



HBarSuite – WebApp and SmartNode

FrontEnd and BackEnd Pentest

Prepared by: Halborn

Date of Engagement: March 27th, 2023 – May 2nd, 2023

Visit: Halborn.com

DOCUMENT REVISION HISTORY	3
CONTACTS	3
1 EXECUTIVE OVERVIEW	4
1.1 INTRODUCTION	5
1.2 AUDIT SUMMARY	5
1.3 SCOPE	6
1.4 TEST APPROACH & METHODOLOGY	6
RISK METHODOLOGY	7
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	9
3 FINDINGS & TECH DETAILS	10
3.1 (HAL-01) DENIAL OF SERVICE AFFECTING SMART NODES - CRITICAL	12
Description	12
Evidences	12
Risk Level	13
Recommendation	13
Remediation Plan	13
3.2 (HAL-02) LACK OF NFT CREATION WHEN ADDING LIQUIDITY TO A POOL - CRITICAL	14
Description	14
Code Location	14
Risk Level	14
Recommendation	14
Remediation Plan	15
3.3 (HAL-03) HTML INJECTION - MEDIUM	16
Description	16

Evidence	17
Risk Level	17
Recommendation	17
Remediation Plan	18
3.4 (HAL-04) UNDEFINED VALUE ON MALICIOUS POOL CREATION - LOW	19
Description	19
Evidences	19
Risk Level	20
Recommendation	20
Remediation Plan	20

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/02/2023	Erlantz Saenz
0.2	Draft Review	05/04/2023	Gabi Urrutia
0.3	Document Additions	05/05/2023	Erlantz Saenz
0.4	Document Additions Review	05/05/2023	Gabi Urrutia
1.0	Remediation Plan	05/25/2023	Erlantz Saenz
1.1	Remediation Plan Review	05/26/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Erlantz Saenz	Halborn	Erlantz.Saenz@halborn.com



EXECUTIVE OVERVIEW



1.1 INTRODUCTION

HBarSuite engaged Halborn to conduct a security audit on their web application and their underlying infrastructure, beginning on March 27th, 2023 and ending on May 2nd, 2023. The security assessment was scoped to the web application provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned a full-time security engineer to audit the security of the web application. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that application functions operate as intended
- Identify potential security issues with the smart node

In summary, Halborn identified two critical vulnerabilities, one medium and one low, that should be addressed by the client.

Smart Nodes used by the application were vulnerable to Denial of Service (DoS) attacks, this could allow an attacker to take down two of the nodes to stop the whole application. Additionally, the application was not minting NFTs when adding liquidity to the pool due to an error in a third-party provider.

The application contained few security flaws in the data validation, this allowed an attacker to introduce malicious characters or information in the request that were not correctly validated by the backend.

Additionally, one of the main characteristics of the platform, the Smart Rebalance Market Maker (SRMM) was tested during the audit and no security flaws were found. The application contained multiple mechanism that would stop an incorrect balance on the application, modifying accordingly the pool.

1.3 SCOPE

The scope of the assessment was provided for the items below and their underlying infrastructure:

- <https://hsuite-finance.web.app>

1.4 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the penetration test. While manual testing is recommended to uncover flaws in logic, process and implementation; automated testing techniques assist enhance coverage of the solution and can quickly identify flaws in it.

The following phases and associated tools were used throughout the term of the audit:

- Mapping application content and functionality
- Technology stack-specific vulnerabilities
- Vulnerable or outdated dependencies
- Exposure of any critical information during user interactions with the blockchain and external libraries
- Attacks that could impact funds, such as draining or manipulating of funds
- Application logic flaws
- Authentication / Authorization flaws
- Cross-environment issues
- Lack of validation on input forms
- Brute force protections
- Input handling
- Rate limiting
- Fuzzing
- Multiple parameter injection techniques (SQL/JSON/HTML/Command/Directories...)

- Sensitive information disclosure

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring 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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
2	0	1	1	0

LIKELIHOOD

IMPACT

				(HAL-01) (HAL-02)
		(HAL-03)		
	(HAL-04)			

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
DENIAL OF SERVICE AFFECTING SMART NODES	Critical	SOLVED - 05/22/2023
LACK OF NFT CREATION WHEN ADDING LIQUIDITY TO A POOL	Critical	SOLVED - 05/22/2023
HTML INJECTION	Medium	SOLVED - 05/25/2023
UNDEFINED VALUE ON MALICIOUS POOL CREATION	Low	SOLVED - 05/23/2023



FINDINGS & TECH DETAILS



3.1 (HAL-01) DENIAL OF SERVICE AFFECTING SMART NODES – CRITICAL

Description:

A Denial-of-Service (DoS) attack is an attack meant to affect an application, making it inaccessible. DoS attacks accomplish this by overloading the target with traffic, or sending it information that could derive in high resources' consumption.

The Smart Nodes were affecting by this issue when an attacker using a single machine could overflow their resources.

Evidences:

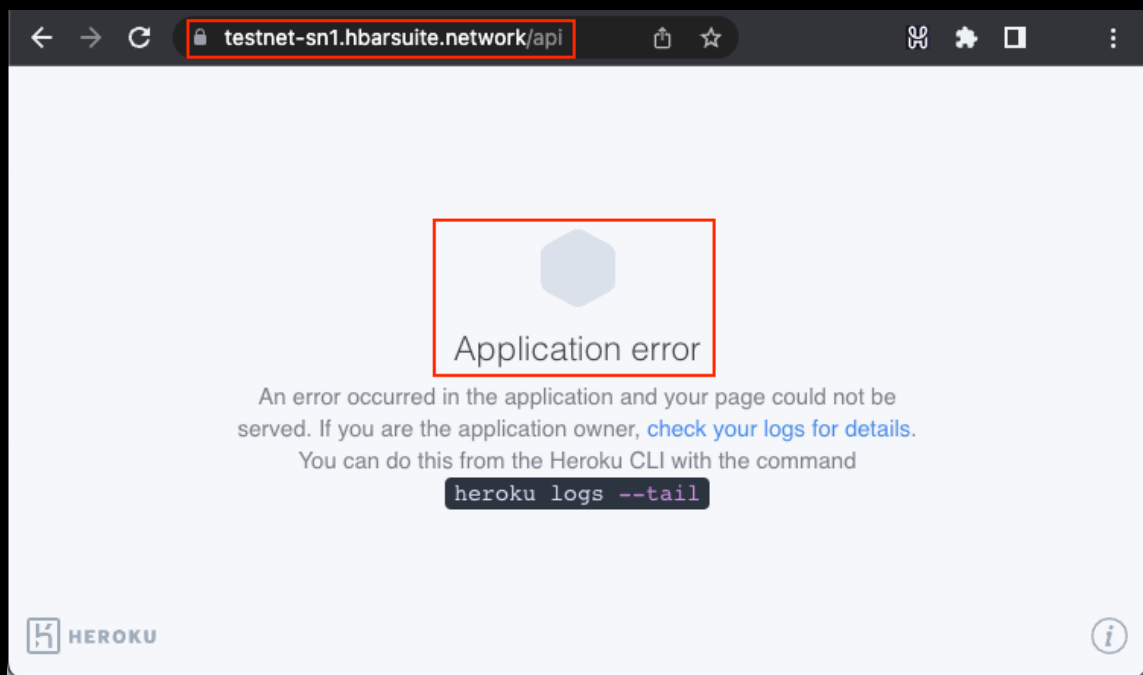


Figure 1: Application error when visiting the resource.

Row	Payload	Status	Words	Length	Time	Label
8589		503	371	1187	102	
8617		503	370	1185	107	
8618		503	367	1179	110	
8620		503	367	1179	99	
8621		503	370	1185	102	
8622		503	373	1191	102	
8623		503	371	1187	113	
8624		503	370	1185	115	
8690		503	368	1181	105	
8736		503	369	1183	103	
8768		503	369	1183	99	
8818		503	370	1185	110	
8862		503	367	1179	103	
8876		503	371	1187	106	
8887		503	374	1193	98	
9015		503	367	1179	104	
9017		503	373	1191	103	

Pretty	Raw	Hex	Render
1 GET /markets/price?tokenId=0.0.34719665 HTTP/1.1			1 HTTP/1.1 503 Service Unavailable
2 Host: testnet-snl.hbarsuite.network			2 Date: Mon, 28 Nov 2022 20:37:18 GMT
3 Accept: application/json			3 Content-Type: text/html; charset=utf-8
4			4 Transfer-Encoding: chunked
5			5 Connection: keep-alive
			6 Cache-Control: no-cache, no-store
			7 CF-Cache-Status: DYNAMIC
			8 Report-To: {
			"endpoints": [{
			"url": "https://a.nel.cloudflare.com/report/v3?s=GbSMt9eIux4gQL%2F0wyjubiwgzkKQ9f%2BxcmFRRstLvzJp7WMRFfidie0becgD8fzSbauf7QryUD8QX0vxU1RnDy%2FrHjUIFVnFpGR004iuld%2BeBnj1cptkbHbnhhFLuoDLqFvFVBiwUb8cJUzp20DgQ%3D%3D"}],
			"group": "cf-nel",
			"max_age": 604800
			}
			9 NEL: {
			"success_fraction": 0,
			"report_to": "cf-nel",
			"max_age": 604800
			}

Reqs: 34872 | Queued: 100 | Duration: 61 | RPS: 572 | Connections: 3770 | Retries: 3 | Fails: 1768 | Next:

Figure 2: Multiple errors on the application

Risk Level:**Likelihood - 5****Impact - 5****Recommendation:**

Configure a WAF or a denial of service protection in order to adjust the number of requests that a user could send to the applications.

Remediation Plan:

SOLVED: The issue was solved by the **HBarSuite team** and verified by Halborn. During the retesting phase, it was possible to denial of service the previously affected items.

3.2 (HAL-02) LACK OF NFT CREATION WHEN ADDING LIQUIDITY TO A POOL - CRITICAL

Description:

One of the functionalities of HBarSuite DEX is the liquidity addition to a pool. This functionality allows a user to add liquidity in to a pool, minting an NFT that is sent to the user's wallet, and it is burned when the user claims back the liquidity provided.

It was found during the audit that the functionality, stopped working during the last days of the audit. The application was depositing the client's liquidity in to the pool, but no NFTs were minted. This situation made impossible to a user to claim back the liquidity or observe the liquidity added in to the different pools.

Note: The HBarSuite team confirmed that the functionality stopped working regarding changes in the 3rd party provider.

Code Location:

Risk Level:

Likelihood - 5

Impact - 5

Recommendation:

Modify the application accordingly to the changes of the 3rd party provider. Additionally, add a functionality that could allow a user to mint the NFT after adding liquidity to the pool, in case that something fails, the user could mint the liquidity NFT afterwards.

Remediation Plan:

SOLVED: The issue was solved by the **HBarSuite team** and verified by Halborn. During the retesting phase, it was possible to observe that the application was minting the corresponding NFT linked to the liquidity added.

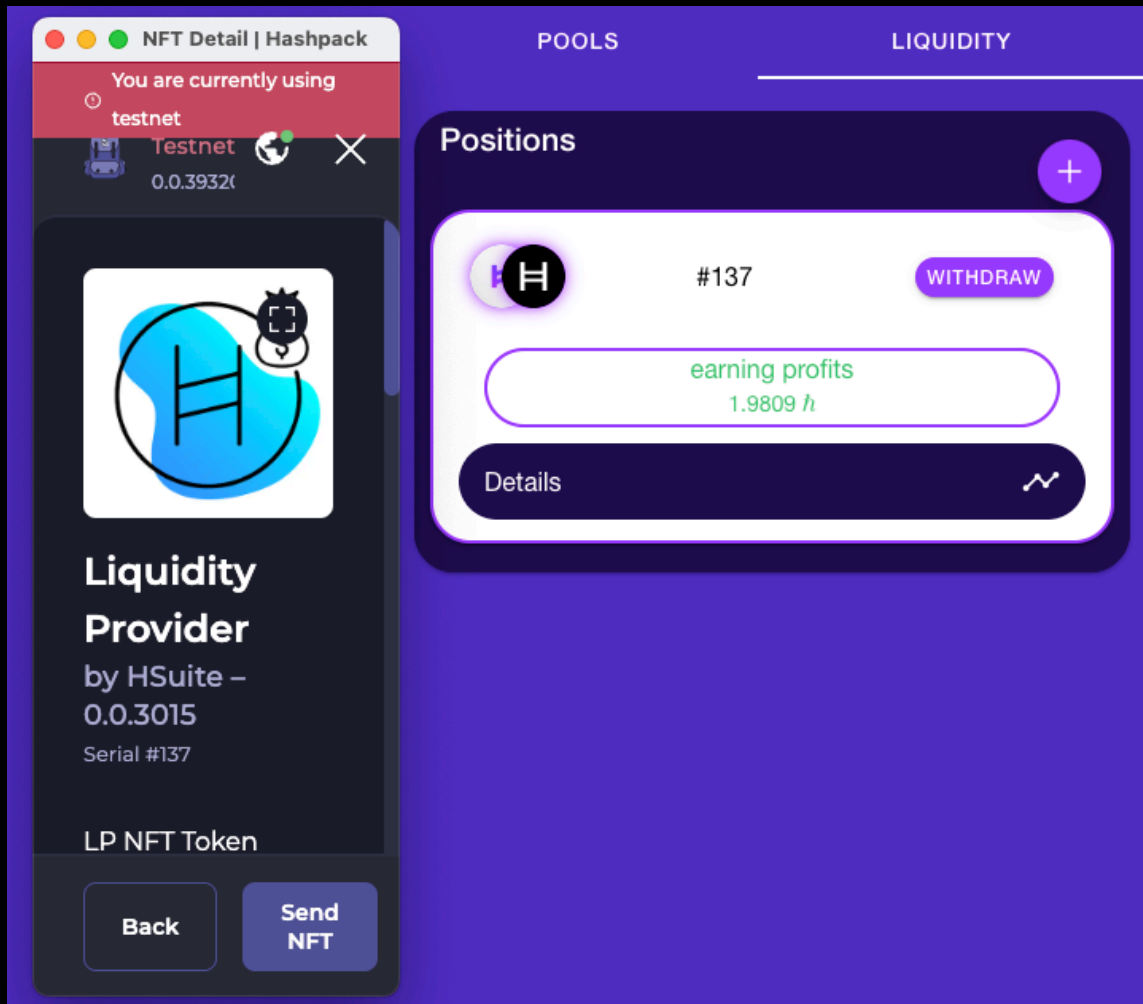


Figure 3: NFT created after adding liquidity to a pool.

3.3 (HAL-03) HTML INJECTION - MEDIUM

Description:

HyperText Markup Language (HTML) injection is a technique used to take advantage of non-validated input to modify the legitimate behavior of the application. When the application fails to validate the input data, it allows an attacker to perform malicious inputs that could be presented to other users. The range of attacks could vary from simple visual modifications to redirects to malicious web applications where the attacker could perform further attacks.

In this case, the web application presented an HTML injection on DAO creation. This could allow an attacker to inject malicious HTML tags and modify the appearance of the application. However, during the assessment not all the available HTML tags were interpreted by the application, reducing the attack surface.

Evidence:

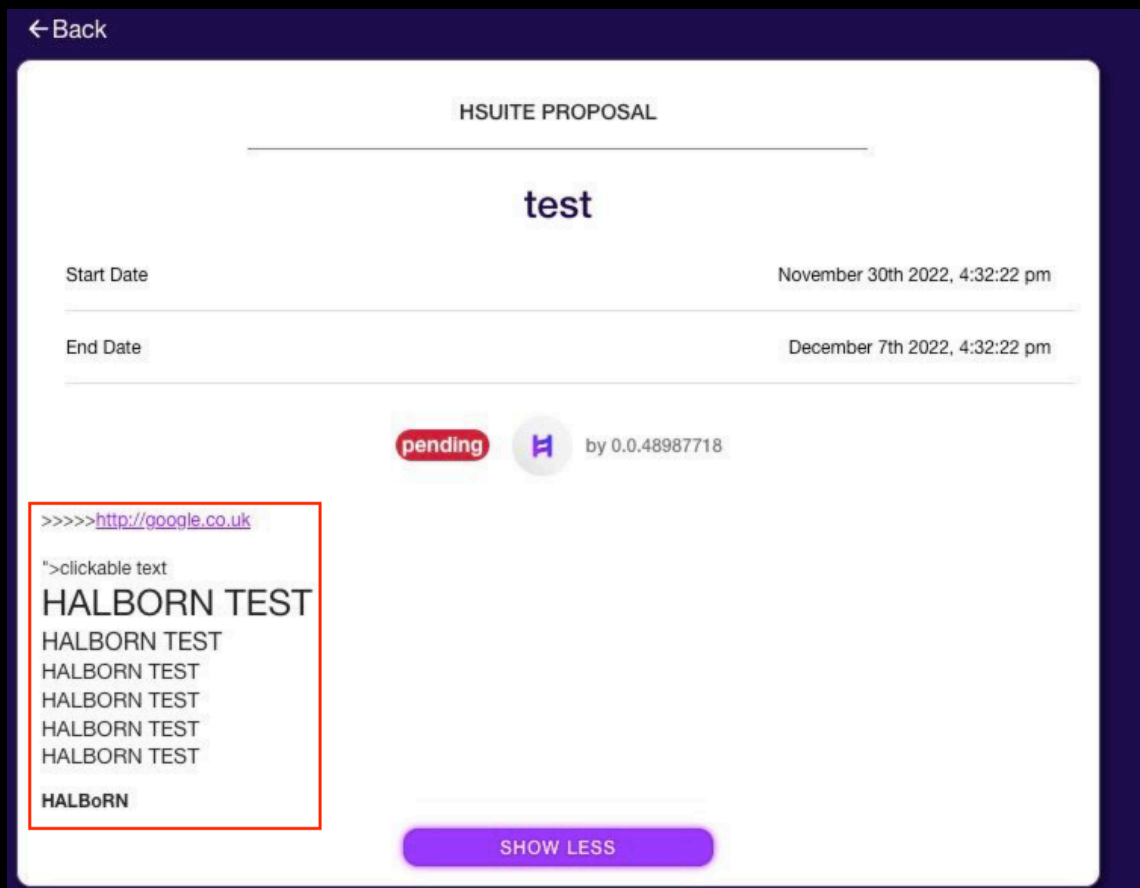


Figure 4: HTML injection on DAO creation

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

Filter any kind of malicious character sent to the server. Additionally, limit the types of characters that a user could inject into the application or escape, everything outside of alphanumeric characters.

Remediation Plan:


SOLVED: The issue was solved by the **HBarSuite team** and verified by Halborn. It was possible to observe that the application was verifying the malicious input data and sanitizing it before executing any action.

3.4 (HAL-04) UNDEFINED VALUE ON MALICIOUS POOL CREATION – LOW

Description:

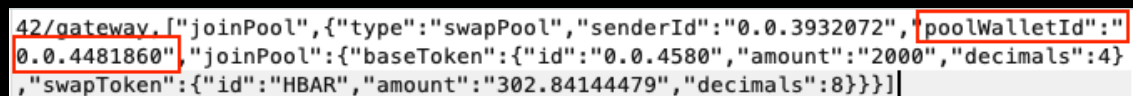
When a user tries to make a swap between two tokens that do not exist on the DEX, the application was offering to the user to create a pool between both currencies. However, when the requests to create a new pool were tampered and modified to create a malicious pool under the attacker's control, the application was detecting this as `undefined`. This could mean that the application did not have a control to catch an exception when a pool was manipulated and rejected by the application. However, no subsequent issues were found regarding this issue.

Evidences:



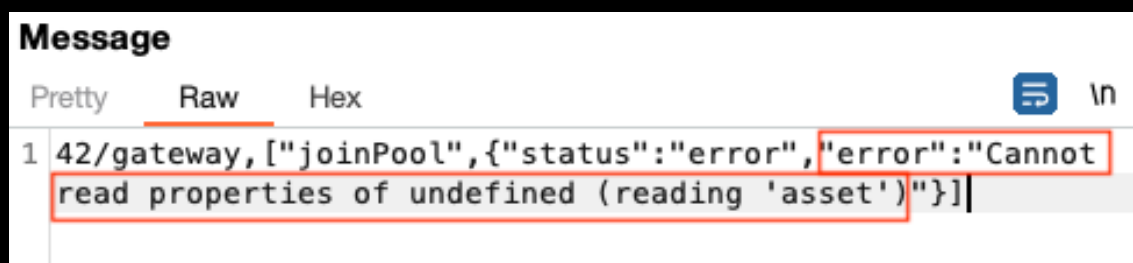
```
42/gateway, [{"joinPool": {"type": "swapPool", "senderId": "0.0.3932072", "poolWalletId": "0.0.4051084", "joinPool": {"baseToken": {"id": "0.0.4580", "amount": "2000", "decimals": 4}, "swapToken": {"id": "HBAR", "amount": "302.84144479", "decimals": 8}}}]
```

Figure 5: Legitimate request to create a new pool



```
42/gateway, [{"joinPool": {"type": "swapPool", "senderId": "0.0.3932072", "poolWalletId": "0.0.4481860", "joinPool": {"baseToken": {"id": "0.0.4580", "amount": "2000", "decimals": 4}, "swapToken": {"id": "HBAR", "amount": "302.84144479", "decimals": 8}}}]
```

Figure 6: Malicious request to create a pool under attacker's control



Message

Pretty Raw Hex

```
1 42/gateway, [{"status": "error", "error": "Cannot read properties of undefined (reading 'asset')"}]
```

Figure 7: Undefined value detected on the backend

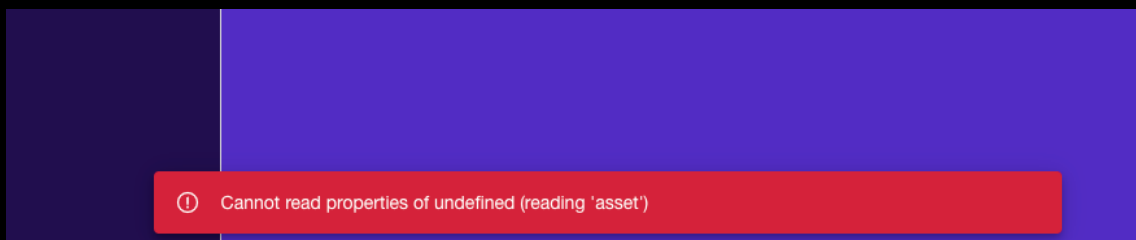


Figure 8: Error on the application

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

Improve the error logging into the application, implementing correct error messages when the backend sanitizes the data or undefined values.

Remediation Plan:

SOLVED: The issue was solved by the **HBarSuite team** and verified by Halborn. During the retesting phase, it was possible to observe that the application was verifying the legitimate creation of a pool.



THANK YOU FOR CHOOSING

// HALBORN

