Snekbot Audit Report

2025-05-30

Table of Contents

1 - Audit Manifest

Please find below the list of pinned software dependencies and repositories which were covered by the audit.

1.a - Software versions and dependencies

Software	Version	Commit
Snekbot	-	9d1c084acdb9cd3369e1e992373b44d71acf2cf1
github.com/go-telegram-bot-api/telegram-bot-api	5.5.1	4fe428c77a68a9903ab5bfcaecb978003ce0424b
github.com/bwmarrin/discordgo	0.28.1	da9e191069d09e1b467145f5758d9b11cb9cca0d

1.b - Repositories

Repository	Git Ref
https://github.com/snekbotio/bot	9d1c084acdb9cd3369e1e992373b44d71acf2cf1

2 - Specification

This specification details the goals of the project which we digest and convert into an informal specification or description of the pieces and how they interact to produce the desired outcomes.

2.a - High Level Objectives

1. Provide a simple chat bot interface for trading Cardano native tokens
2. Provide a custodial wallet associated with a chat user
3. Provide withdrawals to chat user's own wallet

2.b - Informal Specification (what it is/how it works)

1. Discord
2. Hooks (Oura and USD price webhook receiver)
3. Server (Telegram)
4. Alerts
5. Wallet
6. Redis
7. PostgreSQL

3 - Findings Summary

ID	Title	Severity	Status
BL-001	Misuse of ServerConfig.PublicKeyCallback may cause authorization bypass in golang.org/x/crypto	Low	Resolved
BL-002	Non-linear parsing of case-insensitive content in golang.org/x/net/html	Low	Resolved
BL-003	Discord command handler missing type conversion error check	Medium	Acknowledged
BL-004	HTTP Proxy bypass using IPv6 Zone IDs in golang.org/x/net	Low	Resolved
BL-005	go-redis allows potential out of order responses when CLIENT SETINFO times out during connection establishment	Low	Resolved
BL-006	Missing authentication on alerts service endpoints	Low	Acknowledged
BL-007	Use of static initialization vector with AES CBC	High	Resolved
BL-008	Prevent crashes by guarding against nil	Low	In Progress

3.a - BL-001 Misuse of ServerConfig.PublicKeyCallback may cause authorization bypass in golang.org/x/crypto

Category	Commit Fixed	Severity	Status
Auth bypass	26da29fe01481f9bbbdd6b53c40478bccae5f3cc	Low	Resolved

3.a.i - Description

GitHub Advisory: https://github.com/advisories/GHSA-v778-237x-gjrc

Applications and libraries which misuse the ServerConfig.PublicKeyCallback callback may be susceptible to an authorization bypass.

The documentation for ServerConfig.PublicKeyCallback says that "A call to this function does not guarantee that the key offered is in fact used to authenticate." Specifically, the SSH protocol allows clients to inquire about whether a public key is acceptable before proving control of the corresponding private key. PublicKeyCallback may be called with multiple keys, and the order in which the keys were provided cannot be used to infer which key the client successfully authenticated with, if any. Some applications, which store the key(s) passed to PublicKeyCallback (or derived information) and make security relevant determinations based on it once the connection is established, may make incorrect assumptions.

For example, an attacker may send public keys A and B, and then authenticate with A. PublicKeyCallback would be called only twice, first with A and then with B. A vulnerable application may then make authorization decisions based on key B for which the attacker does not actually control the private key.

Since this API is widely misused, as a partial mitigation golang.org/x/crypto@v0.31.0 enforces the property that, when successfully authenticating via public key, the last key passed to ServerConfig.PublicKeyCallback will be the key used to authenticate the connection. PublicKeyCallback will now be called multiple times with the same key, if necessary. Note that the client may still not control the last key passed to PublicKeyCallback if the connection is then authenticated with a different method, such as PasswordCallback, KeyboardInteractiveCallback, or NoClientAuth.

Users should be using the Extensions field of the Permissions return value from the various authentication callbacks to record data associated with the authentication attempt instead of referencing external state. Once the connection is established the state corresponding to the successful authentication attempt can be retrieved via the ServerConn.Permissions field. Note

that some third-party libraries misuse the Permissions type by sharing it across authentication attempts; users of third-party libraries should refer to the relevant projects for guidance.

3.a.ii - Recommendation

This is classified as a critical (CVSS v3 9.1/10) vulnerability. This library is used by github.com/bwmarrin/discordgo which is used for accessing the Discord API. Upon inspection, the vulnerable code paths are not executed. Marking as low priority.

Recommendation is to update using the dependabot provided pull request.

3.a.iii - Resolution

Updated using the provided pull request.

3.b - BL-002 Non-linear parsing of case-insensitive content in golang.org/x/net/html

Category	Commit Fixed	Severity	Status
Denial of Service	1af3be11a53d989c793451967dd115e5e4a3db4a	Low	Resolved

3.b.i - Description

GitHub Advisory: https://github.com/advisories/GHSA-w32m-9786-jp63

An attacker can craft an input to the Parse functions that would be processed non-linearly with respect to its length, resulting in extremely slow parsing. This could cause a denial of service.

3.b.ii - Recommendation

This is classified as a high (CVSS v3 8.7/10) vulnerability. This vulnerability is explicitly in the HTML parsing functions, which is not used in snekbot or dependency libraries. Marking as low priority.

Recommendation is to update using the dependabot provided pull request.

3.b.iii - Resolution

Updated using the provided pull request.

3.c - BL-003 Discord command handler missing type conversion error check

Category	Commit	Severity	Status
Crash		Medium	Acknowledged

3.c.i - Description

There is an unchecked error in infra/handler/discord/command_handler.go line 336, which should be caught by golangci-lint execution. As there is no pull request automation in the repository, it's likely this manual testing step was not always executed. Marking as medium priority due to process improvement.

3.c.ii - Recommendation

Add automatic golangci-lint workflow on pull request.

Change volume is low enough that even with a private repository on a free plan, it will fit well within the allotted GitHub Hosted Runner minutes. Example Workflow

Implement handlers for type conversion.
In examples/slash_commands/main.go, there is an example of converting the slice into a map, and then checking the type and using the StringValue() utility function. Unfortunately, this function panics when given a bad type. However, it does show that we can use the Type field of ApplicationCommandInteractionDataOption to compare against ApplicationCommandOptionString and only performing the type conversion if it matches. This would allow for providing the user with better failure error messages.

3.c.iii - Resolution

“While you're right, this is a special case where we've constructed the ApplicationCommandInteractionDataOption so there is Type is going to be 0. Error handling in general needs to be improved and better communicated to the user, but that will require a significant refactor. I'm OK with the default behavior being page 0 and swallowing the error.”

Automatic golangci-lint workflows were added in 8048ebabc01483316cd036e2fdb94ac406271241 to catch future issues.

3.d - BL-004 HTTP Proxy bypass using IPv6 Zone IDs in golang.org/x/net

Category	Commit	Severity	Status
Info Disclosure	9ea64bef96fcdc6eb3d62e52f3ac9e17e82c4dd7	Low	Resolved

3.d.i - Description

GitHub Advisory: https://github.com/advisories/GHSA-qxp5-gwg8-xv66

Matching of hosts against proxy patterns can improperly treat an IPv6 zone ID as a hostname component. For example, when the NO_PROXY environment variable is set to "*.example.com", a request to "[::1%25.example.com]:80` will incorrectly match and not be proxied.

3.d.ii - Recommendation

This is classified as a moderate (CVSS v3 4.4/10) vulnerability. This vulnerability is explicitly in the HTTP proxy handing function, which is not used in snekbot or dependency libraries. Marking as low priority.

Recommendation is to update using the dependabot provided pull request.

3.d.iii - Resolution

Updated using the provided pull request.

While resolving this issue, another issue was also resolved in an unused code path in the golang.org/x/net library: https://github.com/advisories/GHSA-vvgc-356p-c3xw

3.e - BL-005 go-redis allows potential out of order responses when CLIENT SETINFO times out during connection establishment

Category	Commit	Severity	Status
Info Disclosure	076662869aac844d0adeda9048b6caedbb56cede	Low	Resolved

3.e.i - Description

The issue only occurs when the CLIENT SETINFO command times out during connection establishment. The following circumstances can cause such a timeout:

1. The client is configured to transmit its identity
2. There are network connectivity issues
3. The client was configured with aggressive timeouts

The impact differs by use case:

- Sticky connections: Rather than using a connection from the pool on-demand, the caller can stick with a connection. Then you receive persistent out-of-order responses for the lifetime of the connection.
- Pipelines: All commands in the pipeline receive incorrect responses.
- Default connection pool usage without pipelining: When used with the default ConnPool once a connection is returned after use with ConnPool#Put the read buffer will be checked and the connection will be marked as bad due to the unread data. This means that at most one out-of-order response before the connection is discarded.

3.e.ii - Recommendation

This is classified as a low (CVSS v3 3.7/10) vulnerability. This vulnerability is limited to a connection timeout error condition with limited avenue for remote exploitation. This has the potential to expose ordering issues by returning out of order data from the cache. Marking as low priority.

Recommendation is to update using the dependabot provided pull request.

3.e.iii - Resolution

Updated using the provided pull request.

3.f - BL-006 Missing authentication on alerts service endpoints

Category	Commit	Severity	Status
Auth Bypass		Low	Acknowledged

3.f.i - Description

The alerts service allows unauthenticated POST requests on the incoming HTTP routes. This service is intended as a message gateway between the snekx.io service and Telegram. These endpoints could be used for generating fake messages to Telegram. However, an attacker would need to determine the correct payload shape and discover the service, which limits this to an impersonation vulnerability with an extremely limited attack surface which produces a limited outcome requiring additional social engineering to exploit, such as convincing others to make bad trades based on incorrect information. Marking as low.

3.f.ii - Recommendation

Recommendation is to introduce some form of authentication for these endpoints. If this is not feasible, restricting by IP address or some other manner should be done.

This may not be possible and may simply be an accepted risk due to vendor requirements as we cannot dictate changes to snekx here.

3.f.iii - Resolution

Due to the low probability of this being abused and the limited impact given this feature’s limited use, the risk is accepted.

3.g - BL-007 Use of static initialization vector with AES CBC

Category	Commit	Severity	Status
Info Disclosure	906f01379eccd4bb0b5e71843c16703f8be3ff12	High	Resolved

3.g.i - Description

A static initialization vector (IV) with AES CBC used for multiple encryption operations can be used to reveal patterns in the plaintext. Since this encryption is being used to store seed phrases, which share a common library of allowed words, it's more vulnerable than a typical password to pattern cryptanalysis. For the best security, a randomly generated IV should be used for each encryption operation. The IV is safe to be stored along with the encrypted text, as it's simply a block of dummy data padding. Exploitation would require access to the snekbot database. Marking as high.

3.g.ii - Recommendation

Recommendation is to generate and store the IV along with each encrypted message.

https://en.wikipedia.org/wiki/Initialization_vector

3.g.iii - Resolution

This was resolved by storing the IV along with each message in commit 906f01379eccd4bb0b5e71843c16703f8be3ff12 in the upstream repository.

3.h - BL-008 Prevent crashes by guarding against nil

Category	Commit	Severity	Status
Crash		Low	In Progress

3.h.i - Description

Errors which can lead to application stability problems can be detected using Nilaway, from Uber. The nature of these errors mean they are rarely a security issue, but a deep check was done on areas in the report specifically dealing with encryption, secrets, database, or authentication. Upon further investigation, we found no issues where a nil pointer could result in a bypass of authentication or authorization, leak of secrets or credentials, or manipulation of data into the database. Marking as low.

3.h.ii - Recommendation

Blink Labs uses nilaway regularly to detect possible crash paths in our code. We recommend performing these checks yourself periodically to reduce crash vectors.

https://github.com/uber-go/nilaway

3.h.iii - Resolution

Due to the nature of the deployment infrastructure in place, crashes have negligible impact on users. The current risk is accepted while work is in progress to scan and address the issues over time.

4 - Appendix

4.a - Disclaimer

This Software Repository Security Audit Report (“Report”) is provided on an “as is” basis, for informational purposes only, and should not be construed as investment advice or any other kind of advice on legal, financial, or other matters. The entities and individuals involved in preparing this Report (“Auditors”) do not guarantee the accuracy, completeness, or usefulness of the information provided herein and shall not be held liable for any contents, errors, omissions, or inaccuracies in this Report or for any actions taken in reliance thereon.

The Auditors make no claims, promises, or guarantees about the absolute security of the smart con- tracts audited and the underlying code. The findings, interpretations, and conclusions presented in this Report are based on the best efforts of the Auditors and reflect their professional judgment at the time of the audit. The blockchain and cryptocurrency landscape is rapidly evolving, and new vulnerabilities may emerge that were not identified or considered at the time of the audit. As such, this Report should not be considered as a comprehensive guarantee of the audited code’ security.

The Auditors disclaim, to the fullest extent permitted by law, any and all warranties, whether express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement. The Auditors shall not be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this Report, even if advised of the possibility of such damage.

This Report is not exhaustive and is subject to change without notice. The Auditors reserve the right to update, modify, or revise this Report based on new information, subsequent developments, or further analysis. The Auditors encourage all interested parties to conduct their own independent research and due diligence when evaluating the security of smart contracts.

By using or relying on this Report, you agree to indemnify and hold harmless the Auditors from any claim, demand, action, damage, loss, cost, or expense, including attorney fees, arising out of or relating to your use of or reliance on this Report.

If you have any questions or require further clarification regarding this Report, please contact the support@blinklabs.io.

4.b - Issue Guide

4.b.i - Severity

Severity	Description
Critical	Critical issues highlight exploits, bugs, loss of funds, or other vulnerabilities that prevent the App from working as intended. These issues have no workaround.
High	High issues highlight exploits, bugs, or other vulnerabilities that cause unexpected transaction failures or may be used to trick general users of the App. Apps with High issues may still be functional.
Medium	Medium issues highlight edge cases where a user can purposefully use the App in a non-incentivized way and often lead to a disadvantage for the user.
Low	Low issues highlight cases where user impact is limited and cannot be abused for gain or abuse which negatively impacts other users.
Info	These are not issues. These are just pieces of information that are beneficial to the App creator, or should be kept in mind for the end user. These are not necessarily acted on or have a resolution, they are logged for the completeness of the audit.

4.b.ii - Status Status

Status	Description
Resolved	Issues that have been fixed by the project team.
Mitigated	Issues that have a partial mitigation, and are now vulnerable in only extreme corner cases.
Acknowledged	Issues that have been acknowledged or partially fixed by the project team. Projects can decide to not fix issues for whatever reason.
Identified	Issues that have been identified by the audit team. These are waiting for a response from the project team.

4.d - About Us

Blink Labs is focused on creating open source software and custom solutions on the Cardano blockchain. Our seasoned team at Blink Labs possesses a diverse range of expertise acquired from decades of experience across a variety of industries. Our experience ranges projects of all scales, from one man solopreneurs to global-scale projects in communications, storage, cloud computing, big data analytics, and the highly regulated fields of global life science and advertising. We harness this wealth of experience to craft robust and scalable systems tailored specifically for the Cardano blockchain.

Our primary mission is to deliver top-tier software and services to enhance the Cardano ecosystem, streamlining time-to-market and bolstering reliability through open source solutions. We are dedicated to building high-quality, highly available, and repeatable solutions that empower the Cardano community and contribute to the blockchain's success.

4.d.i - Links

Customer - https://snekbot.io
Blink Labs - https://blinklabs.io