GETTY/IO

Security Audit for:

KleverChain

Version 0.1: August 30th, 2022

About US

Getty/IO is an innovative remote IT security consulting firm that has the expertise to audit and recommendation to build secure web and mobile products. We specialize in modern Javascript technologies, lean and highly scalable backends, and blockchain technologies, which help to build the secure product for our customers worldwide.

Born in South America, GETTY has become the largest remote development firm in the region. We are a global company helping startups and enterprises from all around the world scale their development teams by providing them with the top remote developers and consultants:

- Blockchain - Security Review & Audit
- Smart Contract - Security Review & Audit
- Applications - Security Review & Audit

Our professionals are able to integrate to our Customer’s teams and add value from day one. Agile methodologies, experience helping dozens of teams build products, and constant coaching and mentoring ensure all our engagements help our customers become successful.

In our hands, your entire process is safe, without the hassle, and as seamless as it can be. With the collaboration of our team of experts, you can expect to achieve much more. We are present in the United States, Canada, Portugal, Estonia, and Brazil.

Abstract

Getty/IO has been appointed by the Klever Team to carry out the audit of KleverChain, his main blockchain. **Klever** is a decentralized p2p and self-custody wallet and **KFI** is the Klever blockchain governance token, to be used in Klever's blockchain parameters and kApp upgrades and the **KLV** token is the fuel of the blockchain, used to incentive people to participate in the blockchain thru stake as also pay networks fees"

This document presents the results of an Internal security audit for the KleverChain. This test aimed to identify security vulnerabilities that could negatively affect the systems under the scope, the data they handle, and consequently the business. They were simulated in a systematic way, attacks that were specifically tailored for the engagement's scope to test the resilience against real-life attack scenarios based on a black-box approach.

The analysis focused on vulnerabilities especially related to implementation, and on issues caused by architectural or design errors.

For each vulnerability discovered during the assessment, it was attributed a risk severity rating. The issue's severity classification is based on the potential it presents to provide means for fraud, data leakage, and other harmful events that may bring a direct adverse impact to the business.

Methodology

Tests were conducted using risk factors, such as probability and impact. Each test and tool that has been used was focused on vulnerability's complexity and how it could be mitigated.

		IS Risk Vision		
Exploitation Complexity	Low	Medium	High	Critical
	Medium	Low	Medium	High
	High	Low	Low	High
		Low	Medium	High
		Impact		

Tools and Vectors

The following tools and vectors were applied:

- Fuzzers: Bed, Rfuzz (Ruby), Sfuzz, fuzzing auxiliary modules of Metasploit, Spike (kit for developing fuzzers), etc.
- Brute force: John the Ripper, Hashcat, etc.
- Web applications: W3AF, Websecurify, Accunetix, Metasploit scanning auxiliary modules, CGI
- Scanner, ASP-Auditor, Oscanner, proxies such as Fiddler2 or Webscarab, Firefox browser plugins such as hacking toolbar, Tamper Data, User-Agent switcher, etc.
- Manual search in vulnerability repositories such as CVE, OSVD or NVD.
- Manual code analysis with the aim of finding weaknesses and developing bad practices, making use of editors, debuggers, and decompilers.
- Manual attacks: open port stress tests, SQL injections, CSS, RFI, overflows buffer detection, directory listing, web proxies (ZAP, Burp Suite, and webscarab), etc.
- Network communications analysis, Tshark, Ettercap, Wireshark, etc.
- In addition, this audit phase can be completed by using tools that automate the security analysis process of devices under studies, such as the well-known Nessus scanner and its GPL equivalent OpenVas.

Tests Performed

Information Gathering

- Conduct Search Engine Discovery and Reconnaissance
- Fingerprint Ports, Protocols, and Services
- Review Service Versions
- Enumerate Specific Application Stacks
- Identify Application Entry Points
- Map Execution Paths and Architecture

P2P

- Sybil Attack
- Eclipse Attack
- Eavesdropping Attack

- Denial of Service Attack
- BGP Hijack Attack
- Alien Attack
- Timejacking

RPC

- Eavesdropping Attack
- Denial of Service Attack
- The Ethereum Black Valentine's Day Vulnerability
- Http Input Attack
- Cross-Domain Phishing Attack

Consensus

- Long Range Attack
- Bribery Attack
- Race Attack
- Liveness Denial
- Censorship
- Finney Attack
- Vector76 Attack
- Alternative Historical Attack
- 51% Attack
- Grinding Attack
- Coin Age Accumulation Attack
- Selfing Mining
- Block Double Production

Configuration and Deployment Management Testing

- Test Network/Infrastructure Configuration
- Test Application Platform Configuration
- Test File Extensions Handling of Sensitive Information
- Identify API Exposure/Leakage
- Review Old, Backup, and Unreferenced Files for Sensitive Information
- Enumerate Infrastructure and Application Admin Interfaces

Authentication Testing

- Test Default Credentials

- Test for Authentication Bypass
- Test for Valid User Enumeration

Authorization Gathering

- Test Directory Traversal/File Include
- Test for Bypassing Authorization Schema
- Test for Privilege Escalation
- Test for Insecure Direct Object References (IDORs)
- Test for Sensitive Data Leakage

Data Validation Testing

- Test for Server-Side Request Forgery (SSRF)
- Test for Remote Code Execution (RCE)
- Test for SQL Injection (Oracle, MySQL, SQL Server, PostgreSQL, MS Access,
- NoSQL)
- Test for LDAP Injection
- Test for XML Injection
- Test for IMAP/SMTP Injection
- Test for Code Injection (Local File Inclusion and Remote File Inclusion)
- Test for Command Injection
- Test for Buffer Overflow (Heap, Stack, Format string)
- Test for Incubated Vulnerabilities (i.e.Blind SQL Injection)

Error Handling

- Analyze Error Codes
- Analyze Stack Traces

Cryptography

- Test for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection
- Test for Sensitive Information Sent Via Unencrypted Channels

Encryption

- Cryptographic Attack
- Private Key Prediction
- Length Extension Attack
- Hash collision attack

- Transaction
- Transaction Replay Attack
- Transaction Malleability Attack
- Time-Locked Transaction Attack
- False Top-Up Attack
- Rug Pull Attack

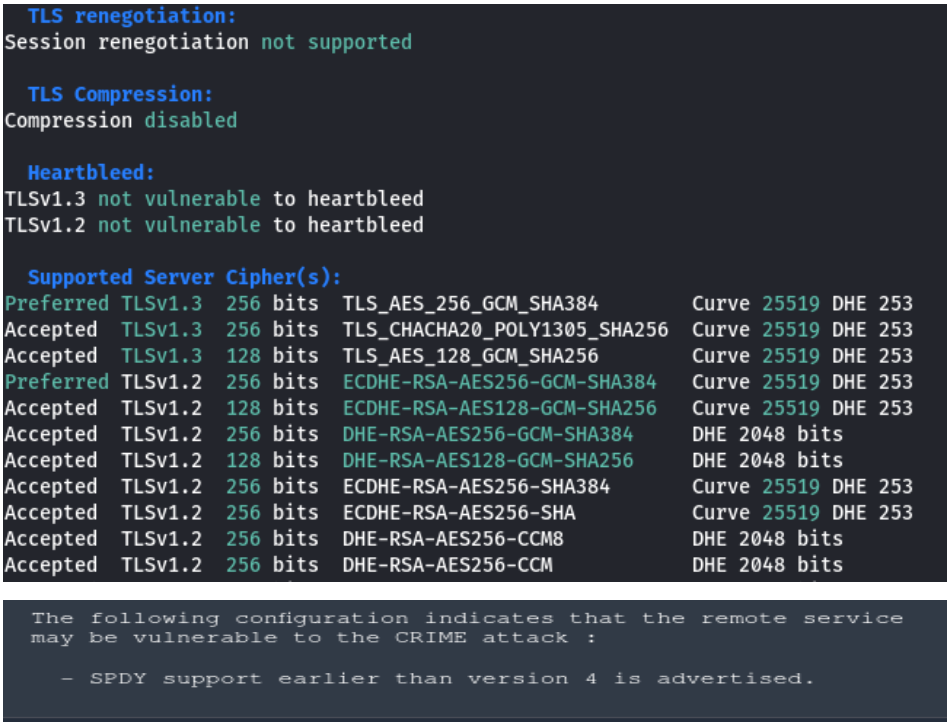
Audit Dashboard

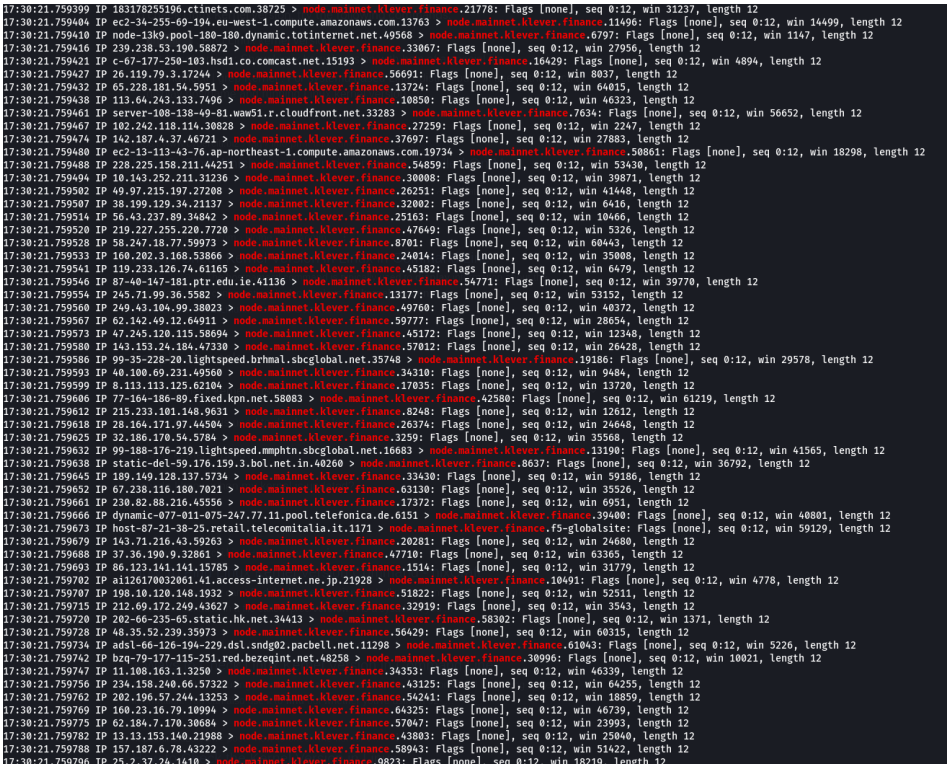
Project	KleverChain
Auditors	Wesley Silva Luis Araujo
Assets	node.mainnet.klever.finance node.testnet.klever.finance
Networks	Node Mainnet /Node Testnet
Date	2021-06-30 to 2021-08-22

Issues Found

	Low	Medium	High	Critical
Open	**2**	**0**	**0**	**0**
Resolved	**1**	**0**	**0**	**0**

Results

Item	Transport Layer Security (TLS) Protocol CRIME Vulnerability
Description	The remote service has one of two configurations that are known to be required for the CRIME attack: • SSL / TLS compression is enabled. • TLS advertises the SPDY protocol earlier than version 4. Since this is an old version of the software, it may be vulnerable to attacks.
Evicence	
Solution	Disable compression and/or the SPDY service.
Risk Factor	Low
Assets	Node.mainnet.klever.finance / node.testnet.klever.finance
Resolved	No. Very low probability to be exploited.

Item	IP Fragmentation Functionality 0 Length Fragment Handling Remote DoS
Description	The remote host is prone to a denial of service attack. The remote host appears to be using a Linux kernel that contains a flaw in its IP fragment handling code. By sending a series of packets with 0-length fragments, an unauthenticated attacker may be able to disable the remote host's IP connectivity.
Evicence	
Solution	Effective protection only happens with a third-party solution. Example: Clouflare.
Risk Factor	Low
Assets	Node.mainnet.klever.finance / node.testnet.klever.finance
Resolved	Yes. Configured DoS protection

Conclusion

After using some tools and testing the environment described in this document, We have not found critical or high vulnerabilities.

All the security requirements have been archived during the architecture and coding phase.

Appendix

Severity

Low	Low issues are generally subjective in nature or potentially deal with topics like "best practices" or "readability". Minor issues will in general not indicate an actual problem or bug in code. The maintainers should use their own judgment as to whether addressing these issues improves the codebase.
Medium	Medium issues are generally objective in nature but do not represent actual bugs or security problems. These issues should be addressed unless there is a clear reason not to.
High	High issues will be things like bugs or security vulnerabilities. These issues may not be directly exploitable or may require a certain condition to arise in order to be exploited. Left unaddressed, these issues are highly likely to cause problems with the operation of the contract or to lead to a situation that allows the system to be exploited in some way.