soken

SMART CONTRACT SECURITY AUDIT

Unichain-core

January, 2022

Website: soken.io

Table of Contents

Disclaimer

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws. We took into consideration smart contract based algorithms, as well. Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Soken and its affiliates shall not be held responsible to you or anyone else, nor shall Soken provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties or other conditions other than as set forth in that exclusion and Soken excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Soken disclaims all responsibility and responsibilities and no claim against Soken is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue —important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue —serious bug causes, must be analyzed and fixed.

Limitations

The security audit cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Objective

The objective of this document is to provide the results of the code review of **Unichaincore**. The objective of this code review is to examine the **Unichaincore**, focusing mainly on its security aspects, the risk that they pose to its users and the integrity and confidentiality of the data contained within.
Unichaincore is UniChain is a highly secure and scalable blockchain platform for Smart Society 5.0. This repos implement the core components of UniChain blockchain including unichain-core node, database and key management tools.

Procedure

The scope of the project is as follows:

Application name	unichain-core		Review start	14/01/2022
Objective	Security Code Review		Review end	28/01/2022
Categories	Data/Input Management	✓	Error Handling / Information Leakage	✓
	Authentication Controls	✓	Software Communications	✓
	Session Management	✓	Logging / Auditing	✓
	Authorization Management	✓	Secure Code Design	✓
	Cryptography	✓	Optimized Mode Controls	✓
Comments	N/A			

Audit Details

Project Name: **Uniworld**

Language: **Java**

Blockchain: **Unichain**

Social Profiles

Project Website: **https://unichain.world/**

Project Twitter: **https://twitter.com/unichain_world**

Project Telegram Official: **https://t.me/UnichainWorldOfficial**

Project LinkedIn: **https://www.linkedin.com/company/unichain/**

Project Medium: **https://medium.com/uniworld-io**

Project Youtube: **https://www.youtube.com/c/UniWorldEcosystem**

Whitepaper Review

Unichain technical white paper key points:

- Intro
- Side chain architecture
- DPOS-Hotstuff consensus algorithm
- UniChain native features
  - The native token (URC-30)
  - The native NFT (URC-721/URC-1155)
- Unichain ecosystem & Incentive model
- UniChain Specifications
- Conclusion

The whitepaper of Unichain has been verified on behalf of Soken team.

Whitepaper link:
https://developers.unichain.world/asset/Unichain-whitepaper-v2.pdf

General Overview and Findings

Categories	Status	Total
Data/Input Management		90%
Authentication Controls		90%
Session Management		90%
Authorization Management		100%
Cryptography		80%
Error Handling / Information Leakage		90%
Software Communications		100%
Logging/Auditing		100%
Secure Code Design		100%
Optimized Mode Controls		100%

Total Findings

Critical	High	Medium	Low	Info
0	0	2	2	1

In relation to the control categories, findings were discovered as follows. The total number of findings (5) can be considered as low.

The remaining categories of controls successfully passed the review with no relevant findings. No **critical or high-risk** findings were detected. Among the remaining findings, two mediums were detected. The remaining three were of low and

informative nature. This shows that the impact of the findings varies from one risk level to another.

ID	Control	Result
BG-001	Cross-Site Scripting (XSS)	Medium
BG-002	Insecure Design Object	Medium
BG-003	Use of a Broken or Risky Cryptographic Algorithm	Low
BG-004	Information disclosure	Low
BG-005	Path Traversal	Info

Findings Details

1. Cross-Site Scripting (XSS)

Description	Unsanitized input from, an HTTP parameter flows into println, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).	Threat	
		Vulnerability	
		Impact	
Path	src/main/java/org/unichain/ore/services/http/fullnode/servlet/GetBlockByIdServlet.java		
Line Number	33		

Proof of concept

src/main/java/org/unichain/core/services/http/fullnode/servlet/GetBlockByIdServlet.java

```
23  public class GetBlockByIdServlet extends HttpServlet {
24    @Autowired
25    private Wallet wallet;
26
27    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
28      try {
29        boolean visible = Util.getVisible(request);
30        String input = request.getParameter("value");
31        Block reply = wallet.getBlockById(ByteString.copyFrom(ByteArray.fromHexString(input)));
32        if (reply != null) {
33          response.getWriter().println(Util.printBlock(reply, visible));
34        } else {
```

Impact

The most common attack performed with cross-site scripting involves the disclosure of information stored in user cookies. Typically, a malicious user will craft a client-side script, which -- when parsed by a web browser -- performs some activity (such as

sending all site cookies to a given Email address). This script will be loaded and run by each user visiting the website. Since the site requesting to run the script has to access the cookies in question, the malicious script does also.

Recommendations

- Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
- Convert special characters such as ?, &, /, <, > and spaces to their respective HTML or URL encoded equivalents.
- Give users the option to disable client-side scripts. Redirect invalid requests.
- Detect simultaneous logins, including those from two separate IP addresses, and invalidate those sessions.

2. Insecure Design Object

Description	This Vulnerability occurs because data (eg: Credentials) is being sent without protection or encryption.	Threat	
		Vulnerability	
		Impact	
Path	unichain-core-master\src\main\java\org\unichain\core\config\args\Storage.java		
Line Number	44:59		

Impact

This code reads a password from a storage file and uses the password to connect to a database. Although this code will run correctly, someone with access to the Storage.java file can read the value of a password. This might include an employee who uses this information to breach the system.

Recommendations

Credential storage should be secure and encrypted. Authentication communication must be secure, but storage must be equally secure. Plaintext credentials, insecure hashing, and improper salt functions could introduce flaws.

3. Use of a Broken or Risky Cryptographic Algorithm

Description	Default AES/ECB algorithm (AES/ECB/NoPadding) used in javax.crypto.Cipher.getinstance may be insecure, because equal messages get encrypted o equal data.	Threat	
		Vulnerability	
		Impact	
Path	src/main/java/org/unichain/common/crypto/SymmEncoder.java		
Line Number	13,48,59,152,150,59		

Impact

The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the exposure of sensitive information.

Recommendations

- Avoid using DES, and use AES or another strong algorithm instead
- Beware of outdated hashes, such as MD5 and SHA1 (along with many others); recognize and implement current cryptographic standards instead.
- Increase developer awareness of proper encryption key management and protection and ensure best practice through the applications.

4. Improper Error Handling

Description		Threat	
		Vulnerability	
		Impact	
Path	unichain-core-master\src\main\java\org\unichain\common\crypto\SymmEncoder.java		
Line Number	52,832,844,63,454,167		

Impact

Improper handling of errors can introduce a variety of security problems for a website. The most common problem is when detailed internal error messages such as stack traces, database dumps, and error codes are displayed to the user (hacker).

Recommendations

Many error handling flaws come from default error handlers, which tend to provide more information than is necessary for users. Finding and replacing these error handlers with more secure approaches is an important first step to a secure system. When detailed error messages are required for developers, a secure error handler writes the error details to a log while providing a friendlier message to users – one that avoids revealing sensitive information.

5. Path Traversal

Description	Unsanitized input from a command-line argument flows into java.io.File, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to manipulate arbitrary files.	
Path	**src/main/java/org/unichain/program/DBConvert.java**	
Line Number	**194**	

Proof of concept

src/main/java/org/unichain/program/DBConvert.java

```
188     dbSrc = "unichain/database";
189     dbDst = "unichain-dst/database";
190   } else {
191     dbSrc = args[0];
192     dbDst = args[1];
193   }
194   File dbDirectory = new File(dbSrc);
195   if (!dbDirectory.exists()) {
```

Impact

The attacker may be able to overwrite or create critical files, such as programs, libraries, or important data. If the targeted file is used for a security mechanism, then the attacker may be able to bypass that mechanism. For example, appending a new account at the end of a password file may allow an attacker to bypass authentication. The attacker may be able to read the contents of unexpected files and expose sensitive data. If the targeted file is used for a security

mechanism, then the attacker may be able to bypass that mechanism. For example, the attacker could conduct brute force password guessing attacks by reading a password file, in order to break into an account on the system.

Recommendations

In order to mitigate the attack mentioned above, we must validate the user input and ensure that it does not contain invalid characters. We can then either strip them from the string or return an error.

Conclusion

To conclude, the code review confirmed that the code has a good level from a security point of view, with only a few findings, none of which were critical or high-risk in nature. It is important to highlight that these findings cannot be directly considered security flaws that can be exploited, given that 'Security' is a set of layers and, therefore, several risky findings are necessary to compromise the software.

Soken Contact Info

Website: www.soken.io

Mob: (+1)416-875-4174

32 Britain Street, Toronto, Ontario, Canada

Telegram: @team_soken

GitHub: sokenteam

Twitter: @soken_team