QuillAudits

Audit Report
June, 2024

For

Table of Content

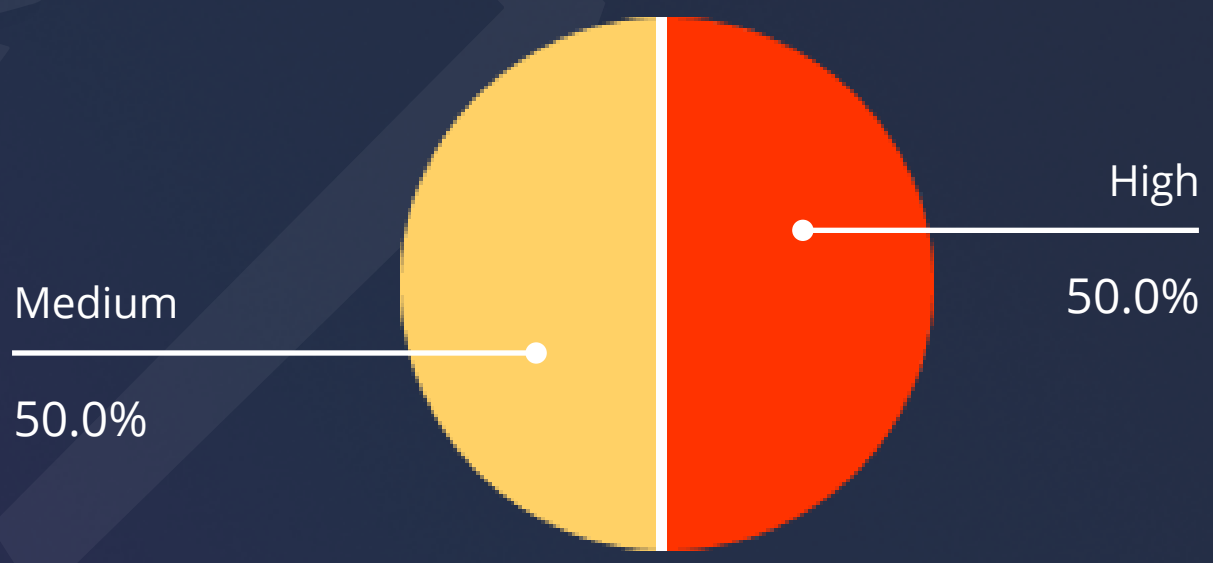
Overview

Project Overview	GariBot is a Telegram Trading Companion f**or Solana (Aptos support coming** soon). Seamlessly conduct trades on the fly with the lowest fees in the market. It has features like DEX Integration, Market Insights, Instant Alerts, Continuous Monitoring and Track your Portfolio.
Scope of Audit	The scope of this pentest was to analyze **GariBot Source Code and TG Bot** for quality, security, and correctness.
Timeline	26th June 2024 - 1st July 2024
Updated Code Received	1st July 2024
Second Review	1st July 2024
In Scope	*T.me/@GariTradingBot* *https://github.com/vikash-chingari/Trading-Bot*
Commit	6de07c9c282d3ebbf5d03fc0e194b515bf91870f
Fixed In	5bfd40ea7744fccc65639932efc508628fec54e9

Number of Issues per Severity

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	1	1	0	0

Security Issues



Checked Vulnerabilities

We scanned the application for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Improper Authentication
- Improper Resource Usage
- Improper Authorization
- Insecure File Uploads
- Insecure Direct Object References
- Client-Side Validation Issues
- Broken Access Controls
- Insecure Cryptographic Storage
- Insufficient Cryptography
- Insufficient Session Expiration
- Information Leakage
- Third-Party Components
- Rate Limit
- Input Validation
- Injection Attacks
- Cross-Site Request Forgery
- Broken Authentication and Session Management
- Insufficient Transport Layer Protection
- Malware
- Denial of Service (DoS) Attacks
- Cross-Site Scripting (XSS)
- Security Misconfiguration
- Unvalidated Redirects and Forwards

And more...

Techniques and Methods

Throughout the pentest of applications, care was taken to ensure:

- Information gathering – Using OSINT tools information concerning the web architecture, information leakage, web service integration, and gathering other associated information related to web server & web services.
- Using Automated tools approach for Pentest like Nessus, Acunetix etc.
- Platform testing and configuration
- Error handling and data validation testing
- Encryption-related protection testing
- Client-side and business logic testing

Tools and Platforms used for Pentest:

- Sonarcube
- Checkmarx
- Postman and many more.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your web app can be exploited. Issues on this level are critical to the web app's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the web app code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Issues Found

High Severity Issues

1. Dependency Confusion Attack

Description

The chingari-aptos package specified in package.json is sourced from a Git repository URL (git+https://bitbucket.org/sumitghosh/web3-utilities.git#chingari-aptos-2.1.5). This setup is vulnerable to a Dependency Confusion attack. An attacker could create a package with the same name (chingari-aptos) and publish it to a public package registry like npm. If the malicious package has a higher version number than the one specified or if there's a mistake in the registry configuration, the package manager might download and install the malicious package instead of the intended one.

Steps to Reproduce

- Review the package.json file for dependencies sourced from Git URLs.
- Note the use of chingari-aptos with a Git repository URL.
- Publish a package named chingari-aptos to the npm registry with a higher version number.
- Run the package manager (e.g., npm install or yarn install).
- Observe that the package manager might download the malicious package from the public registry instead of the intended Git repository.

Impact

- **Remote Code Execution:** An attacker can run arbitrary code in your environment if the malicious package is installed.
- **Data Breach:** Sensitive data can be exfiltrated or compromised.
- **Service Disruption:** Malicious code can cause the application to behave unexpectedly or crash.

Recommendation

Scope Packages: Use scoped packages to reduce the likelihood of name conflicts. Scoped packages include the namespace in the package name (e.g., @company/chingari-aptos).

Status

Resolved

Medium Severity Issues

2. Race Condition in Enable/Disable Solana Settings

Description

A race condition vulnerability exists in the Solana settings enable/disable functionality within the application. This issue occurs when concurrent operations attempt to modify the Solana settings, leading to inconsistent or unintended states. Race conditions can cause data corruption, unexpected behavior, and potential security vulnerabilities.

Steps to Reproduce

1. Open Garibot.
2. Click on Solana Setting and Press on Enable/Disable Function in Auto-Buy.
3. Clicking on it 21 times.

Start at Disable and after 21 times it should be enable. But due to spamming it get's DOS'ed and then result in being disabled.

Impact

Race conditions can lead to-

- **Inconsistent settings:** Enabling or disabling settings may not reflect the actual desired state.

Recommendation

To mitigate this issue, implement the following steps:

- **Mutex or Locking Mechanism:** Use a mutex or locking mechanism to ensure that only one operation can modify the Solana settings at a time.
- **Atomic Operations:** Ensure that the enable/disable operations are atomic, meaning they complete without being interrupted.
- **Concurrency Control:** Implement concurrency control mechanisms like optimistic or pessimistic locking to manage simultaneous requests.

Status

Resolved

Closing Summary

In this report, we have considered the security of the GariBot. We performed our audit according to the procedure described above.

Two issues of High and medium severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.In the End, GariBot Team resolved both Issues.

Disclaimer

QuillAudits Dapp security audit provides services to help identify and mitigate potential security risks in GariBot Platform. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of GariBot Platform. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your Platform for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the GariBot team to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
June, 2024

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com