// HALBORN

Playground Labs - Kapital DAO Guild Service

Browser Extension Pentest

Prepared by: **Halborn**

Date of Engagement: **September 26th, 2022 - October 22nd, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/12/2022	George Skouroupathis
0.2	Draft Review	10/24/2022	Constantin Casmir
0.3	Draft Review	10/26/2022	Gabi Urrutia
1.0	Remediation Plan	11/30/2022	George Skouroupathis
1.1	Remediation Plan Review	11/30/2022	Carlos Polop
1.2	Remediation Plan Review	11/30/2022	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
George Skouroupathis	Halborn	George.Skouroupathis@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Playground Labs engaged Halborn to conduct a security audit on their browser extension, beginning on September 26th, 2022 and ending on October 22nd, 2022 . The security assessment was scoped to the Kapital DAO Guild Service browser extension. To begin the test, the Client's team provided the source code for Halborn to conduct security testing using tools to scan, detect, validate possible vulnerabilities found in the extension and report the findings at the end of the engagement.

The Kapital DAO Guild Service browser extension is a pass-through for the game the user is playing to connect to the backend API. It allows users to authenticate using their Discord credentials and store their JWT on disk, manage their scholarships, forward blockchain requests to the API.

1.2 AUDIT SUMMARY

The team at Halborn was provided approximately three weeks for the engagement and assigned a full-time security engineer to audit the security of the 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 goals of the security audit are to improve the quality of the systems reviewed and to aim for sufficient remediation to help protect users.

In summary, Halborn identified some security risks that were mostly addressed by the Playground Labs team.

In more detail, care should be taken when verifying the origin of Message Events to the extension Content Script, so as not to allow rogue websites to logout or confirm user signatures/transactions.

Moreover, the API (dev-api.kapital.gg) was found to return custom errors, thereby revealing the technologies used to the end user.

Lastly, it was discovered that dependencies are not pinned to exact versions and that the extension's CSP is not as restrictive as it could be. There was also some discovery that the extension does not check for synthetically generated events, but since there are no attack vectors currently available, it was marked as **INFORMATIONAL**.

To remediate the identified security issues, it is recommended that the Playground Labs team implements the following high-level remediation actions:

- Validate the origin of each message before processing the message.
- Returns a custom error from the application's API to not disclose the technologies the application is based on.
- Pin dependencies to exact versions.
- Make the Content Security Policy as strict as possible.

1.3 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:

- Storing assets securely
- Exposure of any critical information during user interactions with the blockchain and external libraries
- Any attack that impacts funds, such as draining or manipulating of funds
- Application Logic Flaws
- Areas where insufficient validation allows for hostile input
- Application of cryptography to protect secrets

- Input Handling
- Fuzzing of all input parameters
- Web extension misconfiguration
- Technology stack-specific vulnerabilities and Code Audit
- Known vulnerabilities in 3rd party / OSS dependencies.

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.

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

1.4 SCOPE

The security assessment was scoped to the following components:

Kapital Guild Service Extension:

- REDACTED

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	2	2

LIKELIHOOD

IMPACT

			(HAL-01)	
(HAL-02)				
(HAL-04) (HAL-05)	(HAL-03)			

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - UNCHECKED ORIGIN IN POSTMESSAGE	High	SOLVED - 11/30/2022
HAL-02 - UNRESTRICTIVE/UNSECURE EXTENSION CONTENT-SECURITY-POLICY	Low	SOLVED - 11/30/2022
HAL-03 - DEPENDENCIES SHOULD BE PINNED TO THE EXACT VERSION	Low	SOLVED - 11/30/2022
HAL-04 - APPLICATION ERROR MAY DISCLOSE SENSITIVE TECHNOLOGY INFORMATION	Informational	ACKNOWLEDGED
HAL-05 - EXTENSION DOES NOT CHECK FOR SYNTHETICALLY GENERATED EVENTS	Informational	FUTURE RELEASE

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNCHECKED ORIGIN IN POSTMESSAGE - HIGH

Description:

During the audit, Halborn discovered that the extension does not check the source of MessageEvents in its EventListeners. This **(considering that it is in Playground Labs' plans to whitelist the use of the extension for all domains)** presents the danger that any website could send message events to the Content Script of the extension.

In particular, each webpage into which the extension's Content Script can be injected, can send message events to the Background Script with the following payload.message:

- login
- logout
- confirm_signature
- reject_signature
- confirm_transaction
- reject_transaction

and the following payload.method:

- eth_sendTransaction
- eth_signTypedData_v4

This means that all websites where the Content Script is injected can either log out the user from the extension or confirm/reject signatures and transactions that originated from other legitimate websites.

Code Location:

Listing 1

```
1 REDACTED
```

Screenshots/Videos:

Confirming signatures generated on another website

Logging the user out of the Kapital DAO Extension from a website

CVSS Vector:

- AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:N/A:H

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

It is recommended while progressively allowing more websites to be injected with the extension's Content Script to add origin checks in the Event Listeners and avoid whitelisting all domains in the check. Instead, it is recommended to have a list of trusted origins and compare the event origin against them:

Listing 2: Checks for a valid message origin in the Content Script

```
1 var trusted_origins = [
2     "https://game.cryptounicorns.fun/",
3     "https://game.cryptopterodactyls.fun/"
4 ];
5
6 if( trusted_origins.includes( event.origin ) ) {
```

```
    do_stuff(event);
}
```

Listing 3: Checks for a valid message origin in the Background Script

```
chrome.runtime.onMessage.addListener(async (payload, sender,
↳ sendResponse) => {
  var messageOrigin = sender.origin;
  if (messageOrigin == ...) {
    handleRequest(payload, sender, sendResponse)
```

Additionally, it is recommended to verify using the message origin that **login/logout/confirm/reject** messages come from the correct origin, (in this case chrome-extension://ID) to avoid malicious websites from the logging in/out of the user and from the confirmation of their signatures and transactions.

Figure 1: A message from the extension's popup

Reference:

Unchecked Origin in postMessage

Remediation Plan:

SOLVED: The Playground Labs team solved this issue by implementing the following fixes:

- Checking the origin of each Message Event that the extension should emit:
  Commit ID: 27b021d00b6c5274f6bd71573cc52a0c05686720

3.2 (HAL-02) UNRESTRICTIVE/UNSECURE EXTENSION CONTENT-SECURITY-POLICY - LOW

Description:

The Kapital DAO Guild Service browser extension has a declared non-restrictive/insecure Content-Security-Policy (CSP).
The CSP cannot declare a default-src directive. This is a fallback directive that ensures that malicious resources from undeclared, untrusted sources are not loaded.

Moreover, elements controlled by object-src are perhaps coincidentally considered legacy HTML elements and aren't receiving new standardized features (such as the security attributes sandbox or allow for <iframe>). Therefore, it is recommended to restrict this fetch-directive (e.g., explicitly set object-src 'none' if possible).

Lastly, the following directives, which do not inherit from the default -src directive, are not defined:

- base-uri
- form-action
- frame-ancestors
- plugin-types
- report-uri
- sandbox
- reflected-xss
- referrer

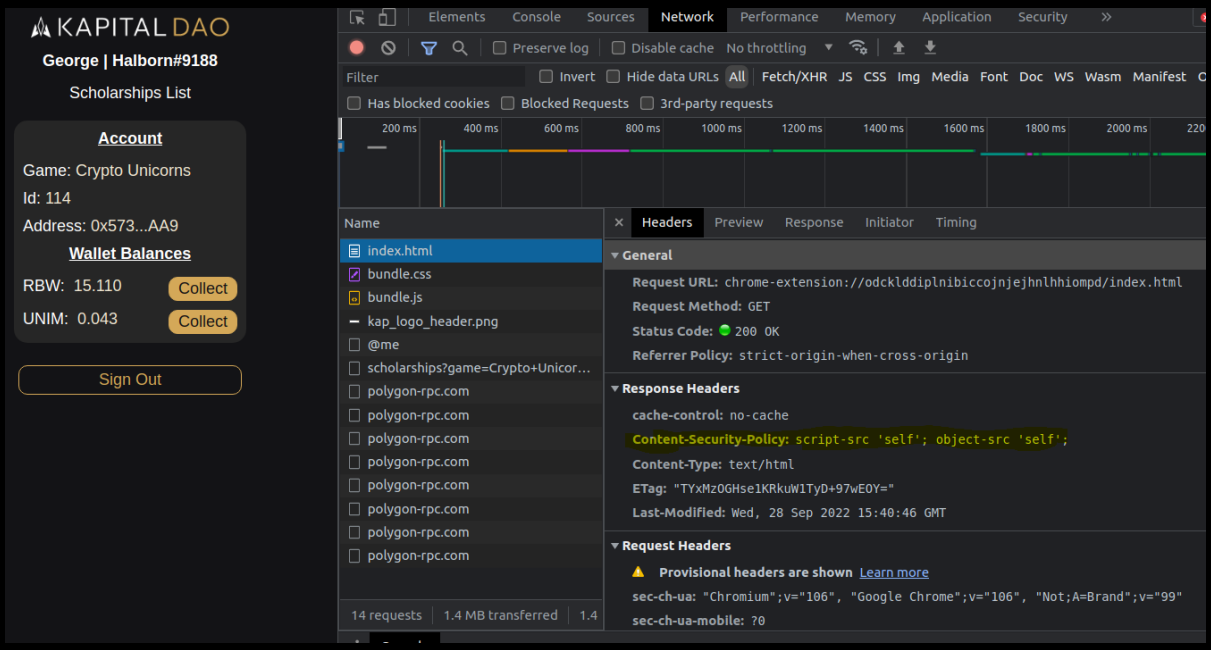

Figure 2: The default CSP the extension is using

CVSS Vector:

- AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:N/A:N

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to add the default-src 'none' directive to the CSP as a fallback. It is also recommended to set the object-src directive to **none** if possible, since the elements it controls are considered to be inherited, and to set missing directives that do not inherit from the default-src directive.

Reference:

Secure Chrome extensions: CSP
object-src directive best practices

Remediation Plan:

SOLVED: The Playground Labs team solved this issue by implementing the following fixes:

- CSP update in manifest.json:
  Commit ID: 82e576964d9c0f19cf45b064b39762cb5d4d4f5f

3.3 (HAL-03) DEPENDENCIES SHOULD BE PINNED TO EXACT VERSIONS - LOW

Description:

The application contains some dependencies that are not pinned to an exact version, but are instead set to compatible version (^x.x.x). This can potentially allow dependency attacks.

Code Location:

Listing 4

```
1 REDACTED
```

CVSS Vector:

- AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Pinning dependencies to an exact version (=x.x.x) can reduce the chance of inadvertently introducing a malicious version of a dependency in the future.

Remediation Plan:

SOLVED: The Playground Labs team solved this issue by implementing the following fixes:

- Pinning all dependencies to exact versions:
  Commit ID: 0cadfbbc5026922f6c2cefbfe602f073cc2fd076

3.4 (HAL-04) APPLICATION ERROR MAY DISCLOSE SENSITIVE TECHNOLOGY INFORMATION - INFORMATIONAL

Description:

During the audit, Halborn discovered that the application at dev-api.kapital.gg generates an error message that includes sensitive information about its environment, technologies, or associated data.

It specifically hints at the fact that the back-end server is possibly using Python and the pydantic library.

Screenshots/Videos:

Figure 3: In the above code, in case the game parameter is not a valid enumeration member, the server will throw a type_error.enum error.

CVSS Vector:

- AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

It is recommended to always return a generic error message to end users, to avoid any possibility of disclosing potentially critical information about the application-related technologies or environment.
The returned error messages can comply with the OpenAPI specification and still hide information about the technology used in the back-end. They can still be able to allow integrations and to facilitate third-party developers' needs.

Reference:

CWE-209: Generation of Error Message Containing Sensitive Information

Remediation Plan:

ACKNOWLEDGED: The Playground Labs team acknowledged this finding.

3.5 (HAL-05) CONTENT SCRIPT DOES NOT CHECK FOR SYNTHETICALLY GENERATED EVENTS - INFORMATIONAL

Description:

During the assessment, Halborn observed that the extension does not check if the events it listens for are trustworthy. Specifically, the `on:click` events that call the `handleConfirm()` and `handleReject` functions in `Confirm.svelte` do not perform this check. The `isTrusted` property states if the event was generated purely from user actions and not synthetically created by an attacker.
The severity of this vulnerability has been downgraded to **INFORMATIONAL** since the attack vector would require someone to already be able to perform an XSS attack on the extension popup window. However, it is good practice to always perform this check on events. This can add security in the future when the extension listens for events generated by the website.

As good coding practice, `MessageEvents` should also check this property, and they are listed below.

Proof of Concept:

The following Svelte snippet exposes two functions, each called by a button. The first button directly calls `handleClick()` and the second calls `triggerClick()`, which synthetically clicks the first button, which in turn calls `handleClick()`:

Listing 5: Trusted/untrusted events handling

```
<script>
    function handleClick(event) {
        console.log('handleClick called');
        console.log(event.isTrusted)
        alert('clicked')
    }
```

```
 7
 8     function triggerClick() {
 9         console.log('triggerClick called first');
10         var btn = document.getElementById("myBtn");
11     btn.click();
12     }
13 </script>
14
15 <button id="myBtn" on:click={handleClick}>
16     Click me Trusted
17 </button>
18 <hr />
19 <button on:click={triggerClick}>
20     Click me Untrusted
21 </button>
```

The image below shows the isTrusted property of each event:

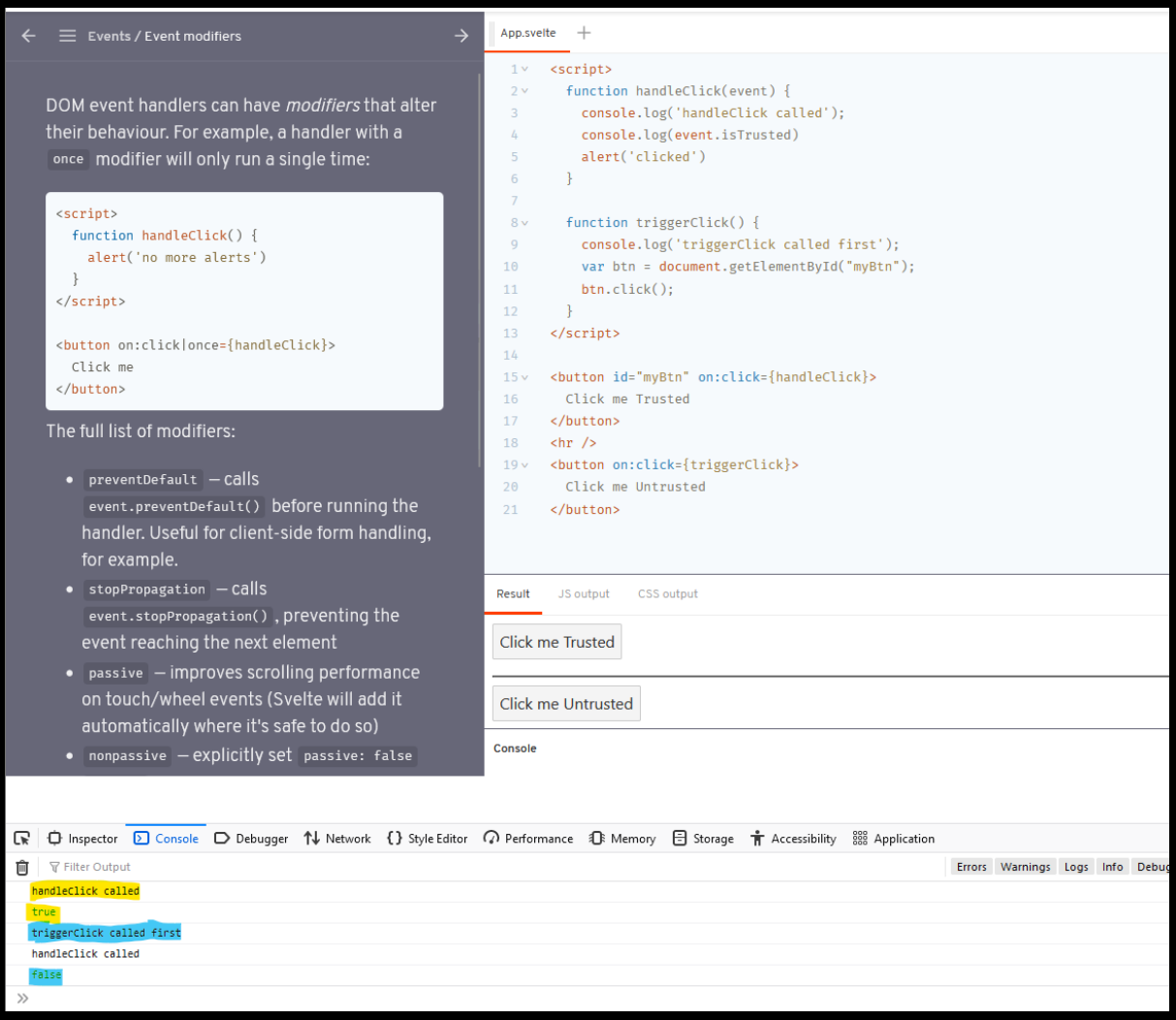

Figure 4: A trusted and an untrusted event

Code Location:

Listing 6

```
1 REDACTED
```

CVSS Vector:

- AV:N/AC:H/PR:H/UI:R/S:U/C:L/I:N/A:N

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

It is recommended to check if each event processed is trusted:

Listing 7

```
1 REDACTED
```

Remediation Plan:

FUTURE RELEASE: The Playground Labs team intend to address this finding in a future release.

THANK YOU FOR CHOOSING

// HALBORN