RD
AUDITORS

Glitter Finance Code Review and Security Analysis Report

Customer: Glitter Finance
Prepared on: 28th August 2022
Platform: Solana
Language: RUST

rdauditors.com

Table of Contents

Disclaimer

This document may contain confidential information about its systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities are fixed - upon the decision of the customer.

Document

Name	Project Code Review and Security Analysis Report of Glitter Finance
Platform	Solana
File 1	Config.rs
MD5 hash	d9975ddcd77c713dd4f78901600a292c
SHA256 hash	1bc7294cbfa15394110d8da16fbec399e98240a9c6e1275c12873ea6d1695321
File 2	Logger.rs
MD5 hash	4b862816bd6f1525e4b5e3d06b3d8480
SHA256 hash	c5fc601d4db8fd328da997a9dc18a3e5dff15b6883c64f5466eb51191fbf71ca
File 3	Main.rs
MD5 hash	7dd3e02d57c823725a2a8b1e164fff0e
SHA256 hash	71a21e776bc220f974e2ab89904b7b024f96168b74962786359065d83498ebb8
File 4	Publisher.rs
MD5 hash	0aabdb4d853ba56d9f107773c4114f42

SHA256 hash	c9497f8763cc9d43cf51584a587f506264a4a822fd74c983c12a9c5eae5fdf13
File 5	Req_handler.rs
MD5 hash	33983910d7476966e79e46eaa3ece497
SHA256 hash	fb8a9b81ff4ef840f6df2f916cf421cd6a105d1d4f056832b0292275b6f0dcdf
File 6	Resp_handler.rs
MD5 hash	93b1a0f9543acce88beb06e4f1e773f7
SHA256 hash	018e7b2545a4ee9cbd561b4e818277e88b011f951951a61ede6578785377f0bd
File 7	Server.rs
MD5 hash	bbee9b56debda4c4ebb78cf68afe2b7d
SHA256 hash	26dd6d567d45d9a81ea6ac6b8a8924ba47bad8f52dbcc2e15c7f2c74cddcf6e1
File 8	Subscriber.rs
MD5 hash	55cb7d48d6ce67d4ab93439cedd187af
SHA256 hash	718f00d924bd8a31326ea64d4756d27fde2b928e3f9d58a82e3f151fb9384f7d
File 9	Ws_handler.rs
MD5 hash	0e4574665ca738d1ebbc62277e588eac

SHA256 hash	1b8b48bc6fd39d44b41db3f30a6199e08ad3744402c2ecca8886bc0c0d241185
File 9	Message.rs
MD5 hash	4f8e6d4f4d793f63e822d3381051579d
SHA256 hash	fc026c41c96a04deaa9cfb871f7f4507a50a9134650314816abe1666b52a6555
File 10	Address.rs
MD5 hash	0e06525b1f304acf86b07cf415ed07aa
SHA256 hash	05210e2e9f9e9b3e0a3d883de78bddd1f93a2b46cb3f8a13c4df00bc616ac2e6
File 11	Endpoints.rs
MD5 hash	daa4b3f57ec529b66f99d3c883723ce0
SHA256 hash	67e69bf3caec83b2bcc8581e368e3869f556843b4497595aa3b1a96ab9557fce
File 12	Instruction.rs
MD5 hash	d9323773f58ecaf529445d6b9df21092
SHA256 hash	08164417f7189ef6d6b2ba6f49857edf2651cf30e877ccb8356cbe091b4324a
File 13	Processor.rs

MD5 hash	511f572fff6f46911327859ff5690f2b
SHA256 hash	39983e8eac6242930bdc92e97b98f3f37b92e6c80149769eb05aaee26ae255cb
File 14	Txn_handler.rs
MD5 hash	38edc744153ae5c7fbaf2f726591589e
SHA256 hash	bafd308fd927bdd8fa69ef82fea363e342d7ce28563d4022088ad4cbdd54db76
File 15	Utils.rs
MD5 hash	5168bbe75be6dc95d0cbf06054a16bbf
SHA256 hash	6cef53ab517ea282369cb79083ec98c72adba6c2e94bbd3893b9acd12f71ca5a
File 16	Validations.rs
MD5 hash	4fcca5f4b6243b9d4b267077f5d6a751
SHA256 hash	13dba6c2401297265e6cfcebd5df054f6b1883e0a66bd3e0081d4cbfc8cdf41f
Date	28/08/2022

Introduction

RD Auditors (Consultant) were contracted by Glitter Finance (Customer) to conduct a RUST Solana based Bridge (Validator Based Bridge System) Code Review and Security Analysis. This report represents the findings of the security assessment of the customer`s project codes review conducted between 6th - 28th August 2022.

This project consists of folders/files provided in the below Github link:
https://github.com/Glitter-Finance/glitter-finance-solana-backend-multithread/tree/main/src

Project Scope

The scope of the project is a bridge application and its associated codes. We have scanned this bridge application for commonly known and more specific vulnerabilities (none were found in this report), below are those considered (the full list includes but is not limited to):

- Missing signer
- Integer overflow & underflow
- Arbitrary signed program invocation
- Account confusions
- Other Known / Possible vulnerabilities

This is a 'Privacy First' based solana network using RUST bridge application as the coding language. It is a relatively modern innovative approach towards applications which are both permissionless and privacy-preserving. The lists of known vulnerabilities within the RUST coding language are relatively low, however we have checked/tested all possible areas including logical conflict and code flow projections. Please keep note we were not requested for extensive audit which requires a bundle of test cases framing and rigorous testing which took months together to complete. Hence we performed a generic audit for bugs/security and all.

Executive Summary

According to the assessment, the customer's RUST based bridge application is **well-secured.**

Manual and localized checks are done. All issues were performed by our team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the AS-IS section and all issues found are located in the audit overview section.

We found the following;

Total Issues	0
Critical	0
High	0
Medium	0
Low	0
Very Low	0
Suggestions	3

Code Quality

The libraries within this bridge application are part of a logical algorithm. A library is a different type of bridge application that contains reusable code. Once deployed on the blockchain (only once), it is assigned to a specific address and its properties/methods can be reused many times by other applications.

The Glitter Finance team provided scenario and unit test codes, which could help to determine the integrity of the code in an automated way.

Also, the code is very less commented. Commenting provides rich documentation for functions, return variables and more.

Documentation

The hash of that file is mentioned in the table. As mentioned above, It's less commented code, which could help anyone to quickly understand the programming flow as well as complex code logic.

Comments are very helpful in understanding the overall architecture of the protocol. It also provides a clear overview of the system components, including helpful details, like the lifetime of the background script.

Use of Dependencies

As per our observation, the libraries and other open source codes are used in this bridge application infrastructure. Those were based on well-known industry standard open source projects and even core code blocks that are written well and systematically.

AS-IS Overview

File And Function Level Report

File: Config.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	index	Passed	All Passed	No Issue	Passed

File: Logger.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	logger	Passed	All Passed	No Issue	Passed

File: Main.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	main	Passed	All Passed	No Issue	Passed

File: Publisher.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	new	Passed	All Passed	No Issue	Passed
2	started	Passed	All Passed	No Issue	Passed
3	stopping	Passed	All Passed	No Issue	Passed
4	stopped	Passed	All Passed	No Issue	Passed
5	handle	Passed	All Passed	No Issue	Passed
6	handle	Passed	All Passed	No Issue	Passed
7	handle	Passed	All Passed	No Issue	Passed
8	handle	Passed	All Passed	No Issue	Passed
9	handle	Passed	All Passed	No Issue	Passed

10	handle	Passed	All Passed	No Issue	Passed
11	sol_logs_subscribe	Passed	All Passed	No Issue	Passed
12	sol_accounts_subscribe	Passed	All Passed	No Issue	Passed
13	sol_logs_rpc	Passed	All Passed	No Issue	Passed
14	sol_collector_rpc	Passed	All Passed	No Issue	Passed
15	algo_logs_api	Passed	All Passed	No Issue	Passed

File: Req_handler.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	ws_connection	Passed	All Passed	No Issue	Passed

File: Resp_handler.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	response_creator	Passed	All Passed	No Issue	Passed

File: Server.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	server	Passed	All Passed	No Issue	Passed

File: Subscriber.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	new	Passed	All Passed	No Issue	Passed
2	hb	Passed	All Passed	No Issue	Passed
3	started	Passed	All Passed	No Issue	Passed
4	stopping	Passed	All Passed	No Issue	Passed
5	handle	Passed	All Passed	No Issue	Passed
6	handle	Passed	All Passed	No Issue	Passed

File: Ws_handler.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	new	Passed	All Passed	No Issue	Passed
2	handle	Passed	All Passed	No Issue	Passed
3	handle	Passed	All Passed	No Issue	Passed
4	handle	Passed	All Passed	No Issue	Passed

File: Address.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	new	Passed	All Passed	No Issue	Passed
2	decode_from_string	Passed	All Passed	No Issue	Passed

3	encode_ as_string	Passed	All Passed	No Issue	Passed
4	from_str	Passed	All Passed	No Issue	Passed
5	fmt	Passed	All Passed	No Issue	Passed
6	fmt	Passed	All Passed	No Issue	Passed
7	from	Passed	All Passed	No Issue	Passed

File: Endpoints.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	from_str	Passed	All Passed	No Issue	Passed
2	from	Passed	All Passed	No Issue	Passed
3	from_str	Passed	All Passed	No Issue	Passed
4	from	Passed	All Passed	No Issue	Passed
5	pubsub_logs	Passed	All Passed	No Issue	Passed
6	pubsub_account	Passed	All Passed	No Issue	Passed
7	rpc_logs	Passed	All Passed	No Issue	Passed
8	rpc_account	Passed	All Passed	No Issue	Passed
9	get_cluster_url	Passed	All Passed	No Issue	Passed

File: Instruction.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	unpack	Passed	All Passed	No Issue	Passed
2	default	Passed	All Passed	No Issue	Passed
3	approve_native_bridge_ix	Passed	All Passed	No Issue	Passed
4	cancel_native_bridge_ix	Passed	All Passed	No Issue	Passed
5	release_native_ix	Passed	All Passed	No Issue	Passed
6	approve_token_bridge_ix	Passed	All Passed	No Issue	Passed
7	cancel_token_bridge_ix	Passed	All Passed	No Issue	Passed
8	release_token_ix	Passed	All Passed	No Issue	Passed

File: Process.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	Parse_tx_info	Passed	All Passed	No Issue	Passed
2	create_msg	Passed	All Passed	No Issue	Passed
3	create_ix	Passed	All Passed	No Issue	Passed
4		Passed	All Passed	No Issue	Passed

File: Txn_handler
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	send_confirm_tx	Passed	All Passed	No Issue	Passed
2	get_blockhash	Passed	All Passed	No Issue	Passed
3	confirm_tx	Passed	All Passed	No Issue	Passed
4	Parse_txn_err	Passed	All Passed	No Issue	Passed

File: Utils.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	calc_bridge_fee	Passed	All Passed	No Issue	Passed
2	read_signature_state	Passed	All Passed	No Issue	Passed

File: Validation.rs
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion Passed

Sl.	Function	Observation	Test Report	Conclusion	Score
1	bridge_msg_validation	Passed	All Passed	No Issue	Passed

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to lost tokens etc.
High	High level vulnerabilities are difficult to exploit; however, they also have a significant impact on bridge application execution, e.g. public access to crucial functions.
Medium	Medium level vulnerabilities are important to fix; however, they cannot lead to lost tokens.
Low	Low level vulnerabilities are most related to outdated, unused etc. These code snippets cannot have a significant impact on execution.
Lowest Code Style/ Best Practice	Lowest level vulnerabilities, code style violations and information statements cannot affect bridge application execution and can be ignored.

Audit Findings

Critical:

No critical severity vulnerabilities were found.

High:

No high severity vulnerabilities were found.

Medium:

No medium severity vulnerabilities were found.

Low:

No low severity vulnerabilities were found.

Very Low:

No very low severity vulnerabilities were found.

Suggestions

1) Double check all hard coded values before going to production.
2) Should use more comments with the code blocks.
3) According to the scope and coverage of the project, extensive audit or rigorous tests recommended with plenty of test-cases.

Conclusion

We have used as many tests as possible based on the project scope. The application is written systematically, so it is now ready to go for production.

Since possible test cases can be unlimited and developer level documentation (SRS, Architecture and function level description) were not provided except code flow diagram, for such an extensive bridge application protocol, we provide no such guarantee of future outcomes nor publicly unknown vulnerabilities (which may be detected in the future).

We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

The security state of the reviewed validator controlled bridge application is now “well-secured”.

Note For Application Users

The Software Requirement Specification and developer level documentation were not provided, so our test cases/observations were limited in scope.

We did not perform an extensive audit, it was not under the scope of the request. We do not guarantee any discrepancy raised from any dependent library/macros (external objects) included and/or tempered.

Technical auditing does not guarantee the project's ethical side. Please do your due diligence before investing. Our audit report is never an investment advice.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities.

We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

The bridge applications given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in bridge application source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Because the total number of test cases are unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the application. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of bridge applications.

Technical Disclaimer

bridge applications are deployed and executed on the blockchain. The platform, its programming language, and other software related to the bridge application can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited bridge applications.

Email: info@rdauditors.com

Website: www.rdauditors.com