·Decurity·

tornado

FRONTEND SECURITY AUDIT REPORT

TORNADO CASH

Decurity, 2022

CONTENTS

1 GENERAL INFORMATION

This report contains information about the results of the security audit of the Tornado Cash (hereafter referred as "Customer") application, conducted by Decurity in the period from 05/11/2022 to 05/31/2022.

1.1 Introduction

Tasks solved during the work are:

- Review the application design and the usage of 3rd party dependencies,
- Audit the UI implementation,
- Develop the recommendations and suggestions to improve the security of the application.

1.2 Scope of Work

The testing scope included the Tornado Cash Classic UI dApp, source code located in the repository https://github.com/tornadocash/tornado-classic-ui (commit fa3d089e44a4aa9a6f30aa11779b0f34b74e0a4e).

The deployed app has been analyzed using the following URLs:

- https://tornadocash.eth.limo
- https://tornadocash.eth.link
- https://cloudflare-ipfs.com/ipns/tornadocash.eth
- https://tornadocash.app.runonflux.io

1.3 Threat Model

The assessment presumes actions of an intruder who might have capabilities of an external user. The privacy risks have been considered as a primary concern upon the request of the Customer.

1.4 Weakness Scoring

An expert evaluation scores the findings in this report, an impact of each vulnerability is calculated based on its ease of exploitation (based on the industry practice and our experience) and severity (for the considered threats).

2 SUMMARY

As a result of this work, we have discovered a single critical exploitable security issue which has been fixed and re-tested in the course of the work.

The other suggestions included fixing the low-risk issues and some best practices (see 3.1).

The Tornado Cash team has given the feedback for the suggested changes and explanation for the underlying code.

2.1 Suggestions

The table below contains the discovered issues, their risk level, and their status as of 28 April 2022.

Table 1. Discovered weaknesses

Issue	Scope	Risk Level	Status
IPFS dApp hijacking	https://cloudflare-ipfs.com/ipns/tornadocash.eth/	High	Mitigated
Insecure Content Security Policy (CSP) and HTTP headers	• tornadocash.eth.limo • tornadocash.eth.link • cloudflare-ipfs.com/ipns/tornadocash.eth	Medium	Acknowledged

Issue	Scope	Risk Level	Status
	• tornadocash.app.runonflux.io		
Lack of Referrer Policy	• cloudflare-ipfs.com/ipns/tornadocash.eth • tornadocash.app.runonflux.io	Medium	Acknowledged / Partially fixed
Window opener hijacking (Tabnabbing)	• components/Navbar.vue • components/EncryptedTx.vue • components/Notices.vue • components/Footer.vue • components/MetamaskNavbarIcon.vue • components/governance/Proposal.vue • components/governance/ma	Low	Fixed

Issue	Scope	Risk Level	Status
	nage/tabs/DelegateTab.vue • components/governance/manage/tabs/UndelegateTab.vue • components/withdraw/Withdraw.vue • components/Tx.vue • components/Job.vue • pages/index.vue • pages/compliance.vue		
Location host spoofing	• store/relayers.js • plugins/detectIPFS.js	Low	Fixed
Potentially redundant external/HTTP links	Source code	Low	Fixed

Issue	Scope	Risk Level	Status
Faulty multiple tab detection	plugins/preventMultiTabs.js	Low	Fixed
Unreachable code	• components/withdraw/Withdraw.vue (lines 380-382) • pages/index.vue (lines 87-89) • pages/index.vue (lines 97-99)	Low	Fixed

3 GENERAL RECOMMENDATIONS

This section contains general recommendations how to fix discovered during the testing weaknesses and vulnerabilities and how to improve overall security level.

Section 3.1 contains a list of general mitigations against the discovered weaknesses, technical recommendations for each finding can be found in section 4.

Section 3.2 describes a brief long-term action plan to mitigate further weaknesses and bring the product security to a higher level.

3.1 Current findings remediation

Follow the recommendations in the section 4.

3.2 Security process improvement

- Keep the whitepaper and documentation updated to make it consistent with the implementation and the intended use cases of the system,
- Perform regular audits for all the new releases and updates,
- Ensure the secure off-chain storage and processing of the credentials (e.g. the privileged private keys),
- Launch a public bug bounty campaign for the application.

4 FINDINGS

4.1 IPFS dApp hijacking

Risk Level: High

CVSS:

AV:Network/AC:Low/PR:None/UI:Required/S:Unchanged/C:High/I:High/A:Low

Status:

In the commit `b91b81f5c9967a2b09116ff3e340e17d2e6c4feb` the LocalStorage usage for the IPFS deployments has been restricted to only store the `netId` value.

Also, a warning has been added:

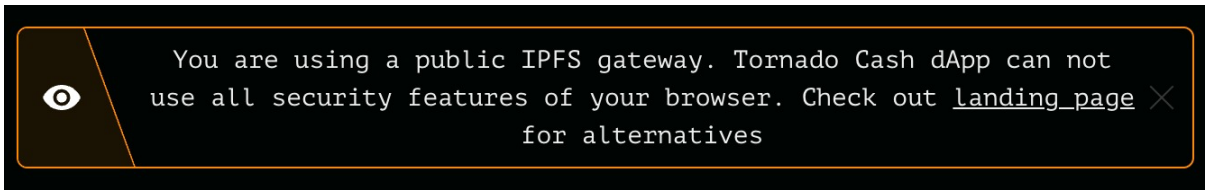


Image 1. IPFS security warning

Scope:

https://cloudflare-ipfs.com/ipns/tornadocash.eth/

References: -

Remediation:

The following remediation scenarios could be possible:

- Discourage users from using IPFS gateways where CID or IPNS is a part of the URL path, such as cloudflare-ipfs.com
- Refuse loading from such IPFS gateways in the application itself

Description:

Persistent Cross-Site Scripting vulnerabilities occur when the application stores user-controlled information in the persistent storage and then uses it to render HTTP response bodies to other clients.

Proofs:

One of the application deployment locations is the IPFS network. The official link to access the application is https://cloudflare-ipfs.com/ipns/tornadocash.eth/.

The problem with such links is that any other IPFS resource can be loaded in the same origin. In the frontend security context this creates a situation similar to the impact of a stored XSS attack.

An attacker can pin a malicious HTML file to an IPFS node and pass the link to the victims. When a victim loads the malicious page, the current URI can be rewritten via Javascript:

```
window.history.pushState({},'','https://cloudflare-ipfs.com/ipns/tornadocash.eth/');
```

Also, the local storage of the Tornado Cash application can be accessed and rewritten which may further lead to privacy issues or expose the app to the attacks from the local storage.

```
console.log(localStorage.getItem('tornadoClassicV2'));
```

Therefore, the IPFS dApp deployment essentially breaks the Same Origin Policy principles by making different untrusted applications share the same browser context.

The screenshot below demonstrates the attack:

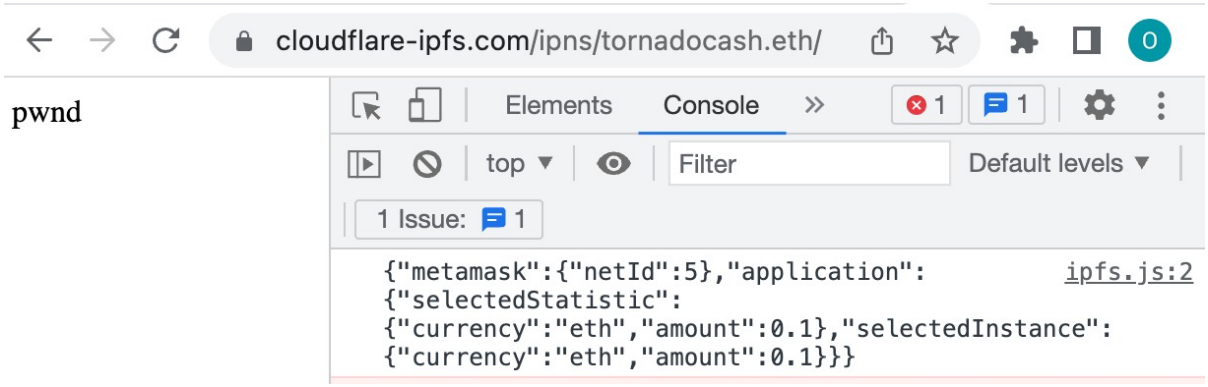


Image 2. Stored XSS attack in the IPFS gateway origin

4.2 Insecure Content Security Policy (CSP) and HTTP headers

Risk Level: Medium

CVSS:

AV:Network/AC:High/PR:None/UI:Required/S:Unchanged/C:Low/I:Low/A:None

Status:

As explained by the Customer, the inline scripts currently cannot be removed because it would break the IPFS installations (the JS files have to be loaded by the relative URL).

The dependencies migration to the compiled/runtime versions (without JS or WASM evaluation) is planned in the future releases.

Scope:

- tornadocash.eth.limo

- tornadocash.eth.link
- cloudflare-ipfs.com/ipns/tornadocash.eth
- tornadocash.app.runonflux.io

References:

- https://csp-evaluator.withgoogle.com/
- https://securityheaders.com/
- https://stackoverflow.com/questions/68459611/how-to-fix-unsafe-eval-error-with-vue3-for-the-client-side-version
- https://github.com/ajv-validator/ajv/issues/406

Remediation:

- Enable CSP using one of the methods such as an HTTP header or a meta-tag,
- Make the policy as strict as possible without breaking the app,
- Evaluate the policy using the website https://csp-evaluator.withgoogle.com/.

If possible set the following HTTP security headers:

- `Content-Security-Policy` recommended directives:
  - `upgrade-insecure-requests` - to protect against MiTM
  - `script-src, style-src, object-src` - to protect against XSS
  - `base-uri, form-action` - XSS and formjacking
- `Strict-Transport-Security` should be set to `max-age=31536000;` on all sites
- `X-Content-Type-Options` should be set to `nosniff` on all sites
- `X-Frame-Options` should be set to `SAMEORIGIN` on all sites
- `Referer-Policy` should be set to `no-referrer` on all sites

- `Permissions-Policy` should disallow `geolocation=()` on all sites

Example of a non-restrictive CSP configuration that works for the app:

```
<meta http-equiv="Content-Security-Policy" content="img-src 'self' data:;font-src data:;style-src 'self' 'unsafe-inline';connect-src *;script-src 'self' 'unsafe-eval' 'unsafe-inline';default-src 'self';object-src 'none';base-uri 'none';upgrade-insecure-requests">
```

To make it more secure, the inline scripts have to be removed and all the libraries that call `eval` have to be migrated to the runtime versions to make it possible to remove the insecure `unsafe-eval` flag.

Description:

Content Security Policy is a browser feature that enables mitigation of some types of Cross-Site Scripting (XSS) or similar attacks. If CSP is not implemented, there is no direct risk but it makes it easier for the attackers to carry out Cross-Site-Scripting attacks.

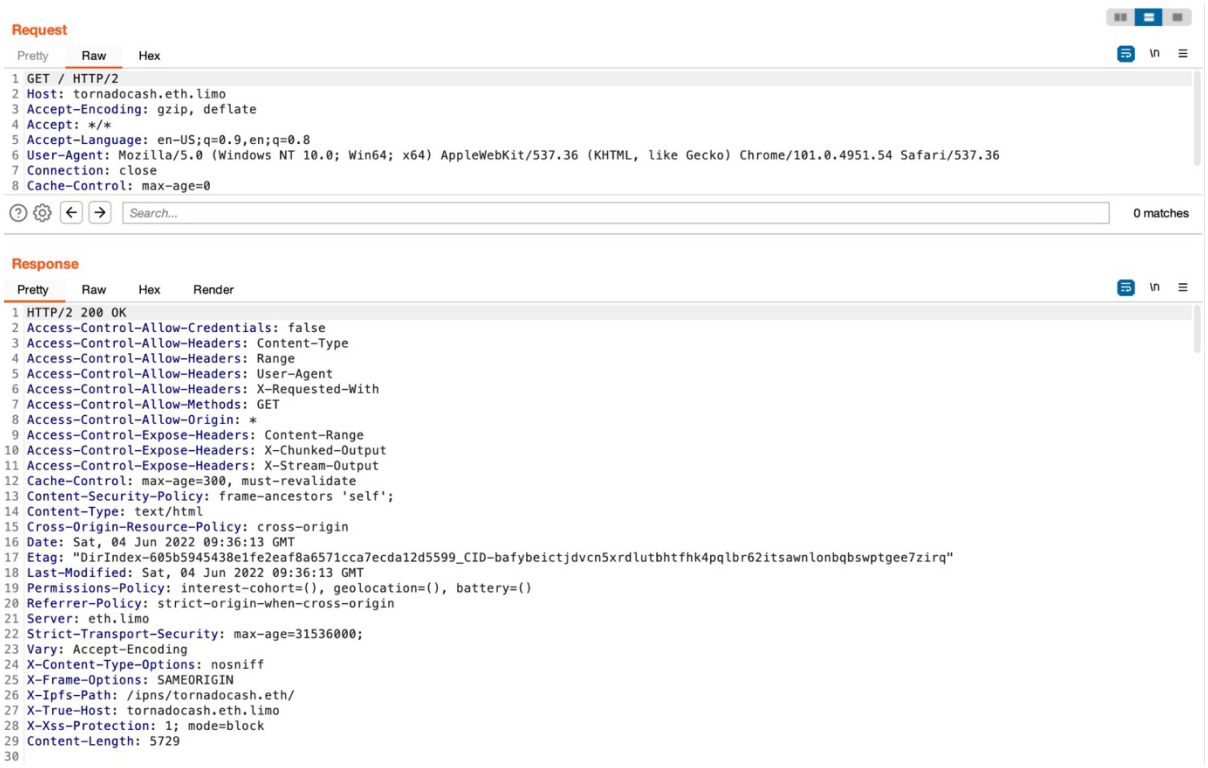
Proofs:

The Tornado Cash UI frontend lacks the HTML-level CSP which could be implemented using the meta-tags.

Additionally, out of four analysed official Tornado Cash web links, the following servers don't have the required HTTP security headers in place:

- https://cloudflare-ipfs.com/ipns/tornadocash.eth/
- https://tornadocash.app.runonflux.io/

The screenshots below show the HTTP responses with misconfigured security headers:



Request

Pretty Raw Hex

```
1 GET / HTTP/2
2 Host: tornadocash.eth.limo
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36
7 Connection: close
8 Cache-Control: max-age=0
```

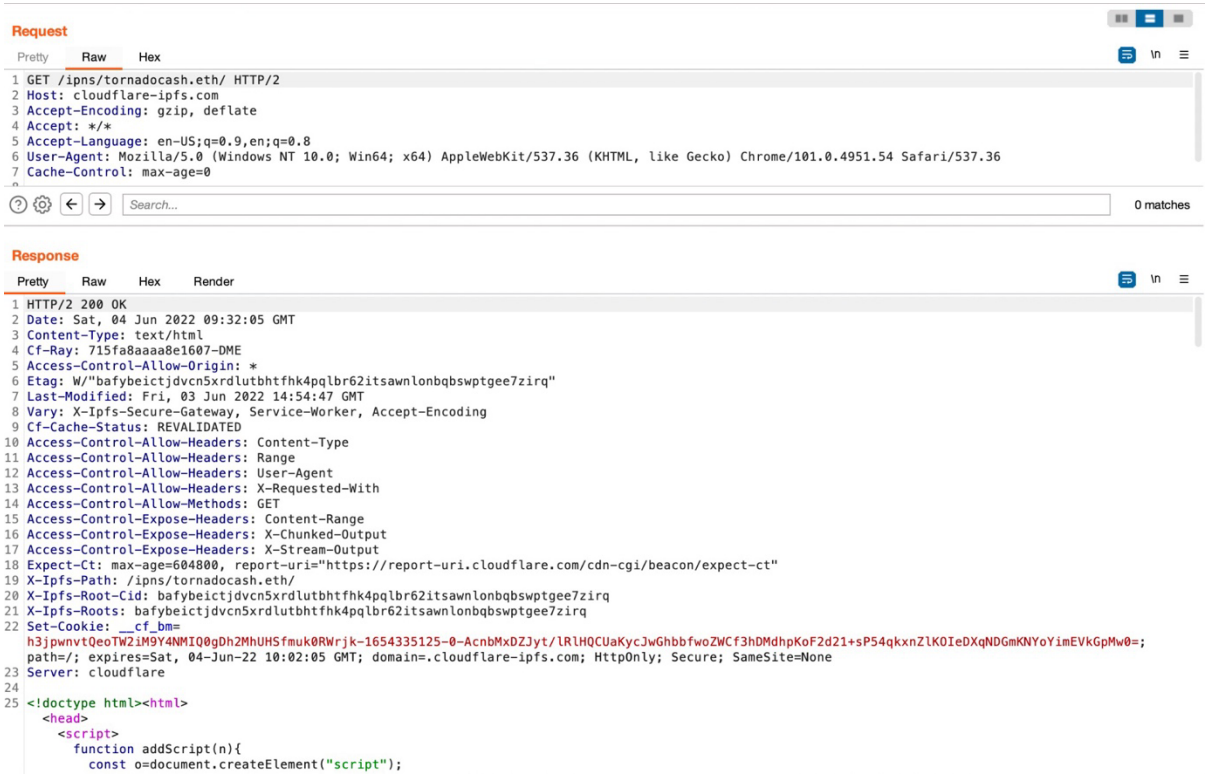
Search... 0 matches

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Access-Control-Allow-Credentials: false
3 Access-Control-Allow-Headers: Content-Type
4 Access-Control-Allow-Headers: Range
5 Access-Control-Allow-Headers: User-Agent
6 Access-Control-Allow-Headers: X-Requested-With
7 Access-Control-Allow-Methods: GET
8 Access-Control-Allow-Origin: *
9 Access-Control-Expose-Headers: Content-Range
10 Access-Control-Expose-Headers: X-Chunked-Output
11 Access-Control-Expose-Headers: X-Stream-Output
12 Cache-Control: max-age=300, must-revalidate
13 Content-Security-Policy: frame-ancestors 'self';
14 Content-Type: text/html
15 Cross-Origin-Resource-Policy: cross-origin
16 Date: Sat, 04 Jun 2022 09:36:13 GMT
17 Etag: "DirIndex-605b5945438e1fe2eaf8a6571cca7ecda12d5599_CID-bafybeictjdvcn5xrdlutbhtfhk4pqlbr62itsawnlonbqbswptgee7zirq"
18 Last-Modified: Sat, 04 Jun 2022 09:36:13 GMT
19 Permissions-Policy: interest-cohort=(), geolocation=(), battery=()
20 Referrer-Policy: strict-origin-when-cross-origin
21 Server: eth.limo
22 Strict-Transport-Security: max-age=31536000;
23 Vary: Accept-Encoding
24 X-Content-Type-Options: nosniff
25 X-Frame-Options: SAMEORIGIN
26 X-Ipfs-Path: /ipns/tornadocash.eth/
27 X-True-Host: tornadocash.eth.limo
28 X-Xss-Protection: 1; mode=block
29 Content-Length: 5729
30
```

Image 3. https://tornadocash.eth.limo



Request

Pretty Raw Hex

```
1 GET /ipns/tornadocash.eth/ HTTP/2
2 Host: cloudflare-ipfs.com
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36
7 Cache-Control: max-age=0
```

Search... 0 matches

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Date: Sat, 04 Jun 2022 09:32:05 GMT
3 Content-Type: text/html
4 Cf-Ray: 715fa8aaaa8e1607-DME
5 Access-Control-Allow-Origin: *
6 Etag: W/"bafybeictjdvcn5xrdlutbhtfhk4pqlbr62itsawnlonbqbswptgee7zirq"
7 Last-Modified: Fri, 03 Jun 2022 14:54:47 GMT
8 Vary: X-Ipfs-Secure-Gateway, Service-Worker, Accept-Encoding
9 Cf-Cache-Status: REVALIDATED
10 Access-Control-Allow-Headers: Content-Type
11 Access-Control-Allow-Headers: Range
12 Access-Control-Allow-Headers: User-Agent
13 Access-Control-Allow-Headers: X-Requested-With
14 Access-Control-Allow-Methods: GET
15 Access-Control-Expose-Headers: Content-Range
16 Access-Control-Expose-Headers: X-Chunked-Output
17 Access-Control-Expose-Headers: X-Stream-Output
18 Expect-Ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
19 X-Ipfs-Path: /ipns/tornadocash.eth/
20 X-Ipfs-Root-Cid: bafybeictjdvcn5xrdlutbhtfhk4pqlbr62itsawnlonbqbswptgee7zirq
21 X-Ipfs-Roots: bafybeictjdvcn5xrdlutbhtfhk4pqlbr62itsawnlonbqbswptgee7zirq
22 Set-Cookie: __cf_bm=
   h3jpwnvtQeoTW2iM9Y4NMIQ0gDh2MhUHSfmuk0RWrjk-1654335125-0-AcnbMxDZJyt/lRlHQCUaKycJwGhbbfwoZWCf3hDMdhpKoF2d21+sP54qkxnZlKOIeDXqNDGmKNYoYimEVkGpMw0=;
   path=/; expires=Sat, 04-Jun-22 10:02:05 GMT; domain=.cloudflare-ipfs.com; HttpOnly; Secure; SameSite=None
23 Server: cloudflare
24
25 <!doctype html><html>
     <head>
       <script>
         function addScript(n){
           const o=document.createElement("script");
```

Image 4. cloudflare-ipfs.com/ipns/tornadocash.eth

4.3 Lack of Referrer Policy

Risk Level: Medium

CVSS:

AV:Network/AC:Low/PR:High/UI:Required/S:Unchanged/C:Low/I:None/A:None

Status:

Partial fix in the commit `b91b81f5c9967a2b09116ff3e340e17d2e6c4feb`: all the "`_blank`" links have the "`noreferrer`" flag in place.

Scope:

- cloudflare-ipfs.com/ipns/tornadocash.eth
- tornadocash.app.runonflux.io

References:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy

Remediation:

Set the Referrer-Policy header or HTML meta tag to "`no-referrer`".

Description:

Referrer Policy controls behaviour of the Referer header, which indicates the origin or web page URL the request was made from. The web application uses insecure Referrer Policy configuration that may leak user's information to third-party sites.

Proofs:

Our of four official app locations, only the 2 of them (tornadocash.eth.limo and tornadocash.eth.link) have the referrer policy in place.

The other two don't implement the policy and therefore might leak the referrer data.

Requests to the following URLs made by the Tornado Cash app during normal interaction contain the referrer data:

https://mainnet.infura.io/v3/9b8f0ddb3e684ece890f594bf1710c88

https://api.thegraph.com/subgraphs/name/tornadocash/mainnet-tornado-subgraph

https://eth-mainnet.alchemyapi.io/v2/Mr-4mfUSoT5xfF6iHbrdAEoGEE9hVzQa

https://registry.walletconnect.com/api/v2/wallets

all relayers (https://mainnet.fi-box.xyz/status, https://mainnet.tornadorelayer.cc/status, https://mainnet.torn-relay.com/status, etc)

4.4 Window opener hijacking (Tabnabbing)

Risk Level: Low

CVSS:

AV:Network/AC:High/PR:Low/UI:Required/S:Unchanged/C:Low/I:None/A:None

Status:

The weakness doesn't need to be fixed in the modern browsers, however, the fix (`noopener` flag added to the links) has been rolled out in the commit `b91b81f5c9967a2b09116ff3e340e17d2e6c4feb`.

Scope:

- components/Navbar.vue
- components/EncryptedTx.vue
- components/Notices.vue
- components/Footer.vue

- components/MetamaskNavbarIcon.vue
- components/governance/Proposal.vue
- components/governance/manage/tabs/DelegateTab.vue
- components/governance/manage/tabs/UndelegateTab.vue
- components/withdraw/Withdraw.vue
- components/Tx.vue
- components/Job.vue
- pages/index.vue
- pages/compliance.vue

References:

- https://developer.mozilla.org/en-US/docs/Web/API/Window/open
- https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types/noopener

Remediation:

When creating a link to an external document using the `<a>` tag with a defined target, for example `"_blank"` or a named frame, provide the `rel` attribute with a value `"noopener noreferrer"`.

If opening the external document in a new window via javascript, then reset the opener by setting the `windowFeatures` parameter with a value `'noopener,noreferrer'`.

Description:

When a user navigates to a new window created by the `window.open` call or clicks a link to an external site ("target"), the `target="_blank"` attribute causes the target site's contents to be opened in a new window or tab,

which runs in the same process as the original page. The `window.opener` object is the reference to the original page that opened the new tab or window. If an attacker can run script on the target page, then they could read or modify certain properties of the `window.opener` object, including the location property — even if the original and target site are not the same origin.

An attacker can modify the location property to automatically redirect the user to a malicious site, e.g. as part of a phishing attack. Since this redirect happens in the original window/tab — which is not necessarily visible, since the browser is focusing the display on the new target page — the user might not notice any suspicious redirection.

Note: the attack has become largely obsolete due to the changes in the major browsers. Chrome shipped the default "`noopener`" behaviour for the "`_blank`" links in the early 2021.

Proofs:

The web application produces links to untrusted external sites outside of its sphere of control, but it does not properly prevent the external site from modifying security-critical properties of the `window.opener` object, such as the `location` property.

Example of such link located in the footer:

```
<a
  class="footer-address__value"
  target="_blank"
  :href="addressExplorerUrl(donationsAddress)"
  rel="noreferrer"
```

```
          >{{ donationsAddress }}</a
        >
```

Here, the "noreferer" flag is set but the "noopener" flag is not. An external website can redirect the application user in the parent window when they click such a link.

More insecure references can be found using the keyword "_blank".

4.5 Location host spoofing

Risk Level: Low

CVSS:

AV:Network/AC:High/PR:None/UI:Required/S:Unchanged/C:None/I:None/A:None

Status:

The occurrences of ".includes" have been replaced with the appropriate regular expression checks in the commit b91b81f5c9967a2b09116ff3e340e17d2e6c4feb.

Image 5. Example of a fixed host check

Scope:

- store/relayers.js
- plugins/detectIPFS.js

References: -

Remediation:

Use strict regular expressions instead of substring match.

Description:

The incorrect hostname detection can lead to various vulnerabilities.

Proofs:

The `plugins/detectIPFS.js` contains a potentially faulty procedure that is used to detect if the application was loaded from IPFS:

```
if (window.location.host.includes('tornadocash.netlify.app')) {
  return false
} else if (!domainWhiteList.includes(window.location.host)) {
  console.warn('The page has been loaded from ipfs.io. LocalStorage is disabled')
  return true
}
```

The usage of `.includes` method makes the procedure match the wider set of hostnames instead of just the hardcoded ones.

Similarly, `store/relayer.js` contains a check if the relayer is located in the Onion network:

```
  async getCustomRelayerData({ rootState, state, getters, rootGetters, dispatch }, {
url, name }) {
```

```
const provider = getters.ethProvider.eth

if (!url.startsWith('https:') && !url.startsWith('http:')) {
  if (url.includes('.onion')) {
    url = `http://${url}`
  } else {
    url = `https://${url}`
  }
}
```

As a result, any relayer whose URL contains a substring ".onion" will be loaded over plaintext HTTP.

4.6 Potentially redundant external/HTTP links

Risk Level: Low

Status:

The insecure link to the documentation has been replaced in the commit `b91b81f5c9967a2b09116ff3e340e17d2e6c4feb`:

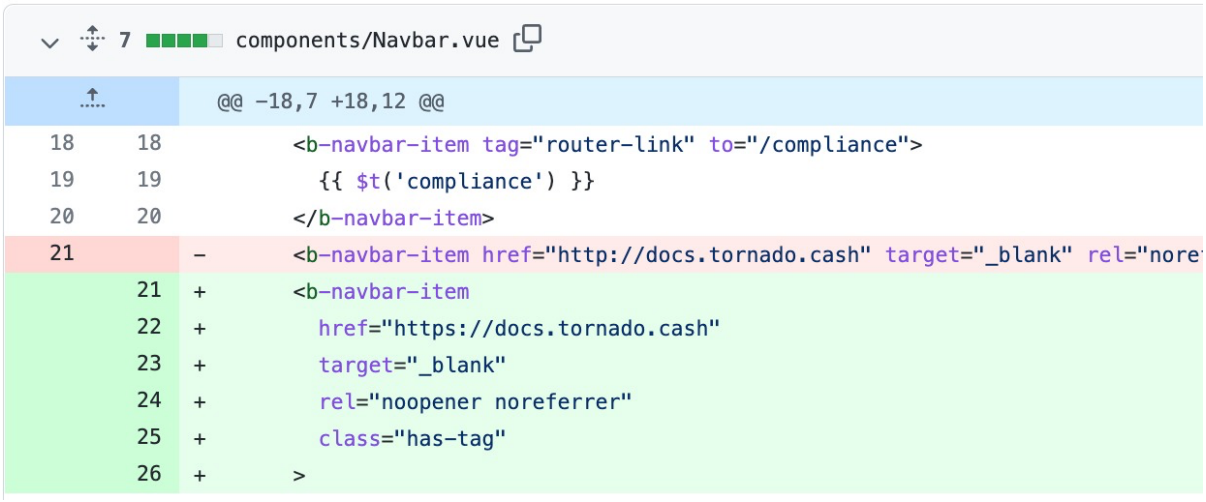


Image 6. Secure documentation link

Scope:

Source code

References: -

Remediation:

- Change the documentation link to https://docs.tornado.cash/
- Review the list of external links, remove unnecessary links.

Description:

Unnecessary external HTTP requests and dependencies increase the attack surface of the application and reduce the users' privacy level.

Proofs:

The application contains a large number of external URLs hardcoded in the source code.

Most of them are necessary for the implementation (e.g. RPC URLs) but some of them may be redundant.

Full list of URLs mentioned in the executable source code is below:

```
http://docs.tornado.cash
https://api.avax.network
https://api.pinata.cloud
https://api.thegraph.com
https://arb-mainnet.g.alchemy.com
https://arb1.arbitrum.io
https://arbiscan.io
https://arbitrum-mainnet.infura.io
https://blockscout.com
https://bsc-dataseed.binance.org
```

https://bsc-dataseed1.defibit.io

https://bsc-dataseed1.ninicoin.io

https://bscscan.com

https://discord.com

https://dune.xyz

https://dweb.link

https://eth-goerli.alchemyapi.io

https://eth-mainnet.alchemyapi.io

https://etherscan.io

https://gateway.pinata.cloud

https://github.com

https://goerli.etherscan.io

https://ipfs.io

https://mainnet.infura.io

https://mainnet.optimism.io

https://nova.tornadocash.eth.link

https://opt-mainnet.g.alchemy.com

https://optimism-mainnet.infura.io

https://optimistic.etherscan.io

https://polygon-mainnet.g.alchemy.com

https://polygon-mainnet.infura.io

https://polygon-rpc.com

https://polygonscan.com

https://rpc.gnosischain.com

https://rpc.xdaichain.com

https://snowtrace.io

https://t.me

https://torn.community

https://tornado-cash.medium.com

https://tornado.cash

`https://twitter.com`

Note that the link to the documentation is insecure as it uses the `http://` scheme.

4.7 Faulty multiple tab detection

Risk Level: Low

Status:

The check has been fixed by adding an event listener in the commit `b91b81f5c9967a2b09116ff3e340e17d2e6c4feb`:

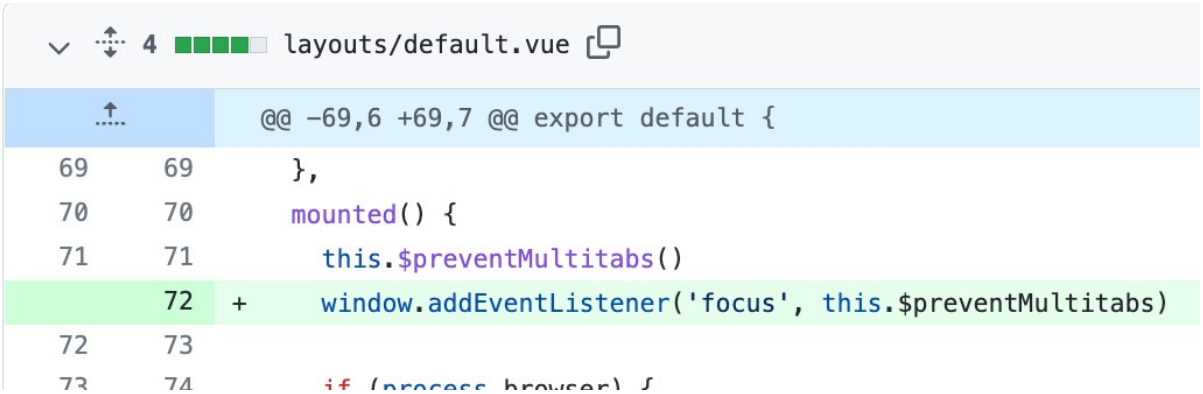


Image 7. Multi-tab prevention fix

Scope:

plugins/preventMultitabs.js

References: -

Remediation:

Fix the implementation if needed (e.g. wrap the first id write in the `setInterval` function).

Description:

When an application is opened in multiple tabs it can lead to undesired consequences such as local storage overwrite.

Proofs:

The multi-tab detection does not work in the following scenario:

- A user loads the application (e.g. https://tornadocash.eth.link/)
- The user opens a new tab and starts loading the same application
- Before the application starts loading, the user quickly moves to a different (non-tornado) tab
- When the user returns to any of the multiple application tabs, both of them are functioning, there're no alerts.

Expected behaviour: the second tab should display a pop-up with the error message and quit: "Multiple tabs opened. Your page will be closed. Please only use single instance of https://tornado.cash".

4.8 Unreachable code

Risk Level: Low

Status:

The redundant code has been deleted in the commit `b91b81f5c9967a2b09116ff3e340e17d2e6c4feb`:

4 components/withdraw/Withdraw.vue

```
@@ -14,6 +14,7 @@
14  14            v-show="!hasErrorNote && depositTxHash"
15  15            :href="txExplorerUrl(depositTxHash)"
16  16            target="_blank"
    17  +         rel="noopener noreferrer"
17  18            class="button is-icon"
18  19          >
19  20            <b-tooltip
@@ -377,9 +378,6 @@ export default {
377 378       this.$emit('get-key', this.getKeys)
378 379     },
379 380     mounted() {
380     -     if (this.$route.query.note) {
381     -       this.withdrawNote = this.$route.query.note
382     -     }
383 381       this.$root.$on('resetWithdraw', () => {
```

Image 8. A snippet of the deleted dead code in the commit diff

Scope:

- components/withdraw/Withdraw.vue (lines 380-382)
- pages/index.vue (lines 87-89)
- pages/index.vue (lines 97-99)

References: -

Remediation:

Consider removing irrelevant code that handles this.$route.query.

Description:

The application contains pieces of code that are not reachable in the execution flow and might degrade the understanding of the code for the maintainers.

Proofs:

The following scenarios behave in accordance with the value of the URL parameter `note`

1. components/withdraw/Withdraw.vue (lines 380-382):

```
if (this.$route.query.note) {
 this.withdrawNote = this.$route.query.note
}
```

2. pages/index.vue (lines 87-89):

```
if (this.$route.query.note) {
 this.activeTab = 1
}
```

3. pages/index.vue (lines 97-99):

```
if (!this.$route.query.note) {
  this.$root.$emit('resetWithdraw')
}
```

However, the entry point app.html performs a redirect removing query parameters when `location.search` is not empty:

```
if (window.location.search) {
  console.log('redirect')
  window.location = window.location.origin + window.location.pathname
}
```

Thus, `this.$route.query` handling in Vue components will not have any effect.

5 APPENDIX

5.1 About us

The Decurity (former DeFiSecurity.io) team consists of experienced hackers who have been doing application security assessments and penetration testing for over a decade.

During the recent years, we've gained an expertise in blockchain field and have conducted numerous audits for both centralized and decentralized projects: exchanges, protocols, and blockchain nodes.

Our efforts have helped to protect hundreds of millions of dollars and make web3 a safer place.