// Security Assessment

10.10.2025 - 10.23.2025

Frontend Staking Application

Babylon Labs

HALBORN

Frontend Staking Application - Babylon Labs

Prepared by: HALBORN

Last Updated 12/04/2025

Date of Engagement: October 10th, 2025 - October 23rd, 2025

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
7	0	0	4	0	3

TABLE OF CONTENTS

1. Summary

2. INTRODUCTION

`Babylon` engaged `Halborn` to conduct a security assessment on their Frontend Staking Application. Halborn was provided access to the application, and its respective source code to conduct security testing using tools to scan, detect, validate possible vulnerabilities found in the application and report the findings at the end of the engagement.

3. ASSESSMENT SUMMARY

The team at Halborn was provided a timeline for the engagement and assigned a full-time security engineer to verify the security of the assets in scope. The security engineer is a penetration testing expert with advanced knowledge in web, blockchain, mobile, recon, discovery & infrastructure penetration testing.

In summary, Halborn identified several medium to low risk vulnerabilities and configuration weaknesses. The findings primarily relate to client-side security controls, dependency management, and web hardening practices. Medium severity issues included the application being susceptible to clickjacking due to the absence of proper frame protection headers, support for obsolete TLS cipher suites, and the use of outdated dependencies containing known vulnerabilities. Additional medium-risk observations involved the absence of a Content Security Policy which could increase the likelihood of client-side injection attacks. Finally, an informational observation was made regarding client side logic of address screening and CDN cache configuration, where arbitrary URLs could be cached, and telemetry logging of wallet related data, representing a potential hardening opportunity. Overall, no critical or high-risk issues were discovered, and the identified items can be remediated through standard configuration updates and adherence to secure development best practices.

The Client team resolved the issue and implemented recommended fixes; some findings were acknowledged or accepted based on design and business requirements.

4. SCOPE

The following URL and its respective repository were in scope:

- https://staking.canon-devnet.babylonlabs.io/
- Release: **https://github.com/babylonlabs-io/babylon-toolkit/releases/tag/v1.2.66**

5. TEST APPROACH & METHODOLOGY

Halborn followed Whitebox and Blackbox methodology as per the scope and performed a combination of manual and automated security testing with both to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the pentest. While manual testing is recommended to uncover flaws in logic, process and implementation; automated testing techniques assist enhance coverage of the infrastructure and can quickly identify flaws in it.

Throughout the assessment, we followed the following phases, but not limited to:

- Mapping application content and functionality
- Identifying application logic flaws
- Reviewing access control and handling of sensitive flows
- Testing authentication and authorization logic
- Rate limiting and brute force resilience testing
- Input validation and response manipulation checks
- Source code review for implementation issues
- Fuzzing of input parameters
- Testing for common injection classes (SQL, JSON, HTML, command)

6. RISK METHODOLOGY

Halborn assesses the severity of findings using either the Common Vulnerability Scoring System (CVSS) framework or the Impact/Likelihood Risk scale, depending on the engagement. CVSS is an industry standard framework for communicating characteristics and severity of vulnerabilities in software. Details can be found in the CVSS Specification Document published by F.I.R.S.T.

Vulnerabilities or issues observed by Halborn scored on the Impact/Likelihood Risk scale are measured by the LIKELIHOOD of a security incident and the IMPACT should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

- **10** - CRITICAL
- **9 - 8** - HIGH
- **7 - 6** - MEDIUM
- **5 - 4** - LOW
- **3 - 1** - VERY LOW AND INFORMATIONAL

7. SCOPE

REPOSITORY

(a) Repository: babylon-toolkit

(b) Assessed Commit ID: 8c1a639

REMEDIATION COMMIT ID:

- https://github.com/babylonlabs-io/babylon-toolkit/commit/b651204
- https://github.com/babylonlabs-io/babylon-toolkit/pull/528

Out-of-Scope: New features/implementations after the remediation commit IDs.

8. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	4	0	3

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
WEBSITE FRAMED BY THIRD PARTY (CLICKJACKING)	MEDIUM	SOLVED - 11/03/2025
OBSOLETE TLS CIPHERS SUPPORTED	MEDIUM	SOLVED - 11/03/2025
OUTDATED AND VULNERABLE DEPENDENCIES	MEDIUM	SOLVED - 11/18/2025

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
MISSING CONTENT SECURITY POLICY	MEDIUM	SOLVED - 11/10/2025
CDN CACHE KEY HARDENING	INFORMATIONAL	SOLVED - 11/10/2025
TELEMETRY CAPTURES WALLET IDENTIFIERS	INFORMATIONAL	SOLVED - 11/10/2025
CLIENT SIDE ENFORCEMENT OF BTC ADDRESS SCREENING	INFORMATIONAL	ACKNOWLEDGED - 10/16/2025

9. FINDINGS & TECH DETAILS

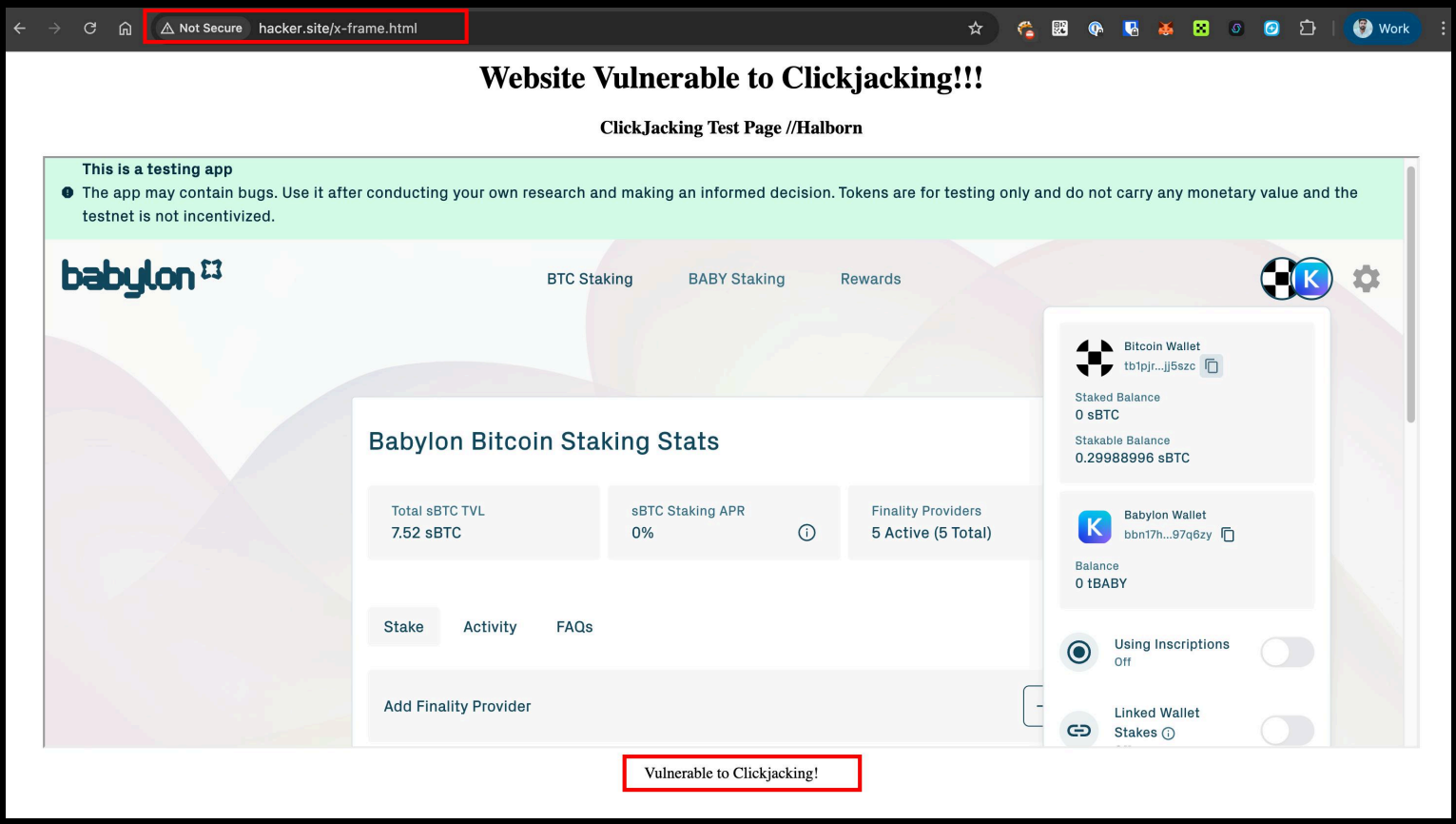
9.1 WEBSITE FRAMED BY THIRD PARTY (CLICKJACKING)

// MEDIUM

Description

It was observed that the Babylon Simple-Staking web application could be rendered inside a third-party iframe because no `X-Frame-Options` header was returned by the server. This allowed the staking interface and wallet interaction components to be displayed within an attacker-controlled domain. The absence of frame restrictions created a risk where a malicious site could visually overlay the legitimate staking dashboard and trick users into performing unintended clicks, such as wallet approvals or staking confirmations. The vulnerability exposed the application to clickjacking and user-interface redress attacks, potentially leading to unauthorized staking actions or sensitive wallet interactions being triggered without informed user consent.

Proof of Concept



Score

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N (5.4)

Recommendation

The application should implement frame protection by including the X-Frame-Options header with a value of DENY or SAMEORIGIN in all HTTP responses. This ensures that Babylon staking pages cannot be embedded within external sites. The header should be configured at the application or reverse proxy level to apply globally across all routes. This control prevents clickjacking by blocking unauthorized framing of the user interface.

Remediation Comment

SOLVED: The **Babylon Labs team** confirmed this issue was limited to the internal development environment used for testing.
Note: This has been retested and verified on the following URL:
`https://staking.testnet.babylonlabs.io/`

References

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
https://owasp.org/www-community/attacks/Clickjacking

9.2 OBSOLETE TLS CIPHERS SUPPORTED

// MEDIUM

Description

It was observed that the frontend endpoint accepted obsolete TLS cipher suites that relied on CBC block chaining. These ciphers are considered outdated and weaker than modern AEAD-based suites used in TLS 1.3. Supporting obsolete ciphers increases the cryptographic attack surface and may expose encrypted frontend traffic to potential downgrade or side-channel risks. Although exploitation is complex, the configuration did not align with current best practices for secure HTTPS communication.

Proof of Concept

```
Testing all IPv4 addresses (port 443): 104.18.1.7 104.18.0.7
---------------------------------------------------------------------------------------------------------------
 Start 2025-10-14 19:34:17        -->> 104.18.1.7:443 (staking.canon-devnet.babylonlabs.io) <<--

 Further IP addresses:   104.18.0.7 2606:4700::6812:107 2606:4700::6812:7
 rDNS (104.18.1.7):      --
 Service detected:       HTTP

 Testing protocols via sockets except NPN+ALPN

 SSLv2      not offered (OK)
 SSLv3      not offered (OK)
 TLS 1      not offered
 TLS 1.1    not offered
 TLS 1.2    offered (OK)
 TLS 1.3    offered (OK): final
 NPN/SPDY   h2, http/1.1 (advertised)
 ALPN/HTTP2 h2, http/1.1 (offered)

 Testing cipher categories

 NULL ciphers (no encryption)                      not offered (OK)
 Anonymous NULL Ciphers (no authentication)        not offered (OK)
 Export ciphers (w/o ADH+NULL)                     not offered (OK)
 LOW: 64 Bit + DES, RC[2,4], MD5 (w/o export)      not offered (OK)
 Triple DES Ciphers / IDEA                         not offered
 Obsoleted CBC ciphers (AES, ARIA etc.)            offered
 Strong encryption (AEAD ciphers) with no FS       offered (OK)
 Forward Secrecy strong encryption (AEAD ciphers)  offered (OK)


 Testing vulnerabilities

 Heartbleed (CVE-2014-0160)                not vulnerable (OK), no heartbeat extension
 CCS (CVE-2014-0224)                       not vulnerable (OK)
 Ticketbleed (CVE-2016-9244), experiment.  not vulnerable (OK)
 ROBOT                                     not vulnerable (OK)
 Secure Renegotiation (RFC 5746)           supported (OK)
 Secure Client-Initiated Renegotiation     not vulnerable (OK)
 CRIME, TLS (CVE-2012-4929)                not vulnerable (OK)
 BREACH (CVE-2013-3587)                    potentially NOT ok, "gzip" HTTP compression detected. - only supplied "/" tested
                                           Can be ignored for static pages or if no secrets in the page
 POODLE, SSL (CVE-2014-3566)               not vulnerable (OK), no SSLv3 support
 TLS_FALLBACK_SCSV (RFC 7507)              No fallback possible (OK), no protocol below TLS 1.2 offered
 SWEET32 (CVE-2016-2183, CVE-2016-6329)    not vulnerable (OK)
 FREAK (CVE-2015-0204)                     not vulnerable (OK)
 DROWN (CVE-2016-0800, CVE-2016-0703)      not vulnerable on this host and port (OK)
                                           make sure you don't use this certificate elsewhere with SSLv2 enabled services, see
                                           https://search.censys.io/search?resource=hosts&virtual_hosts=INCLUDE&q=018A8B338AC4391C670D1CFFF8687EE933CAA45B5E3D0A34285635F687230AF0
 LOGJAM (CVE-2015-4000), experimental      not vulnerable (OK): no DH EXPORT ciphers, no DH key detected with <= TLS 1.2
 BEAST (CVE-2011-3389)                     not vulnerable (OK), no SSL3 or TLS1
 LUCKY13 (CVE-2013-0169), experimental     potentially VULNERABLE, uses cipher block chaining (CBC) ciphers with TLS. Check patches
 Winshock (CVE-2014-6321), experimental    not vulnerable (OK)
 RC4 (CVE-2013-2566, CVE-2015-2808)        no RC4 ciphers detected (OK)
```

Score

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N (4.8)

Recommendation

The TLS configuration should be updated to disable all CBC-based cipher suites and prefer modern AEAD ciphers such as AES-GCM or ChaCha20-Poly1305 under TLS 1.2 and 1.3. It is recommended to review the load balancer or CDN TLS settings and apply a cipher policy that enforces strong encryption and forward secrecy.

Remediation Comment

SOLVED: The **Babylon Labs team** solved the issue as per the recommendation by implementing wallet data redaction in telemetry to reduce privacy exposure.
Note: This has been retested and verified on the following URL:
`https://staking.testnet.babylonlabs.io/`

9.3 OUTDATED AND VULNERABLE DEPENDENCIES

// MEDIUM

Description

It was identified that the Babylon Simple-Staking frontend contained multiple outdated dependencies affected by publicly disclosed vulnerabilities. The project's npm audit output reported 13 issues, including critical and high-severity vulnerabilities. Make sure these components were not directly used in production code; relying on vulnerable versions introduces supply chain risk and may expose the project to future exploitation if affected packages become part of the runtime environment.

Proof of Concept

```
└─λ npm audit
No fix available
node_modules/babel-core/node_modules/lodash
node_modules/babel-plugin-proto-to-assign/node_modules/lodash
  babel-plugin-proto-to-assign  *
  Depends on vulnerable versions of lodash
  node_modules/babel-plugin-proto-to-assign

minimatch  <=3.0.4
Severity: high
Regular Expression Denial of Service in minimatch - https://github.com/advisories/GHSA-hxm2-r34f-qmc5
minimatch ReDoS vulnerability - https://github.com/advisories/GHSA-f8q6-p94x-37v3
No fix available
node_modules/babel-core/node_modules/minimatch

tmp  <=0.2.3
tmp allows arbitrary temporary file / directory write via symbolic link `dir` parameter - https://github.com/advisories/GHSA-52f5-9888-hmc6
fix available via `npm audit fix`
node_modules/external-editor/node_modules/tmp
node_modules/tmp
  external-editor  >=1.1.1
  Depends on vulnerable versions of tmp
  node_modules/external-editor
    @changesets/cli  2.10.0 - 2.29.5 || >=3.0.0-next.0
    Depends on vulnerable versions of external-editor
    node_modules/@changesets/cli

13 vulnerabilities (5 low, 3 moderate, 4 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues possible (including breaking changes), run:
  npm audit fix --force

Some issues need review, and may require choosing
a different dependency.
```

```
13 vulnerabilities (5 low, 3 moderate, 4 high, 1 critical)
```

Score

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:L/A:N (4.7)

Recommendation

The project dependencies should be reviewed and updated to the latest secure versions to reduce supply chain risk. Unmaintained or vulnerable packages should be replaced or removed where possible. Continuous dependency monitoring should be implemented using `npm audit` or similar tooling to ensure newly disclosed vulnerabilities are identified and remediated promptly.

Remediation Comment

SOLVED: The **Babylon Labs team** solved the issue by updating all relevant dependencies to their latest stable versions.

Remediation Hash

https://github.com/babylonlabs-io/babylon-toolkit/commit/b651204

9.4 MISSING CONTENT SECURITY POLICY

// MEDIUM

Description

It was observed that the Babylon Simple-Staking frontend did not include a Content-Security-Policy (CSP) header in its HTTP responses. The absence of a CSP meant that the browser was not instructed to restrict the sources of executable scripts, styles, images, or other resources loaded by the application. Without this control, the frontend was exposed to a higher risk of client-side attacks such as cross-site scripting, data injection, and malicious third-party content execution. The lack of a defined CSP weakened the browser's ability to enforce trusted origins and increased the potential impact of any injected or compromised script.

Proof of Concept

```
└─λ http --headers https://staking.canon-devnet.babylonlabs.io/btc
HTTP/1.1 200 OK
CF-RAY: 98e7bf6b4b52e345-IST
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html
Date: Tue, 14 Oct 2025 14:25:45 GMT
Permissions-Policy: accelerometer=(), autoplay=(), cross-origin-isolated=(), display-capture=(), encrypted-media=(), fullscreen=(), geolocation=(), gyroscope=(), keyboard-map=(), magn
etometer=(), microphone=(), midi=(), payment=(), picture-in-picture=(), publickey-credentials-get=(), screen-wake-lock=(), sync-xhr=(), usb=(), web-share=(), xr-spatial-tracking=()
Server: cloudflare
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Transfer-Encoding: chunked
X-Content-Type-Options: nosniff
cf-cache-status: DYNAMIC
last-modified: Mon, 13 Oct 2025 16:32:38 GMT
via: 1.1 75373f3f77c169166bbce98d302dff7c.cloudfront.net (CloudFront)
x-amz-cf-id: vvvYhWoH1pwevZGXH-ohZOrjUyUAy000seHGzhX3kaHu-7t0FQCDgw==
x-amz-cf-pop: VIE50-P1
x-cache: Error from cloudfront
```

Score

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N (4.2)

Recommendation

A Content Security Policy (CSP) should be implemented to restrict the sources from which scripts, styles, and other resources can be loaded. The policy should allow content only from trusted domains and block inline or unsafe code execution. It should be applied at the application or web server level to ensure consistent enforcement across all frontend responses, reducing the risk of client-side script injection and data compromise.

Remediation Comment

SOLVED: The **Babylon Labs team** resolved the issue and updated the CSP. The 'wasm-unsafe-eval' directive was added to support the required WASM dependency, as no known bypasses exist at this time, with plans to review and refine this configuration in future releases.

Note: This has been retested and verified on the following URL:

`https://staking.testnet.babylonlabs.io/`

9.5 CDN CACHE KEY HARDENING

// INFORMATIONAL

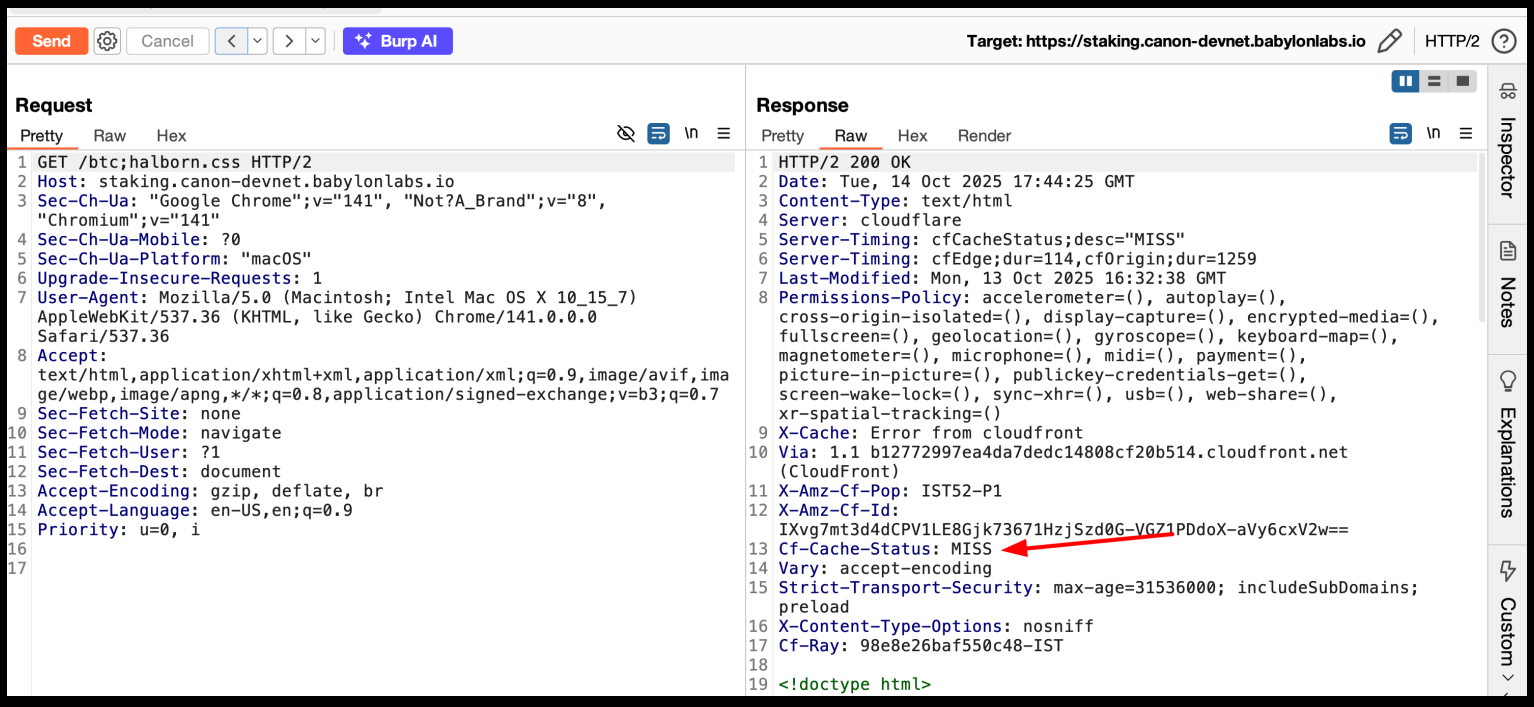
Description

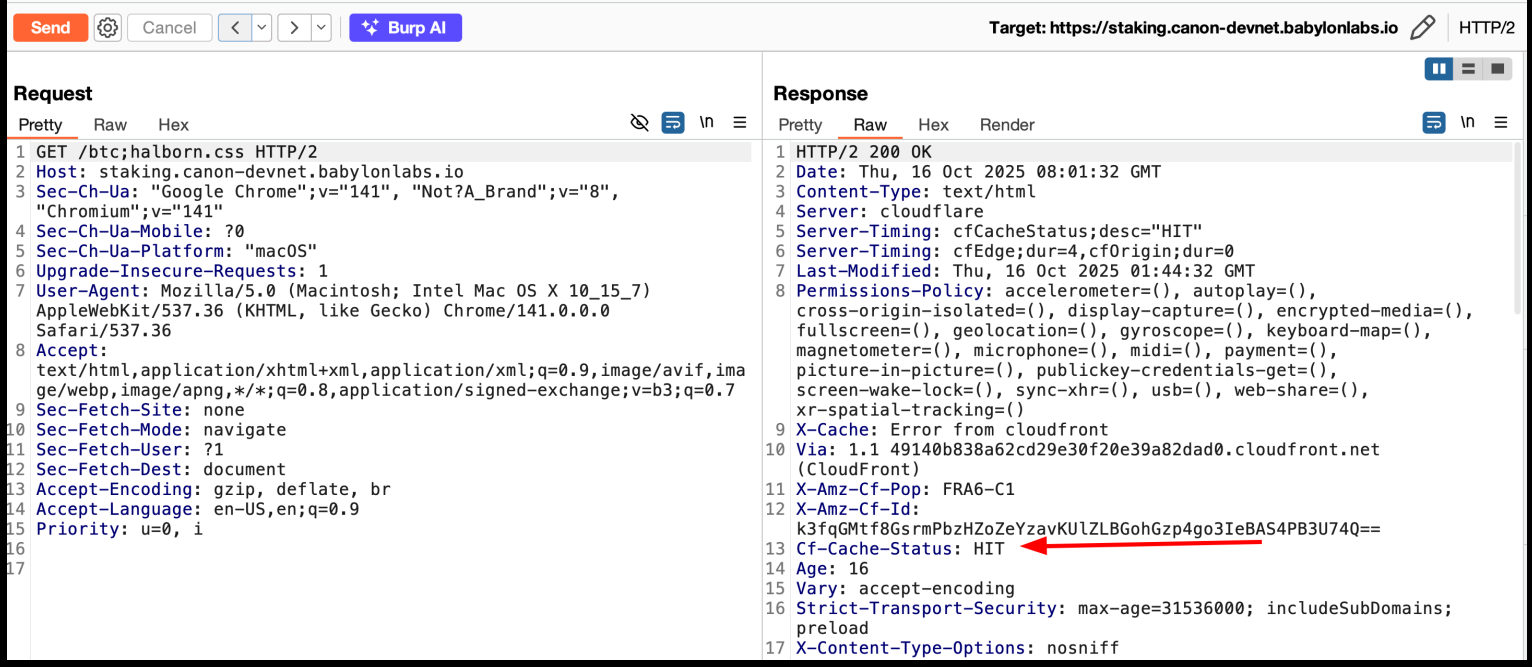
It was observed that the Babylon staking frontend allowed arbitrary URL variants (e.g., `/btc;halborn.css`) to be cached by Cloudflare/CDN. During testing, the first request returned a cache miss (CF-Cache-Status: MISS), and subsequent identical requests returned a cache hit (CF-Cache-Status: HIT), confirming the behavior.

This indicates that the CDN treats unexpected URL patterns as cacheable static content instead of dynamic or invalid paths. While no user-specific or sensitive data was observed in the cached responses. Since the application serves a static React frontend and retrieves wallet data client-side from public blockchain sources, the misconfiguration may still allow cache poisoning or stale content delivery in the future.

If later versions introduce server-rendered data or API prefetching, this caching behavior could cause sensitive or dynamic responses to be unintentionally stored and served to other users.

Proof of Concept





Score

(0.0)

Recommendation

Restrict CDN caching to known static asset paths. Disable caching for HTML responses by using Cache-Control: no-store or equivalent CDN rules. Normalize or reject unusual path suffixes and other invalid extensions to prevent arbitrary cache entries. This ensures caching remains performance-optimized without risk of poisoning or unintended storage in future builds.

Remediation Comment

SOLVED: The issue was fixed through CDN configuration updates.

Note: This has been retested and verified on the following URL:

`https://staking.testnet.babylonlabs.io/`

9.6 TELEMETRY CAPTURES WALLET IDENTIFIERS

// INFORMATIONAL

Description

During the review of the wallet connection flow, it was observed that both BTC and Babylon wallet providers log wallet metadata such as addresses, derived public keys, and wallet names through `useLogger`, which forwards data to `@sentry/react`. These logs are sent alongside other connection and retry metadata to Sentry. While wallet addresses are public on the blockchain, logging them to a centralized telemetry service can correlate on-chain identifiers with off-chain context such as browser sessions, device metadata, and timestamps. This correlation could expand privacy exposure if the telemetry platform or associated access credentials were ever compromised.

Proof of Concept

The following files were observed logging wallet identifiers through the useLogger utility.

Code Location:

```
src/ui/common/context/wallet/BTCWalletProvider.tsx
src/ui/common/context/wallet/CosmosWalletProvider.tsx
src/ui/common/hooks/services/useAddressScreeningService.ts
src/ui/common/hooks/useLogger.ts
```

No sensitive data exposure or exploitability was identified, but the current telemetry design increases the privacy footprint and may introduce compliance considerations.

Score

(0.0)

Recommendation

Implement data minimization controls within the telemetry flow. Hash or redact wallet identifiers before sending them to Sentry, or enable beforeSend filtering to prevent full identifiers from being logged. Restrict detailed identifier logging to debugging or non-production environments where necessary. These measures preserve observability and debugging capabilities while reducing the risk of correlating on-chain identities with off-chain session data.

Remediation Comment

SOLVED: The **Babylon Labs team** implemented wallet data redaction in telemetry, reducing privacy exposure.

Remediation Hash

https://github.com/babylonlabs-io/babylon-toolkit/pull/528

9.7 CLIENT SIDE ENFORCEMENT OF BTC ADDRESS SCREENING

// INFORMATIONAL

Description

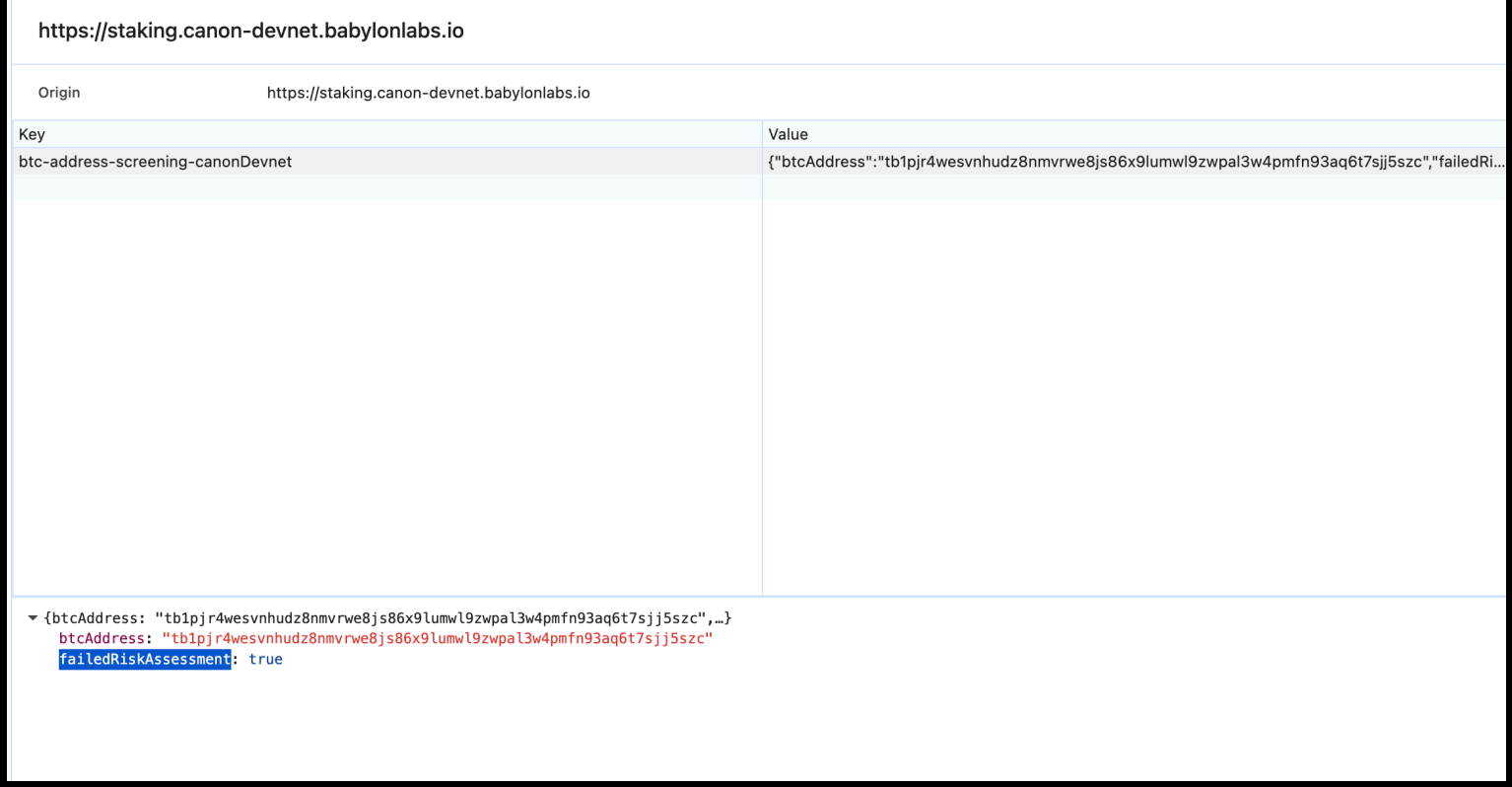
The BTC address screening logic was implemented entirely on the frontend, and the screening result was cached in localStorage under a key such as `btc-address-screening-${network}` . This value determined whether staking actions were allowed within the dApp. Because the decision was stored and trusted client-side, it could be easily modified by an end user or injected script to bypass the screening restriction. Although this behavior does not directly compromise blockchain integrity, it weakens the intended compliance control and could allow restricted or high-risk addresses to use the official staking interface without proper verification. The absence of a defined TTL or revalidation mechanism further increases the risk of stale or tampered results being used to approve staking actions.

Proof of Concept

```
export const BTC_ADDRESS_SCREENING_KEY = btc-address-screening-${network};

export interface BtcAddressScreening {
  btcAddress: string;
  failedRiskAssessment: boolean;
}

export function getBtcAddressScreeningResult():
  | BtcAddressScreening
  | undefined {
  try {
    const value = localStorage.getItem(BTC_ADDRESS_SCREENING_KEY);
    if (!value) return undefined;
    const parsed = JSON.parse(value);
    return parsed;
  } catch {
    return undefined;
  }
}
```



Score

(0.0)

Recommendation

It is recommended to have a backend reverification before staking rather than relying solely on frontend logic. Further, to have a more robust approach, implement a short-lived, server-signed token or TTL-based recheck to ensure results remain valid and tamper-resistant. Ensure that, on any error or failed screening, the default is to treat the address as risky (not safe), to avoid false negatives and potential bypass. This approach strengthens enforcement within the official dApp while maintaining flexibility for decentralized users.

Remediation Comment

ACKNOWLEDGED: The Babylon Labs team confirmed that this behavior matches the intended architecture of the dApp, which currently relies on client initiated requests.

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.