Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Furucombo Web & API 04.-05.2021

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Ing. A. Inführ, BSc. T.-C. “Filedescriptor” Hong

Index

Introduction

> *“Furucombo is a tool built for end-users to optimize their DeFi strategy simply by drag and drop. It visualizes complex DeFi protocols into cubes. Users setup inputs/outputs and the order of the cubes (a “combo”), then Furucombo bundles all the cubes into one transaction and sends out.”*
>
> From https://furucombo.app/

This report describes the results of a security assessment carried out by Cure53 and targeting the Furucombo complex. The project, which took place in spring 2021, encompassed a source code-assisted penetration test, as well as audits of the Dinngo’s Furucombo web application, API and related server-side infrastructure. To discuss the context of this cooperation, the work was requested by Dinngo Pte. Ltd. in March 2021 and then scheduled. The project was then carried out by Cure53 in the second half of April 2021, namely in CW16. Further commenting on the resources, a total of seven days were invested to reach the coverage expected for this project. A team of three

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

senior testers were assigned to this project's preparation, execution and finalization. To optimally structure the work and track progress, the work was split into three separate work packages (WPs):

- **WP1**: Source-Code-assisted Penetration-Tests against Furucombo Web App
- **WP2**: Source-Code-assisted Penetration-Tests against Furucombo Backend API
- **WP3**: Grey-Box Infrastructure-Review and external Server-side Penetration-Test

White-box methodology was used in this engagement to help reach good coverage and depth. Cure53 was given access to the platform rolled-out on production, as well as all relevant sources, API token for testing and everything else that was needed. All preparations were done in early and mid-April, namely in CW15, so that to make a smooth start possible to the testing team. Preparations by the Dinngo team were very thorough and no roadblocks were encountered. The work started on time and moved forward at a good pace. Communications during the test were done using a shared Slack channel which connected two workspaces of Dinngo and Cure53. All partaking team members could join the discussions which were productive and lean as not many questions were needed in light of the well-prepared scope. Nevertheless, Cure53 offered frequent status updates about the test and the findings. Live-reporting was not requested but was not needed in the light of no items of noteworthy severity spotted.

The Cure53 team managed to get very good coverage over the WP1-3 scope items. Only four security-relevant discoveries were made. These are split into two vulnerabilities and two general weaknesses marked by lower exploitation potential. One item received a *Low* score, while all other flaws were only marked as *Informational.* This is a great result for the Dinngo Furucombo platform, showing that the attack surface of the application in scope is rather small and the exposed parts are hardened properly. In the following sections, the report will first shed light on the scope and key test parameters, as well as the structure and content of the WPs. After that, a chapter on test coverage and methods is included to showcase what has been inspected by the Cure53 team. Next, all findings will be discussed in grouped vulnerability and miscellaneous categories, then following a chronological order in each group. Alongside technical descriptions, PoC and mitigation advice are supplied when applicable. Finally, the report will close with broader conclusions about this April 2021 project. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team's observations and collected evidence. Tailored hardening recommendations for the Furucombo complex and Dinngo team are also incorporated into the final section.

Note: *This report was updated with fix notes for each addressed ticket in early May 2021. All of those fixes have been inspected and successfully verified by the Cure53 team in May 2021.*

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Penetration Tests & Code Audits against Dinngo's Furucombo Web App & API**
  - **WP1**: Code-assisted penetration tests against Furucombo Web UI
    - Furucombo Web UI;
      - https://furucombo.app/
  - **WP2**: Code-assisted penetration tests against Furucombo backend & API
    - Furucombo API URL:
      - https://api-next.furucombo.app/
    - Furucombo API Key:
      - `LWOgveSTYN51eCqwsJETC8unt5qcMtiD7OC9y7AV`
    - TestNet
      - RPC URL:
        - https://geth-beta.furucombo.app/
      - Explore URL:
        - https://geth-beta.furucombo.app:8443/
  - **WP3**: Grey-box infrastructure review and external server-side penetration test
    - **.furucombo.app*
  - **Allow-listed Cure53 IPs**
    - 82.102.25.226
    - 37.120.155.34

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Test Methodology

The following section documents the testing methodology applied during this engagement and sheds light on various areas of the Furucombo application subject to inspection and audit. It further clarifies which areas were examined by Cure53 but did not yield any findings.

Client-side security observations

- Cure53 examined the client-side security of the Furucombo web application as the first step of the assessment.
- The initial area to be examined concerned the possibilities to introduce XSS issues. The client-side is mainly written in ReactJS, which automatically HTML-encodes user-controlled variables. Further, no use of *dangerouslySetInnerHTML* was found, however, it was spotted that external links do not sanitize the *javascript:* protocol handler from JSON responses (see FUR-01-002), which can be abused to cause XSS.
- As DOM XSS-related issues are often overlooked in ReactJS-based web applications, the utilized JavaScript resources were studied for this potential threat. In the end, all examined sinks were correctly handled by the Furucombo web application.
- Then, it was checked if proper HTTP security headers were implemented. While most headers (*HSTS, CORS* and *X-Frame-Options*) are correctly deployed, the *Content-Security-Policy* header does not contain adequate directives that can effectively mitigate client-side attacks like XSS (FUR-01-001).
- The handling of user-controlled path or query parameters was assessed. This unveiled a client-side path traversal issue due to the fact that the code does not deploy any validation (FUR-01-003).
- Next, the integration to the Ethereum network was examined. This was to ensure the RPC node provided by the web application could be judged as trustworthy. Cure53 concluded this to be safe given that the node is hosted by Furucombo and has been correctly set up.

Server-side security observations

- Cure53 examined the API host used to aggregate data sources from various on/off chain analytics. It must be noted that the backend does not offer any kind of log-in mechanism to Furucombo users, thereby eliminating the possibility of common ACL-related issues and drastically reducing the overall attack surface.
- Cure53 focused on common and critical issues, namely SSRF and SQL injections. Outbound API calls from the server were examined but no SSRF possibilities were spotted due to HTTP host being fixed.
- One function call was spotted in the ORM and confirmed to use an unsafe pattern that could lead to SQL injection. However, it turned out properly secured as the backend

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

deploys strong input validation. All other SQL queries correctly include user-variables via *parameterized queries* and, therefore, no SQL injections were discovered.

- Another often overlooked threat is related to Denial-of-Service attacks via maliciously crafted HTTP requests. The backend does not support compressed HTTP bodies via the *Client-Encoding* or *Transfer-Encoding* header, which blocks the option of abusing compression bombs to crash the backend. While assessing available features in this context, a potential threat was discovered in one of the endpoints (FUR-01-004), as the amount of data retrieved from the database is not restricted by the backend.
- Finally, the configurations were checked. Sensitive credentials/API keys are properly encrypted and provided as environment variables.

Attack surface exposed via additional infrastructure

- Cure53 examined other real-life attacks against Defi architecture and other systems hosted on the Furucombo domain.
- Subdomains were enumerated through both active and passive methods. Additional subdomains were also extracted from the provided source code. These systems were then checked for exposed services. HTTP-related service was also tested for hidden endpoints or resources, which would impact the security of the Furucombo application or its users. Despite extensive testing, this endeavor did not reveal any additional security issues.
- Similarly, discovered AWS buckets were tested for common misconfigurations, which could allow to list, modify or upload resources hosted by Furucombo.
- The defense against DNS poisoning attack, which has happened multiple times against other Dapps, was examined. It was determined that the NS is delegated to Vercel, whereas existing SSL certificates are properly issued to Furucombo-owned domains. Cure53 concludes this signifies adequate protections against DNS poisoning attacks.
- The same can be said about the deployed CloudFront configuration. No issue in the configured DNS name setup was discovered, eradicating the option to hijack a Furucombo-related domain in this context.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *FUR-01-001*) for the purpose of facilitating any future follow-up correspondence.

FUR-01-002 WP2: Stored XSS via JavaScript link in prebuilt combo *(Low)*

Note: *This issue was verified as properly fixed in May 2021 by the Cure53 team, the problem no longer exists. User-controlled URLs are now being validated properly.*

A potentially stored XSS was spotted in the *combo explore* page. Although ReactJS is usually safe in relation to injection based XSS, additional care has to be applied in handling external links. Here, the affected code simply uses the link in the JSON response in the *href* attribute of a <*Link*>. When a *javascript:* link is supplied as a link, it can introduce XSS.

Affected File:
furucombo-interface-master/src/pages/explore/subpages/details/Introduction.tsx

Affected Code:

```
<ExternalLink href={url} onClick={handleClickExternalLink.bind(null, text)}>
```

Steps to reproduce:

1. Navigate to https://furucombo.app/explore/combo_uniswapv2_00001
2. Intercept the request to https://api-next.furucombo.app/v1/combos/combo_uniswapv2_00002
3. Modify a *url* property in *related* to a *javascript:* URL as demonstrated below
4. Hold CTRL key or use the mouse wheel, to click the relevant link
5. Observe an alert pop up.

```
{"combo":{"id":"combo_compound_00001","title":"Instantly Swap
cTokens","main_defi_name":"compound","short_description":"Swap between cTokens
for higher APY","long_description":"For Compound users who have cTokens on hand
and would like to swap them to another cTokens in order to earn higher APY, such
as swapping your cUSDC to cDAI. Note: this combo is only for cToken holders who
don't have debt on Compound. If you have debt and you want to swap your cTokens,
use the combo \"Compound Collateral Swap\" in the link below.","steps":
[{"defi_name":"compound","text":"Withdraw cToken A"},
{"defi_name":"uniswapv2","text":"Swap Token A to Token B"},
{"defi_name":"compound","text":"Supply Token B and get cToken
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de


```
B"}],"combo_log_id":"bvvmocvv36nc72ksb3gg","apy":0.11129623,"rewards":
["COMBO","COMP"],"tags":["Compound","APY"],"related":[{"text":"🗃 AAVE Get
Higher APY","url":"https://furucombo.app/explore/combo_aave_00001"},{"text":"🗃
Compound Collateral
Swap","url":"https://furucombo.app/explore/combo_compound_00003"},{"text":"🔗
Compound website","url":"https://compound.finance/"},{"text":"🔗 What is Tx
Mining","url":"javascript:alert(1)"}]}}
```


The impact is greatly reduced by the fact that only the team can modify *combo*'s data. It is nevertheless recommended to patch the issue, as further minor issues might exist (e.g. FUR-01-003) and allow non-team members to manipulate the JSON responses. This issue can be addressed by verifying that an external link always starts with *http(s):*.

FUR-01-003 WP1: Client-side path traversal in API request *(Info)*

***Note:** This issue was verified as properly fixed in May 2021 by the Cure53 team, the problem no longer exists. User-controlled path segments are now being validated.*

During the assessment, it was discovered that user-controlled path variables are placed into HTTP API paths without any kind of validation. By using URL encoding, it is possible to modify the HTTP path and target a different API endpoint. The impact of this issue is only rated as *Info*, as it was not possible to abuse this behavior to load arbitrary data into the web application.

Affected File:
furucombo-interface-master/src/apis/furucombo-next-api/client.ts

Affected Code:

```
public v1GetComboLogs = (params: FurucomboNextAPI.V1GetComboLogsParams):
FurucomboNextAPI.V1GetComboLogsResponse =>
this.instance.get('/v1/combo_logs', { params });

public v1GetComboLog = (id: string): FurucomboNextAPI.V1GetComboLogResponse =>
this.instance.get(`/v1/combo_logs/${id}`);
[...]
public v1GetCombo = (id: string): FurucomboNextAPI.V1GetComboResponse =>
this.instance.get(`/v1/combos/${id}`);
```

Example URL:
https://furucombo.app/combo/abcd%5c..%5c..%5c..%5ctest
https://furucombo.app/explore/aaaa%5c..%5c..%5ctest

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Sent HTTP request:
https://api-next.furucombo.app/test

It is recommended to validate the user-controlled *id* parameter via an allow-list which only contains alphanumeric characters. This ensures that a malicious *id* cannot modify the created HTTP path.

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

FUR-01-001 WP1: Insufficient directives of *Content Security Policy* header (*Info*)

Note: *This issue was verified as properly fixed in May 2021 by the Cure53 team, the problem no longer exists. The CSP directives now make use of hashes to verify permitted scripts.*

It was found that the Furucombo website does not adequately leverage the *Content Security Policy* header, except for the *frame-ancestors* directive. The CSP header acts as a defense-in-depth against various client-side attacks.

Implementing it ensures that allow-listed resources are limited to client features which are actually needed by the application. For example, this could prevent XSS issues like FUR-01-002 even when there is an injection.

It is recommended to add the *Content Security Policy*[1] header on the website and implement the *script-src* directive to ensure the possibilities of XSS are eliminated.

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

FUR-01-004 WP2: No default limit for database queries *(Info)*

Note: *This issue was verified as properly fixed in May 2021 by the Cure53 team, the problem no longer exists. Limits for database queries are now properly enforced.*

During the assessment of the backend and its API endpoints, it was discovered that no default limit for database queries is set. It is, therefore, possible to request all items stored in the corresponding database table.

An attacker could abuse this behavior to constantly consume a lot of resources in the backend and potentially cause a Denial-of-Service. The *combo_logs* endpoint would be the most likely target, as it currently contains over 140 000 entries.

Example Request:

```
GET /v1/combo_logs?page=1&page_size=600 HTTP/2
Host: api-next.furucombo.app
[...]
```

Response:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 2497472
[...]
"count":144346
```

It is recommended to consider setting a default limit for any query to ensure the backend does not get blocked by processing and returning a huge number of entries stored in the database. As the backend already supports paging via the *page* and *page_size* parameter, this should be easy to implement without preventing a user from accessing all of the available data.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

Cure53 was tasked with assessing the security of the Furucombo web application as well as systems hosted on subdomains. The overarching conclusion is that the Furucombo web app, API and related server-side infrastructure have withstood scrutiny of the testing team very well. Despite spending seven days on the scope in April 2021, three members of the Cure53 team only spotted four very minor issues.

The following conclusion notes will describe the impression the testers obtained for each item subject to investigations. Generally speaking, the available feature-set as well as the exposed system is rather limited and, therefore, marked by a small attack surface. Importantly, no flaws beyond *Low* have been spotted, indicating that the topic of web security seems very well under control.

It should be noted that the client-side is built upon the ReactJS framework, which ensures that user-controlled variables are properly HTML-encoded. This handling prevents common reflected or stored XSS vulnerabilities. However, a JavaScript link provided in a specific JSON response was found to be directly rendered, which could lead to XSS (FUR-01-002). Additionally, a flaw was discovered in the creation of HTTP requests, which allows to modify the targeted endpoint of the API (FUR-01-003). On a positive note. No DOM XSS vulnerabilities were observed.

Lastly, the deployed HTTP security headers were checked to see if they are correctly implemented. Although most headers are present, the Content Security Policy header was found to only implement the *frame-ancestors* directive and not the *script-src* directive (FUR-01-001). This greatly limits the usefulness of the header. All in all, however, the frontend could benefit from some minor improvements but - in its current design - properly protects its users from XSS related attacks.

The backend only exposes a small set of API endpoints. Only one of these actually parses a HTTP JSON body. Moreover, user-controlled parameters are properly mapped and validated by the backend, which prevents type-confusion-related security issues as well as mass-assignment attacks.

Given the aforementioned validation design paired with the usage of parameterized queries, no SQL injection was spotted in the code. Other potential attack vectors like SSRF were prevented due to hostnames of external API calls being fixed. In the end, only a minor issue was discovered in regard to resource consumption via large HTTP responses (see FUR-01-004).

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Finally, the utilized infrastructure, which included subdomains and their exposed services like HTTP or WebSocket, was examined. It was determined that the Ethereum RPC node provided is trustworthy due to the fact that it is hosted by Furucombo. The serverless configurations were also examined and they are properly encrypted and stored. Similarly the exploration of discovered systems and common misconfiguration - like DNS entries or AWS bucket ACL settings - did not unveil any issues.

All in all, the current security state of the Furucombo application is really good. The choices pertaining to the client-side as well as backend coding languages and frameworks have been sound. Although the application does not offer many features currently, the code reflects that security is part of the development-lifecycle. In case this design is continued for newly developed features, the security of the application should continue on a strong path.

Cure53 would like to thank Hsuan-Ting, Andy Chou and Jay Hsu from the Dinngo Furucombo team for their excellent project coordination, support and assistance, both before and during this assignment.