$",
1031702 "(concatenated string)",
1031703 "system / Context",
1031704 "w-full h-full overscroll-y-contain pb-20 rounded-t-[.625rem] viewportBlock",
1031705 "w-4 p-1 ",
1031706 "2,56",

```

Figure 4: User wallet password leaked from Chrome memory dump – Case II

- **Case III:** After introducing the password to reveal the private key, wallet unlocked:

```

956423 "tNgCS1DZQVGmain",
956424 "<svg xmlns=\\"http://www.w3.org/2000/svg\\" viewBox=\\"0 0 64 64\\" fill=\\"none\\" shape-rendering=
under \\"CC BY 4.0\\". / Remix of the original. - Created with dicebear.com</desc><metadata xmlns
ns#\\" xmlns:rdf=\\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\\"><rdf:RDF><cc:Work><dc:title>Per
rdf:about=\\"https://draftbit.com/\\"><dc:title>Draftbit - draftbit.com</dc:title><cc:Agent></dc
rdf:resource=\\"https://creativecommons.org/licenses/by/4.0/\\" /></cc:Work></rdf:RDF></metadata>
x=\\"0\\" y=\\"0\\" fill=\\"#fff\\" /></mask><g mask=\\"url(#viewboxMask)\\"><path d=\\"M37 46.08V52a5 5
0 1 1 28 0v1.03a4.5 4.5 0 0 1-.58 8.97A14.04 14.04 0 0 1 ",
956425 "0xC8d9AaD730987d29276f24D1b12AFDB6eDa91f91",
956426 "absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2",
956427 "chrome-extension://lccbohghfkdikahanoclbmaolidjdf1/icons/network/ethereum.png",
956428 "NATIVE_0_0",
956429 "123456Aa!",
956430 "chrome-extension://lccbohghfkdikahanoclbmaolidjdf1/icons/nativeToken/ethereum.png",
956431 "w-4 p-1 transition py-0 pt-5 pb-20",
956432 "bg-white/[.07] border border-brand-main/5 rounded-[.625rem] relative cursor-pointer w-2",
956433 "system / Foreign",
956434 "native_bind",
956435 "system / JSProxy",
956436 "(script line ends)",
956437 "1_0xC8d9AaD730987d29276f24D1b12AFDB6eDa91f91_NATIVE_0_0",
956438 "n.d.M",
956439 "system / BytecodeArray",
956440 "(source position table)",
956441 "(BASELINE code)",
956442 "(BASELINE instruction stream)",

```

Figure 5: User wallet password leaked from Chrome memory dump – Case III

- **Case IV:** After introducing the password to reveal the private key, wallet locked:

```

827294 "junk",
827295 "decide",
827296 "mix",
827297 "[a-z0-9!#$%&'*/=?^_`{|}~.-]+@[a-z0-9](?:[a-z0-9-]{0,61}[a-z0-9])?(?:\\.\\.[a-z0-9]([a-z0-9
827298 "tNgCS1DZQVGmain",
827299 "<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64" fill="none" shape-render
under "CC BY 4.0". / Remix of the original. - Created with dicebear.com</desc><metadata
ns#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"><rdf:RDF><cc:Work><dc:tit
rdf:about="https://draftbit.com/"><dc:title>Draftbit - draftbit.com</dc:title></cc:Agen
rdf:resource="https://creativecommons.org/licenses/by/4.0/" /></cc:Work></rdf:RDF></met
x="0" y="0" fill="#fff" /></mask><g mask="url(#viewboxMask)"><path d="M37 46.08V
0 1 1 28 0v1.03a4.5 4.5 0 0 1-.58 8.97A14.04 14.04 0 0 1 ",
827300 "0xC8d9AaD730987d29276f24D1b12AFDB6eDa91f91",
827301 "absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2",
827302 "chrome-extension://lccb0hhgfkdikahanoclbmaolidjdf1/icons/network/ethereum.png",
827303 "NATIVE_0 0",
827304 "123456Aa!",
827305 "chrome-extension://lccb0hhgfkdikahanoclbmaolidjdf1/icons/nativeToken/ethereum.png",
827306 "system / Foreign",
827307 "native_bind",
827308 "system / JSProxy",
827309 "(bytecode offset table)",
827310 "system / CallHandlerInfo",
827311 "system / AccessorPair",
827312 "(code relocation info)",
827313 "(BASELINE code)",
827314 "(BASELINE instruction stream)",
827315 "system / BytecodeArray",
827316 "(code for ko)",
827317 "(instruction stream for ko)",
827318 "(code for s)",
827319 "(instruction stream for s)".

```

Figure 6: User wallet password leaked from Chrome memory dump – Case IV

- **Case V:** After introducing the password to reveal the mnemonic, copied using button, wallet unlocked:


```

969153 "tNgCS1DZQVGmain",
969154 "<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 64 64" fill="none" shape-rendering
under \"CC BY 4.0\". / Remix of the original. - Created with dicebear.com</desc><metadata xmln
ns#\" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"><rdf:RDF><cc:Work><dc:title>Pe
rdf:about="https://draftbit.com/"><dc:title>Draftbit - draftbit.com</dc:title></cc:Agent></d
rdf:resource="https://creativecommons.org/licenses/by/4.0/" /></cc:Work></rdf:RDF></metadata
x="0" y="0" fill="fff" /></mask><g mask="url(#viewboxMask)"><path d="M37 46.08V52a5
0 1 1 28 0v1.03a4.5 4.5 0 0 1-.58 8.97A14.04 14.04 0 0 1 ",
969155 "0xC8d9AaD730987d29276f24D1b12AFDB6eDa91f91",
969156 "absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2",
969157 "chrome-extension://lccbohghgfkdkahanoclbdmaolidjdf1/icons/network/ethereum.png",
969158 "NATIVE_0_0",
969159 "chrome-extension://lccbohghgfkdkahanoclbdmaolidjdf1/icons/nativeToken/ethereum.png",
969160 "123456Aa!",
969161 "bg-white/[.07] border border-brand-main/5 rounded-[.625rem] relative cursor-pointer w-2",
969162 "w-4 p-1 transition py-0 pt-5 pb-20",
969163 "system / Foreign",
969164 "native_bind",
969165 "system / JSProxy",
969166 "system / FeedbackVector",
969167 "(code)",
969168 "(code deopt data)",
969169 "system / LoadHandler",
969170 "system / CallHandlerInfo",
969171 "system / FunctionTemplateRareData",
969172 "(code for validate)",
969173 "(instruction stream for validate)",
969174 ".validate",
969175 "(concatenated string)",
969176 "(code for render)",
969177 "(instruction stream for render)",

```

Figure 8: User wallet password leaked from Chrome memory dump - Case VI

CVSS Vector:

- CVSS:3.1/AV:P/AC:H/PR:L/UI:R/S:U/C:H/I:H/A:N

Risk Level:

Likelihood - 4

Impact - 4

Recommendation:

The values of variables that store sensitive information should be cleared/dereferenced in the code. This action will facilitate the removal of data from memory by the garbage collector. In situations where the data needs to be managed, an obfuscated variable can be utilized with a function that will reconstruct the original data precisely at the point where it is required.

Remediation Plan:

SOLVED: Wigwam Team solved this issue in the following commit ID:
[af2f322c83f3d14d27f956c99f6b1bacd85674c9](#)

3.3 (HAL-03) PLAIN TEXT CONNECTIONS OVER HTTP - MEDIUM

Description:

This vulnerability arises when sensitive information is transmitted over HTTP in plain text, lacking encryption. This exposes the data to potentially eavesdropping and interception by malicious actors. Without the protection of secure communication protocols such as HTTPS, sensitive data, including login credentials and confidential information, becomes susceptible to unauthorized access and compromise.

Code Location:

Listing 1: src/fixtures/networks/ethereum.ts (Line 57)

```
53 faucetUrls: [  
54     "https://goerlifaucet.com",  
55     "https://goerli-faucet.slock.it",  
56     "https://faucet.goerli.mudit.blog",  
57     "http://fauceth.komputing.org?chain=5",  
58 ],  
59   infoUrl: "https://goerli.net/#about",  
60 ],
```

Listing 2: src/fixtures/networks/ethereum.ts (Line 82)

```
77   explorerUrls: [  
78     "https://sepolia.etherscan.io",  
79     "https://sepolia.otterscan.io",  
80   ],  
81   explorerApiUrl: "https://api-sepolia.etherscan.io/api",  
82   faucetUrls: ["http://fauceth.komputing.org?chain=11155111"],  
83   infoUrl: "https://sepolia.otterscan.io",  
84   },  
85 ];
```

Listing 3: src/core/back/services/inpageContentScript.ts (Line 13)

```
8 export async function startInpageContentScript() {
9   try {
10    await browser.scripting.registerContentScripts([
11      {
12        id: "inpage",
13        matches: ["file://*/**", "http://*/**", "https://*/**"],
14        js: ["scripts/inpage.js"],
15        runAt: "document_start",
16        allFrames: true,
17        ["world" as any]: "MAIN",
18      },
19    ]);
```

Listing 4: public/manifest.json (Lines 25,27)

```
24 "host_permissions": [
25   "http://localhost:8545/",
26   "file://*/**",
27   "http://*/**",
28   "https://*/**"
29 ],
```

Listing 5: public/manifest.json (Line 69)

```
67 "content_scripts": [
68   {
69     "matches": ["file://*/**", "http://*/**", "https://*/**"],
70     "js": ["scripts/content.js"],
71     "run_at": "document_start",
72     "all_frames": true
73   },
74   {
75     "matches": ["{{website}}/*"],
76     "js": ["scripts/version.js"],
77     "run_at": "document_start"
78   }
79 ]
80 }
```

CVSS Vector:

- 5.5 – Medium CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L/E:U/RL:O/RC:C

Risk Level:**Likelihood - 2****Impact - 4****Recommendation:**

Ensure that every connection that the wallet makes is over HTTPS and encrypted channels.

Remediation Plan:

SOLVED: The Wigwam team solved this issue in the following GitHub Pull Request: <https://github.com/wigwamapp/local-wigwam/pull/374>

Listing 6: src/lib/nft-metadata/uri/fetch.ts

```
11 export function forceHttps(source: string) {
12   return source.replace("http://", "https://");
13 }
14
```

3.4 (HAL-04) LACK OF USER INPUT SANITATION – MEDIUM

Description:

The user inputs are not properly sanitized or validated, exposing the system to potential malicious activities. In such cases, an attacker may exploit this weakness by injecting malicious input, leading to various security risks, including but not limited to code injection, SQL injection, or other forms of attacks that manipulate the application's intended behavior.

Proof of Concept:

Wigwam Wallet did not validate properly the RPC URL field in the Wallet Settings Page, allowing values with malicious payloads, like empty links, XSS payloads, and causing the application crash.

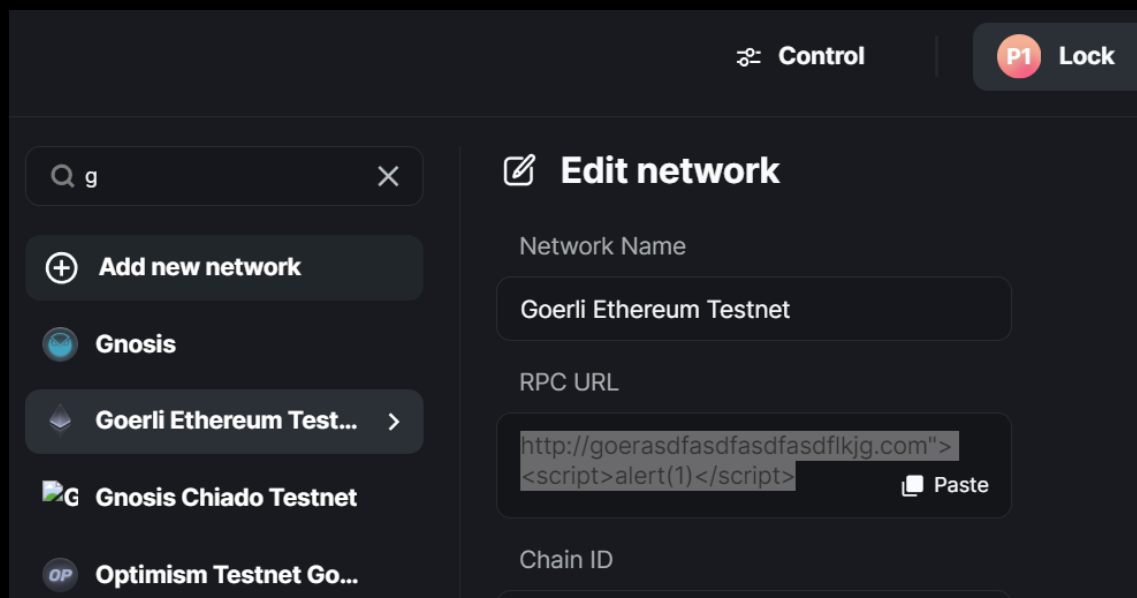


Figure 9: RPC URL with Cross-Site Scripting payload (I)

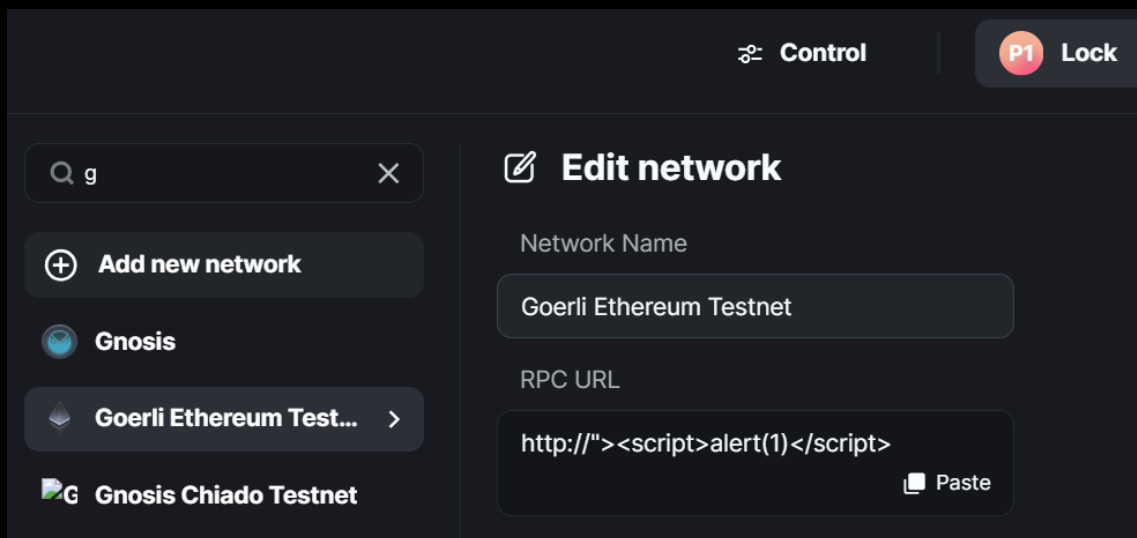


Figure 10: RPC URL with Cross-Site Scripting payload (II)

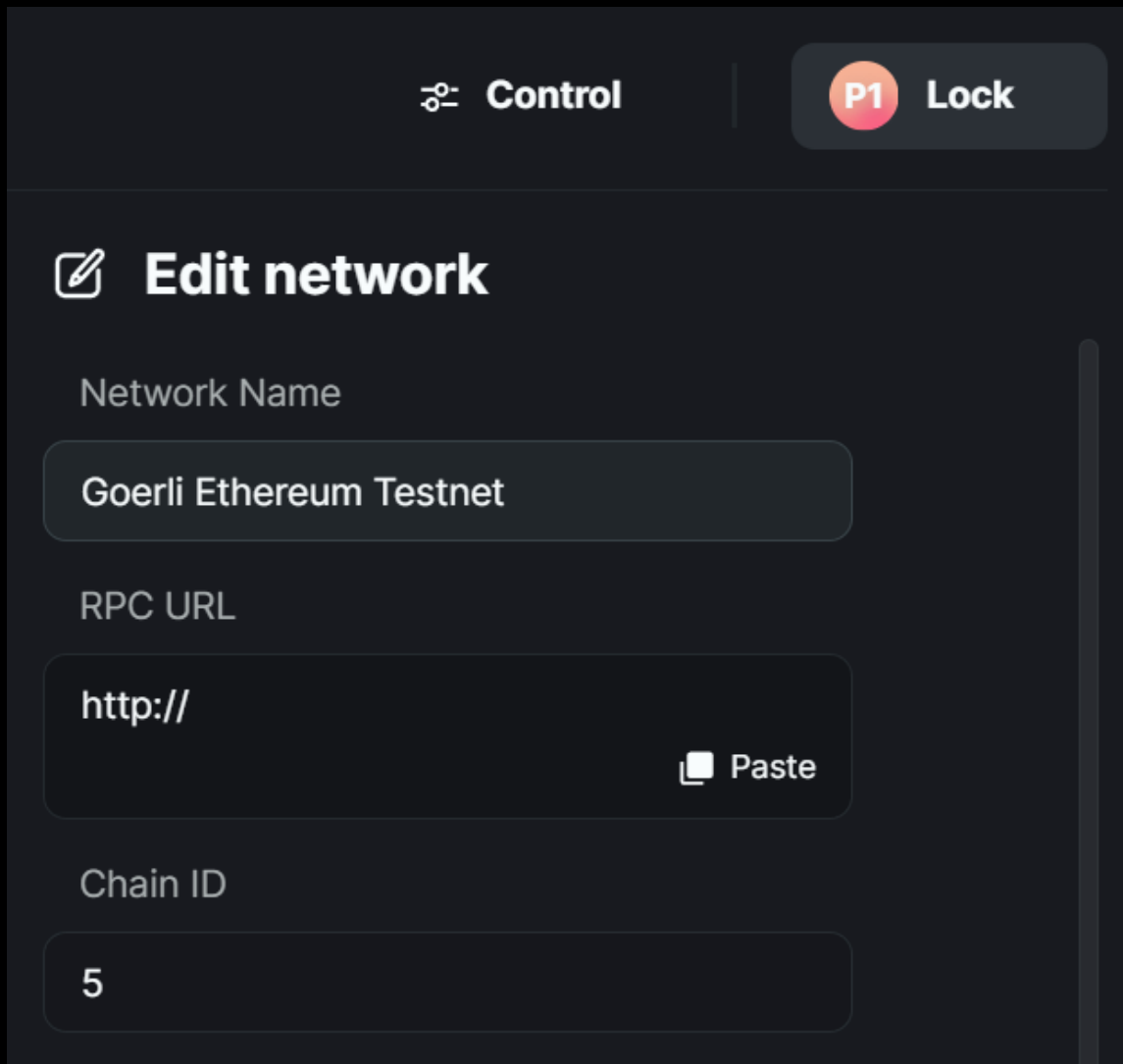


Figure 11: Empty RPC URL

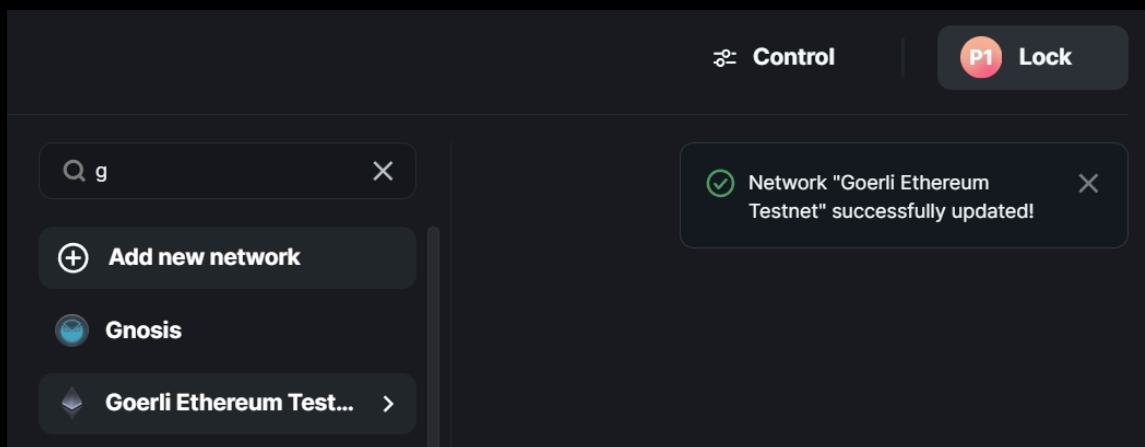


Figure 12: Changes saved successfully

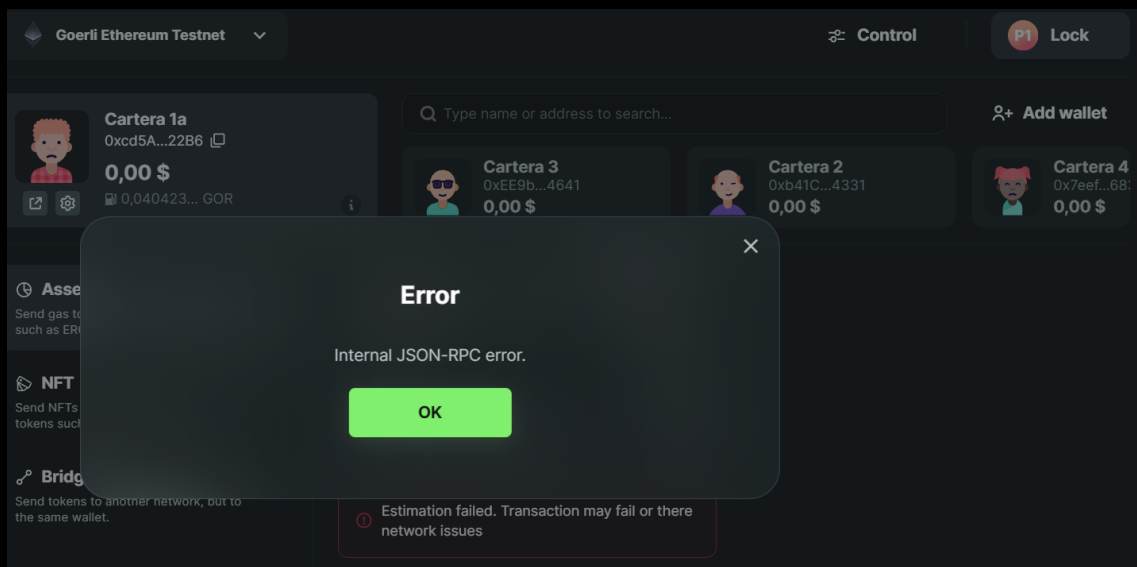


Figure 13: JSON-RPC error when interacting with the wallet

This issue was re-escalated to Medium instead of High because no further exploitation was achieved after testing several malicious payloads. However, it is important to not allow this kind of dangerous characters and payloads to be configured by users.

CVSS Vector:

- 4.9 – Medium `CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L/E:U/RL:O/RC:C`

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

- **Input Validation:** Implement robust input validation mechanisms to ensure that user inputs adhere to expected formats and constraints. Utilize server-side validation as the primary line of defense against malicious input.
- **Parameterized Queries:** When interacting with databases, utilize

parameterized queries or prepared statements to prevent SQL injection attacks. This ensures that user input is treated as data rather than executable code.

- Output Encoding: Apply proper output encoding techniques to sanitize user inputs before rendering them in the user interface. This helps prevent cross-site scripting (XSS) attacks by neutralizing potentially harmful scripts.
- Security Headers: Implement security headers, such as Content Security Policy (CSP), to mitigate risks associated with code injection and other client-side attacks.

Remediation Plan:

SOLVED: The Wigwam team solved this issue in the following GitHub Pull Request: <https://github.com/wigwamapp/local-wigwam/pull/374>

3.5 (HAL-05) PACKAGES WITH KNOWN VULNERABILITIES – MEDIUM

Description:

Wigwam Wallet used multiple third-party dependencies. However, some of them were affected by public-known vulnerabilities that may pose a risk to the global application security level. Although performed tests were mainly carried out from a black-box perspective, multiple vulnerable dependencies were found during the code review phase. Halborn considered them to be reported.

In the image below, it is possible to observe the output from `snyk test` command, showing some vulnerable package dependencies.

```
Testing /media/_share/_wigwam/local-wigwam-31618e9a64ab8d584f2997743cca3f17b745cbe5...
Tested 319 dependencies for known issues, found 1 issue, 1 vulnerable path.

Issues with no direct upgrade or patch:
x Missing Release of Resource after Effective Lifetime [Medium Severity][https://snyk.io/vuln/SNYK-JS-INFLIGHT-6095116] in inflight@1.0.6
  introduced by @dicebear/core@7.0.1 > @dicebear/converter@7.0.1 > tmp-promise@3.0.3 > tmp@0.2.1 > rimraf@3.0.2 > glob@7.2.3 > inflight@1.0.6
  No upgrade or patch available
```

Figure 14: Vulnerable dependencies for **Wigwam wallet** (I)

Besides, in the following pictures, it is possible to observe the output from `yarn audit` command, showing more vulnerable package dependencies.

```

└─$ yarn audit
yarn audit v1.22.19

```

moderate	@adobe/css-tools Improper Input Validation and Inefficient Regular Expression Complexity
Package	@adobe/css-tools
Patched in	>=4.3.2
Dependency of	@testing-library/jest-dom
Path	@testing-library/jest-dom > @adobe/css-tools
More info	https://www.npmjs.com/advisories/1095152

```

1 vulnerabilities found - Packages audited: 1376
Severity: 1 Moderate
Done in 3.83s.

```

Figure 15: Vulnerable dependencies for `Wigwam wallet` (II)

CVSS Vector:

- 4.1 – Medium CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:C/C:N/I:L/A:L/E:U/RL:O/RC:C

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

Update all affected packages to the latest version. It is also recommended to conduct an automated analysis of the dependencies from the inception of the project to determine potential security issues. Developers need to be aware of these potential risks and implement appropriate countermeasures to safeguard the affected application.

Remediation Plan:

SOLVED: The Wigwam team solved this issue in the following GitHub Pull Request: <https://github.com/wigwamapp/local-wigwam/pull/381>

3.6 (HAL-06) EXCESSIVE TIME FOR WALLET AUTO-LOCK – MEDIUM

Description:

Wigwam Wallet had an excessive auto-lock time period set by default. During the analysis, it has been identified that the auto-lock timer was set up to one week by default. Setting the auto-lock time to that high amount of time, diminishes the purpose of this extra auto lock security feature.

Code Location:

Listing 7: src/fixtures/settings.ts (Line 18)

```
18 export const DEFAULT_AUTO_LOCK_TIMEOUT = ONE_WEEK;
```

It was also checked that there were more values for the LOCK_TIMEOUTS, but Halborn did not detect any functionality from the wallet GUI to configure them:

Listing 8: src/fixtures/settings.ts

```
7 export const AUTO_LOCK_TIMEOUTS: number[] = [  
8   0, // off  
9   60_000 * 5, // 5 min  
10  60_000 * 15, // 15 min  
11  ONE_HOUR, // 1 hour  
12  ONE_HOUR * 3, // 3 hours  
13  ONE_DAY, // 1 day  
14  ONE_DAY * 2, // 2 days  
15  ONE_WEEK, // 1 week  
16 ];
```

CVSS Vector:

- 5.4 – Medium CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N/E:U/RL:O/RC:C

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

Check the activity on the extension and set up an auto-lock functionality if the wallet extension has not been actively used for a while.

Remediation Plan:

SOLVED: The Wigwam team solved this issue in the following GitHub Pull Request: <https://github.com/wigwamapp/local-wigwam/pull/374>

Listing 9: src/fixtures/settings.ts

```
8 export const AUTO_LOCK_TIMEOUTS = new Map([
9   [0, "None"],
10  [60_000 * 5, "5 min"],
11  [60_000 * 15, "15 min"],
12  [ONE_HOUR, "1 hour"],
13  [ONE_HOUR * 3, "3 hours"],
14  [ONE_DAY, "1 day"],
15  [TWO_DAYS * 2, "2 days"],
16  [ONE_WEEK, "1 week"],
17 ]);
18
19 export const DEFAULT_AUTO_LOCK_TIMEOUT = TWO_DAYS;
```

3.7 (HAL-07) INSECURE AUTHENTICATION METHODS – LOW

Description:

Wigwam Wallet used HTTP Basic Authentication as the source code revealed. Basic HTTP authentication transmits credentials in an easily decipherable format, exposing sensitive user information to potential interception and unauthorized access. This vulnerability poses a significant security risk as it allowed malicious actors to capture and exploit authentication credentials, compromising the confidentiality of user accounts.

Proof of Concept:

Listing 10: .vendor/axios-fetch-adapter/index.js (Line 99)

```
90 function createRequest(config) {
91   const headers = new Headers(config.headers.toJSON());
92
93   // HTTP basic authentication
94   if (config.auth) {
95     const username = config.auth.username || "";
96     const password = config.auth.password
97       ? decodeURI(encodeURIComponent(config.auth.password))
98       : "";
99     headers.set("Authorization", `Basic ${btoa(username + ":" +
100     password)}`);
100   }
```

CVSS Vector:

- 3.8 – Low CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N/E:U/RL:O/RC:C

Risk Level:

Likelihood – 1

Impact - 3**Recommendation:**

Implement secure authentication methods and avoid using authentication methods that transmit credentials in plaintext.

Remediation Plan:

SOLVED: The Wigwam team solved this issue in the following GitHub Commit ID: [78c6c0a04d41d11b6fd6dddfe461cbdd5c1e04d4](https://github.com/wigwam/axios-fetch-adapter/commit/78c6c0a04d41d11b6fd6dddfe461cbdd5c1e04d4)

Listing 11: .vendor/axios-fetch-adapter/index.js

```
90 function createRequest(config) {
91   const headers = new Headers(config.headers.toJSON());
92
93   // HTTP basic authentication
94   if (config.auth) {
95     throw new Error("Basic auth is not supported. Use a different
↳ behaviour");
96
97     // const username = config.auth.username || "";
98     // const password = config.auth.password
99     //   ? decodeURI(encodeURIComponent(config.auth.password))
100     //   : "";
101     // headers.set("Authorization", `Basic ${btoa(username + ":" +
↳ password)}`);
102   }
```


3.8 (HAL-08) LACK OF DEFAULT CLAUSE ON SWITCH STATEMENT - LOW

Description:

Some `switch` statements were detected with lack of `default` clause. This may pose a risk for the application because if a non contemplated value is passed to the application, this may cause unexpected and unstable behavior of the application, rendering it unusable in the worst cases (DoS).

Source Code Snippets:

Listing 12: src/version.ts (Line 18)

```
10 window.addEventListener(  
11   "message",  
12   (evt) => {  
13     if (  
14       evt.source === window &&  
15       evt.origin === location.origin &&  
16       evt.data?.salt === salt  
17     ) {  
18       switch (evt.data.type) {  
19         case "wigwam.reply":  
20           ext.runtime.sendMessage({  
21             type: "__APPLY_WEBSITE_DATA",  
22             data: evt.data.data,  
23           });  
24           break;  
25  
26         case "wigwam.openapp":  
27           ext.runtime.sendMessage({ type: "__OPEN_OR_FOCUS_TAB" })  
28           ↵ ;  
29           break;  
30         }  
31       },  
32       false,  
33     );
```

Listing 13: src/app/components/blocks/EditWalletSection.tsx (Line 549)

```
548 const getSocialIcon = (social: SocialProvider) => {  
549   switch (social) {  
550     case "google":  
551       return GoogleIcon;  
552     case "facebook":  
553       return FacebookIcon;  
554     case "twitter":  
555       return TwitterIcon;  
556     case "reddit":  
557       return RedditIcon;  
558   }  
559 };
```

Listing 14: src/app/components/blocks/transfer/TokenTransfer.tsx (Line 378)

```
378 switch (standard) {
379     case TokenStandard.ERC20:
380         {
381             const contract = ERC20__factory.connect(
382               ↪ address, signer);
383             gasLimit = await contract.transfer.
384               ↪ estimateGas(
385                 recipientAddr,
386                 value,
387             );
388             break;
389         }
390     case TokenStandard.ERC721:
391         {
392             const contract = ERC721__factory.connect(
393               ↪ address, signer);
394             gasLimit = await contract.transferFrom.
395               ↪ estimateGas(
396                 currentAccount.address,
397                 recipientAddr,
398                 id,
399             );
400             break;
401         }
402     case TokenStandard.ERC1155:
403         {
404             const contract = ERC1155__factory.connect(
405               address,
406               signer,
407             );
408             gasLimit = await contract.safeTransferFrom.
409               ↪ estimateGas(
410                 currentAccount.address,
411                 recipientAddr,
412                 id,
413                 value,
414                 new Uint8Array(),
```

```
415         );  
416     }  
417     break;  
418 }  
419 }
```

Listing 15: src/app/components/elements/AutoIcon.tsx (Line 106)

```
105 function generateDicebearIconSvg(type: DicebearStyleType, seed:  
↳ string) {  
106     switch (type) {  
107         case "avataaars":  
108             return createAvatar(avataaarsStyle, {  
109                 seed,  
110                 mouth: [  
111                     "default",  
112                     "disbelief",  
113                     "eating",  
114                     "grimace",  
115                     "screamOpen",  
116                     "serious",  
117                     "smile",  
118                     "tongue",  
119                     "twinkle",  
120                 ],  
121             }).toString();  
122  
123         case "personas":  
124             return createAvatar(personasStyle, {  
125                 seed,  
126             }).toString();  
127     }  
128 }
```

Listing 16: src/app/components/elements/BackButton.tsx (Line 37)

```
28 const handleClick = useCallback<MouseEventHandler <  
↳ HTMLButtonElement>>(  
29     async (evt) => {  
30         if (onClick) {  
31             await onClick(evt);  
32             if (evt.defaultPrevented) {  
33                 return;  
}
```

```

34     }
35   }
36
37   switch (true) {
38     case historyPosition > 0:
39       goBack();
40       break;
41
42     case !inHome:
43       setCurrentValue([initialValue, "replace"]);
44       break;
45   }
46   },
47   [onClick, historyPosition, initialValue, inHome,
48     ↵ setCurrentValue],
49   );

```

Listing 17: src/app/components/elements/WalletName.tsx (Line 69)

```

67 const getIcon = (wallet: Account) => {
68   if (wallet.source === AccountSource.OpenLogin) {
69     switch (wallet.social) {
70       case "google":
71         return GoogleIcon;
72       case "facebook":
73         return FacebookIcon;
74       case "twitter":
75         return TwitterIcon;
76       case "reddit":
77         return RedditIcon;
78     }
79   }

```

Listing 18: src/app/components/screens/approvals/Signing.tsx (Line 68)

```

66 const message = useMemo(() => {
67   try {
68     switch (approval.standard) {
69       case SigningStandard.PersonalSign:
70         try {
71           return toUtf8String(approval.message);
72         } catch {
73           return approval.message;

```

```

74     }
75
76     case SigningStandard.SignTypedDataV1:
77         return approval.message;
78
79     case SigningStandard.SignTypedDataV3:
80     case SigningStandard.SignTypedDataV4:
81         return JSON.parse(approval.message);
82     }
83 } catch (err) {
84     console.error(err);
85 }
86
87 return null;
88 }, [approval]);

```

Listing 19: src/app/components/screens/approvals/Signing.tsx (Line 108)

```

108 switch (approval.standard) {
109     case SigningStandard.PersonalSign:
110         sig = await ledgerEth.signPersonalMessage(
111             account.derivationPath,
112             hexlify(approval.message).substring(2),
113         );
114         break;
115
116     case SigningStandard.SignTypedDataV1:
117     case SigningStandard.SignTypedDataV3:
118         throw new Error(
119             "Ledger: Only version 4 of typed data signing
120             ↳ is supported",
121         );
122
123     case SigningStandard.SignTypedDataV4:
124         let domainSeparatorHex, hashStructMessageHex;
125
126         try {
127             const {
128                 domain,
129                 types,
130                 primaryType,
131                 message: sanitizedMessage,

```

```

131         } = TypedDataUtils.sanitizeData(
132             JSON.parse(approval.message),
133         );
134
135         domainSeparatorHex = TypedDataUtils.hashStruct
136         ↪ (
137             "EIP712Domain",
138             domain,
139             types,
140             SignTypedDataVersion.V4,
141         ).toString("hex");
142         hashStructMessageHex = TypedDataUtils.
143         ↪ hashStruct(
144             primaryType as any,
145             sanitizedMessage,
146             types,
147             SignTypedDataVersion.V4,
148         ).toString("hex");
149     } catch {
150         throw new Error("Invalid message");
151     }
152
153     sig = await ledgerEth.signEIP712HashedMessage(
154         account.derivationPath,
155         domainSeparatorHex,
156         hashStructMessageHex,
157     );
158     break;
159 }

```

Listing 20: src/app/components/screens/approvals/Signing.tsx (Line 168)

```

159 if (sig) {
160     signedMessage = ethers.Signature.from({
161         v: sig.v,
162         r: "0x" + sig.r,
163         s: "0x" + sig.s,
164     }).serialized;
165
166     let addressSignedWith: string | undefined;
167
168     switch (approval.standard) {

```

```
169         case SigningStandard.PersonalSign:
170             addressSignedWith = recoverPersonalSignature({
171                 data: message,
172                 signature: signedMessage,
173             });
174             break;
175
176         case SigningStandard.SignTypedDataV4:
177             addressSignedWith = recoverTypedSignature({
178                 version: SignTypedDataVersion.V4,
179                 data: message,
180                 signature: signedMessage,
181             });
182             break;
183     }
184
185     if (
186         getAddress(addressSignedWith!) !== getAddress(
187             ↪ account.address)
188     ) {
189         throw new Error(
190             ↪ "Ledger: The signature doesnt match the right
191             ↪ address",
192         );
193     } catch (err) {
194         ledgerError = err;
195     }
196 });
```

Listing 21: src/app/components/screens/approvals/Transaction.tsx (Line 295)

```
292 useEffect(() => {
293     if (!action) return;
294
295     switch (action.type) {
296         case TxActionType.TokenTransfer:
297             action.tokens.forEach(({ slug }) => {
298                 findToken(chainId, accountAddress, slug);
299             });
300             break;
```



```

301
302     case TxActionType.TokenApprove:
303         if (action.tokenSlug) {
304             findToken(chainId, accountAddress, action.tokenSlug);
305         }
306         break;
307     }
308 }, [action, chainId, accountAddress]);

```

Listing 22: src/core/back/rpc/wallet/signing.ts (Line 30)

```

30 switch (standard) {
31     case SigningStandard.EthSign:
32         throw ethErrors.provider.unsupportedMethod();
33
34     case SigningStandard.PersonalSign:
35         accountAddress = params[1];
36         message = params[0];
37
38         if (!isAddress(accountAddress)) {
39             accountAddress = params[0];
40             message = params[1];
41         }
42         break;
43
44     case SigningStandard.SignTypedDataV1:
45         accountAddress = params[1];
46         message = params[0];
47         break;
48
49     case SigningStandard.SignTypedDataV3:
50     case SigningStandard.SignTypedDataV4:
51         accountAddress = params[0];
52         message = params[1];
53         break;
54 }

```

Listing 23: src/core/back/rpc/wallet/signing.ts (Line 59)

```

56 try {
57     accountAddress = getAddress(accountAddress);
58
59     switch (standard) {

```

```
60     case SigningStandard.PersonalSign:
61         message = isHexString(message)
62             ? message
63             : hexlify(toUtf8Bytes(message));
64         break;
65
66     case SigningStandard.SignTypedDataV1:
67         assert(
68             Array.isArray(message) &&
69             message.every(
70                 (item: any) =>
71                     typeof item.type === "string" &&
72                     typeof item.name === "string" &&
73                     typeof item.value === "string",
74             ),
75         );
76         break;
77
78     case SigningStandard.SignTypedDataV3:
79     case SigningStandard.SignTypedDataV4:
80         assert(
81             message &&
82             typeof message === "string" &&
83             typeof JSON.parse(message) === "object",
84         );
85         break;
86     }
87 } catch {
88     throw ethErrors.rpc.invalidParams();
89 }
```

Listing 24: src/core/back/services/pageServer.ts (Line 88)

```
85 internalStateSubs.set(port, (type) => {
86     const perm = currentPermission;
87
88     switch (type) {
89         case "walletStatus":
90             notifyPermission(port, perm);
91             resolveConnectionApproval(perm);
92             break;
93
94         case "chainId":
95             if (!perm) notifyPermission(port);
```

```

96         break;
97
98         case "accountAddress":
99             if (perm) notifyPermission(port, perm);
100            break;
101        }
102    });

```

Listing 25: src/core/back/sync/chain.ts (Line 30)

```

29 try {
30     switch (standard) {
31         case TokenStandard.ERC20: {
32             const contract = ERC20__factory.connect(tokenAddress,
33             ↪ provider);
34
35             return await retry(
36                 () =>
37                 props({
38                     decimals: contract.decimals(),
39                     symbol: contract.symbol(),
40                     name: contract.name(),
41                 }),
42                 { retries: 2 },
43             );
44         }
45         case TokenStandard.ERC721:
46         case TokenStandard.ERC1155: {
47             const agent = new NFTMetadataAgent(chainId, provider);
48             const [contractName, parsed] = await Promise.all([
49                 standard === TokenStandard.ERC721
50                 ? ERC721__factory.connect(tokenAddress, provider)
51                   .name()
52                   .catch(() => null)
53                 : null,
54                 agent.fetchMetadata(tokenAddress, tokenId).catch(() =>
55             ↪ null),
56             ]);
57
58             if (!returnBroken && !parsed) {
59                 return null;
60             }

```

```

61     const metadata: Partial<NFT> = omitEmptyFields({
62       contractAddress: tokenAddress,
63       tokenId: tokenId,
64       name:
65         parsed?.name ??
66         `${contractName ? `${contractName} ` : ""}#${tokenId}`,
67       collectionName: contractName ?? undefined,
68       collectionId: contractName ? slugify(contractName) :
↳ undefined,
69       description: parsed?.description,
70       thumbnailUrl: parsed?.imageUrl,
71       contentUrl: parsed?.contentURL,
72       detailUrl: parsed?.externalURL,
73       contentType: parseContentType(parsed?.contentURLMimeType)
↳ ,
74       attributes: parsed?.attributes as any,
75     });
76
77     return metadata;
78   }
79 }
80 } catch (err) {
81   console.error(err);
82 }

```

Listing 26: src/core/back/vault/index.ts (Line 460)

```

460   switch (source) {
461     case AccountSource.SeedPhrase: {
462       const { derivationPath } = params;
463
464       const rootHDNode = getRootHDNode();
465       const { address, privateKey, publicKey } =
466         rootHDNode.derivePath(derivationPath);
467
468       const account: HDAccount = {
469         ...base,
470         source,
471         address,
472         derivationPath,
473       };
474
475       const keys: AccountKeys = {
476         privateKey: ProtectedValue.fromString(privateKey),

```

```
477     publicKey: ProtectedValue.fromString(publicKey),
478   };
479
480   return { account, keys };
481 }
482
483 case AccountSource.PrivateKey: {
484   const privateKey = add0x(
485     importProtected(params.privateKey).getText(),
486   );
487   const publicKey = ethers.SigningKey.computePublicKey(
488     privateKey,
489     true,
490   );
491   const address = ethers.computeAddress(publicKey);
492
493   const account: PrivateKeyAccount = {
494     ...base,
495     source,
496     address,
497   };
498
499   const keys: AccountKeys = {
500     privateKey: ProtectedValue.fromString(privateKey),
501     publicKey: ProtectedValue.fromString(publicKey),
502   };
503
504   return { account, keys };
505 }
506
507 case AccountSource.OpenLogin: {
508   const privateKey = add0x(
509     importProtected(params.privateKey).getText(),
510   );
511   const publicKey = ethers.SigningKey.computePublicKey(
512     privateKey,
513     true,
514   );
515   const address = ethers.computeAddress(publicKey);
516
517   const { social, socialName, socialEmail } = params;
518
519   const account: SocialAccount = {
520     ...base,
```

```
521     source ,
522     address ,
523     social ,
524     socialName ,
525     socialEmail ,
526   };
527
528   const keys: AccountKeys = {
529     privateKey: ProtectedValue.fromString(privateKey),
530     publicKey: ProtectedValue.fromString(publicKey),
531   };
532
533   return { account, keys };
534 }
535
536 case AccountSource.Ledger: {
537   const derivationPath = params.derivationPath;
538   const publicKey = ethers.SigningKey.computePublicKey(
539     add0x(importProtected(params.publicKey).getText()),
540     true ,
541   );
542   const address = ethers.computeAddress(publicKey);
543
544   const account: LedgerAccount = {
545     ...base ,
546     source ,
547     address ,
548     derivationPath ,
549   };
550
551   const keys: AccountKeys = {
552     publicKey: ProtectedValue.fromString(publicKey),
553   };
554
555   return { account, keys };
556 }
557
558 case AccountSource.Address: {
559   let { address } = params;
560
561   address = ethers.getAddress(address);
562
563   const account: WatchOnlyAccount = {
564     ...base ,
```

```
565     source ,
566     address ,
567   };
568
569   const keys: AccountKeys = {};
570
571   return { account, keys };
572 }
573 }
574 });
```

Listing 27: src/core/common/account.ts (Line 17)

```
17  switch (params.source) {
18    case AccountSource.SeedPhrase:
19      validateDerivationPath(params.derivationPath);
20      break;
21
22    case AccountSource.Ledger:
23      validateDerivationPath(params.derivationPath);
24      validatePublicKey(fromProtectedString(params.publicKey));
25      break;
26
27    case AccountSource.PrivateKey:
28    case AccountSource.OpenLogin:
29      validatePrivateKey(fromProtectedString(params.privateKey));
30      break;
31
32    case AccountSource.Address:
33      validateAddress(params.address);
34      break;
35  }
36 }
```

Listing 28: src/core/common/balance.ts (Line 18)

```
18  switch (standard) {
19    case TokenStandard.ERC20: {
20      const contract = ERC20__factory.connect(address, provider)
21      ↪ ;
22      return await contract.balanceOf(accountAddress);
23    }
```

```

24
25     case TokenStandard.ERC721: {
26         const contract = ERC721__factory.connect(address, provider
↳ );
27         const owner = await contract.ownerOf(id);
28
29         return ethers.getAddress(owner) === accountAddress ? 1n :
↳ 0n;
30     }
31
32     case TokenStandard.ERC1155: {
33         const contract = ERC1155__factory.connect(address,
↳ provider);
34
35         return await contract.balanceOf(accountAddress, id);
36     }
37 }

```

Listing 29: src/lib/ext/porter/client.ts (Line 78)

```

76 return new Promise((resolve, reject) => {
77     const handleMessage = (msg: any) => {
78         switch (true) {
79             case msg?.reqId !== reqId:
80                 return;
81
82             case msg?.type === PorterMessageType.Res:
83                 resolve(msg.data);
84                 break;
85
86             case msg?.type === PorterMessageType.Err:
87                 reject(deserializeError(msg.data));
88                 break;
89         }
90
91         cleanup();
92     };

```

CVSS Vector:

- 3.5 – Low CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L/E:U/RL:O/RC:C

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

According to coding best practices, `default` clause should be added to every `switch` statement to avoid unexpected and unpredictable application behaviors.

Remediation Plan:

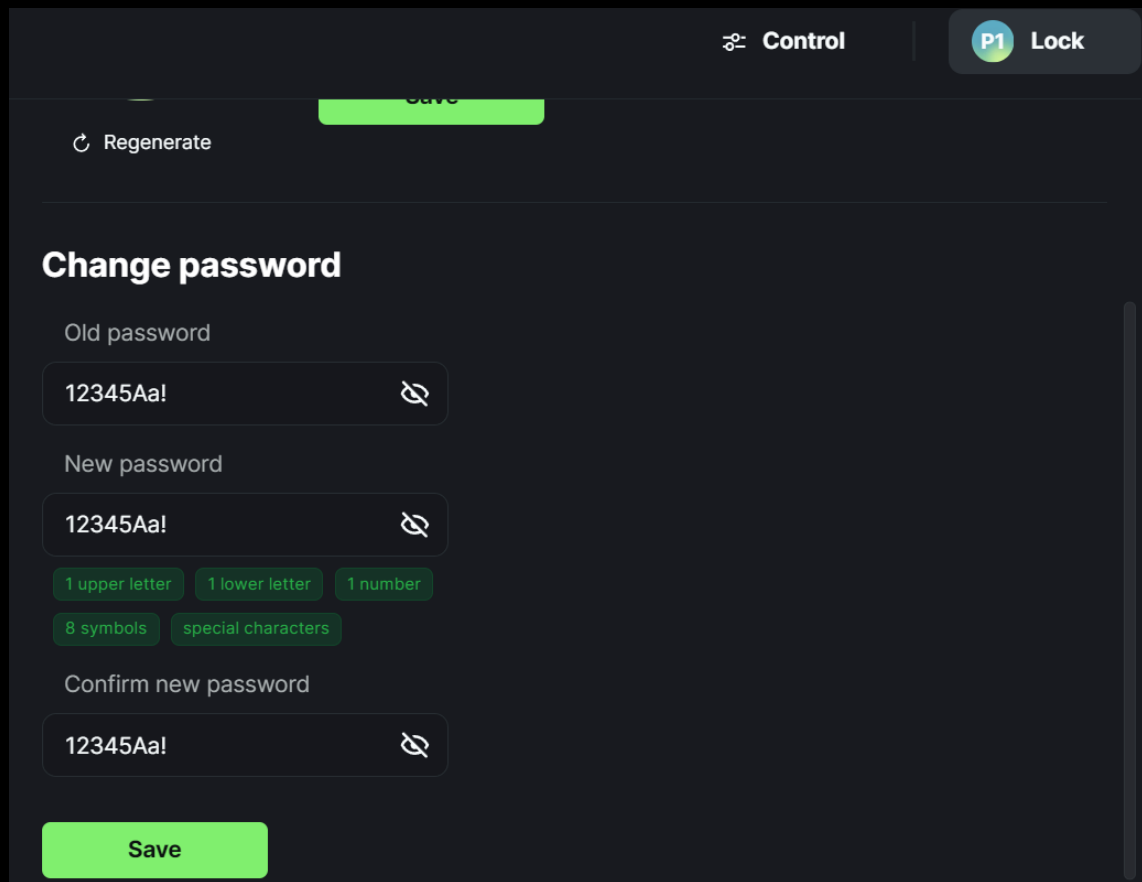
SOLVED: The Wigwam team solved this issue in the following GitHub Pull Request: <https://github.com/wigwamapp/local-wigwam/pull/374>

3.9 (HAL-09) OLD PASSWORD RE-USAGE - LOW

Description:

A user could repetitively set up previously used passwords after changing them. This vulnerability arises from users reverting to familiar, potentially compromised, or weak passwords when updating their credentials. It introduces a security risk as attackers may exploit patterns in password reuse over time, compromising the security of user accounts.

Proof of Concept:



The screenshot shows a dark-themed user interface for changing a password. At the top right, there are two buttons: 'Control' with a gear icon and 'Lock' with a 'P1' indicator. Below these is a 'Save' button. A 'Regenerate' button with a circular arrow icon is visible on the left. The main section is titled 'Change password' and contains three input fields, each with a toggle icon on the right:

- 'Old password' field containing '12345Aa!'
- 'New password' field containing '12345Aa!'. Below this field are four green indicator boxes: '1 upper letter', '1 lower letter', '1 number', and '8 symbols special characters'.
- 'Confirm new password' field containing '12345Aa!'

At the bottom of the form is a large green 'Save' button.

Figure 16: Setting up the old password as new one

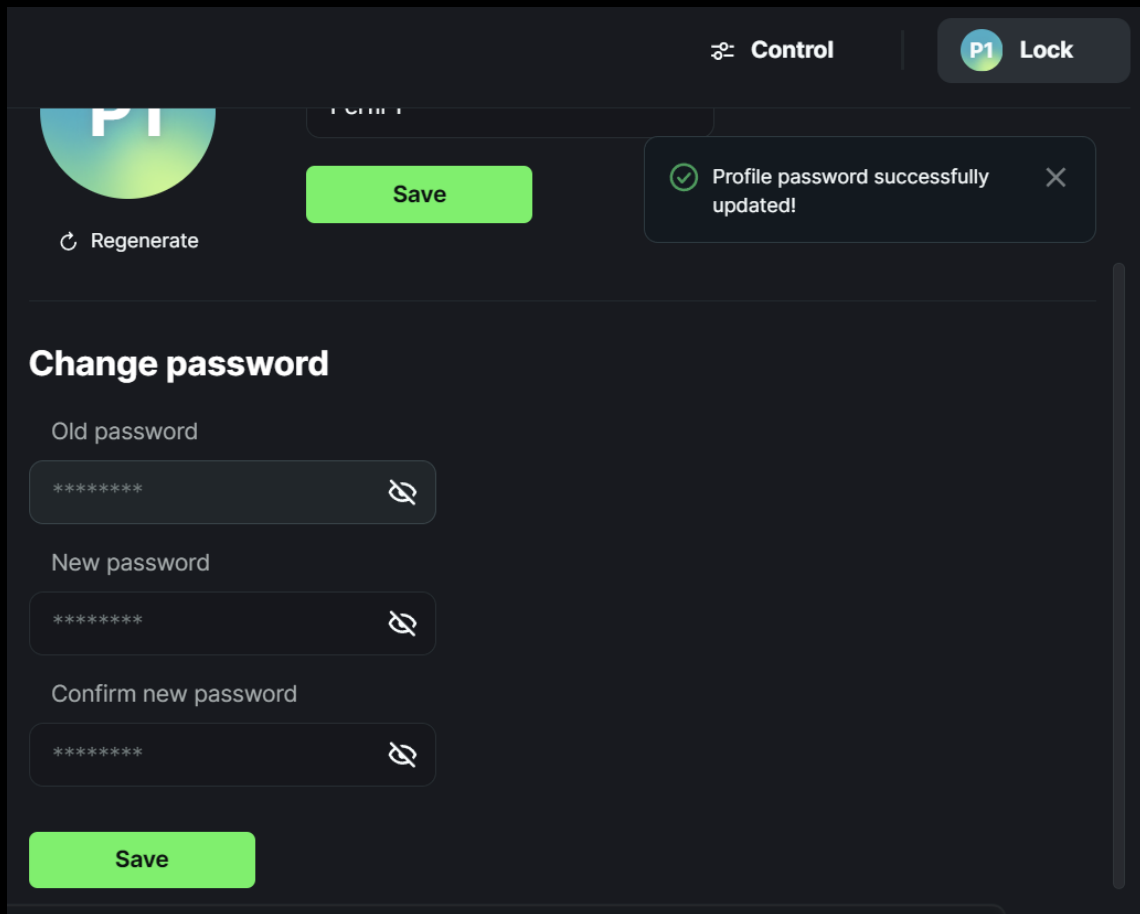


Figure 17: Setting up the old password as new one. Changes confirmed

CVSS Vector:

- 2.1 - Low `CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N/E:U/RL:O/RC:C`

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

Implement a password history policy that prohibits users from reusing a certain number of their most recent passwords. This prevents users from reverting to old passwords and enhances overall security.

Remediation Plan:

SOLVED: The Wigwam team solved this issue in the following GitHub Pull Request: <https://github.com/wigwamapp/local-wigwam/pull/374>

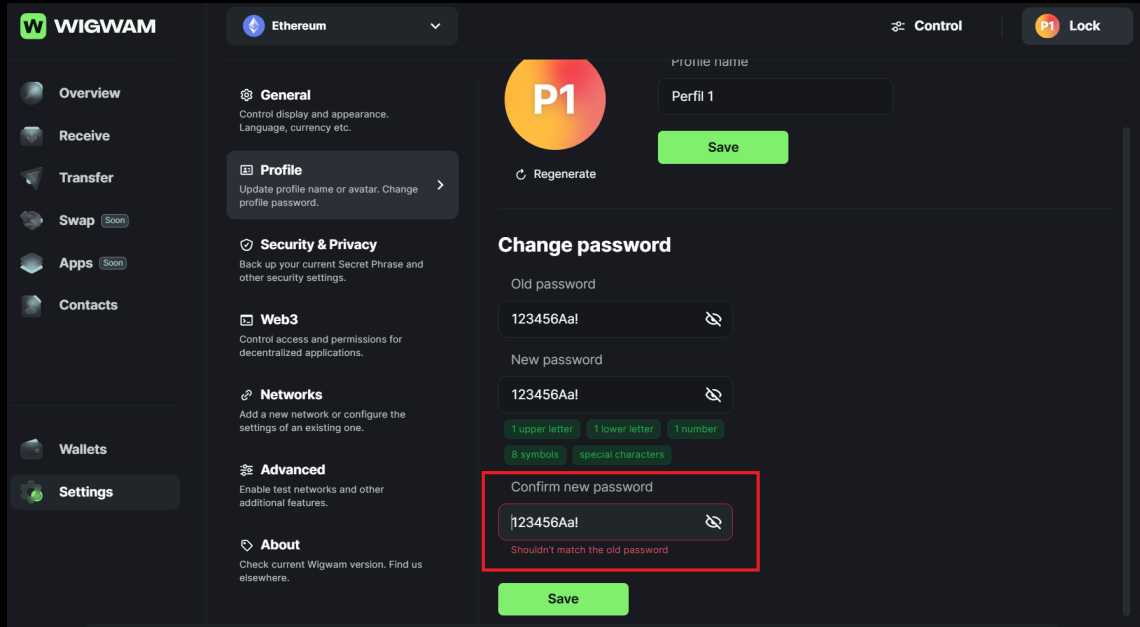


Figure 18: Cannot re-use password

3.10 (HAL-10) BROKEN LINKS - INFORMATIONAL

Description:

There were several broken links or deprecated websites in the `Wigwam wallet` interface and workflows during the security assessment. Although this is not a vulnerability itself, this type of issue could indicate a poor code maturity or lack of maintenance.

Proof of Concept:

Goerli Ethereum Testnet:

- <http://fauceth.komputing.org/?chain=11155111>
- <https://faucet.goerli.mudit.blog/>
- <https://goerli-faucet.slock.it> redirected to <https://www.blockchains.com/>

Huobi:

- <https://scan-testnet.hecochain.com/faucet>

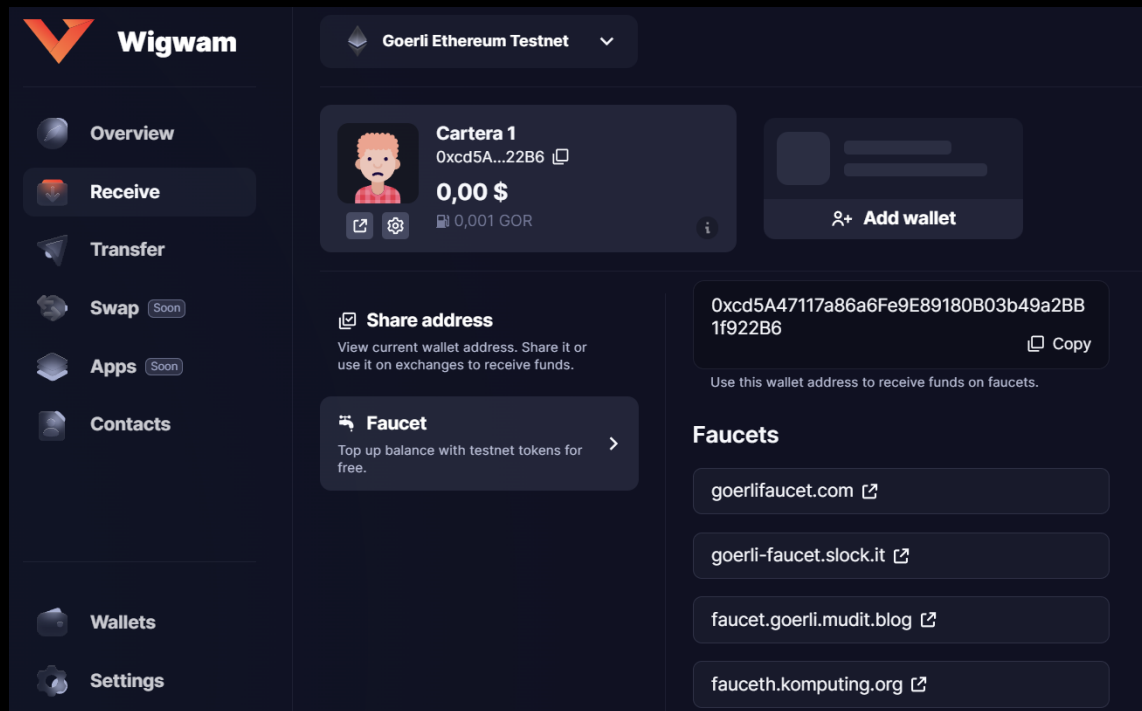


Figure 19: Broken links in Faucet section

CVSS Vector:

- Informational `CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:N/A:N/E:U/RL:O/RC:C`

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

Review the source code and update all broken/deprecated links to third-party entities.

Remediation Plan:

ACKNOWLEDGED: The Wigwam team acknowledged this finding.



PERFORMED TESTS

4.1 STATIC ANALYSIS

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped repositories. Among the tools used were `nodeJSScan` and `SonarQube`. These tools used to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement.

NodeJSScan:

Part of the assessment was Static Code Analysis, which Halborn performed using the `NodeJSScan` tool. NodeJSScan is a Static security code scanner (SAST) specially built for Node.js applications.

`NodeJSScan` only discovered informational issues which do not pose any risk for the `Wigwam Wallet` browser extension, or they do not apply for a browser extension.

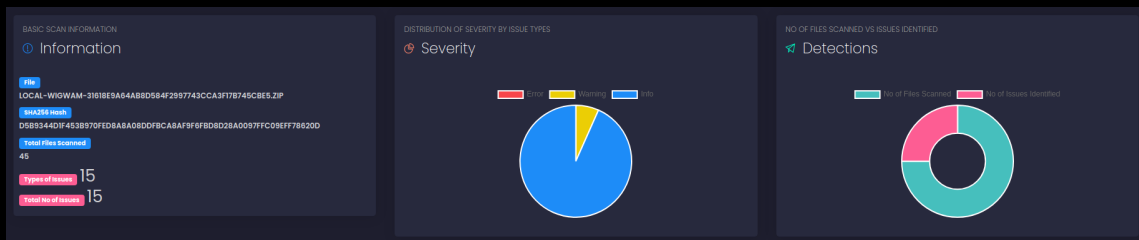


Figure 20: NodeJSScan Results (I)

Findings Summary

ISSUE	DESCRIPTION	SEVERITY	STANDARDS
NODE MD5	MD5 is a weak hash which is known to have collision. Use a strong hashing function.	WARNING	CWE-327: Use of a Broken or Risky Cryptographic Algorithm
RATE LIMIT CONTROL	This application does not have API rate limiting controls.	INFO	CWE-770: Allocation of Resources Without Limits or Throttling
HELMET HEADER CHECK CROSSDOMAIN	Helmet X Permitted Cross Domain Policies header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER XSS FILTER	Helmet XSS Protection header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER CHECK CSP	Helmet Content Security Policy header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER CHECK EXPECT CT	Helmet Expect CT header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER REFERRER POLICY	Helmet Referrer Policy header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER FRAME GUARD	Helmet X Frame Options header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
ANTI CSRF CONTROL	This application does not have anti CSRF protection which prevents cross site request forgery attacks.	INFO	CWE-352: Cross-Site Request Forgery (CSRF)
HELMET HEADER FEATURE POLICY	Helmet Feature Policy header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER IENOOOPEN	Helmet IE No Open header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER NOSNIFF	Helmet No Sniff header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER HSTS	Helmet HSTS header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER X POWERED BY	Helmet X Powered By header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure
HELMET HEADER DNS PREFETCH	Helmet DNS Prefetch header is not configured for this application.	INFO	CWE-693: Protection Mechanism Failure

Figure 21: NodeJSScan Results (II)

JavaScript Issues

▼ NODE MD5 - 1

Description: MD5 is a weak hash which is known to have collision. Use a strong hashing function.

Severity: WARNING

OWASP:

CWE: CWE-327: Use of a Broken or Risky Cryptographic Algorithm

File: local-wigwam-31618e9a64ab8d584f2997743cca3f17b745cbe5/.vendor/swc-jest/index.js

Lines: [68, 69]

Hide Code View File Not Applicable False Positive

```
return crypto
  .createHash("md5")
```

Figure 22: NodeJSScan Results (III)

- No major vulnerabilities were found by `NodeJSScan`.

SonarQube:

`SonarQube` results are not exportable. So Halborn would recommend running the tool locally in the repository folder and check the interactive results in the SonarQube web GUI.

- Step 1: Run SonarQube locally

Listing 30: Running Docker image of SonarQube locally

```
1 $ docker run -d --name sonarqube -e
↳ SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:
↳ latest
```

- Step 2: Access the SonarQube web interface in <http://localhost:9000> and create the scan

- Step 3: Launch the scan by running command provided in the SonarQube web GUI within the project local folder
- Step 4: Check the results in the SonarQube web GUI.



THANK YOU FOR CHOOSING

// HALBORN

