SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

AUDIT

SECURITY ASSESSMENT

23. July, 2024

FOR

VITRA

Introduction

SolidProof.io is a brand of the officially registered company FutureVisions Deutschland, based in Germany. We mainly focus on Blockchain Security, such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assesses potential security issues in the smart contracts implementations, reviews for inconsistencies between the code base and the whitepaper/documentation, and provides suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should they be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should they be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io does not cover testing or auditing the integration with external contracts or services (such as Unicrypt, Uniswap, PancakeSwap, etc.).

SolidProof.io Audits does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed or any indication of the technology proprietors. SolidProof Audits should not be used to make decisions about investment or involvement with any particular project. These reports do not provide investment advice, nor should they be leveraged as investment advice.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual is responsible for their own due diligence and continuous security. SolidProof does not claim any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Vitra
Website	https://vitrastudios.com/
About the project	Vitra Studios is more than just blocks and transactions. Stay updated with the latest news and advancements in our cutting-edge blockchain network. Join us as we pioneer innovations and drive progress in the world of decentralized technology.
Chain	Vitra, powered by go-opera
Language	Go
Codebase Link	Provided by file
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/SHG_VITRA
Twitter	https://x.com/VitraStudios
Facebook	https://www.facebook.com/people/Vitra-Studios/61554658419906/
Instagram	https://www.instagram.com/vitrastudios/
Github	TBA
Reddit	N/A
Medium	N/A
Discord	https://discord.com/invite/ng3pAnswEa
Youtube	https://www.youtube.com/@VitraStudios
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	17. July 2024	• Layout Project • Automated- /Manual-Security Testing • Summary

This audit report provides a detailed security analysis of the Go codebase used in the project, particularly focusing on potential vulnerabilities to external malicious interference with the program's functions. This analysis did not cover functional testing (or unit testing) of the program's logic. Therefore, we cannot assure complete logical correctness of the code, as we did not perform functional tests on it. This includes the internal calculations in the algorithms implemented in the codebase.

Scope of Work

We aim to conduct a comprehensive security assessment of the provided repository, including a fork of the go-opera and custom modifications made by the Vitra team. This assessment aims to identify and mitigate potential security vulnerabilities, ensure code integrity, and verify the robustness of modifications to support secure, reliable, and efficient operations.

The repository consists of:

- The original go-opera source code
- Custom modifications and enhancements made to the codebase
- Integration points and dependencies with external systems or libraries

Repository	Commit
TBA	TBA

Imported packages

Used code from other Frameworks. More are imported indirectly.

- github.com/Fantom-foundation/lachesis-base v0.0.0-20230817040848-1326ba9aa59b
- github.com/cespare/cp v1.1.1
- github.com/davecgh/go-spew v1.1.1
- github.com/deckarep/golang-set v1.7.1
- github.com/docker/docker v1.13.1
- github.com/dvyukov/go-fuzz v0.0.0-20201127111758-49e582c6c23d
  - Replaced with: github.com/guzenok/go-fuzz v0.0.0-20210201043429-a8e90a2a4f88
- github.com/ethereum/go-ethereum v1.10.8
  - Replaced with: github.com/Fantom-foundation/go-ethereum v1.10.8-ftm-rc12
- github.com/evalphobia/logrus_sentry v0.8.2
- github.com/fjl/memsize v0.0.0-20190710130421-bcb5799ab5e5
- github.com/golang/mock v1.6.0
- github.com/hashicorp/golang-lru v0.5.5-0.20210104140557-80c98217689d
- github.com/holiman/bloomfilter/v2 v2.0.3
- github.com/mattn/go-colorable v0.1.8
- github.com/mattn/go-isatty v0.0.12
- github.com/naoina/toml v0.1.2-0.20170918210437-9fafd6967416
- github.com/opentracing/opentracing-go v1.1.0
- github.com/pkg/errors v0.9.1
- github.com/sirupsen/logrus v1.6.0
- github.com/status-im/keycard-go v0.0.0-20190424133014-d95853db0f48
- github.com/stretchr/testify v1.8.1
- github.com/syndtr/goleveldb v1.0.1-0.20210305035536-64b5b1c73954
- github.com/tyler-smith/go-bip39 v1.0.2
- github.com/uber/jaeger-client-go v2.20.1+incompatible
- github.com/uber/jaeger-lib v2.2.0+incompatible
- golang.org/x/crypto v0.7.0
- golang.org/x/sys v0.6.0
- gopkg.in/urfave/cli.v1 v1.20.0

Note for Investors: We have only audited the Go dependencies listed in the above scope. We have not reviewed any additional dependencies related to the project that are not included in our audit scope, and we cannot comment on their security. We are not responsible for any security issues arising from these unreviewed dependencies.

Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit vulnerability and the impact of that event on the organization or system. The risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the Vitra Studios blockchain repository audit, our focus was meticulously dedicated to identifying security vulnerabilities, ensuring code quality, and verifying adherence to specifications and best practices. Our audit was conducted by a seasoned team of penetration testers and blockchain developers with extensive experience in the go-opera and go-ethereum framework and smart contract security.

Each file within the repository was subjected to a thorough manual examination. While automated tools were employed, their usage was strategically limited to augment the efficiency and effectiveness of the manual review process rather than replace it.

Methodology

The auditing process follows a routine series of steps:

- Specification Review: Our team meticulously reviewed all relevant documentation, specifications, and guidelines provided initially. This ensured a profound understanding of the blockchain's architecture, scope, and intended functionalities.

- Manual Code Inspection: The source code was examined line by line. This intensive review aimed to uncover potential security vulnerabilities that could be exploited maliciously.

- Specification Conformance: The code was rigorously compared against the provided specifications to confirm its fidelity in performing as described, ensuring the implementation accurately reflected the intended design and requirements.

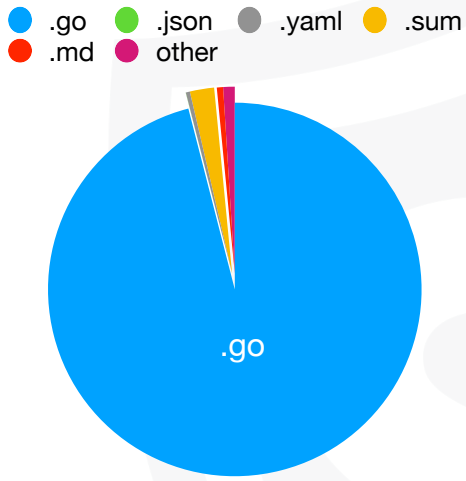
- Test Coverage Analysis: We analyzed the data to ascertain the extent of the test cases' coverage over the codebase. This involved determining how much code was executed during these tests to identify untested paths.

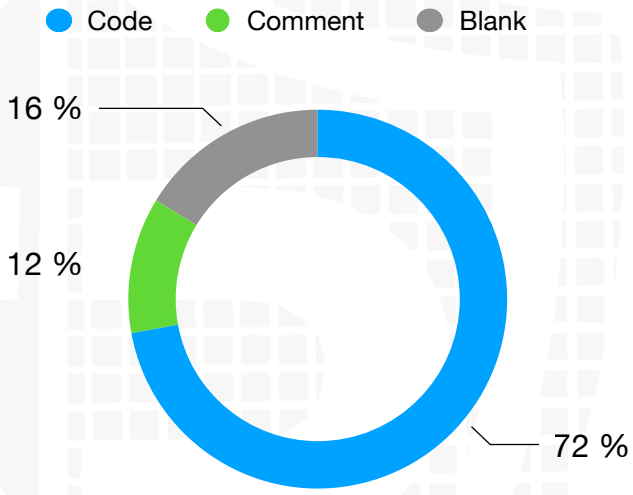
- Symbolic Execution: This technique was applied to analyze how different inputs affect the code execution paths. It helped understand the conditions under which various program parts would execute, revealing potential vulnerabilities.

Metrics

Codebase Composition Overview

This report section analyzes the project's codebase, primarily composed of Go (81%), with extensive comments and documentation, especially in .proto files. The distribution shows 77% code, 14% comments, and 10% blank lines, indicating a well-documented and maintainable structure.





Extension	Total Code	Total Comment	Total Blank	Percent
.go	49.807	8.194	11.477	96
.sum	1.078	5	0	2.1
.md	280	15	98	0.54
.yml	184	2	35	0.35
other				

Cyclomatic Complexity

Cyclomatic Complexity is a software metric used to quantify the complexity of a program. This metric, introduced by Thomas J. McCabe in 1976, measures the number of linearly independent paths through a program's source code. In practical terms, it is determined by the control flow graph of the program, where nodes represent blocks of code that do not contain any control flow keywords (like if, for, switch, case, etc.), and edges represent the flow of execution. The cyclomatic complexity is calculated using the formula: $V(G)=E-N+2P$

Top 25, Highest Cyclomatic Complexity Score

Score	Function	File and Line Number
77	gossip (*handler).handleMsg	gossip/handler.go:989:1
60	gossip consensusCallbackBeginBlockFn	gossip/c_block_callbacks.go:83:1
50	(*Checker).validateMP	eventcheck/basiccheck/ basic_check.go:77:1
43	launcher exportGenesis	cmd/opera/launcher/ genesiscmd.go:205:1
40	filters TestUnmarshalJSONNewFilterArgs	gossip/filters/api_test.go:28:1
32	evmstore (*Store).CheckEvm	gossip/evmstore/utils.go:31:1
30	evmwriter (PreCompiledContract).Run	opera/contracts/evmwriter/ evm_writer.go:56:1
30	ethapi DoEstimateGas	ethapi/api.go:1076:1
29	integration migrateLegacyDBs	integration/legacy_migrate.go:196:1
29	evmcore testTransactionQueueTimeLimiting	evmcore/tx_pool_test.go:978:1
29	heavycheck (*Checker).ValidateEvent	eventcheck/heavycheck/ heavy_check.go:210:1
28	emitter (*Emitter).createEvent	gossip/emitter/emitter.go:290:1
28	evmcore TestTransactionPostponing	evmcore/tx_pool_test.go:675:1
26	emitter (*Emitter).isAllowedToEmit	gossip/emitter/control.go:44:1
26	ethapi (*TransactionArgs).setDefaults	ethapi/transaction_args.go:79:1
24	filters TestFilters	gossip/filters/filter_test.go:117:1
24	filters (*FilterCriteria).UnmarshalJSON	gossip/filters/api.go:484:1
24	evmcore TestTransactionPoolRepricingDynamicFee	evmcore/tx_pool_test.go:1473:1

Score	Function	File and Line Number
24	evmcore/tx_pool_test.go:1349:1	evmcore/tx_pool_test.go:1349:1
23	cser TestVals	utils/cser/binary_test.go:150:1
23	evmcore (*TxPool).validateTx	evmcore/tx_pool.go:585:1
23	launcher dbTransform	cmd/opera/launcher/db-transform.go:23:1
22	(*fastReflection_Proposal).ProtoMethods	integration/assembly.go:165:1
22	integration makeEngine	integration/assembly.go:165:1

For the project under review, the Average Cyclomatic Complexity score is **2.76**, indicating a low overall complexity and suggesting that the codebase is relatively straightforward to manage and maintain. A lower ACC score generally reflects simpler and more easily understandable code, which can lead to fewer bugs, easier code reviews, and more efficient development processes. This improvement in complexity can be beneficial for long-term project sustainability and developer productivity.

Overall tested functions

In the comprehensive audit conducted on the project, both automated tools and manual inspection techniques scrutinized **587** public functions from **391** files.

Call graph

The call graph generated during the audit is extensive and detailed, reflecting the complex interactions within the codebase. Due to its size and complexity, it is too large to be effectively displayed within the confines of this report.

Audit Results

Final Words & Comments

As we conclude this audit of the Vitra Studios blockchain project, we would like to commend the development team's efforts in creating a robust and multifaceted platform. Our detailed analysis focused on identifying security vulnerabilities, ensuring code quality, and verifying adherence to specifications and best practices. Throughout this process, we have noted several key points and would like to offer some final thoughts and recommendations.

Dependency Review

Our analysis identified 14 potential vulnerabilities in 14 dependencies. Ten were deemed significant enough to be explicitly listed in the issues table for immediate attention, with 13 high issues. Additionally, four dependencies were identified with a total of 6 medium issues. These dependencies pose potential security risks and should be addressed promptly to mitigate any adverse impacts on the system's security and stability.

Current Codebase Observations

The Vitra Studios blockchain is based on the go-opera project from the Fantom Foundation. However, it is important to highlight that the original go-opera project is in the process of being archived. The development and maintenance of this project have significantly slowed, with the master branch not receiving any commits in the past two years. Additionally, all other branches are listed as stale.

Security Vulnerabilities

Our audit revealed that the current codebase utilizes an outdated version of geth, which is known to have multiple security vulnerabilities. These vulnerabilities include issues with p2p networking, ping functionality, and potential goroutine exhaustion, which pose significant risks to the stability and security of the blockchain network.

Recommendation for Codebase Transition

Given the aforementioned issues and the fact that the development focus has shifted, we strongly recommend transitioning to the project's new codebase, which is now maintained at https://github.com/Fantom-foundation/Sonic. The Sonic project represents the next evolution of the Fantom blockchain and addresses many of the security concerns present in the older go-opera codebase.

Future Development

The upgrade to Sonic is already in progress, and we encourage the Vitra Studios development team to participate actively in this transition. Once fully implemented, the new chain will provide enhanced security features, improved performance, and a more sustainable development environment. Embracing this change will mitigate current vulnerabilities and position Vitra Studios at the forefront of blockchain technology, ensuring a secure and scalable platform for its users.

Conclusion

In conclusion, while the Vitra Studios blockchain project has demonstrated considerable potential, the necessity of transitioning to the Sonic codebase cannot be overstated. This strategic move will address current security vulnerabilities and align the project with the latest advancements in blockchain technology. We commend the Vitra Studios team for their dedication and recommend prioritizing this upgrade to ensure their platform's continued success and security.

Critical issues

No critical issues

High issues

#1 | Outdated go-ethereum Version

File	Severity	Location	Status
Main	High	Dependency	Open

Description:
The current codebase utilizes an outdated version of go-ethereum (v1.10.8). The latest version is v1.14.7, which includes several important security fixes and improvements. Using the older version poses significant security risks, including vulnerabilities related to p2p networking, ping functionality, and potential goroutine exhaustion.

Changelog go-ethereum v1.14.7:
https://github.com/ethereum/go-ethereum/releases/tag/v1.14.7

Advisory:
To mitigate these security vulnerabilities, upgrade to go-ethereum v1.14.7 or later. For more information, please refer to the advisory.

#2 | Deprecated dependencies

File	Severity	Location	Status
Main	High	Dependency	Open

Description:
The current codebase uses several deprecated dependencies, each with known security vulnerabilities. It is critical to update these dependencies to mitigate potential security risks. Below is a list of the identified issues in the dependencies, along with their CVEs and descriptions:

Dependency: go:github.com/btcsuite/btcd:v0.20.1-beta

- Upgrade to: 0.23.3
- **CVE-2022-39389, Score: 6.5:** Lightning Network Daemon (lnd) is an implementation of a lightning bitcoin overlay network node. All lnd nodes versions prior to 0.15.4-beta are vulnerable to a block parsing bug that can cause a node to enter a degraded state once encountered. In this degraded state, nodes can continue to make payments and forward HTLCs, and close out channels. Opening channels is prohibited, and also on chain transaction events will be undetected. This can cause loss of funds if a CSV expiry is researched during a breach attempt or a CLTV delta expires forgetting the funds in the HTLC. Users are advised to upgrade.

Users unable to upgrade may use the lncli updatechanpolicy RPC call to increase their CLTV value to a very high amount or increase their fee policies. This will prevent nodes from routing through your node, meaning that no pending HTLCs can be present. Note: This vulnerability affects the base package btcd versions prior to 0.22.3 and 0.23.x prior to 0.23.3.

- **CVE-2022-44797, Score: 9.8:** btcd before 0.23.2, as used in Lightning Labs lnd before 0.15.2-beta and other Bitcoin-related products, mishandles witness size checking.

Dependency: go:github.com/consensys/gnark-crypto:v0.4.1-0.20210426202927-39ac3d4b3f1f

- Upgrade to: 0.12.1
- **Cx42455c0d-6fe9, Score: 9.8:** github.com/consensys/gnark-crypto exponentiation in the pairing target group GT using GLV can give incorrect results in versions prior to 0.12.1. When the exponent is bigger than r, the group order of the pairing target group GT, the exponentiation à la GLV (ExpGLV) can sometimes give incorrect results compared to normal exponentiation (Exp). The issue impacts all users using ExpGLV for exponentiations in GT. This does not impact Exp and ExpCyclotomic which are sound. Also note that GLV methods in G1 and G2 are sound and not impacted.
- **CVE-2023-44273, Score: 9.8:** The github.com/consensys/gnark-crypto in versions prior to 0.12.0 allows Signature Malleability. This occurs because the deserialization of "EdDSA" and "ECDSA" signatures does not ensure that the data is in a certain interval.

Dependency: go:github.com/dgrijalva/jwt-go:v3.2.0+incompatible

- **CVE-2020-26160, Score: 7.5:** jwt-go before 4.0.0-preview1 allows attackers to bypass intended access restrictions in situations with []string{} for m["aud"] (which is allowed by the specification). Because the type assertion fails, "" is the value of aud. This is a security problem if the JWT token is presented to a service that lacks its own audience check.

Dependency: go:github.com/docker/docker:v1.13.1

- Upgrade to: 26.0.0-rc3
- **CVE-2023-28840, Score: 8.7:** Moby is an open-source container framework developed by Docker Inc. that is distributed as Docker, Mirantis Container Runtime, and various other downstream projects/ products.

Dependency: go:github.com/gin-gonic/gin:v1.4.0

- Upgrade to: 1.9.0
- **CVE-2023-26125, Score: 9.8:** Versions of the package github.com/gin-gonic/gin prior to 1.9.0 are vulnerable to Improper Input Validation by allowing an attacker to use a specially crafted request via the "X-Forwarded-Prefix" header, potentially leading to cache poisoning. Note: Although this issue does not pose a significant threat on its own, it can serve as an input vector for other more impactful vulnerabilities. However, successful exploitation may depend on the server configuration and whether the header is used in the application logic.
- **CVE-2020-28483, Score: 7.1:** This affects all versions of package github.com/gin-gonic/gin. When gin is exposed directly to the internet, a client's IP can be spoofed by setting the X-Forwarded-For header.
- **CVE-2020-36567, Score: 7.5:** Unsanitized input in the default logger in github.com/gin-gonic/gin prior to v1.6.0 allows remote attackers to inject arbitrary log lines.

Dependency: go:github.com/kataras/iris/v12:v12.0.1

- Upgrade to: 12.2.1
- **CVE-2021-23772, Score: 8.8:** All versions of github.com/kataras/iris and github.com/kataras/iris/v12 before 12.2.0-alpha8 are vulnerable to unsafe handling of file names during upload using UploadFormFiles method. This vulnerability allows attackers to write to arbitrary locations outside the designated target folder.

Dependency: go:github.com/labstack/echo/v4:v4.2.1

- **CVE-2022-40083, Score: 9.6:** Labstack Echo versions prior to 4.9.0 was discovered to contain an open redirect vulnerability via the Static Handler component. This vulnerability can be leveraged by attackers to cause a Server-Side Request Forgery (SSRF).

Dependency: go:github.com/valyala/fasthttp:v1.6.0

- Upgrade to: 1.34.0
- **CVE-2022-21221, Score: 7.5:** The package github.com/valyala/fasthttp before 1.34.0 are vulnerable to Directory Traversal via the ServeFile function, due to improper sanitization. It is possible to be exploited by using a backslash %5c character in the path.
- **Note:** This security issue only impacts Windows users.

Dependency: go:golang.org/x/crypto:v0.7.0

- Upgrade to: 0.23.0
- **CVE-2023-42818, Score: 9.8:** JumpServer is an open-source bastion host. When users enable MFA and use a public key for authentication, the Koko SSH server does not verify the corresponding SSH private key. An attacker could exploit a vulnerability by utilizing a disclosed public key to attempt brute-force authentication against the SSH service. This issue has been patched in versions 3.6.5 and 3.5.6. Users are advised to upgrade. There

are no known workarounds for this issue.

Dependency: go:google.golang.org/protobuf:v1.27.1

- Upgrade to: 1.33.0
- **CVE-2024-24786, Score: 7.5:** In the package google.golang.org/protobuf versions prior to 1.33.0, the "protojson.Unmarshal" function can enter an infinite loop when unmarshaling certain forms of invalid JSON. This condition can occur when unmarshaling into a message which contains a "google.protobuf.Any" value, or when the "UnmarshalOptions.DiscardUnknown" option is set.

Medium issues

#1 | Deprecated dependencies

File	Severity	Location	Status
Main	Medium	Dependency	Open

Description:
The current codebase uses several deprecated dependencies, each with known security vulnerabilities. It is critical to update these dependencies to mitigate potential security risks. Below is a list of the identified issues in the dependencies, along with their CVEs and descriptions:

Dependency: go:github.com/graph-gophers/graphql-go:v0.0.0-20201113091052-beb923fada29

- **CVE-2022-21708, Score: 6.5:** graphql-go is a GraphQL server with a focus on ease of use. In versions prior to 1.3.0 there exists a DoS vulnerability that is possible due to a bug in the library that would allow an attacker with specifically designed queries to cause stack overflow panics. Any user with access to the GraphQL handler can send these queries and cause stack overflows. This in turn could potentially compromise the ability of the server to serve data to its users. The issue has been patched in version v1.3.0. The only known workaround for this issue is to disable the graphql.MaxDepth option from your schema which is not recommended.

Dependency: go:github.com/microcosm-cc/bluemonday:v1.0.2

- Upgrade to: 1.0.5
- **CVE-2021-29272, Score: 6.1:** bluemonday before 1.0.5 allows XSS because certain Go lowercasing converts an uppercase Cyrillic character, defeating a protection mechanism against the "script" string.

Dependency: go:golang.org/x/image:v0.0.0-20190802002840-cff245a6509b

- Upgrade to: 0.10.0
- **CVE-2022-41727, Score: 5.5:** An attacker can craft a malformed TIFF image which will consume a significant amount of memory when passed to "DecodeConfig". This could lead to a denial of service. The vulnerable versions are prior to 0.5.0.
- **CVE-2023-29407, Score: 6.5:** A maliciously-crafted image can cause excessive CPU consumption in decoding. A tiled image with a height of 0 and a very large width can cause excessive CPU consumption, despite the image size (width * height) appearing to be zero. This vulnerability affects golang.org/x/image package versions prior to 0.10.0. This has the same fix as CVE-2023-29408.
- **CVE-2023-29408, Score: 6.5:** The TIFF decoder does not place a limit on the size of compressed "tile" data. A maliciously-crafted image can exploit this to cause a small image (both in terms of pixel width/height, and encoded size) to make the decoder decode large amounts of compressed data, consuming excessive memory and CPU. This vulnerability affects golang.org/x/image package versions prior to 0.10.0. This has the same fix as CVE-2023-29407.

Dependency: go:golang.org/x/net:v0.8.0

- Upgrade to: 0.17.0
- **CVE-2023-44487, Score: 5.3:** The HTTP/2 protocol allows a denial of service (server resource consumption) because request cancellation can reset many streams quickly, as exploited in the wild in August through October 2023.

Low issues

No low issues

Informational issues

No informational issues

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY