



Audit

Security Assessment

Date: **26. August 2024**

Client: **Vitra Studios**

Index

Introduction.....	3
Disclaimer.....	3
Project Overview.....	4
Overview.....	4
Social Media Information.....	4
Scope of Work.....	5
Imported Packages.....	6
Audit Information.....	7
Vulnerability & Risk Management Level.....	7
Auditing Strategy and Techniques Applied.....	7
Methodology.....	8
Audit Results	10
Critical Issues - [0].....	10
High Issues - [2].....	10
Medium Issues - [1].....	12
Low Issues - [0].....	14
Informational - [0].....	14



FIRST LINE OF DEFENSE

Introduction

line1.io is a distinguished brand under the officially registered entity, FutureVisions Deutschland, based in Germany. Our expertise lies in Blockchain Security, providing comprehensive services including Smart Contract Audits and KYC verification for project teams. At line1.io, we rigorously evaluate smart contracts for security vulnerabilities, verify the alignment between the codebase and related whitepapers/documentation, and deliver detailed recommendations for improvement.

Disclaimer


line1.io reports are neither an endorsement nor a disapproval of any specific project or team. These reports do not reflect the economic viability or value of any product or asset created by any team. line1.io does not test or audit the integration with external contracts or services.

line1.io audits do not offer any warranty or guarantee regarding the absolute absence of bugs in the analyzed technology, nor do they provide any information about the proprietors of the technology. These audits should not be used as a basis for making investment decisions or for involvement in any particular project. The reports do not constitute investment advice and should not be relied upon as such.

line1.io reports represent a comprehensive auditing process designed to assist our clients in enhancing the quality of their code while mitigating the significant risks associated with cryptographic tokens and blockchain technology. Given the high level of inherent risk in blockchain technology and cryptographic assets, line1.io emphasizes that each company and individual must conduct their own due diligence and maintain continuous security. line1.io does not claim any guarantee of the security or functionality of the technologies we analyze.

Project Overview

Overview

Project Name	Vitra Studios
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	

Social Media Information

Telegram	https://t.me/SHG_VITRA
X / Twitter	https://x.com/VitraStudios
Facebook	https://www.facebook.com/people/Vitra-Studios/61554658419906/
Instagram	https://www.instagram.com/vitrastudios
Discord	https://discord.com/invite/ng3pAnswEa
YouTube	https://www.youtube.com/@VitraStudios

Audit Summary

Version	Date	Changelog
V1.0	11th August 2024	Initial Report
V1.1	26th August 2024	Adjustments and Publication

Note: 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 VITRA-BLOCKCHAIN repository, which includes a fork of the opera-go framework and custom modifications made by the Vitra Studios team. This assessment seeks 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 opera-go source code
- Custom modifications and enhancements made to the framework by Vitra Studios
- Integration points and dependencies with external systems or libraries

Repository	Commit
https://github.com/VitraStudios/VITRA-BLOCKCHAIN	da33fb4

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/matttn/go-colorable v0.1.8
- github.com/matttn/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 Management 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	Description	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 as possible.
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 audit of the Vitra Studios blockchain repository, our primary focus was on identifying potential security vulnerabilities, ensuring high code quality, and verifying compliance with established specifications and best practices. The audit was carried out by a highly skilled team of penetration testers and blockchain developers, each with extensive expertise in the go-opera and go-ethereum frameworks, as well as smart contract security.

Every file within the repository was subjected to a meticulous manual review. While automated tools were employed, their use was strategically limited to complement the manual examination process, enhancing both its efficiency and effectiveness, without substituting it.

Methodology

The auditing process follows a structured series of steps to ensure thorough examination and assessment:

- **Specification Review:** Our team meticulously reviewed all relevant documentation, specifications, and guidelines provided initially. This step ensures a deep understanding of the blockchain's architecture, scope, and intended functionalities.
- **Manual Code Inspection:** The source code was examined line by line in an intensive review aimed at uncovering potential security vulnerabilities that could be exploited maliciously. This careful inspection ensures no detail is overlooked.
- **Specification Conformance:** The code was rigorously compared against the provided specifications to confirm its fidelity in performing as described. This ensures that the implementation accurately reflects the intended design and requirements.
- **Test Coverage Analysis:** We analyzed the extent of test cases' coverage over the codebase. This involved determining how much code was executed during these tests to identify untested paths and ensure comprehensive test coverage.
- **Symbolic Execution:** This technique was applied to analyze how different inputs affect the code execution paths. It helped understand the conditions under which various parts of the program would execute, revealing potential vulnerabilities and ensuring robust security.

During this audit, certain aspects, such as unused constants, functions, exported functions, global variables, and parameters, were excluded from the scope of our review. While these elements were not directly evaluated, we suggest revisiting these areas in future audits to ensure that they do not introduce security concerns or affect the overall quality of the codebase.

Metrics

Codebase Composition Overview

This report section provides an in-depth analysis of the project's codebase, which is primarily composed of Go (96%). The codebase is well-structured, with extensive comments and documentation, particularly within the .proto files. This level of documentation is crucial for maintaining clarity and understanding across the development team and for any future developers who may work on the project.

The code distribution reveals that 77% of the total lines are dedicated to the actual code, while 14% consist of comments, and 10% are blank lines. This distribution highlights a balanced approach to coding, emphasizing not only functionality but also the importance of clear documentation. The substantial proportion of comments demonstrates a commitment to making the code understandable and maintainable. These comments provide insights into the code's logic, functionality, and any potential edge cases that developers need to be aware of.

Additionally, the presence of 10% blank lines indicates that the codebase follows good coding practices, such as separating logical blocks of code for better readability and maintainability. This practice helps in reducing complexity and making the code more approachable for review and debugging.

Overall, the structure of the codebase, with its emphasis on thorough documentation and clear separation of code, comments, and blank lines, suggests a high level of professionalism and foresight in its development. This meticulous approach not only facilitates easier maintenance and updates but also ensures that the project can be effectively scaled and adapted as needed in the future.



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
.yaml	184	2	35	0.35

Audit Results

Critical Issues - [0]

No Critical Issues

High Issues - [2]

#H-1 Outdated go-ethereum Version

Severity	Location / Line	Status
High	Dependency	Open
Description	<p>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.</p> <p>Changelog go-ethereum v1.14.7: https://github.com/ethereum/go-ethereum/releases/tag/v1.14.7</p>	
Advisory	To mitigate these security vulnerabilities, upgrade to go-ethereum v1.14.7 or later. For more information, please refer to the advisory.	

#H-2 Deprecated dependencies

Severity	Location / Line	Status
High	Dependency	Open
Description	<p>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:</p> <p>Dependency: go:github.com/btcsuite/btcd:v0.20.1-beta Upgrade to: 0.23.3</p> <p>CVE-2022-39389, Score: 6.5: Lightning Network Daemon (Ind) is an implementation of a lightning bitcoin overlay network node. All Ind 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 <code>Incli updatechanpolicy</code> 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.</p> <p>CVE-2022-44797, Score: 9.8: btcd before 0.23.2, as used in Lightning Labs Ind before 0.15.2-beta and other Bitcoin-related products, mishandles witness size checking.</p> <p>Dependency: go:github.com/consensus/gnark-crypto:v0.4.1-0.20210426202927-39ac3d4b3f1f Upgrade to: 0.12.1</p> <p>Cx42455c0d-6fe9, Score: 9.8: github.com/consensus/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</p>	

sound. Also note that GLV methods in G1 and G2 are sound and not impacted.

CVE-2023-44273, Score: 9.8: The `github.com/consensus/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

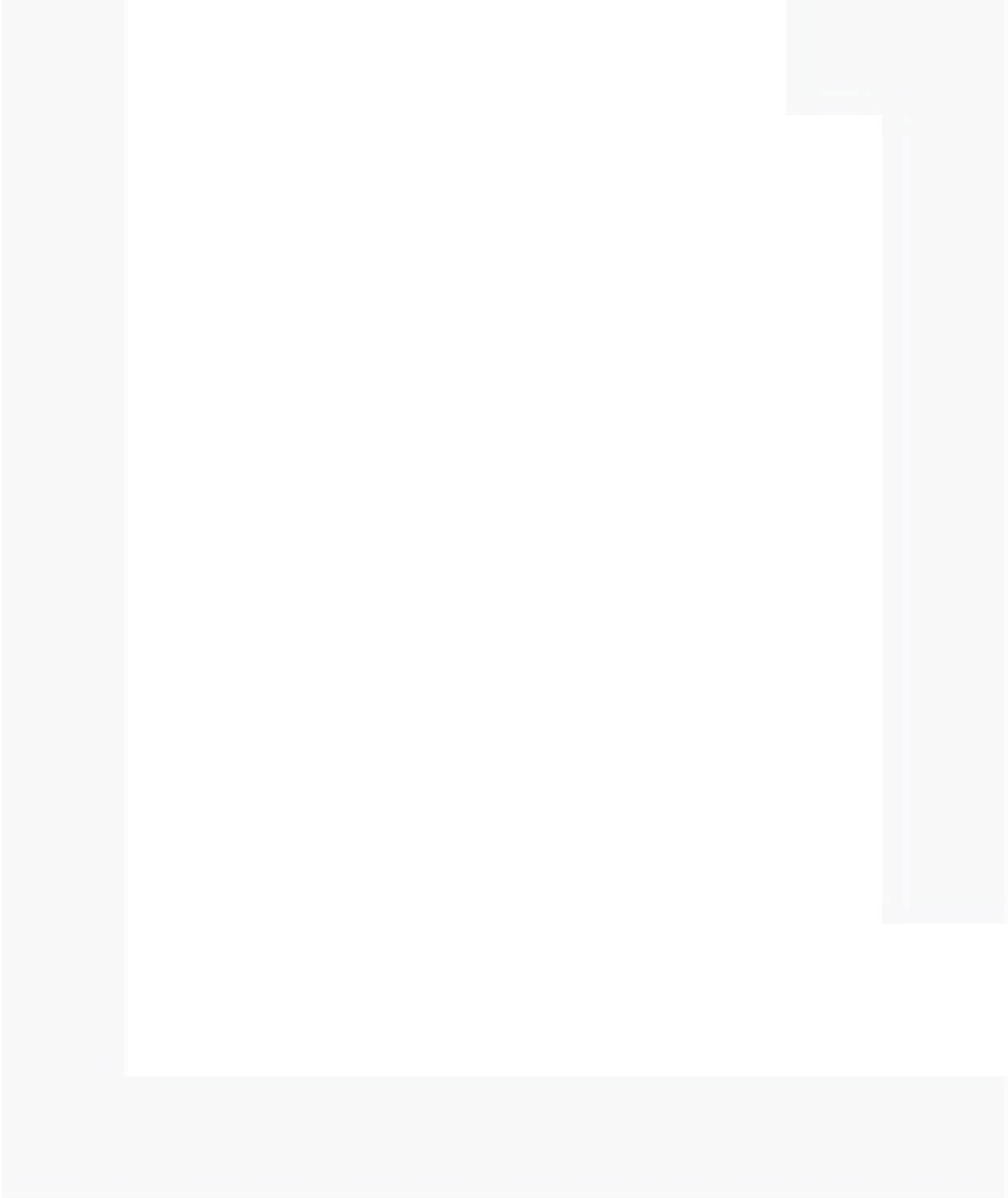
CVE-2023-42818, Score: 9.8: JumpServer is an open-source bastion host. When users enable MFA and

	<p>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.</p> <p>Dependency: <code>go:google.golang.org/protobuf:v1.27.1</code></p> <p>Upgrade to: 1.33.0</p> <p>CVE-2024-24786, Score: 7.5: In the package <code>google.golang.org/protobuf</code> 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.</p>
Advisory	/

Medium Issues - [1]

#M-1 Deprecated dependencies		
Severity	Location / Line	Status
Medium	Dependency	Open
Description	<p>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:</p> <p>Dependency: <code>go:github.com/graph-gophers/graphql-go:v0.0.0-20201113091052-beb923fada29</code></p> <p>CVE-2022-21708, Score: 6.5: <code>graphql-go</code> 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 <code>graphql.MaxDepth</code> option from your <code>schemaxm</code> which is not recommended.</p> <p>Dependency: <code>go:github.com/microcosm-cc/bluemonday:v1.0.2</code></p> <p>Upgrade to: 1.0.5</p> <p>CVE-2021-29272, Score: 6.1: <code>bluemonday</code> before 1.0.5 allows XSS because certain Go lowercasing converts an uppercase Cyrillic character, defeating a protection mechanism against the "script" string.</p> <p>Dependency: <code>go:golang.org/x/image:v0.0.0-20190802002840-cff245a6509b</code></p> <p>Upgrade to: 0.10.0</p> <p>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.</p> <p>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 <code>golang.org/x/image</code> package versions prior to 0.10.0. This has the same fix as CVE-2023-29408.</p> <p>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 <code>golang.org/x/image</code> package versions prior to 0.10.0. This has the same fix as CVE-2023-29407.</p>	

	<p>Dependency: go:golang.org/x/net:v0.8.0</p> <p>Upgrade to: 0.17.0</p> <p>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.</p>
Advisory	/



Low Issues - [0]

No Low Issues

Informational - [0]

No Informational Issues

