Borg Security

Security Report

Biconomy

By Borg Security

Monday, 6th of October 2025

About Borg Security

Established in 2024 and based in Trondheim, Norway, Borg Security delivers specialized security services to some of the most security-sensitive organizations worldwide. Our approach blends rigorous security research with a deep understanding of adversarial tactics to identify and mitigate risks in web applications and networks. Borg Security has played a critical role in reinforcing the security of major applications used by millions.

Borg Security began as a solution to secure Web3 infrastructure that was vulnerable to Web2 attacks. We aim to be the bridge between Web2 and Web3 Security. We are recognized for our leadership in blockchain and traditional security.

To stay current with our latest projects and insights, follow us on X at @borg_security, or explore our case studies at borgsecurity.io/case-studies.

For inquiries or to connect with us directly, please reach out through our contact form at borgsecurity.io/contact or send an email to contact@borgresearch.io.

Borg Research AS
Kjøpmannsgata 59
Trondheim, 7011
https://borgsecurity.io
contact@borgresearch.io

Notices & Remarks

This report has been designated as public information by Borg Security. Biconomy has requested that the report be made public. No part of this material may be copied, shared, or republished, either in part or in full, without written permission from Borg Security.

Following the submission of this report, Biconomy acknowledged all 5 findings and remediated one. The effectiveness of this mitigation has been confirmed through a retest conducted by Borg Security. As a result, no outstanding security issues remain from the initial assessment.

Vulnerability	Status
#1 Inconsistent Input Validation Between Frontend and Backend Allows for Copy Trades Below $50 Limit	Acknowledged
#2 Outdated NextJS Weakens Hypersignals' Security Posture	Acknowledged
#3 Overly Permissive Onboarding Progress API Enables Arbitrary Key/Value Injection	Acknowledged
#4 Access Token Doesn't Expire on Logout	Acknowledged
#5 Broken Access Control on POST /api/auth/v2/login	Mitigated ✔

Table of Contents | Hypersignals

Summary

Executive Summary

This report is a security analysis conducted by Borg Security to identify security vulnerabilities in Hypersignals' infrastructure. This report contains all relevant findings from a black-box penetration test. Recommended steps, potential impacts, and mitigation suggestions accompany all findings.

Vulnerability Summary

5	0	0	0	1	4
Total Findings	Critical	High	Medium	Low	Informational

0	Critical	Critical risks impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
0	High	High risks can include centralization issues and logical errors. Under specific circumstances, these high risks can result in the loss of funds and/or control of the project.
0	Medium	Medium risks may not pose a direct risk to users' funds, but they can affect a platform's overall functioning.
1	Low	Low risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
4	Informational	Informational errors often include recommendations to improve the style of the code or suggest specific operations that align with industry best practices. They usually do not affect the overall functioning of the code.

Scope

Assets

Asset	Type	Identified Tech
www.hypersignals.ai	Domain (Web Faced)	React
staging.hypersignals.ai	Domain (Web Faced)	Vercel
dev.hypersignals.ai	Domain (Web Faced)	Vercel
competition.hypersignals.ai	Domain (Web Faced)	NextJS, Vercel, React
app.hypersignals.ai	Domain (Web Faced)	NextJS, Vercel, React
blog.hypersignals.ai	Domain (Web Faced)	
u002fblog.hypersignals.ai		

Attack Vectors Discovered

Vulnerability Type	Notes
Insufficient Session Expiration (CWE-612)	Access Token does not expire on a user's logout.
Using Components With Known Vulnerabilities (CWE-937)	NextJS 14.2.12 with known vulnerabilities is used on the Hypersignals main app.
Client-Side Enforcement of Server-Side Security (CWE-602)	Backend trusts frontend validation for the minimum copy trade amount.
Improper Input Validation (CWE-20)	The onboarding status allows for arbitrary keys and values.

Approach & Methods

Generic

Borg Security conducted a comprehensive security assessment using manual reviews and static analysis techniques. Our black-box penetration testing focused on the following key areas:

- Evaluating Authentication Flows and Token Management.
- Analyzing requests and responses for adherence to industry standards and best practices.
- Verifying the integrity of Hypersignal's infrastructure to ensure user funds.

Targeted

We focused on ensuring that user funds are safe and that data is handled properly, not being leaked or transmitted unsafely. The tests consisted of all web-related vulnerabilities that could apply to Hypersignals' infrastructure and features.

There was extensive testing on access control, such as IDORs, authorization misconfigurations, parameter pollution, and HTTP request smuggling, to name a few. We also did server-side injections to spot any places where SQL injections, remote code execution, server-side template injection, and similar vulnerabilities could exist. Of which we found no indication.

Observations

Export Wallet

Hypersignals' export wallet functions trust Privy's integration to make sure an attacker cannot access a user's private key. Although it is Privy's infrastructure, we made sure that the integration does not add any security risk to Hypersignals. The integration makes sure that private keys are not transmitted in plaintext, and only the user with the access token can manage to retrieve the parts needed to decode their private key. This process means that the user's access token is enough in itself to export a user wallet, which is why we extra carefully validated the access token management, making sure that even small misconfigurations do not exist on Hypersignals.

Withdrawals

Hypersignals' withdrawal also relies on Privy's integration to ensure security and reliability for users. The inputs cannot be tampered with, and since this is an on-chain transaction, few attack vectors, web-wise, apply here. The withdrawal is overall safe using Privy, ensuring that all vectors are handled properly. Nonetheless, the access token is the only factor needed to withdraw money, as with the export wallet.

Copy Trading

Hypersignals' copy trading flow relies on Privy tokens for authentication. To start a copy trade, the user sends a `POST` request to `/api/copy-trade/start` with parameters including `address_to_copy` and `capital_allocation_proportion`, which determines what percentage of the user's portfolio is allocated. The frontend enforces a $50 minimum allocation, but the backend does not perform this validation. By sending a percentage close to zero, making the $ amount $0.007, trades can be created below the intended threshold. In these cases, the frontend may incorrectly display the allocation as "10%", but on the backend, the trade is actually initiated with

the low value. Once a copy trade has been initiated, the process runs independently of the user and cannot be further modified outside of the provided edit and stop endpoints.

Editing a copy trade happens via `POST /api/copy-trade/edit?trading_process_id=ID`. The body allows updates to parameters, including `capital_allocation_proportion`. The same frontend/backend mismatch exists here: the frontend enforces a $50 minimum, but the backend does not. The `trading_process_id` values are user-specific and tied to the authenticated account, preventing IDOR attacks. Without the correct user's Privy token, another user cannot edit or stop an existing trade.

Stopping a copy trade is handled by `POST /api/copy-trade/stop?trading_process_id=ID`, which again requires the correct Privy token. Only the owner of the process can successfully stop it.

Race conditions were tested on the start endpoint. In one case, two 200 OK responses were received from concurrent requests, but only one trade was executed, deducting funds once. Further attempts to replicate this behaviour failed, suggesting that either backend state checks or an on-chain mechanism prevents duplication.

Overall, the copy trading system functions correctly with appropriate access controls tied to the Privy token, and trading processes cannot be modified by unauthorized users. However, the reliance on frontend validation of the minimum trade size could be seen as a weakness, as backend validation is absent. This allows users to initiate copy trades with values far below the intended threshold.

Findings

#1 | Inconsistent Input Validation Between Frontend and Backend Allows for Copy Trades Below $50 Limit

Type	Client-Side Enforcement of Server-Side Security (CWE-602)
Severity	**CVSS 0** (Informational)
Asset	https://app.hypersignals.ai/

We discovered that Hypersignals relies on client-side enforcement on the minimum threshold of the copy trade amount. The endpoints `/api/copy-trade/start` and `api/copy-trade/edit?trading_process_id=ID` both have the same client-side enforcement, meaning that you could both start a copy trade process and edit a copy trade process with amounts under $50.

This means users can bypass the minimum-allocation control and initiate/edit copy trade with near-zero funds. While this doesn't grant unauthorized access, it weakens the integrity of trading policies and might affect accounting and monitoring.

Fix/Suggested Mitigation:

We recommend enforcing the $50 minimum server-side for both start and edit paths. Reject requests where the computed dollar allocation is below $50. This would mitigate this issue, but this is still informational, since it has no direct security impact.

Remediation Status

Acknowledged. The development team has reviewed the finding and decided not to address it at this stage.

#2 | Outdated NextJS Weakens Hypersignals' Security Posture

Type	Using Components with Known Vulnerabilities (CWE-937)
Severity	**CVSS 1** (Low)
Asset	https://app.hypersignals.ai/

Hypersignals is currently running an outdated NextJS release. Relying on an older framework increases exposure to publicly known vulnerabilities in NextJS and its transitive dependencies. We did not confirm an exploit path specific to the deployed version during testing, hence the low severity score. However, using outdated tech on a platform is not considered best practice, and it weakens Hypersignals' security posture.

Fix/Suggested Mitigation:

While no direct impact on Hypersignals appears to be present due to the known vulnerabilities, we would still recommend updating to the newest version without any known vulnerabilities. This would strengthen Hypersignals' security posture.

Remediation Status

Acknowledged. The development team has reviewed the finding and decided not to address it at this stage.

#3 | Overly Permissive Onboarding Progress API Enables Arbitrary Key/Value Injection

Type	Improper Input Validation (CWE-20)
Severity	**CVSS 0** (Informational)
Asset	https://app.hypersignals.ai/

The endpoint `/v1/onboarding/visit` accepts client-supplied JSON and persists it with minimal validation. By sending crafted POST requests, a user can create or modify arbitrary onboarding sections, which are reflected on the endpoint `/v1/onboarding/status`. This is only reflected on the authenticated user's onboarding profile, meaning that no one other than the user setting the onboarding status will see it.

This does not serve as a direct security risk; however, seeing as improper input validation has been spotted multiple times, this weakens Hypersignals' security posture.

Fix/Suggested Mitigation:

We recommend adding proper input validation and distrusting client-side input, making sure that all fields provided by the user are both expected values and keys. By only allowing keys that are expected, such as whale-moves and top-trader-moves, a user cannot add arbitrary key names and values.

Remediation Status

Acknowledged. The development team has reviewed the finding and decided not to address it at this stage.

#4 | Access Token Doesn't Expire on Logout

Type	Insufficient Session Expiration (CWE-613)
Severity	**CVSS 0** (Informational)
Asset	https://app.hypersignals.ai/

The access token provided on login, privy-token, lacks proper invalidation on a user's logout. When a user logs out of their Hypersignals account, the access token remains valid, although removed from the user's local storage and cookie storage.

This increases the impact of access token leakage, since a malicious actor could have a valid session that the user is unaware of. The access token expires automatically after one hour, making this informational, based on the prerequisites that need to be in place in a short period of time.

Fix/Suggested Mitigation:

We recommend invalidating the user's access token server-side, not only removing it from the user's local storage and cookie storage. This would strengthen the security posture of Hypersignals and reduce the severity of access token leakage.

Remediation Status

Acknowledged. The development team has reviewed the finding and decided not to address it at this stage.

#5 | Broken Access Control on POST /api/auth/v2/login

Type	Authorization Bypass Through User-Controlled Key (CWE-639)
Severity	**CVSS 0** (Informational)
Asset	https://app.hypersignals.ai/

The `POST /api/auth/v2/login` endpoint accepts a client-supplied `walletAddress` and returns the corresponding `user_id` and wallets. By changing `walletAddress` to an address not owned by the authenticated session, the API responds with another user's `user_id` and wallet list. The server is using the client-supplied identifier for lookups without verifying ownership.

This does not constitute a serious security risk on its own, but it discloses user IDs and wallet addresses, which can be used for targeted reconnaissance and may be chained with other flaws to escalate impact.

Fix/Suggested Mitigation:

Verify that the authenticated user is the owner of the supplied `walletAddress` before returning any data.

Remediation Status

Fixed. The development team implemented a patch addressing this issue, and retesting confirms that the vulnerability is no longer present.

Borg Security | **Secure** your Most **Valuable** Assets

Borg Security was established as a response to the need for enhanced security measures within Web3 infrastructure. Despite the decentralized nature of many Dapps, vulnerabilities stemming from Web2 technologies persist. Borg Security aims to bridge this gap by providing comprehensive security solutions.

80M+	**< 24h**	**100%**
Users Secured	Exploit Mitigation / Prevention	Satisfaction Guarantee



borgsecurity.io | @borg_security