CredShields
Audit Report For Noft Games

Aug 5th, 2022 • CONFIDENTIAL

Description

This document details the process and result of the security audit performed by CredShields Technologies PTE. LTD. on behalf of Noft Games between May 12th, 2022, and May 22nd, 2022.

Author

Shashank (Co-founder, CredShields)

shashank@CredShields.com

Reviewers

Aditya Dixit (Research Team Lead)

aditya@CredShields.com

Prepared for

Noft Games

Table of Contents

1. Executive Summary

Noft Games engaged CredShields to perform a Web application audit from May 12th, 2022, to May 22nd, 2022. During this timeframe, Elevn (11) vulnerabilities were identified.

During the audit, four (4) vulnerabilities were found that had a severity rating of either High or Critical. These vulnerabilities represent the greatest immediate risk to "Noft Games" and should be prioritized for remediation.

The table below shows the in-scope assets and breakdown of findings by severity per asset. *Section 2.3* contains more information on how severity is calculated.

	Critical	High	Medium	Low	Info	Σ
Noft Games	1	3	4	3	0	11
	1	**3**	**4**	**3**	**0**	**11**

Table: Findings Overview

The security audit was conducted by the CredShields team to focus on identifying vulnerabilities in Noft Games scope during the testing window while abiding by the policies set forth by Noft Games.

Maintaining a healthy security posture requires constant review and refinement of existing security processes. Running a CredShields audit allows Noft Games' internal security team

and development team to not only uncover specific vulnerabilities but gain a better understanding of the current security threat landscape.

Reviewing the remaining resolved reports for a root cause analysis can further educate Noft Games' internal development and security teams and allow manual or automated procedures to be put in place to eliminate entire classes of vulnerabilities in the future. This proactive approach helps contribute to future-proofing the security posture of Noft Games' assets.

2. Methodology

Noft Games engaged CredShields to perform a security audit of Noft Games' web and external network. The following sections cover how the engagement was put together and executed.

2.1 Preparation phase

CredShields reviewed all the documents like API and product documentation to understand the features, functionalities, and access control of the Noft Games. The team reviewed all the functions and prepared maps and graphs to better understand the logic flow and access control.

A testing window from May 12th, 2022, to May 22nd, 2022, was agreed upon during the preparation phase.

2.1.1 Scope

During the preparation phase, the following scope for the engagement was agreed-upon:

ASSETS IN SCOPE
Asset 1:

Noft Games Web Applications **- *.noftgames.io** **Asset 2:** **External networks**

Table: Asset(s) in Scope

2.1.2 Audit Goals

Audit procedures at CreditShields involve both automated (in-house) tools and manual analysis. However, the majority of audit methods require a manual review of the application's source code.

The testing was done in accordance with the standards of the OWASP, along with an extended self-developed checklist based on industry standards and best practices for blockchain RPC security. The team focused heavily on the core concept behind all the functionalities, along with preparing test and edge cases. This included understanding the business logic and how it could have been exploited.

During the audit, the focus was on verifying the security, resilience, and compliance of the codebase. The audit activities can be categorized into the following:

- **Security** - The identification of security issues in each application and its interaction with other projects.
- **Sound Architecture** - Analyzing this system's architecture through the lens of established best practices and general software best practices.

2.2 Retesting phase

Noft Games is actively partnering with CredShields to validate the remediations implemented towards the discovered vulnerabilities.

2.3 Vulnerability Classification and Severity

Discovering vulnerabilities is important, but estimating the associated risk to the business is just as important.

To adhere to industry guidelines, CredShields follows OWASP's Risk Rating Methodology. This is calculated using two factors - **Likelihood** and **Impact**. Each of these parameters can take three values - **Low**, **Medium**, and **High.**

These depend upon multiple factors such as Threat agents, Vulnerability factors (Ease of discovery and exploitation, etc.), and Technical and Business Impacts. The likelihood and the impact estimate are put together to calculate the overall severity of the risk.

CredShields also define an **Informational** severity level for vulnerabilities that do not align with any of the severity categories and usually have the lowest risk involved.

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Overall, the categories can be defined as described below -

1. **Informational**

   We believe in the importance of technical excellence and pay a great deal of attention to its details. Our coding guidelines, practices, and standards help ensure that the software we audit is stable and reliable.

   Informational vulnerabilities should not be a cause for alarm but rather a chance to improve the quality of the codebase by emphasizing readability and good practices.

   They do not represent a direct risk to the product but rather suggest improvements and best practices that can not be categorized under any of the other severity categories.

   Code maintainers should use their own judgment as to whether to address such issues.

2. **Low**

   Vulnerabilities in this category represent a low risk to the product and the organization. The risk is either relatively small and may or may not be exploited on a recurring basis, or a risk that the client indicates is not important or significant, given the client's business circumstances.

3. **Medium**

   Medium severity issues are those that are usually introduced due to weak or erroneous logic in the code.

These issues may lead to exfiltration or modification of some of the private information belonging to the end-user, and exploitation would be detrimental to the client's reputation under certain unexpected circumstances or conditions. These conditions are outside the control of the adversary.

These issues should eventually be fixed under a certain timeframe and remediation cycle.

4. **High**

   High severity vulnerabilities represent a greater risk to the product and the organization. These vulnerabilities may lead to a limited loss of confidentiality, integrity, and availability for some of the end-users.

   They may or may not require external conditions to be met, or these conditions may be manipulated by the attacker, but the complexity of exploitation will be higher.

   These vulnerabilities, when exploited, will impact the client's reputation negatively.

   They should be fixed immediately.

5. **Critical**

   Critical issues are directly exploitable bugs or security vulnerabilities. These issues do not require any external conditions to be met.

   The issue puts the vast majority of, or large numbers of, users' sensitive information at risk of modification or compromise.

   The client's reputation will suffer a severe blow, or there will be serious financial repercussions.

2.4 CredShields staff

The following individual at CredShields managed this engagement and produced this report:

- **Shashank, Co-founder CredShields**
  - shashank@CredShields.com

Please feel free to contact this individual with any questions or concerns you have around the engagement or this document.

3. Findings

This chapter contains the results of the security assessment. Findings are sorted by their severity and grouped by the asset and CWE classification if applicable. Each asset section will include a summary. The table in the executive summary contains the total number of identified security vulnerabilities per asset per risk indication.

3.1 Findings Overview

3.1.1 Vulnerability Summary

During the security assessment, a total of eleven (11) security vulnerabilities were identified in the assets.

VULNERABILITY TITLE	SEVERITY	SWC | Vulnerability Type
IDOR to disclose PII	High	CWE-284: Improper Access Control
XSS in Profile Picture	Critical	CWE-79: Cross-Site Scripting
Missing Validation when Creating Battles	High	CWE-20: Improper Input Validation

Login Bruteforce	High	CWE-307: Improper Restriction of Excessive Authentication Attempts
Wp-cron.php Denial of Service	Medium	CWE-400: Denial of Service
XMLRPC Login Bruteforce	Medium	CWE-307: Improper Restriction of Excessive Authentication Attempts
WordPress User Enumeration	Low	CWE-799: Improper Control of Interaction Frequency
XMLRPC Pingback Vulnerabilities	Medium	CWE-918: Server-Side Request Forgery (SSRF)
GraphQL Introspection Enabled	Low	CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
AMQP Cleartext Authentication enabled	Low	CWE-319: Cleartext Transmission of Sensitive Information
Postgres Credential Bruteforce	Medium	CWE-307: Improper Restriction of Excessive Authentication Attempts

3.1.2 Findings Summary

The Credshields security team found multiple security issues on the gaming platform that leaked other user's data. One issue related to unrestricted file upload allowed any user to conduct stored cross-site scripting attacks to steal the user's account credentials. The team also noticed a stale WordPress installation on the server that had multiple security misconfigurations leading to Dos and password brute force attacks.

The team also performed an external network security check and found an exposed Postgres instance with a lack of brute force protection.

4. Remediation Status

Noft Games is actively partnering with CredShields from this engagement to validate the discovered vulnerabilities' remediations. The table shows the remediation status of each finding.

VULNERABILITY TITLE	SEVERITY	REMEDIATION STATUS
IDOR to disclose PII	High	**Fixed [05.08.2022]**
XSS in Profile Picture	Critical	**Fixed [05.08.2022]**
Missing Validation when Creating Battles	High	Invalid Bug
Login Bruteforce	High	**Fixed [05.08.2022]**
Wp-cron.php Denial of Service	Medium	**Fixed [05.08.2022]**
XMLRPC Login Bruteforce	Medium	**Fixed [05.08.2022]**
WordPress User Enumeration	Low	**Fixed [05.08.2022]**
XMLRPC Pingback Vulnerabilities	Medium	**Fixed [05.08.2022]**

GraphQL Introspection Enabled	Low	Won't Fix
AMQP Cleartext Authentication enabled	Low	**Fixed [05.08.2022]**
Postgres Credential Bruteforce	Medium	**Fixed [05.08.2022]**

Table: Summary of findings and status of remediation

5. Bug Reports

Bug ID#1 [Fixed]

IDOR to disclose PII

Vulnerability Type
CWE-284: Improper Access Control

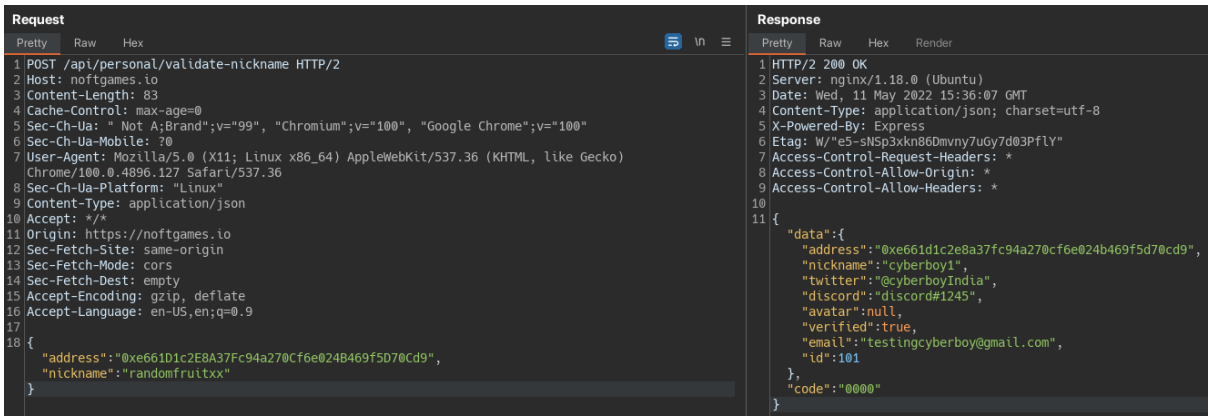
Severity
High

Description:
Insecure Direct Object References occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability, attackers can bypass authorization and access resources in the system directly, for example, database records or files.
Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user-supplied input and uses it to retrieve an object without performing sufficient authorization checks.
The API endpoint to validate the nickname was found to be vulnerable, which allowed the team to view all the user PII including their email addresses stored in the database, just by providing their wallet address which is publicly available.

Vulnerable Endpoint:

PoC:

1. Log in to the application and edit your nickname. A request like the one mentioned below will be sent.
2. Change the value of the “address” parameter in the response with the address of the victim.
3. Their personal details, including their email address, will be displayed in the response.




```
POST /api/personal/validate-nickname HTTP/2
Host: noftgames.io
Content-Length: 83
Cache-Control: max-age=0
Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="100", "Google Chrome";v="100"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/100.0.4896.127 Safari/537.36
Sec-Ch-Ua-Platform: "Linux"
Content-Type: application/json
Accept: */*
Origin: https://noftgames.io
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{"address":"0xe661D1c2E8A37Fc94a270Cf6e024B469f5D70Cd9","nickname":"randomfruitxx"}
```

Impacts:

This vulnerability exposes email address, Twitter, and discord handles, and all the other private details a user stores in their profile for all the users existing on the platform.

Remediation:

Implement proper input validations when validating nicknames, probably adding another signature parameter so that the malicious actors can't access details belonging to other accounts by supplying their addresses.
It is recommended to refer to OWASP's Authorization Cheatsheet.

Retest:
This is fixed. The user's personal details are not being disclosed in the response anymore.

Bug ID#2 [Fixed]

XSS in Profile Picture

Vulnerability Type
CWE-79: Cross-Site Scripting

Severity
Critical

Description:
Cross-Site Scripting (XSS) attacks are a type of injection in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser-side script, to a different end-user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.
The feature to upload the profile picture is vulnerable. A malicious file containing an XSS payload can be uploaded, which is then stored on the server in the context of the application, therefore exposing the user's cookies containing the address and the signature.

Vulnerable URL:

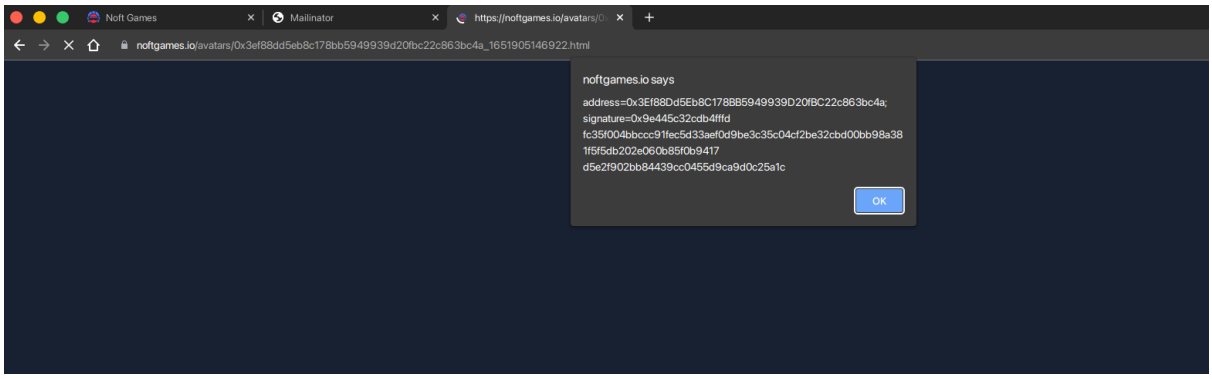
1. https://noftgames.io/avatars/0x3ef88dd5eb8c178bb5949939d20fbc22c863bc4a_1651905146922.html

PoC:

1. Log in to the application and head over to the "Edit Profile" section and upload a new display picture.
2. Use the following payload and save it in an HTML file and upload it.
3. Open the uploaded URL, and the script will execute, showing the user's cookies as shown below.

```
<html>
```

```
<body>
<script>alert(document.cookie);</script>
</body>
</html>
```



Impacts:

This vulnerability can have critical impacts considering this is an application that relies on user signatures to identify them. If the cookies and signatures are stolen, the attackers will be able to masquerade as their victims, stealing their funds or nofts and executing other malicious actions.

Remediation:

Follow and implement the below-mentioned points to remediate and prevent XSS attacks -

1. Always treat all user input as untrusted data.
2. Never insert untrusted data except in allowed locations.
3. Always input or output-encode all data coming into or out of the application.
4. Always whitelist allowed characters and seldom use blacklisting of characters except in certain use cases.
5. Always use a well-known and security encoding API for input and output encoding, such as the OWASP ESAPI.
6. Never try to write input and output encoders unless absolutely necessary. Chances are that someone has already written a good one.
7. Never use the DOM function innerHtml and instead use the functions innerText and textContent to prevent DOM-based XSS.

8. As a best practice, consider using the HTTPOnly flag on cookies that are session tokens or sensitive tokens.
9. As a best practice, consider implementing Content Security Policy to protect against XSS and other injection-type attacks.
10. As a best practice, consider using an auto-escaping templating system.
11. As a best practice, consider using the X-XSS-Protection response header.

Retest:
This is fixed. The application does not allow uploading malicious files with XSS payloads as profile pictures.

Bug ID#3 [Invalid bug]

Missing Validation when Creating Battles

Vulnerability Type
CWE-20: Improper Input Validation

Severity
High

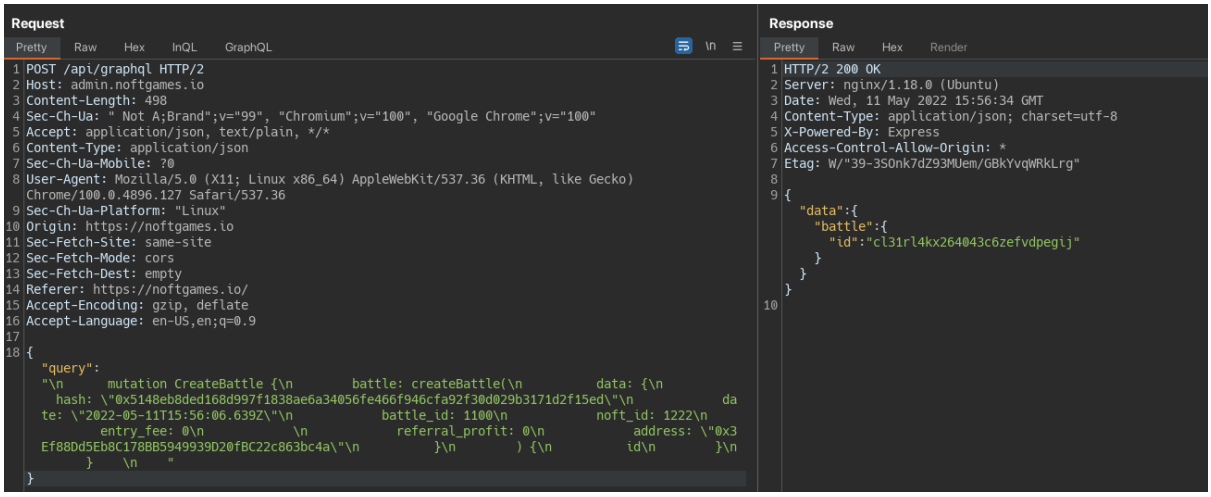
Description:
The application sends a call to an administrative GraphQL interface when creating battles. This endpoint does not have any access control validation on the parameters and the user making the request.
This can be abused to modify the parameters according to the attacker's choice, such as increasing referral profit, modifying the battle, or Noft IDs.

Vulnerable URL:

- https://admin.noftgames.io/api/graphql

PoC:

1. Log in to the application and go to the "Battles" tab and join a battle.
2. Select a Noft and fight. Note a request being sent to the endpoint mentioned above.
3. After signing the transaction, this request will contain all the details related to the battle being created.
4. Modify the values present in the request, and you'll note that the request will still go through successfully without any validation errors.




```
POST /api/graphql HTTP/2
Host: admin.noftgames.io
Content-Length: 498
Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="100", "Google Chrome";v="100"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/100.0.4896.127 Safari/537.36
Sec-Ch-Ua-Platform: "Linux"
Origin: https://noftgames.io
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://noftgames.io/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{"query":"\n      mutation CreateBattle {\n        battle: createBattle(\n
data: {\n            hash:
\"0x5148eb8ded168d997f1838ae6a34056fe466f946cfa92f30d029b3171d2f15ed\"\n
date: \"2022-05-11T15:56:06.639Z\"\n            battle_id: 1100\n
noft_id: 1222\n            entry_fee: 0\n            \n            referral_profit:
0\n            address: \"0x3Ef88Dd5Eb8C178BB5949939D20fBC22c863bc4a\"\n
}\n        ) {\n          id\n        }\n      }     \n    "}
```

Impacts:

This vulnerability can be abused to modify the battle parameters bypassing all the UI based input validations. Multiple input parameters like the referral profit, entry fee, Noft, and battle IDs can be modified to increase the attacker's profit.

Remediation:

Do not allow requests to the backend admin endpoint to be sent from the client-side. This information can just be communicated to an API in the backend.
If this is absolutely required, make sure that the request parameters can not be manipulated by users.

Retest:
After discussion with the team, we were informed that this is an invalid bug, and it is working as expected and used for referral purposes.

Bug ID#4 [Fixed]

Login Bruteforce

Vulnerability Type
CWE-307: Improper Restriction of Excessive Authentication Attempts

Severity
High

Description:
The software does not properly limit the number or frequency of interactions that it has with an actor, such as the number of incoming requests.

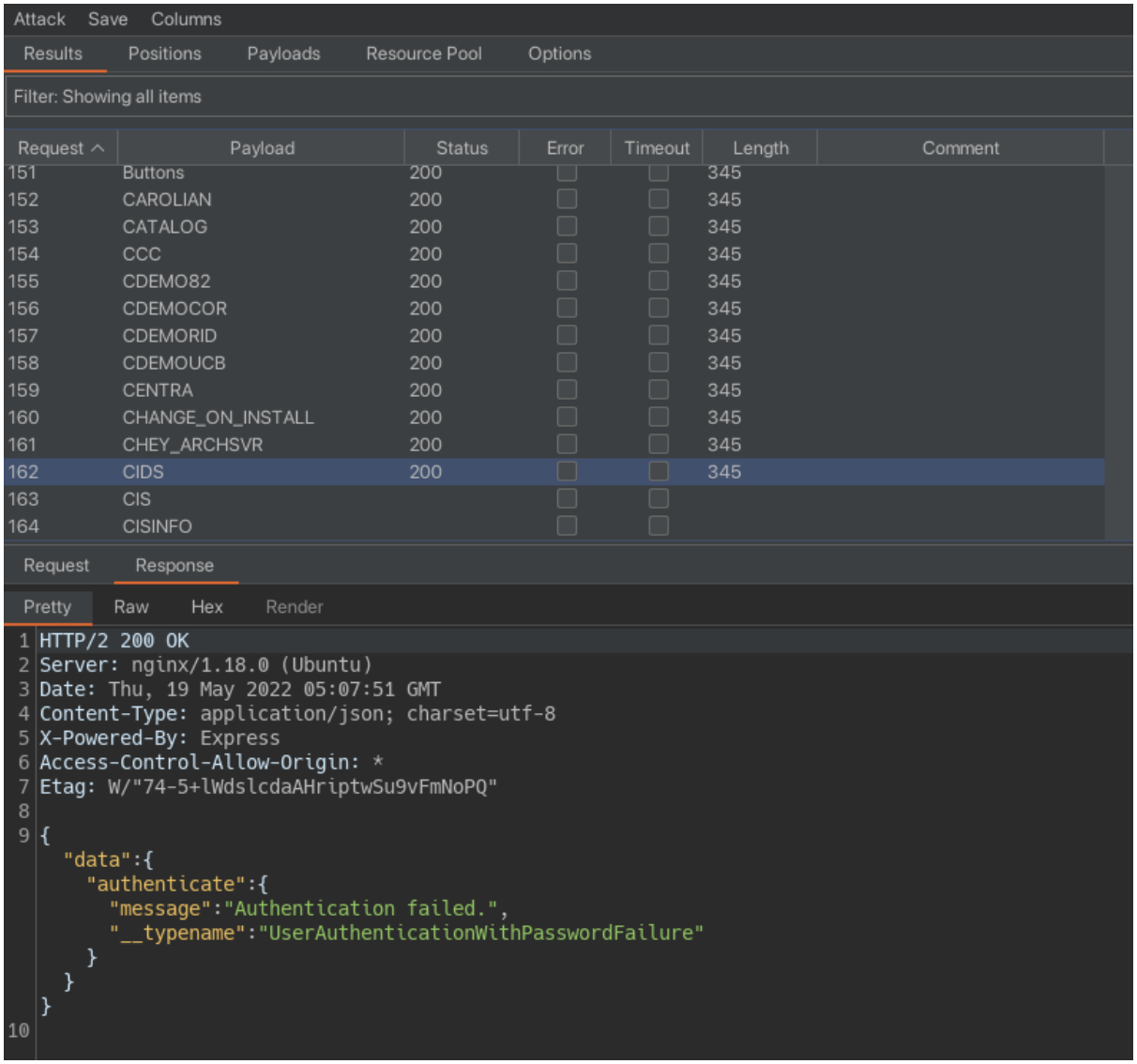
This can allow the actor to perform actions more frequently than expected. The actor could be a human or an automated process such as a virus or bot. This could be used to cause a denial of service, compromise program logic (such as limiting humans to a single vote), or other consequences. For example, an authentication routine might not limit the number of times an attacker can guess a password.
The login forms on the admin portal and the WordPress blog were affected by this issue.

Vulnerable URLs:

- http://blog.beta.noftgames.io/wp-login.php
- https://admin.noftgames.io/

PoC:

1. Navigate to the login page and enter an existing email and a random password.
2. Capture the request with your desired intercepting proxy. I'll be using Burp Suite's Intruder to simulate a brute force attack.
3. It can be seen in the below request that even after 150+ successive authentication attempts with different passwords, the login requests were not blocked.
4. Similar issue is present on the WordPress instance as well.




```
POST /api/graphql HTTP/1.1
Host: admin.noftgames.io
Content-Length: 445
Accept: */*
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/101.0.4951.64 Safari/537.36
Content-Type: application/json
Sec-Gpc: 1
Origin: https://admin.noftgames.io
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
```

```
Referer: https://admin.noftgames.io/signin
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close

{"variables":{"identity":"admin","secret":"admin"},"query":"mutation ($identity:
String!, $secret: String!) {\n  authenticate: authenticateUserWithPassword(name:
$identity, password: $secret) {\n    ... on UserAuthenticationWithPasswordSuccess
{\n      item {\n        id\n        __typename\n      }\n      __typename\n    }\n
... on UserAuthenticationWithPasswordFailure {\n      message\n      __typename\n
}\n    __typename\n  }\n}\n"}
```

Impacts:

This vulnerability can be used to guess the user's passwords using a bruteforce attack, therefore, compromising their account. In this case, the administrator's accounts.

Remediation:

Implement a rate limit or CAPTCHA protection on the endpoint.

Retest:

The website has been removed.

Bug ID#5 [Fixed]

Wp-cron.php Denial of Service

Vulnerability Type
Denial of Service

Severity
Medium

Description:
wp-cron.php is the WordPress task scheduler that takes care of things like checking for updates and publishing scheduled posts. It runs on every single page load.
If your website has not been visited in a while, it will have a lot of missed tasks to catch up with which can greatly increase the loading time and could cause additional resource usage issues.
When this file is accessed a "heavy" MySQL query is performed, so it could be used by attackers to cause a DoS.

Vulnerable URL:

- https://blog.beta.noftgames.io/wp-cron.php

PoC:

1. Open the URL mentioned above in the browser to verify that wp-cron.php is not disabled and can be accessed.
2. Check the response headers to make sure you get a 200 response.

Impacts:
External users and attackers may cause a Denial of Service by calling the endpoint simultaneously from multiple addresses, thereby taking down the website.

Remediation:
The only real alternative and the much better solution is to configure a regular system

cronjob that executes the wp-cron.php script directly through PHP every minute. This ensures that any scheduled tasks are indeed executed at their scheduled time. It also should not be done via an HTTP request but a direct execution of PHP to avoid hindering the web server's capacity or generating additional memory overhead on the network layer.

Reference:
https://medium.com/@thecpanelguy/the-nightmare-that-is-wpcron-php-ae31c1d3ae30

Retest:
The blog has been removed.

Bug ID#6 [Fixed]

XMLRPC Login Bruteforce

Vulnerability Type
CWE-307: Improper Restriction of Excessive Authentication Attempts

Severity
High

Description:
XML-RPC uses XML encoding over HTTP to provide a remote procedure call protocol. It's commonly used to execute various functions in a WordPress instance for APIs and other automated tasks. MySQL query is performed, so it could be used by attackers to cause a DoS.
wp.getUserBlogs, **wp.getCategories**, or **metaWeblog.getUsersBlogs** are some of the methods that can be used to brute-force credentials. **system.multicall** method can be used to amplify this attack.

Vulnerable URL:

- https://blog.beta.noftgames.io/xmlrpc.php

PoC:

1. Send a POST request to the "/xmlrpc.php" as shown below with the credentials to guess in the request body.
2. This can be abused further to result in an amplification attack by using system.multicall method to guess multiple credentials at a time.

```
POST /xmlrpc.php HTTP/2
Host: blog.beta.noftgames.io
Cookie: _color_system_schema=default
Sec-Ch-Ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/92.0.4515.159 Safari/537.36
```

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Length: 164
<methodCall>
<methodName>wp.getUsersBlogs</methodName>
<params>
<param><value>admin</value></param>
<param><value>pass</value></param>
</params>
</methodCall>
```

Impacts:

Attackers can exploit this vulnerability to bruteforce the credentials for the admin account and other users by using a wordlist of usernames and passwords and sending simultaneous requests.

Remediation:

The file xmlrpc.php should be blocked for external access, but it should be noted that this breaks some plugins.

Another way to mitigate this attack is by disabling the ability to call the system.multicall method in your WordPress installation by editing your **functions.php** file. Adding the function **mmx_remove_xmlrpc_methods()** will alleviate the problem, like so:

```
function mmx_remove_xmlrpc_methods( $methods ) {
unset( $methods['system.multicall'] );
return $methods;
}
add_filter( 'xmlrpc_methods', 'mmx_remove_xmlrpc_methods');
```

Reference:

https://blog.cloudflare.com/a-look-at-the-new-wordpress-brute-force-amplification-attack

Retest:

The blog has been removed.

Bug ID#7 [Fixed]

WordPress User Enumeration

Vulnerability Type
CWE-799: Improper Control of Interaction Frequency

Severity
Medium

Description:
User Enumeration is an attack where an attacker thoroughly scans a web application to discover the login names of the web application. These discovered usernames could then be used along with the password brute force vulnerability to guess their passwords.

Vulnerable URL:

- https://blog.beta.noftgames.io/?author=1
- https://blog.beta.noftgames.io/wp-json/wp/v2/users
- https://blog.beta.noftgames.io/wp-login.php

PoC:

1. Visit the URLs shown above to reveal the username for the existing user account on the WordPress blog.
2. It's also possible to guess usernames on the admin login page when a wrong username is entered.

Impacts:
By knowing the existing usernames on the platform it becomes really easy for an attacker to brute-force their passwords and take over admin accounts.

Remediation:
You can reduce the attack surface and make user enumeration harder by following the below

steps:
1. Disable the WordPress REST API if you are not using it,
2. Disable WordPress XML-RPC if you are not using it,
3. Configure your webserver to block requests to /?author=<number>,
4. Don’t expose /wp-admin and /wp-login.php directly to the public Internet.
It is recommended to install a plugin called WP Hardening to prevent user enumeration among other common WordPress vulnerabilities and misconfigurations. - https://wordpress.org/plugins/wp-security-hardening/

Retest:
The blog has been removed.

Bug ID#8 [Fixed]

XMLRPC Pingback Vulnerabilities

Vulnerability Type
CWE-918: Server-Side Request Forgery (SSRF)

Severity
Medium

Description:
XML-RPC uses XML encoding over HTTP to provide a remote procedure call protocol. It's commonly used to execute various functions in a WordPress instance for APIs and other automated tasks.
It exposes a method called **pingback.ping** that allows the server to make external calls

Vulnerable URL:

- https://blog.beta.noftgames.io/xmlrpc.php

PoC:

1. Create a POST request to the xmlrpc.php as shown below.
2. Check the URL in the request body with the URL of your server to view the incoming connection.

```
POST /xmlrpc.php HTTP/2
Host: blog.beta.noftgames.io
Cookie: _color_system_schema=default
Sec-Ch-Ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/92.0.4515.159 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
```

```
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Length: 336
<methodCall>
<methodName>pingback.ping</methodName>
<params><param>
<value><string>http://<your-server></string
></value>
</param><param><value><string>https://<a_valid_blog_post_link></string>
</value></param></params>
</methodCall>
```

Impacts:

An attacker can leverage the default XML-RPC API in order to perform callbacks for the following purposes:

- Distributed denial-of-service (DDoS) attacks - An attacker executes the pingback.ping the method from several affected WordPress installations against a single unprotected target (botnet level).
- Cloudflare Protection Bypass - An attacker executes the pingback.ping the method from a single affected WordPress installation which is protected by CloudFlare to an attacker-controlled public host (for example a VPS) in order to reveal the public IP of the target, therefore bypassing any DNS level protection.
- XSPA (Cross Site Port Attack) - An attacker can execute the pingback.ping the method from a single affected WordPress installation to the same host (or other internal/private host) on different ports. An open port or an internal host can be determined by observing the difference in time of response and/or by looking at the response of the request.

Remediation:

Disable the XMLRPC.php entirely to prevent these vulnerabilities.

It's also possible to just disable this particular pingback method by making the changes to your functions.php file.

```
add_filter( 'xmlrpc_methods', function( $methods ) {
    unset( $methods['pingback.ping'] );
    return $methods;
} );
```

Retest:

The blog has been removed.

Bug ID#9 [Won't Fix]

GraphQL Introspection Enabled

Vulnerability Type
CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

Severity
Low

Description:
With GraphQL the relations between objects, their types, and properties are defined in the schema. The schema itself supports meta properties which makes it highly introspective. It means that you can retrieve the whole structure of the application API with a single request to GraphQL endpoint.

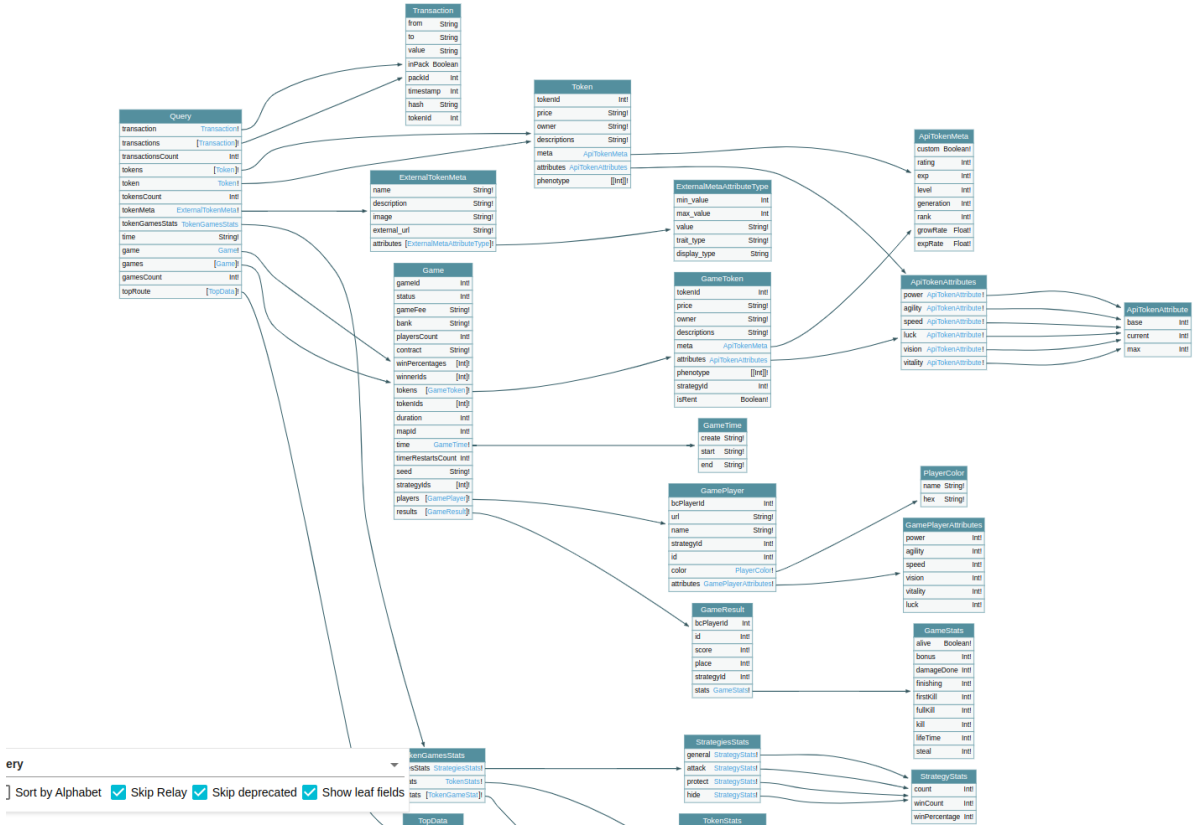
While this is not a vulnerability in itself, it does provide an attacker a complete map of the GraphQL API, and potentially exposes queries that are not necessarily intended for use by the application.

Vulnerable URL:

- https://noftgames.io/api/graphql

PoC:

1. Head over to GraphQL Voyager to copy the introspection query - https://apis.guru/graphql-voyager/
2. Paste the query in the GraphQL API mentioned above.



Impacts:

An introspection system can completely reveal the back-end system defined by developers including arguments, fields, types, descriptions, deprecated status of types, and so on.
This could easily give out the complete map of the back-end system along with the schema and directives.

Remediation:

Disabling the introspection queries is the best practice for information disclosure through graphql.

Retest:

The team decided to leave it as it is because it is not exploitable at this stage.

Bug ID#10 [Fixed]

AMQP Cleartext Authentication enabled

Vulnerability Type
CWE-319: Cleartext Transmission of Sensitive Information

Severity
Low

Description:
The remote Advanced Message Queuing Protocol (AMQP) service supports one or more authentication mechanisms that allow credentials to be sent in the cleartext.

Vulnerable URL:

- 136.243.55.232:5672
- 5.9.116.242:5672

PoC:

1. The default installation for RabbitMQ is clear text. It is up to the site administrator to correctly configure Authentication/Encryption for their environment.

Impacts:
Supporting cleartext authentication may compromise the credentials in case of a Man in middle attack.

Remediation:
Disable cleartext authentication mechanisms in the AMQP configuration.

Retest:
The ports are not reachable externally now

Bug ID#11 [Fixed]

Postgres Credential Bruteforce

Vulnerability Type
CWE-307: Improper Restriction of Excessive Authentication Attempts

Severity
High

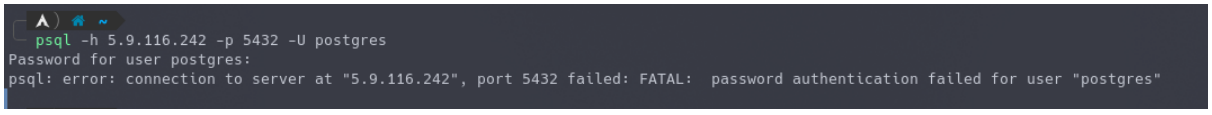
Description:
The organization was exposing two of its Postgres instances to the public and had external connections and authentications enabled allowing malicious actors to brute force the database password.

Vulnerable IPs:

- 136.243.55.232:5432
- 5.9.116.242:5432

PoC:

1. It can be seen in the below screenshot that the Postgres database can be reached externally.
2. This can be automated to brute-force the login credentials.



Impacts:
An attacker may be able to guess the database's password and therefore may be able to gain access to the complete database.

Remediation:
Disable external access to the Postgres instance. The access can also be restricted to certain IP addresses.

Retest:

The ports are not reachable externally now.

6. Appendix 1

6.1 Disclosure:

The Reports neither endorse nor condemn any specific project or team, nor do they guarantee the security of any specific project. The contents of this report do not, and should not be interpreted as having any bearing on, the economics of tokens, token sales, or any other goods, services, or assets.

Emerging technologies such as Blockchain, Smart Contracts, and Solidity carry a high level of technical risk and uncertainty. There is no warranty or representation made by this report to any Third Party in regards to the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business.
In no way should a third party use these reports to make any decisions about buying or selling a token, product, service, or any other asset. It should be noted that this report is not investment advice, is not intended to be relied on as investment advice, and has no endorsement of this project or team. It does not serve as a guarantee as to the project's absolute security.
CredShields Audit team owes no duty to any third party by virtue of publishing these Reports.