// HALBORN

Casper - Management Snap App

WebApp Pentest

Prepared by: **Halborn**

Date of Engagement: **June 7th, 2023 - June 16th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/12/2023	Carlos Polop
0.2	Draft Review	06/15/2023	Gabi Urrutia
1.0	Remediation Plan	06/20/2023	Carlos Polop
1.1	Remediation Plan Review	06/20/2023	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
Carlos Polop	Halborn	carlos.polop@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

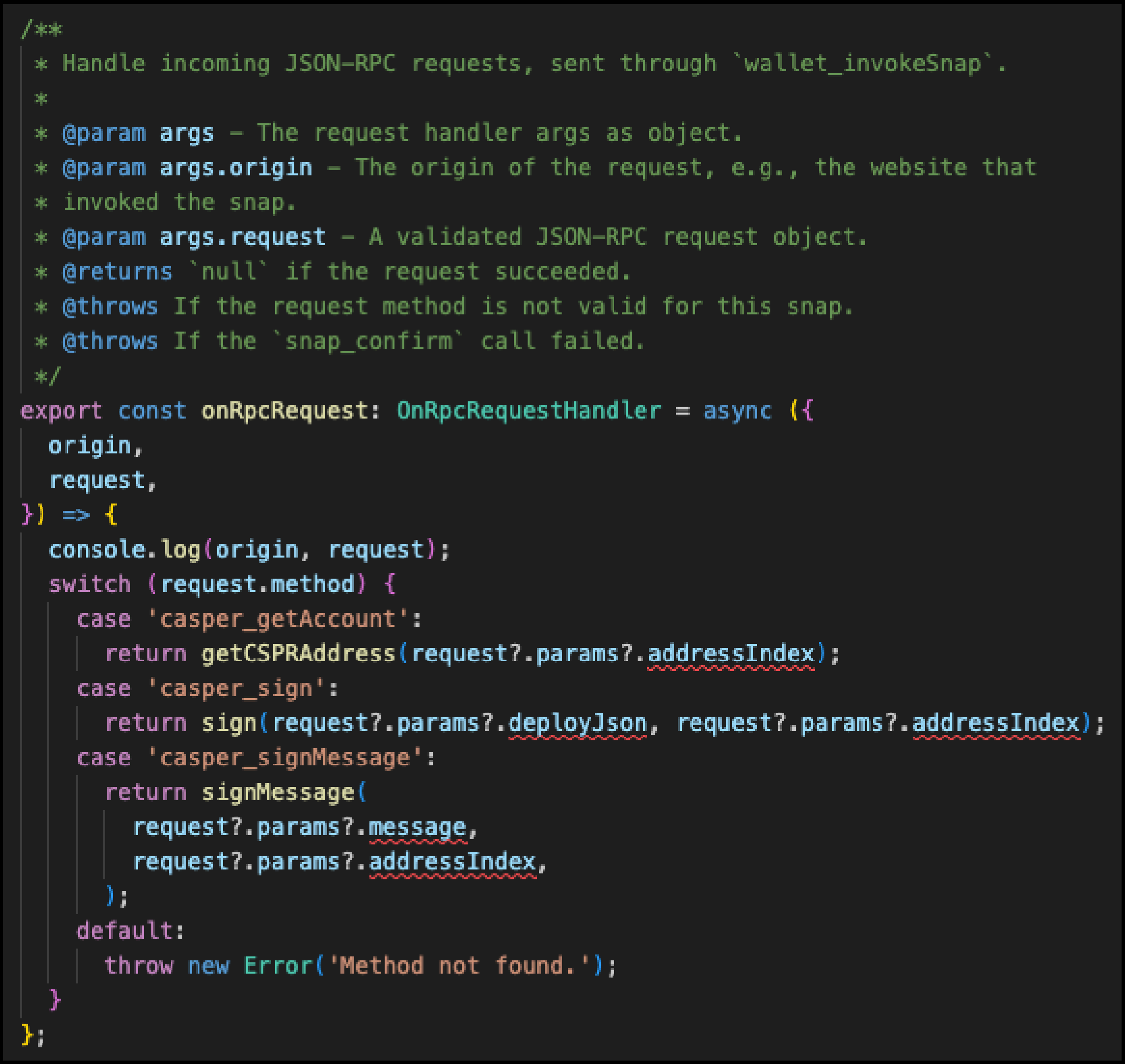
Casper engaged Halborn to conduct a penetration test on their web application, beginning on June 7th, 2023 and ending on June 16th, 2023. The security assessment was scoped to the snap application repository provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided a week and a half for the engagement and assigned a full-time security engineer to audit the security of the snap application. The security engineer is a penetration testing expert with advanced knowledge in web, recon, discovery & infrastructure penetration testing.

The purpose of this audit is to ensure that the Metamask snap application is secure and cannot be unexpectedly abused to get access to users Casper's assets.

After a careful review of the code and a dynamic analysis of the snap application, it was determined that the attack surface was highly limited as only 3 functions were being exposed by the snap application: `casper_getAccount`, `casper_sign`, `casper_signMessage`.



```
/**
 * Handle incoming JSON-RPC requests, sent through `wallet_invokeSnap`.
 *
 * @param args - The request handler args as object.
 * @param args.origin - The origin of the request, e.g., the website that
 * invoked the snap.
 * @param args.request - A validated JSON-RPC request object.
 * @returns `null` if the request succeeded.
 * @throws If the request method is not valid for this snap.
 * @throws If the `snap_confirm` call failed.
 */
export const onRpcRequest: OnRpcRequestHandler = async ({
  origin,
  request,
}) => {
  console.log(origin, request);
  switch (request.method) {
    case 'casper_getAccount':
      return getCSPRAddress(request?.params?.addressIndex);
    case 'casper_sign':
      return sign(request?.params?.deployJson, request?.params?.addressIndex);
    case 'casper_signMessage':
      return signMessage(
        request?.params?.message,
        request?.params?.addressIndex,
      );
    default:
      throw new Error('Method not found.');
  }
};
```

Figure 1: Exported functions

Although those functions allow performing privileged actions, Metamask prompts the user to authorize the installation of the Casper's snap application and to authorize webs to be able to communicate with the snap application. Therefore, only pre-authorized websites by the user will be able to contact the snap application.

Even with Metamask out of scope, the pentester performed some checks to confirm that a web page cannot install or contact the Casper's snap application without Metamask asking for permission to the user.

Some test scenarios consisted on **trying to access the Casper's snap app even if it was disabled** :

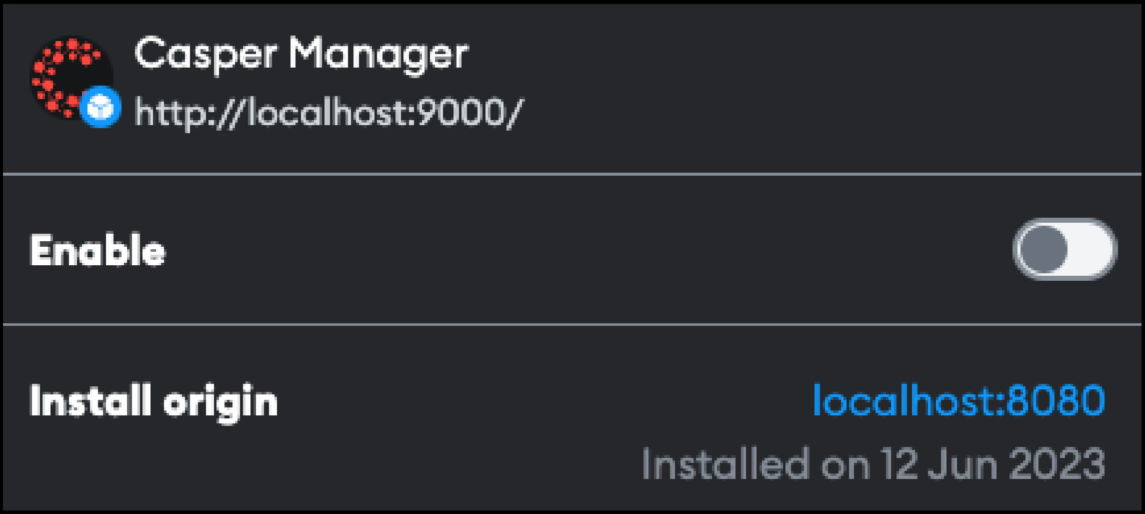

Casper Manager
http://localhost:9000/
Enable
Install origin
localhost:8080
Installed on 12 Jun 2023

Figure 2: Disabled app

Or with an **empty list of allowed sites to contact it**:

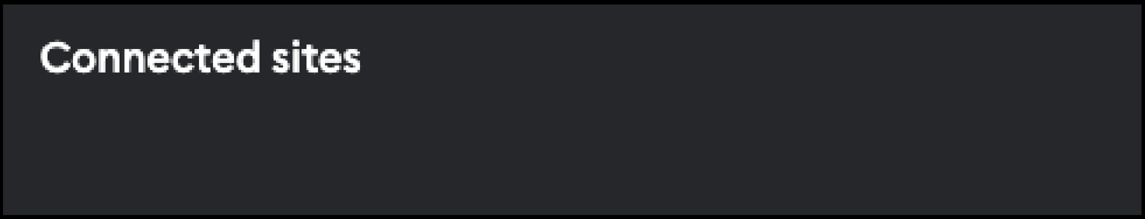

Connected sites

Figure 3: Empty allowed list

In every scenario tested, Metamask correctly throw an error, avoiding the web page to contact the snap application.

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

1.3 SCOPE

The analyzed snap application was https://github.com/casper-ecosystem/casper-manager/ with the commit id c19720a4612eb5224d5d6603917ea3162c2c07d4

1.4 TEST APPROACH & METHODOLOGY

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
0	0	0	1	2

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
DEPENDENCIES SHOULD BE PINNED TO EXACT VERSIONS	Low	SOLVED - 06/20/2023
PROMPT USER WHEN REQUESTING PUBLIC KEYS	Informational	ACKNOWLEDGED
RESTRICT SITES	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) DEPENDENCIES SHOULD BE PINNED TO EXACT VERSIONS - LOW

Description:

The application contains some external dependencies, some of which are not pinned to an exact version but set to a compatible version (ˆx.x.x). This can potentially enable dependency attacks, as observed with the event-stream package with the Copay Bitcoin Wallet.

Details:

Main package.json:

Listing 1

```
"devDependencies": {
    "@metamask/eslint-config": "^10.0.0",
    "@metamask/eslint-config-jest": "^10.0.0",
    "@metamask/eslint-config-nodejs": "^10.0.0",
    "@metamask/eslint-config-typescript": "^10.0.0",
    "@playwright/test": "^1.34.3",
    "@types/isomorphic-fetch": "^0.0.36",
    "@types/jest": "^29.5.1",
    "@typescript-eslint/eslint-plugin": "^5.33.0",
    "@typescript-eslint/parser": "^5.33.0",
    "eslint": "^8.21.0",
    "eslint-config-prettier": "^8.1.0",
    "eslint-plugin-import": "^2.26.0",
    "eslint-plugin-jest": "^26.8.2",
    "eslint-plugin-jsdoc": "^39.2.9",
    "eslint-plugin-node": "^11.1.0",
    "eslint-plugin-prettier": "^4.2.1",
    "jest": "^29.5.0",
    "prettier": "^2.2.1",
    "prettier-plugin-packagejson": "^2.2.18",
    "ts-jest": "^29.1.0",
    "typescript": "^4.7.4"
}
```

Snap package.json:

Listing 2

```
"dependencies": {
  "@ethersproject/bignumber": "^5.7.0",
  "@metamask/key-tree": "^6.2.1",
  "@metamask/snaps-ui": "^0.32.2",
  "casper-js-sdk": "^2.13.0",
  "ethereum-cryptography": "^1.2.0",
  "tweetnacl-ts": "^1.0.3"
},
"devDependencies": {
  "@chainsafe/dappeteer": "5.2.0",
  "@lavamoat/allow-scripts": "^2.0.3",
  "@metamask/auto-changelog": "^2.6.0",
  "@metamask/eslint-config": "^10.0.0",
  "@metamask/eslint-config-jest": "^10.0.0",
  "@metamask/eslint-config-nodejs": "^10.0.0",
  "@metamask/eslint-config-typescript": "^10.0.0",
  "@metamask/snaps-cli": "^0.32.2",
  "@metamask/snaps-types": "^0.32.2",
  "@types/jest": "^29.5.1",
  "@typescript-eslint/eslint-plugin": "^5.33.0",
  "@typescript-eslint/parser": "^5.33.0",
  "eslint": "^8.21.0",
  "eslint-config-prettier": "^8.1.0",
  "eslint-plugin-import": "^2.26.0",
  "eslint-plugin-jest": "^26.8.2",
  "eslint-plugin-jsdoc": "^39.2.9",
  "eslint-plugin-node": "^11.1.0",
  "eslint-plugin-prettier": "^4.2.1",
  "jest": "^29.5.0",
  "playwright": "^1.34.3",
  "prettier": "^2.2.1",
  "prettier-plugin-packagejson": "^2.2.11",
  "release-it": "^15.10.3",
  "rimraf": "^3.0.2",
  "ts-results": "npm:@casperlabs/ts-results@^3.3.4",
  "typescript": "^4.7.4"
}
```

Lib package.json:

Listing 3

```
"dependencies": {
    "@metamask/providers": "^10.2.1",
    "buffer": "^5.7.1",
    "casper-js-sdk": "^2.11.1"
  },
  "devDependencies": {
    "esdoc-typescript-plugin": "^1.0.1",
    "release-it": "^15.10.3",
    "typedoc": "^0.23.28"
  }
```

CVSS Vector:

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

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

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

Remediation Plan:

SOLVED: The vulnerability was fixed by using exact versions (fixed in commit 8d4384ac69c35ff637979e257c7730f94a191f4b).

3.2 (HAL-02) PROMPT USER WHEN REQUESTING PUBLIC KEYS - INFORMATIONAL

Description:

The application is currently prompting the user for permission to call 2 out of the 3 exposed endpoints. Therefore, it is possible to call casper_getAccount without asking the user.

The information given by the action isn't very sensitive (the public key given an index), however, if an attacker managed to get access to a web application with access to this snap application he would be able to **leak all the public keys**.

Recommendation:

If possible, it is recommended to ask permission to the user to call any function being exposed by the snap application. If not, it is recommended to add some kind of limiting into the function to prevent potential attackers from abusing it.

Remediation Plan:

ACKNOWLEDGED: The Casper team acknowledged this vulnerability and might fix it in the future.

3.3 (HAL-03) RESTRICT SITES - INFORMATIONAL

Description:

Metamask snap applications are protected by default by Metamask, which will control which origins can communicate with each installed snap.

However, with some social engineering, an attacker could manage to convince a user to provide access to a malicious origin, so it can manage the user’s Casper’s assets.

This could be made impossible if the snap could only allow verified origins to communicate with it.

Recommendation:

If possible, only allow to communicate with the snap web origins that are verified and specified in a white-list. If Casper does not expect third-parties websites to be communicating with the snap, only the known Casper websites using the snap should be allowed.

Remediation Plan:

ACKNOWLEDGED: The Casper team acknowledged this vulnerability and might fix it in the future.

THANK YOU FOR CHOOSING

// HALBORN