



AUDIT REPORT

January 2026

For

Swarm[®]

Table of Content

Executive Summary	03
Number of Security Issues per Severity	04
Summary of Issues	05
Checked Vulnerabilities	06
Techniques and Methods	07
Types of Severity	08
Types of Issues	09
Severity Matrix	10
 Medium Severity Issues	11
1. CSRF Attack due to lack of state in OAUTH	11
 Low Severity Issues	12
2. Clickjacking Attack	12
3. No Input Validation in Title	13
Closing Summary & Disclaimer	14

Executive Summary

Project Name	VAULT by Swarm Network
Protocol Type	Dapp Pentest
Project URL	https://testnet.tryvault.xyz/
Overview	<p>VAULT is a decentralized secrets management platform built on the Sui blockchain, offering zero-knowledge encryption for secure storage of sensitive credentials and confidential data. The platform employs military-grade encryption protocols ensuring that only authorized users can decrypt their stored secrets, while maintaining complete data sovereignty. The solution enables organizations to securely share vaults among team members through blockchain-based access controls, eliminating traditional centralized key management vulnerabilities. As a Web3-native application, VAULT leverages Sui's high-performance blockchain infrastructure to provide cryptographically secure, distributed storage with no single point of failure. The platform targets enterprises and development teams requiring robust secrets management without compromising on decentralization principles or privacy guarantees.</p>
Review 1	12th December 2025 - 24th December 2025
Updated Code Received	15th January 2025
Review 2	15th January 2026 - 20th Januray 2025
Fixed In	https://testnet.tryvault.xyz/

Verify the Authenticity of Report on QuillAudits Leaderboard:

<https://www.quillaudits.com/leaderboard>

Number of Issues per Severity



Critical	0(0.0%)
High	0(0.0%)
Medium	1 (33.3%)
Low	2 (66.6%)
Informational	0(0.0%)

Issues	Severity				
	Critical	High	Medium	Low	Informational
Open	0	0	0	0	0
Acknowledged	0	0	0	0	0
Partially Resolved	0	0	0	0	0
Resolved	0	0	1	2	0

Summary of Issues

Issue No.	Issue Title	Severity	Status
1	CSRF Attack due to lack of state in OAUTH	Medium	Resolved
2	Clickjacking Attack	Low	Resolved
3	No Input Validation in Title	Low	Resolved

Checked Vulnerabilities

Improper Authentication

Improper Resource Usage

Improper Authorization

Insecure File Uploads

Insecure Direct Object References

Client-Side Validation Issues

Rate Limit

Input Validation

Injection Attacks

Cross-Site Scripting (XSS)

Cross-Site Request Forgery

Security Misconfiguration

Broken Access Controls

Insecure Cryptographic Storage

Insufficient Cryptography

Insufficient Session Expiration

Insufficient Transport Layer Protection

Unvalidated Redirects and Forwards

Information Leakage

Broken Authentication and Session Management

Denial of Service (DoS) Attacks

Malware

Third-Party Components

And More..

Techniques and Methods

Throughout the pentest of application, 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:

Burp Suite

Acunetix

Nmap

DNSenum

Neucli

Metasploit

Dirbuster

Nabbu

Horusec

SQLMap

Turbo Intruder

Postman

Netcat

Nessus

And Many more..

Types of Severity

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

Critical: Immediate and Catastrophic Impact

Critical issues are the ones that an attacker could exploit with relative ease, potentially leading to an immediate and complete loss of user funds, a total takeover of the protocol's functionality, or other catastrophic failures. Critical vulnerabilities are non-negotiable; they absolutely must be fixed.

High (H): Significant Risk of Major Loss or Compromise

High-severity issues represent serious weaknesses that could result in significant financial losses for users, major malfunctions within the protocol, or substantial compromise of its intended operations. While exploiting these vulnerabilities might require specific conditions to be met or a moderate level of technical skill, the potential damage is considerable. These findings are critical and should be addressed and resolved thoroughly before the contract is put into the Mainnet.

Medium (M): Potential for Moderate Harm Under Specific Circumstances

Medium-severity bugs are loopholes in the protocol that could lead to moderate financial losses or partial disruptions of the protocol's intended behavior. However, exploiting these vulnerabilities typically requires more specific and less common conditions to occur, and the overall impact is generally lower compared to high or critical issues. While not as immediately threatening, it's still highly recommended to address these findings to enhance the contract's robustness and prevent potential problems down the line.

Low (L): Minor Imperfections with Limited Repercussions

Low-severity issues are essentially minor imperfections in the smart contract that have a limited impact on user funds or the core functionality of the protocol. Exploiting these would usually require very specific and unlikely scenarios and would yield minimal gain for an attacker. While these findings don't pose an immediate threat, addressing them when feasible can contribute to a more polished and well-maintained codebase.

Informational (I): Opportunities for Improvement, Not Immediate Risks

Informational findings aren't security vulnerabilities in the traditional sense. Instead, they highlight areas related to the clarity and efficiency of the code, gas optimization, the quality of documentation, or adherence to best development practices. These findings don't represent any immediate risk to the security or functionality of the contract but offer valuable insights for improving its overall quality and maintainability. Addressing these is optional but often beneficial for long-term health and clarity.

Types of Issues

Open	Resolved
Security vulnerabilities identified that must be resolved and are currently unresolved.	These are the issues identified in the initial audit and have been successfully fixed.
Acknowledged	Partially Resolved
Vulnerabilities which have been acknowledged but are yet to be resolved.	Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Severity Matrix

		Impact		
		■ High	■ Medium	■ Low
Likelihood	■ High	Critical	High	Medium
	■ Medium	High	Medium	Low
	■ Low	Medium	Low	Low

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- **Low** - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

Likelihood

- **High** - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- **Medium** - only a conditionally incentivized attack vector, but still relatively likely.
- **Low** - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

Medium Severity Issues

CSRF Attack due to lack of state in OAUTH

Resolved

Description

The Google OAuth implementation lacks a state parameter for CSRF protection, allowing attackers to forge authentication callbacks. An attacker can trick users into linking their SWARM VAULT account to the attacker's Google account, potentially giving the attacker access to the victim's vaults.

Vulnerable

File: frontend/src/services/zkLoginService.ts
Lines: 93-103
Function: buildGoogleOAuthUrl

```
// Vulnerable Code (Lines 93-103)
function buildGoogleOAuthUrl(nonce: string): string {
    const params = new URLSearchParams({
        client_id: zkLoginConfig.googleClientId,
        redirect_uri: zkLoginConfig.redirectUri,
        response_type: 'id_token',
        scope: zkLoginConfig.googleScopes.join(' '),
        nonce: nonce,
        // MISSING: state parameter for CSRF protection!
    });
    return `https://accounts.google.com/o/oauth2/v2/auth?${params.toString()}`;
}
```

Impact

- Account takeover through forced OAuth linking
- Unauthorized access to encrypted vaults via attacker's Google account

Recommendation

- Generate cryptographically secure state parameter for each OAuth request
- Store state server-side with user session
- Validate state parameter on OAuth callback
- Implement state expiration (5-10 minutes)

Low Severity Issues

Clickjacking Attack

Resolved

Description

The SWARM VAULT testnet application lacks proper X-Frame-Options and Content Security Policy (CSP) frame-ancestors directives, allowing the entire application to be embedded in malicious iframes. Attackers can overlay invisible frames over the legitimate interface, tricking users into performing unintended actions such as approving transactions, sharing vaults, or revealing encryption keys.

Vulnerable URL

URL: <https://testnet.tryvault.xyz/>

Impact

- Unauthorized vault access through UI redressing attacks
- Trick users into approving malicious blockchain transactions
- Credential theft via invisible form overlays
- Unintended sharing of encrypted secrets with attackers

Recommendation

- Add X-Frame-Options: DENY header to prevent all framing
- Implement CSP header with frame-ancestors 'none' directive
- Deploy JavaScript frame-busting code as defense-in-depth

POC

<https://clickjacker.io/test?url=https://testnet.tryvault.xyz/>

No Input Validation in Title

Resolved

Description

The vault title field accepts unsafe characters without proper sanitization or validation. While XSS is not currently triggered due to output encoding, the lack of input validation allows storage of potentially malicious payloads that could be exploited if output encoding is accidentally removed in future updates or in other contexts where the data is used.

Vulnerable Code Location

Title input fields across the application

Backend API endpoints accepting vault title without validation

Impact

- Potential for stored XSS if output encoding is removed in future
- Database pollution with malformed or malicious data

Recommendation

Implement strict input validation with allowlist of permitted characters

Sanitize all special characters before storage

POC

<https://testnet.tryvault.xyz/vault/>

0x102ffab752329f6a82d0ddcb5163434be75f3e5e86627edfa7360e27f03d399

Closing Summary

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

Some issues of medium and low severities were found and the Swarm Network resolved all the issues.

Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code.

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.

Stay proactive. Stay secure.

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 seven years of expertise, we've secured over 1400 projects globally, averting over \$3 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.



7+ Years of Expertise	1M+ Lines of Code Audited
50+ Chains Supported	1400+ Projects Secured

Follow Our Journey



AUDIT REPORT

January 2026

For

Swarm

 QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillaudits.com