Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Pentest-Report Firedancer GUI & Backend API 11.2024

Cure53, Dr.-Ing. M. Heiderich, A. Kahla, Y. Yuan, M. Pedhapati

Index

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Introduction

This report describes the results of a penetration test and source code audit against the Firedancer web application, with focus on its frontend aspects and GUI, as well as its backend components.

To give some context regarding the assignment's origination and composition, the Firedancer team contacted Cure53 in November 2024. The test execution was scheduled for later that same month, namely in CW46. A total of nine days were invested to reach the coverage expected for this project, and a team of four senior testers was assigned to its preparation, execution, and finalization.

The methodology conformed to a white-box strategy, whereby assistive materials such as sources, a testing environment, as well as all further means of access required to complete the tests were provided to facilitate the undertakings.

The work was split into two separate work packages (WPs), defined as:

- **WP1**: White-box pen.-tests & source code audits against Firedancer web GUI
- **WP2**: White-box pen.-tests & source code audits against Firedancer backend

All preparations were completed in November 2024, specifically during CW45, to ensure a smooth start for Cure53. Communication throughout the test was conducted through a dedicated and shared Slack channel, established to combine the teams of Firedancer and Cure53. All personnel involved from both parties were invited to participate in this channel. Communications were smooth, with few questions requiring clarification, and the scope was well-prepared and clear. No significant roadblocks were encountered during the test. Cure53 provided frequent status updates and shared their findings through the aforementioned Slack channel. Live reporting was not specifically requested for this audit.

The Cure53 team achieved good coverage over the scope items, and identified a total of three findings. Of the three security-related findings, two were classified as security vulnerabilities, and one was categorized as a general weakness with lower exploitation potential.

The overall very small amount of findings made during this assignment speaks for itself with regard to the security posture of the tested scope. This is especially so, given that this audit marked the first iteration of testing conducted by Cure53, which usually results in a higher number of vulnerabilities being identified.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Furthermore, it can be positively acknowledged that no findings of *High* or *Critical* severity were identified during this audit, indicating that the application was developed with security in mind, and already appears to be well-strengthened against more severe threats.

All-in-all, it should be noted that even though the application left the testing team with a good impression, it is still recommended to address and resolve the small number of weaknesses discovered here, in order to ensure the upkeep of the generally good level of security.

The report will now shed more light on the scope and testing setup, and will provide a comprehensive breakdown of the available materials. Next, the report will detail the *Test Methodology* used in this exercise. This is intended to show the client which areas of the software in scope have been covered, and which tests have been executed, despite only limited findings having been made. Following this, the report will list all findings identified in chronological order, starting with the *Identified Vulnerabilities* and followed by the *Miscellaneous Issues* unearthed. Each finding will be accompanied by a technical description, Proof-of-Concepts (PoCs) where applicable, plus any fix or preventative advice to action.

In summation, the report will finalize with a *Conclusions* chapter in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the Firedancer web application, with focus on its frontend aspects and GUI, as well as its backend components.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Scope

- **Penetration-tests & code audits against Firedancer GUI & Backend API**
  - **WP1:** White-box pen.-tests & source code audits against Firedancer GUI
    - **Environment URL:**
      - ssh -L 8089:localhost:80 user@65.21.19.170
    - **Sources in scope:**
      - URL:
        - https://github.com/firedancer-io/firedancer/tree/main/src/disco/gui
      - Commit:
        - 18171b52cd29f086963dbc957891b210a4169c11
  - **WP2:** White-box pen.-tests & source code audits against Firedancer backend
    - **Sources in scope:**
      - URL:
        - https://github.com/firedancer-io/firedancer/tree/main/src/ballet/http
      - Commit:
        - 18171b52cd29f086963dbc957891b210a4169c11
  - **Test-supporting material was shared with Cure53**
  - **All relevant sources were shared with Cure53**

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Test Methodology

This section documents the testing methodology applied by Cure53 during this project and discusses the resulting coverage, shedding light on how various components were examined. Further clarification concerning areas of investigation subjected to deep-dive assessment is offered, especially in the absence of significant security vulnerabilities having been detected.

- The security assessment of the Firedancer validator's GUI and HTTP server code, specifically targeting the commit *18171b52cd29f086963dbc957891b210a4169c11*, was conducted with the aim of identifying potential vulnerabilities and unexpected behaviors. The primary objectives were to ensure the robustness of the GUI's interaction with the HTTP server, secure WebSocket communication, and proper handling of HTTP requests and responses.
- The evaluation commenced with the Web GUI (*src/disco/gui*), which manages real-time updates via WebSockets. The review emphasized the accuracy of WebSocket message handling and input validation. Data handling security was assessed, so as to prevent the exposure of sensitive information through the GUI. Particular attention was paid to memory management, especially where string operations were concerned. The use of an old version of cJSON was noted, as outlined in ASY-01-002.
- The backend API, responsible for implementing the HTTP server and WebSocket connections, was subsequently audited. The review initially focused on identifying potential resource exhaustion issues, particularly those stemming from manual memory management, handling multiple WebSocket connections, and the lack of explicit message size limits. To address these concerns, the connection eviction mechanism under high load was closely examined, with particular attention being paid to its impact on legitimate connections.
- Specifically, attempts were made to test for resource exhaustion Denial of Service (DoS), caused by long header / body portions in HTTP requests or WebSocket frames. However, buffer lengths were found to be correctly validated, and any long requests / frames were immediately closed before any further data was read into memory.
- Efforts were made to ensure correct socket configuration and proper connection closure procedures, with the aim of preventing resource leaks. The audit also confirmed the enforcement of maximum connection limits, and investigated potential exploits involving multiple *Content-Length* headers, specifically analyzing the system's behavior of processing only the first instance. Lastly, the team verified the proper handling of HTTP requests and responses, along with the accurate formatting and processing of WebSocket frames, ensuring compliance with expected standards.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

- In-depth analysis was performed to ensure that buffer overflow and over-read attacks were not possible on any of the HTTP / WebSocket handling code, with a focus on ensuring that all buffer lengths were correctly validated and consistent between different functions.
- Fuzz testing was used to look for any rare edge cases that might cause the HTTP parser to report incorrect content lengths, as well as possible cases where a small number of requests might be able to cause a DoS condition. No such findings were noted.
- Analysis was performed on the way the WebSocket frame-handling code validated frame headers and frame lengths, to ensure that when ping packets were processed, an attacker could not specify a shorter response length than they actually sent (i.e. similar to CVE-2014-0160).
- Attempts to leverage directory traversal attacks on the static files being served were not successful, due to the fact that the files were built into the binary itself, rather than being served out of a directory on-disk.
- On the web GUI, analysis was performed on both the React code, as well as on request-handling on the server side. The team's focus where the React frontend was concerned was to look for relevant client-side vulnerabilities. Here the team primarily looked for Cross-Site Scripting (XSS) attacks. This led to the identification of the XSS vulnerability described in ASY-02-003.
- On the backend, validation of headers to prevent Cross-Site Request Forgery (CSRF) vulnerabilities was performed, in addition to input validation checks. While not directly a CSRF issue, one vulnerability caused by lack of origin validation was found (ASY-01-001)
- Most other web-related vulnerabilities were deemed to be irrelevant to the test, since there were no access controls, and the web user interface was entirely a read-only website. This excluded most types of Insecure Direct Object Reference (IDOR) and Access Control List (ACL) issues.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, all tickets are given a unique identifier (e.g., ASY-01-001) to facilitate any future follow-up correspondence.

ASY-01-001 WP1: Lack of *Origin* validation on WebSockets *(Low)*

While testing the web interface WebSocket endpoint, it was discovered that there is no validation for the *Origin* header. If a victim is on the same network as a Firedancer instance with the GUI enabled, an attacker that is able to get the victim to click on a malicious link could access the WebSocket endpoint and receive all events.

Steps to reproduce:

1. Visit *http://http.badssl.com/*.

2. Open the Chrome DevTools and run the following:

   Command:
   ```
   new WebSocket('ws://{firedancer ui endpoint}/websocket')
   ```

3. Notice in the “Network” tab that the WebSocket connection is open, and is receiving all messages.

It is recommended to (by default) validate that all WebSocket requests come from origins such as *127.0.0.1* or *localhost*, and to allow the user to add additional allow-listed origins to the configuration file.

ASY-01-003 WP1: XSS on peer information view *(Medium)*

Client note: *After report submission, it was determined that the affected file is in fact unused and not deployed in production.*

When reviewing the Firedancer Web GUI, it was discovered that the peer information view has links to websites without any validation on the URL. A malicious validator could advertise their website as a JavaScript URI (e.g. *javascript:alert(1)*), and cause XSS on the web GUI page if a user clicked on the link.

Since the exploit requires unlikely user interaction (the user would see the *javascript:* URI before clicking on it), the severity of this vulnerability has been reduced.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Affected file:
firedancer-frontend/src/features/LeaderSchedule/ValidatorSummary.tsx

Affected code:

```
<Flex direction="column" gap="1">
  <Text size="7" style={{ color: "#B2BCC9" }}>
    {peer.info?.name}
  </Text>
  <Text style={{ color: "#67696A" }}>{peer.identity_pubkey}</Text>
  {peer.info?.website && (
    <Link href={peer.info.website}>
      <Text>{peer.info.website}</Text>
    </Link>
  )}
</Flex>
```

It is recommended to validate the link on the client side, to only allow HTTP(S) links, as well as to employ a Content Security Policy (CSP) as a first line of defense against XSS attacks.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit, but which may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, while a vulnerability is present, an exploit may not always be possible.

ASY-01-002 WP1: Outdated vulnerable cJSON version *(Low)*

While evaluating the application's source code, Cure53 identified that the application includes the third-party library cJSON, for JSON parsing purposes. However, this particular iteration was published back in December 2023, and is affected by a double free vulnerability. While the application is not currently vulnerable to this issue, its exploitation by attackers could pose a significant risk. Such an attack could lead to serious security flaws, including crashes, or, in the worst-case scenario, arbitrary code execution.

The issue's URL:
https://github.com/DaveGamble/cJSON/issues/833

To mitigate this issue, Cure53 recommends updating cJSON to the latest version, in order to introduce the latest available safeguards and to prevent potential exploitation attempts.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Conclusions

As noted in the *Introduction*, this November 2024 penetration test and source code audit was conducted by Cure53 against the Firedancer web application, with focus on its frontend aspects and GUI, as well as its backend components, emphasizing WebSocket message handling, HTTP request management, and resource control. The audit revealed an overall robust architecture where the Firedancer validator's GUI and backend API were concerned.

From a contextual perspective, nine working days were allocated to reach the coverage expected for this project. The methodology used conformed to a white-box strategy, and a team of four senior testers was assigned to the project's preparation, execution, and finalization.

In general, the source code was of high quality and always correctly used the C standard library functions for bounded buffers. This prevented all types of buffer overflow and buffer over-read attacks that were tested against the application.

Due to the lack of dynamic allocation (i.e. almost no use of *malloc*), it was determined that an entire class of vulnerabilities (heap vulnerabilities) were not possible in the tested portion of the Firedancer codebase. This significantly reduced the attack surface of the HTTP and WebSocket handling portions of the backend. Due both to the static allocation techniques used, as well as correct buffer boundary validation, all types of resource exhaustion attacks failed against the server.

On the web interface, one finding resulted from a lack of URL validation on the client and server (ASY-01-003). Malicious validators could advertise themselves as having a *javascript:* URI as a website, in order to cause an XSS vulnerability if a user clicked on the link on the validator information panel. This was deemed to not be of high severity, due to the fact that exploitation was difficult, given that it required very unlikely user interaction. No other client-side vulnerabilities were discovered on the website portion of the Firedancer GUI. The React code was idiomatic, and never used insecure primitives like *dangerouslySetInnerHtml*.

The *src/ballet/http* module, which implements the HTTP server and WebSocket connections, was thoroughly tested for potential vulnerabilities. The audit focused on resource exhaustion risks caused by manual memory management, handling of multiple WebSocket connections, and the absence of message size checks. The connection eviction mechanism was examined under high load, to ensure that legitimate connections were not dropped inadvertently.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Socket configuration and connection closure procedures were reviewed, in order to verify proper implementation and to prevent resource leaks. Maximum connection limits were confirmed to be correctly enforced, and handling of multiple *Content-Length* headers was analyzed to assess exploitability. HTTP request validation, response handling, and WebSocket frame formatting were found to be implemented securely and in compliance with the expected standards.

The *src/disco/gui* module, which was found to be responsible for managing GUI WebSocket messages, was also tested for security issues. Input validation for WebSocket messages was confirmed to be robust, and checks for integer overflow were verified to function correctly. Functions such as *fd_memcpy* and *strncpy* were analyzed in order to ensure safe memory handling, with no overlaps or buffer overflow risks identified by the team. One relatively minor issue here was the use of an outdated version of the cJSON library (ASY-01-002).

Particular attention was given to the *fd_gui_handle_validator_info_update* function, where assumptions about the size of *name* and *icon_url* fields were found to be verified in order to prevent potential vulnerabilities. Broadcast functionality - including *fd_http_server_ws_broadcast* - was tested to confirm secure and efficient message transmission. These findings collectively highlighted the application's adherence to secure coding practices, ensuring resilience against common threats.

The team made a further finding on the backend, in that if a victim user was on the same network as a Firedancer instance with the GUI enabled, then an attacker who was able to get the victim to click on a malicious link could access the WebSocket endpoint and receive all events (ASY-01-001). This was due to a lack of *Origin* header validation on the backend, and was deemed to be *Low* impact, since it would require the validator GUI to either be running on the same device as the user's web browser, or otherwise for the validator GUI be exposed (e.g. over a VPN) to the victim.

Overall, upon completing the security audit of the Firedancer GUI and Backend API, the Cure53 team had a positive impression of the items within the scope. This was supported by the minimal number of issues identified. The reported issues are recommended for resolution, in order to further strengthen the security posture.

Cure53 would like to thank Cynthia Burke, Liam Wachter, and Nick Lang from the Firedancer team for their excellent project coordination, support and assistance, both before and during this assignment.