Radically Open Security

Penetration Test Report

OnionShare

V 1.0
Amsterdam, June 30th, 2023
Confidential

Document Properties

Client	OnionShare
Title	Penetration Test Report
Target	Onionshare iOS application
Version	1.0
Pentester	Abhinav Mishra
Authors	Abhinav Mishra, Stefan Vink
Reviewed by	Stefan Vink
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	June 28th, 2023	Abhinav Mishra	Initial draft
0.2	June 29th, 2023	Stefan Vink	Reviewed
1.0	June 30th, 2023	Abhinav Mishra	Final version

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between May 17, 2023 and June 9, 2023, Radically Open Security B.V. carried out a penetration test for OnionShare on their Onionshare iOS application

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the penetration test was limited to the following target(s):

- Onionshare iOS application

The Tor implementation was out of scope for this audit. Only the iOS app and it's code was audited.

The scoped services are broken down as follows:

- Pentest iOS application (including reporting): 4 days
- (Optional) retest: 0-2 days
- PM/Review: 1 days
- **Total effort: 5 - 7 days**

1.3 Project objectives

ROS will perform a penetration test of the `Onionshare iOS app` with OnionShare in order to assess the security of the application. To do so ROS will access code at github (https://github.com/onionshare/onionshare-ios) and guide OnionShare in attempting to find vulnerabilities, exploiting any such found to try and gain further access and elevated privileges.

1.4 Timeline

The Security Audit took place between May 17, 2023 and June 9, 2023. The ROS pentest team spent around 24 hours on the auditing of the iOS app and it's code.

1.5 Results In A Nutshell

During this crystal-box penetration test we found 2 Moderate-severity issues.

The security issues discovered during the assessment, are related to lack of input validation. One of the issue refers to the input validation of the `title` field, when a file is being shared. It was discovered that it is possible to provide HTML or Javascript code as a part of the title. The other input validation issue is in the `Resources/static/js/receive.js` code, where the application is trusting the input and using it in constructing the HTML page, without sanitizing it.

Addressing these vulnerabilities and implementing our recommended security measures will significantly strengthen the overall security posture of the app. By pro-actively resolving these issues, you can enhance the app's resilience against potential attacks, safeguard sensitive data, and instill confidence in your users.

The report provides comprehensive insights into the identified vulnerabilities, accompanied by actionable recommendations to guide the remediation process. It is essential to prioritize these recommendations, allocate resources for necessary updates and improvements, and regularly reassess the app's security to ensure ongoing protection against emerging threats.

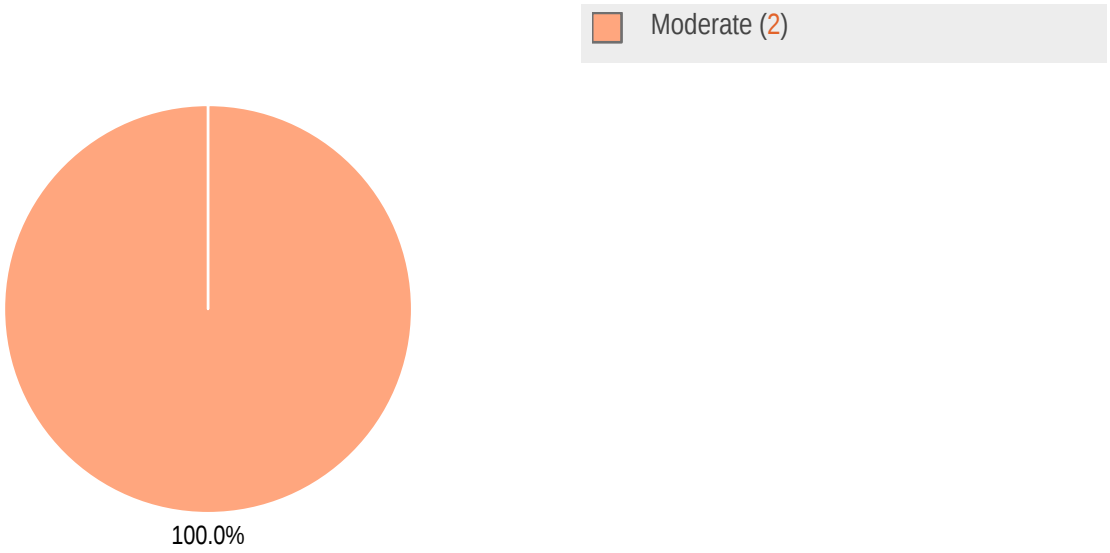
It is important to note that while the app utilizes the Tor network for certain functionalities, our assessment specifically examined the iOS app, configurations, and security measures. Our evaluation aimed to identify vulnerabilities and weaknesses within the app's code, functionality, and associated components and did not encompass the internal implementation of Tor.

As part of the assessment, we thoroughly examined the behaviour of the app, scrutinizing the app's handling of data and the security of the local web server used for file sharing. Our goal was to identify any vulnerabilities or security gaps that may compromise the app's security and the privacy of its users.

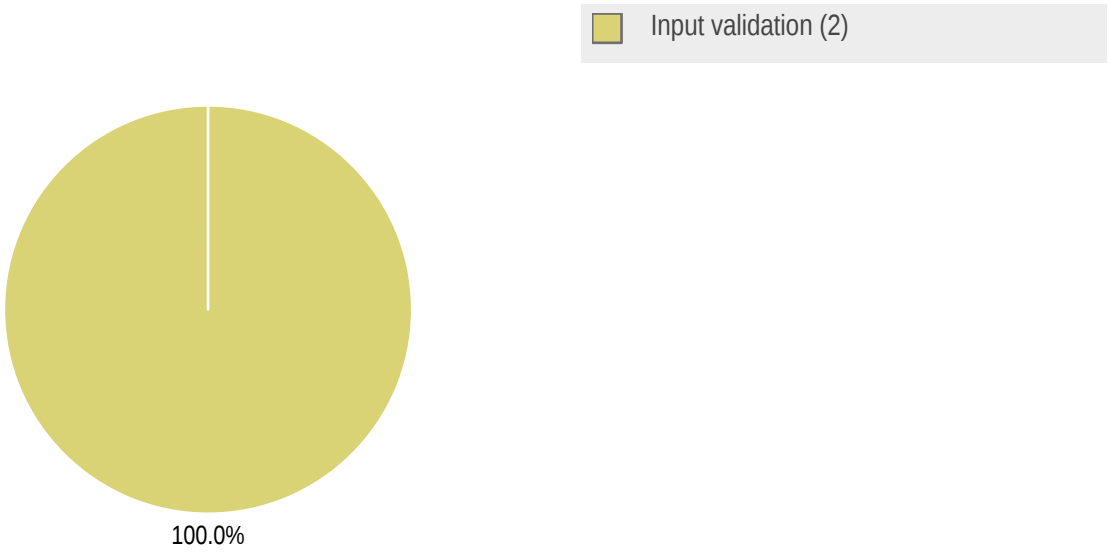
1.6 Summary of Findings

ID	Type	Description	Threat level
ONS-003	Input validation	The application allows users to provide a custom title value while sharing a file. However this field does not implement any check or input validation.	Moderate
ONS-001	Input validation	Unsanitized input is being used to dynamically construct the HTML page on client side. This might lead to cross site scripting.	Moderate

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

ID	Type	Recommendation
ONS-003	Input validation	• Implement input validation on the `custom title` field to restrict the type and size of input.
ONS-001	Input validation	• Implement input validation and ensure it is sanitized and validated before parsing it as JSON. • Use proper error handling to gracefully handle any parsing errors and prevent potential information leakage. • Apply appropriate sanitization or encoding to any user-generated content before inserting it into the HTML document to prevent XSS attacks.

2 Methodology

2.1 Planning

Our general approach during penetration tests is as follows:

1. **Reconnaissance**
   We attempt to gather as much information as possible about the target. Reconnaissance can take two forms: active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection afforded to the app or network. This usually involves trying to discover publicly available information by visiting websites, newsgroups, etc. An active form would be more intrusive, could possibly show up in audit logs and might take the form of a social engineering type of attack.
2. **Enumeration**
   We use various fingerprinting tools to determine what hosts are visible on the target network and, more importantly, try to ascertain what services and operating systems they are running. Visible services are researched further to tailor subsequent tests to match.
3. **Scanning**
   Vulnerability scanners are used to scan all discovered hosts for known vulnerabilities or weaknesses. The results are analyzed to determine if there are any vulnerabilities that could be exploited to gain access or enhance privileges to target hosts.
4. **Obtaining Access**
   We use the results of the scans to assist in attempting to obtain access to target systems and services, or to escalate privileges where access has been obtained (either legitimately though provided credentials, or via vulnerabilities). This may be done surreptitiously (for example to try to evade intrusion detection systems or rate limits) or by more aggressive brute-force methods. This step also consist of manually testing the application against the latest (2017) list of OWASP Top 10 risks. The discovered vulnerabilities from scanning and manual testing are moreover used to further elevate access on the application.

2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.

- **High**

  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**

  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**

  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.
- **Low**

  Low risk of security controls being compromised with measurable negative impacts as a result.

3 Findings

We have identified the following issues:

3.1 ONS-003 — Input validation missing on custom title field

Vulnerability ID: ONS-003

Vulnerability type: Input validation

Threat level: Moderate

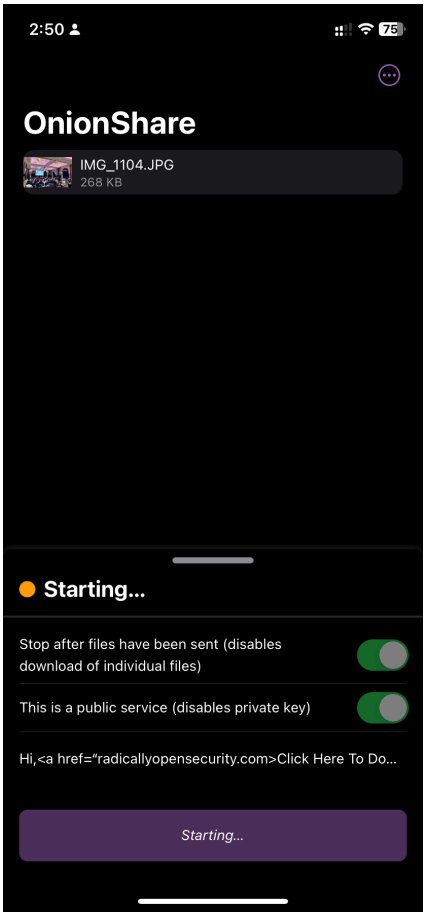
Description:

The application allows users to provide a custom title value while sharing a file. However this field does not implement any check or input validation.

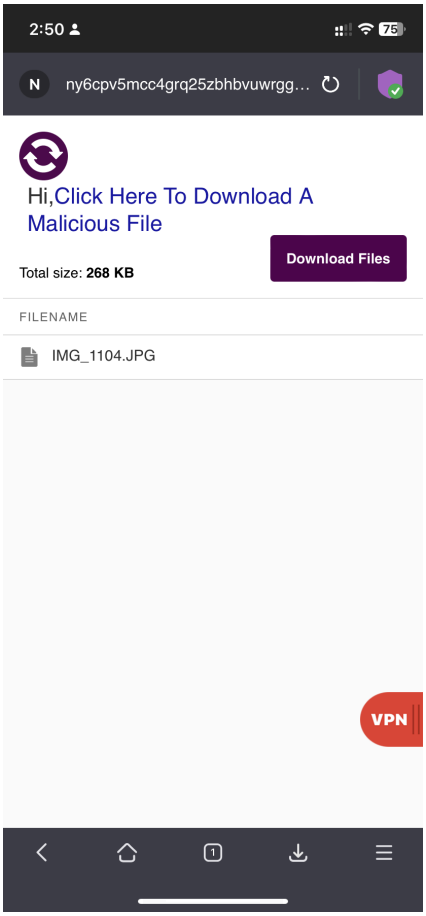
Technical description:

As there is no validation, a user is allowed to share files on a page with HTML code injected inside the `custom title` field or input a really long string inside the `custom title` field.

Injecting HTML in custom title value



HTML code injected on download webpage



Impact:

An attacker would be able to inject custom HTML code (even Javascript) inside the web page. This code will run on the user's browser, who could open the malicious download link.

`Note:` It's important to note that implementing input validation or a filter will not completely mitigate this issue since the webpage is generated client-side and the user has full control. Even with input validation or filtering in place, the attacker can still manipulate the HTML page during its generation.

Recommendation:

- Implement input validation on the `custom title` field to restrict the type and size of input.

3.2 ONS-001 — Unsanitized input is being used to dynamically construct the HTML page on client side which could result in XSS

Vulnerability ID: ONS-001

Vulnerability type: Input validation

Threat level: Moderate

Description:

Unsanitized input is being used to dynamically construct the HTML page on client side. This might lead to cross site scripting.

Technical description:

Vulnerable code

`Resources/static/js/receive.js`

```
// Parse response
      try {
        var response = JSON.parse(ajax.response);

        // The 'new_body' response replaces the whole HTML document and ends
        if ('new_body' in response) {
          $('body').html(response['new_body']);
          return;
        }
```

Impact:

Using unsanitized data to construct HTML might allow an attacker to inject malicious code in the HTML, leading to cross site scripting attack on client side.

Recommendation:

- Implement input validation and ensure it is sanitized and validated before parsing it as JSON.
- Use proper error handling to gracefully handle any parsing errors and prevent potential information leakage.
- Apply appropriate sanitization or encoding to any user-generated content before inserting it into the HTML document to prevent XSS attacks.

4 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

4.1 NF-004 — Non finding test cases

During the audit of the app, we diligently executed an extensive array of test cases to evaluate its security posture. While certain tests successfully identified vulnerabilities, it is noteworthy that a substantial portion of the test cases did not expose any vulnerabilities. Below, we highlight a selection of these non-vulnerable test cases to provide insight into the thoroughness of our assessment.

- Verify if the webserver can be accessed from any other device with in the network
- Verify the web page created to share the file, does not introduce any security issue like injection, directory traversal etc.
- Verify that an attacker can not access any other file, other than the one being shared, through the web page
- Test for insecure storage on device
- Test for unintentional information leakage
- Verify if the application handles malicious input properly, and does not crash
- Analysing the app runtime behaviour while sharing the file
- Reviewed the source code for the iOS application
- Reviewed the process memory for any unintentional info leakage
- Analysed the iOS logs during app is sharing a file

4.2 NF-002 — Internal web server bound to localhost

The app sets up a custom web server by extending a base class, configures a template renderer, and overrides a method to render templates. To confirm whether the web server is bound to the localhost, we looked at the implementation of the `AdParticipesCumCepis.WebServer` class.

In the `AdParticipesCumCepis.WebServer` class, there is an option `GCDWebServerOption_BindToLocalhost` set to `true` in the try webServer.start(options: [...]) block. This option indicates that the web server will be bound to the localhost.

5 Future Work

- **Review Of Tor Components**

  We highly recommend conducting a comprehensive code review of the Tor component as part of your future work. This additional assessment will provide valuable insights into the security posture of the Tor implementation within the app. By thoroughly examining the code, potential vulnerabilities and weaknesses can be identified, helping to mitigate risks and enhance the overall security of your application. In addition, it is important to note that certain security issues may remain hidden within the Tor component, undetectable through standard penetration testing techniques alone. These hidden vulnerabilities could potentially pose a significant risk to the security of your system. By conducting a thorough code review of the Tor component, these concealed security issues can be exposed and addressed promptly, minimizing the chances of exploitation by malicious actors.
- **Retest of findings**

  When mitigations for the vulnerabilities described in this report have been deployed, a repeat test should be performed to ensure that they are effective and have not introduced other security problems.
- **Regular security assessments**

  Security is an ongoing process and not a product, so we advise undertaking regular security assessments and penetration tests, ideally prior to every major release or every quarter.

6 Conclusion

We discovered 2 Moderate-severity issues during this penetration test.

In conclusion, this pentest focused on the limited functionalitities of the app and a correspondingly limited attack surface. This is an encouraging finding as it indicates that the scope for potential security vulnerabilities is relatively narrow. Throughout the assessment, the team identified two security issues. While these vulnerabilities require immediate attention and remediation, the overall low number is a testament to the robustness of the application's security measures. However, it is crucial to note that even a small number of security issues can have severe consequences if left unaddressed. Therefore, we recommend swift remediation of the identified vulnerabilities and the implementation of robust security measures to further fortify the application against potential future threats.

We recommend fixing all of the issues found and then performing a retest in order to ensure that mitigations are effective and that no new vulnerabilities have been introduced.

Finally, we want to emphasize that security is a process – this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Appendix 1 Testing team

Abhinav Mishra	With over a decade of experience, Abhinav is a seasoned professional specializing in penetration testing for web, mobile, and infrastructure systems. Additionally, Abhinav is highly regarded for his expertise in delivering comprehensive training programs focused on enhancing security measures for web, mobile, and infrastructure technologies.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.