



Penetration Test Report

OnionShare

V 1.0
Amsterdam, June 30th, 2023
Confidential

Document Properties

Client	OnionShare
Title	Penetration Test Report
Target	Onionshare Android application
Version	1.0
Pentesters	Abhinav Mishra, Aidan Elphick-Miner
Authors	Abhinav Mishra, Aidan Elphick-Miner, Stefan Vink
Reviewed by	Stefan Vink
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	June 27th, 2023	Abhinav Mishra, Aidan Elphick-Miner	Initial draft
0.2	June 29th, 2023	Stefan Vink	Reviewed
1.0	June 30th, 2023	Abhinav Mishra	Final

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

Radically Open Security B.V. is registered at the trade register of the Dutch chamber of commerce under number 60628081.

Table of Contents

1	Executive Summary	4
1.1	Introduction	4
1.2	Scope of work	4
1.3	Project objectives	4
1.4	Timeline	4
1.5	Results In A Nutshell	5
1.6	Summary of Findings	5
1.6.1	Findings by Threat Level	6
1.6.2	Findings by Type	7
1.7	Summary of Recommendations	7
2	Methodology	9
2.1	Planning	9
2.2	Risk Classification	9
3	Findings	11
3.1	ONS-002 — Accessing shared file without Tor	11
3.2	ONS-004 — Vulnerable software version being used	14
3.3	ONS-001 — Internal webserver listens on 0.0.0.0	15
3.4	ONS-003 — Webserver details being logged	16
4	Non-Findings	20
4.1	NF-005 — Non finding test cases	20
5	Future Work	21
6	Conclusion	22
Appendix 1	Testing team	23

1 Executive Summary

1.1 Introduction

Between June 12, 2023 and June 23, 2023, Radically Open Security B.V. carried out a penetration test for OnionShare of their Onionshare Android app.

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 Android application

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

The scoped services are broken down as follows:

- Code audit & pentest Android application (including reporting): 3-4 days
- (Optional) retest: 0-2 days
- PM/Review: 1 days
- **Total effort: 4 - 7 days**

1.3 Project objectives

ROS will perform a penetration test of `Onionshare Android app, tag 0.1.11` 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-android/releases/tag/0.1.11>) and guide OnionShare in attempting to find vulnerabilities, exploiting any such found to try and gain further access and elevate privileges.

1.4 Timeline

The Security Audit took place between June 12, 2023 and June 23, 2023. The ROS pentest team spent around 24 hours on the auditing of android app, and around 8 hours on the reporting.

1.5 Results In A Nutshell

During this crystal-box penetration test we found 1 Elevated, 2 Moderate and 1 Low-severity issues.

The elevated severity issue discovered during the assessment, allows an attacker to access the shared files directly through the webserver address `127.0.0.1:17638`, instead of going through the Tor URL. However, this is only possible under certain conditions, when the device already has an app which can be used as a proxy.

The two moderate severity issues are about vulnerable software components being used and the webserver listening on `0.0.0.0`. However, the finding about internal webserver listening on `0.0.0.0` has already been resolved.

The low severity vulnerability is about insecure logging practices within the app.

Addressing these vulnerabilities and implementing our recommended security measures will significantly strengthen the overall security posture of the Android app. By proactively 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 Android 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 the Tor.

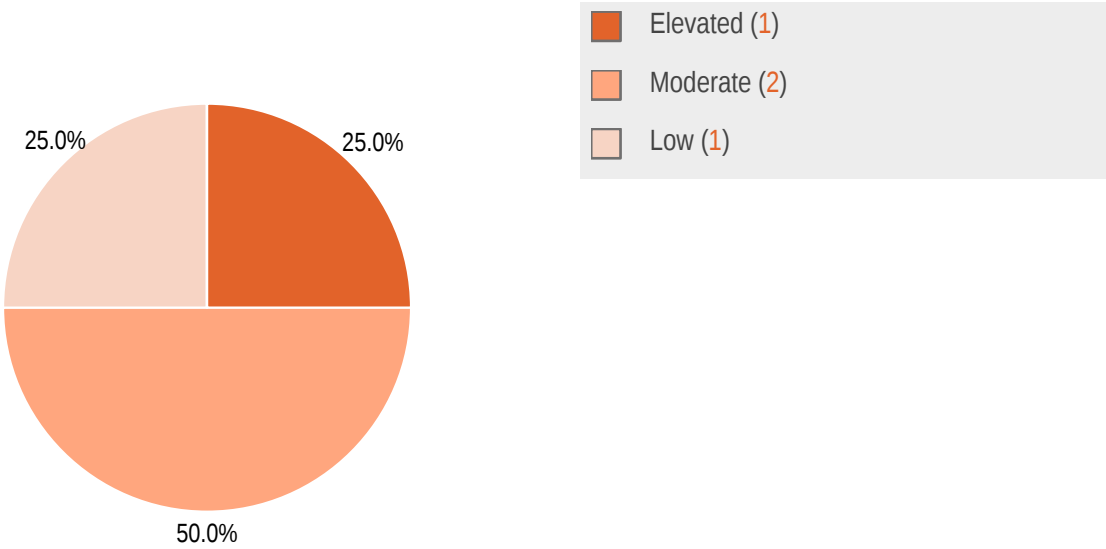
As part of the assessment, we thoroughly examined the behaviour of the app, scrutinising the app's handling of data and the security of the local web server used for file sharing. Our goal was to identify any vulnerabilities or security gaps that may compromise the app's security and the privacy of its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

1.6 Summary of Findings

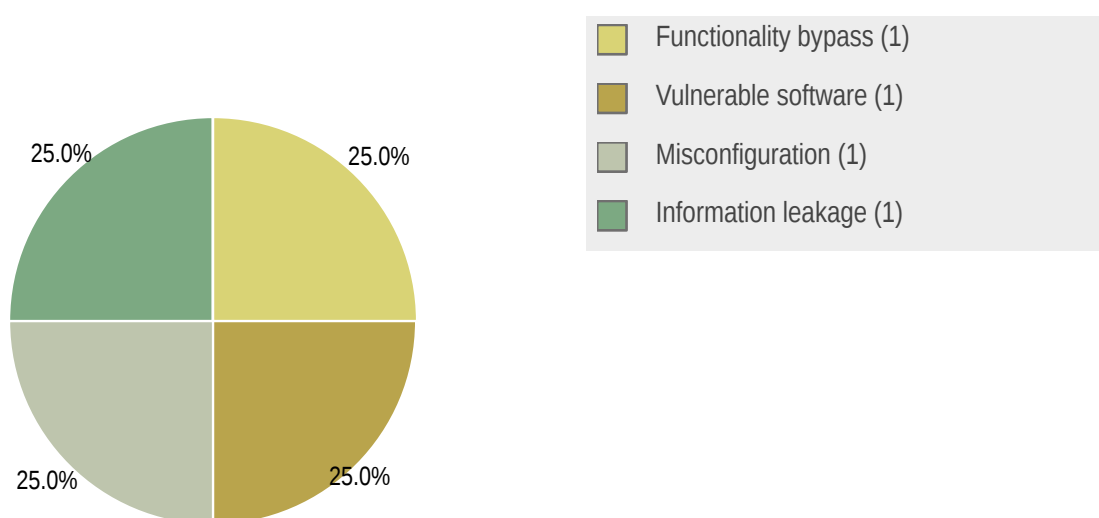
ID	Type	Description	Threat level
ONS-002	Functionality bypass	Since the application hosts a web server accessible at <code>127.0.0.1</code> , it opens up the possibility of using a local proxy app on the device to directly access the hosted files, bypassing the Tor connection.	Elevated
ONS-004	Vulnerable software	The android app uses vulnerable components. The used version has publicly known vulnerabilities.	Moderate
ONS-001	Misconfiguration	Anyone with network level access to the Onionshare host device can interact with the internal webserver and download shared files without going through Tor.	Moderate

ONS-003	Information Leakage	The Android app org.onionshare.android.nightly sends the details of web server and port number in the Android logs.	Low
---------	---------------------	---	-----

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

ID	Type	Recommendation
ONS-002	Functionality bypass	<p>Considering the way this application behaves, fully mitigating the risk of someone directly accessing the webpage might not be possible. But the following might help in reducing the risk, and makes it more difficult for the attacker:</p> <ul style="list-style-type: none"> Instead of having a fixed port for webserver, run the server on random ports Fix the ONS-003 (page 16) finding, so that other apps can't read the open port from logs Password protect the webpage, so that even after the page is accessed, the visitor would need to submit a password or key.
ONS-004	Vulnerable software	<ul style="list-style-type: none"> It is recommended to upgrade the version of affected software to a secure, stable and recent version.
ONS-001	Misconfiguration	<ul style="list-style-type: none"> Call <code>embeddedServer</code> with the <code>host</code> argument set to <code>127.0.0.1</code>.
ONS-003	Information Leakage	<ul style="list-style-type: none"> Sending sensitive information to the Logcat is generally discouraged due to security and privacy concerns. Logcat is a debugging tool that captures system logs, including app logs, and can be accessed by developers and potentially other parties with sufficient privileges. By sending sensitive information to Logcat, the app expands its attack surface, increasing the chances of successful exploitation by malicious actors. If an attacker (or malicious app) gains access to Logcat logs, they could potentially harvest valuable data, exploit vulnerabilities, or even launch targeted attacks against users. Given these risks, app developers should avoid logging any sensitive information. Instead, focus on logging relevant application events and

		error messages that assist in debugging without compromising user data.
--	--	---

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 through 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-002 — Accessing shared file without Tor

Vulnerability ID: ONS-002

Vulnerability type: Functionality bypass

Threat level: Elevated

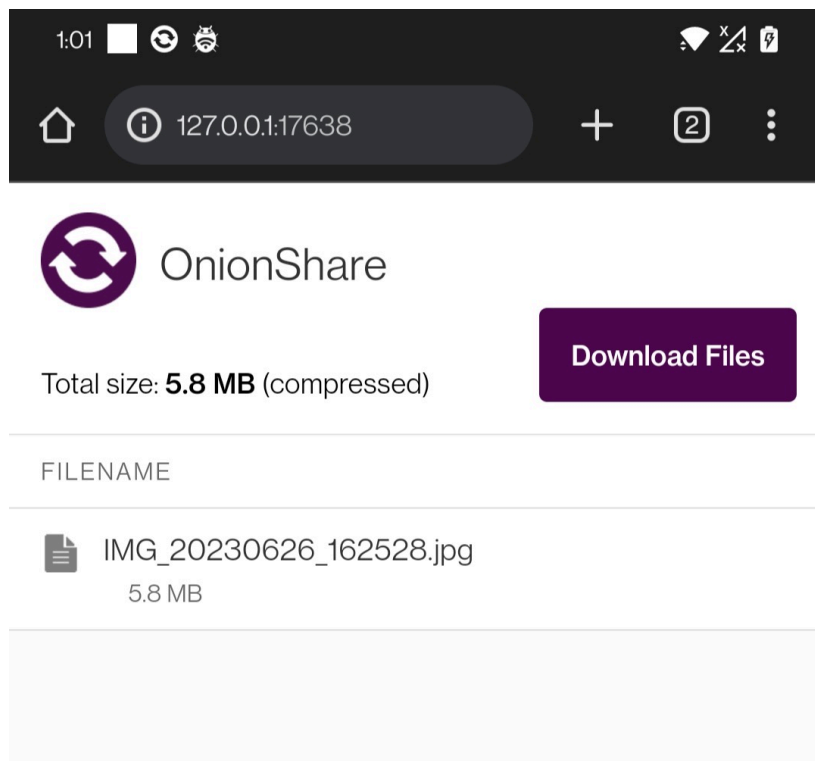
Description:

Since the application hosts a web server accessible at 127.0.0.1, it opens up the possibility of using a local proxy app on the device to directly access the hosted files, bypassing the Tor connection.

Technical description:

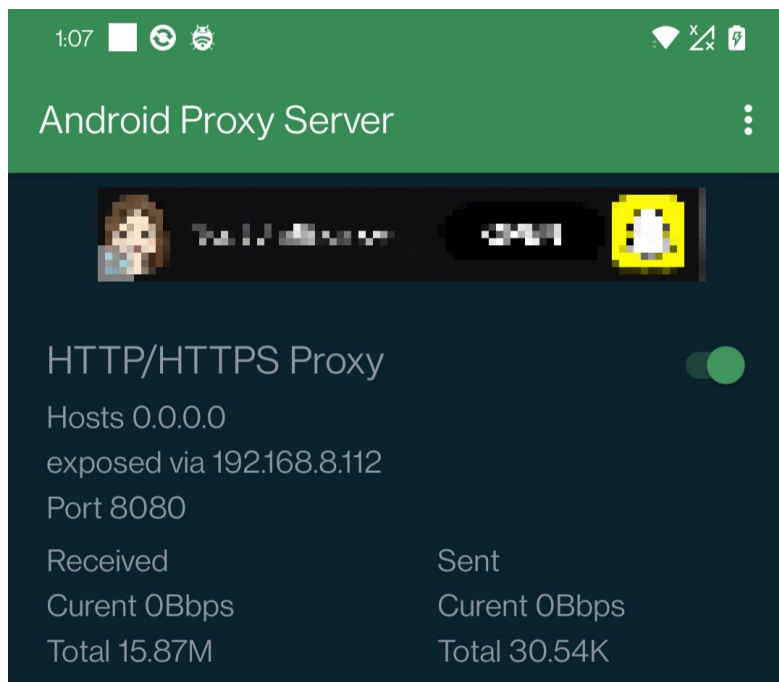
When the Android app initiates the sharing process and starts a web server on the device, it becomes accessible at the IP address 127.0.0.1 and port number 17638. However, it is important to note that during this process, the app inadvertently exposes the port number in the Android logs (check [ONS-003](#) (page 16)).

Local webserver hosting the file

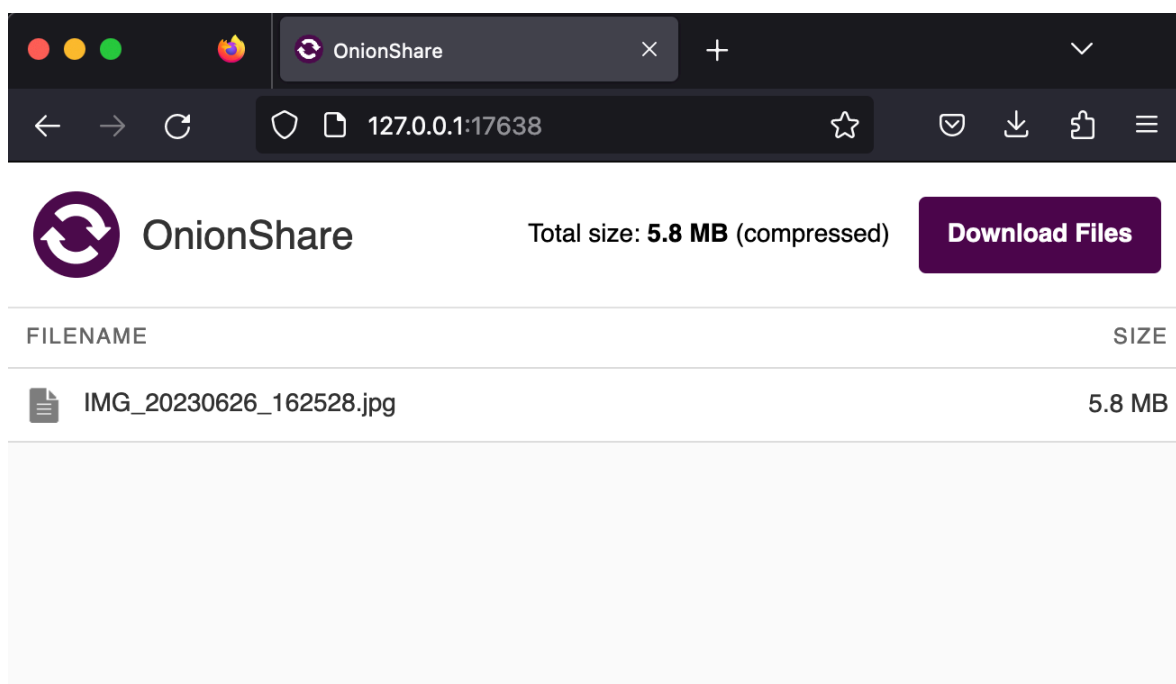


For an attacker, it is possible to access this shared file without going through Tor. This can be done through an app on the same device, running a proxy.

Running a proxy app



Accessing the file, through the proxy app, from Firefox on different device



Note: Onionshare Desktop faces the same issue, however the attackers on the same device are considered out of scope of the threat model.

Impact:

In the event of a successful exploitation, an attacker would gain access to the files being shared through the Android app's web server. The attacker can leverage the knowledge of the web server's IP address (127.0.0.1) and port number (17638) obtained from the Android logs. Armed with this information, they can establish a direct connection to the web server from the local network or even remotely, using a proxy app installed on the device. By bypassing the Tor connection, the attacker avoids any anonymisation or encryption layers provided by the Tor network. Moreover, this access to the shared files not only violates user privacy but also poses a significant security risk.

Recommendation:

Considering the way this application behaves, fully mitigating the risk of someone directly accessing the webpage might not be possible. But the following might help in reducing the risk, and makes it more difficult for the attacker:

- Instead of having a fixed port for webserver, run the server on random ports
- Fix the **ONS-003** (page 16) finding, so that other apps can't read the open port from logs

- Password protect the webpage, so that even after the page is accessed, the visitor would need to submit a password or key.

3.2 ONS-004 — Vulnerable software version being used

Vulnerability ID: ONS-004

Vulnerability type: Vulnerable software

Threat level: Moderate

Description:

The android app uses vulnerable components. The used version has publicly known vulnerabilities.

Technical description:

The Android app uses the following software components with vulnerable version:

- io.ktor:ktor-server-core@2.2.4 (line 131, <https://github.com/onionshare/onionshare-android/blob/main/app/build.gradle>)

```
def ktor_version = '2.2.4'
implementation "io.ktor:ktor-server-core:$ktor_version"
implementation "io.ktor:ktor-server-netty:$ktor_version"
implementation "io.ktor:ktor-server-pebble:$ktor_version"
implementation "io.ktor:ktor-server-status-pages:$ktor_version"
implementation "io.ktor:ktor-server-call-logging:$ktor_version"
implementation 'org.slf4j:slf4j-api:2.0.6'
implementation 'com.github.tony19:logback-android:3.0.0'
```

Known vulnerability - Directory Traversal: [CVE-2022-48476](#)

Impact:

The software version is affected by Directory Traversal vulnerability CVE-2022-48476. Even though a public exploit for this vulnerability is not available at the moment, the vulnerability was fixed in version 2.3.0

Recommendation:

- It is recommended to upgrade the version of affected software to a secure, stable and recent version.

3.3 ONS-001 — Internal webserver listens on 0.0.0.0

Vulnerability ID: ONS-001

Vulnerability type: Misconfiguration

Threat level: Moderate

Description:

Anyone with network level access to the Onionshare host device can interact with the internal webserver and download shared files without going through Tor.

Technical description:

In `app/src/main/java/org/onionshare/android/server/WebserverManager.kt` Line 64:

```
server = embeddedServer(Netty, PORT, watchPaths = emptyList(), configure = {  
    // disable response timeout  
    responseWriteTimeoutSeconds = 0  
})
```

`embeddedServer` will default to listening on 0.0.0.0 unless the `host` argument is set.

Impact:

The Onionshare web interface and files shared by a user will be accessible directly by any attacker on the same network as the device.

Recommendation:

- Call `embeddedServer` with the `host` argument set to `127.0.0.1`.

3.4 ONS-003 — Webserver details being logged

Vulnerability ID: ONS-003

Vulnerability type: Information Leakage

Threat level: Low

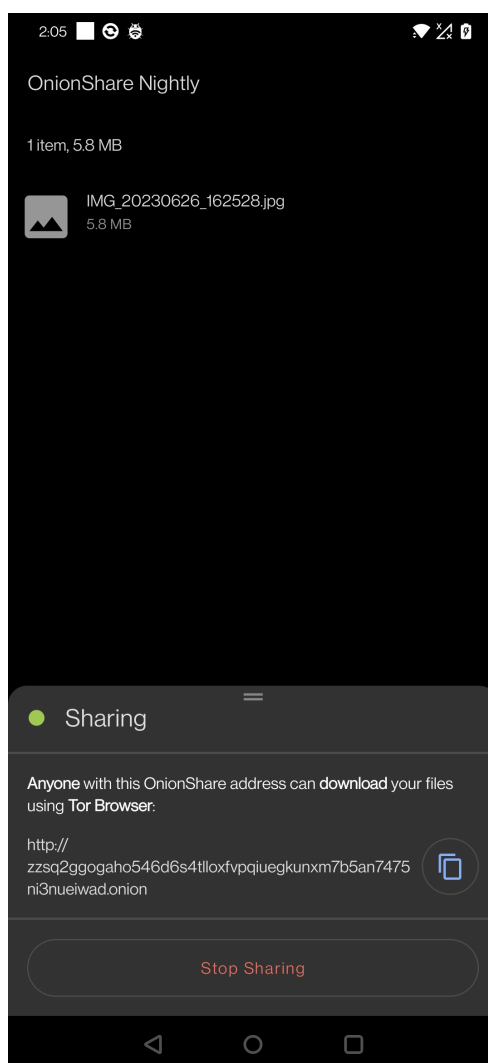
Description:

The Android app `org.onionshare.android.nightly` sends the details of web server and port number in the Android logs.

Technical description:

When the webserver starts on the device, the Android app sends the details to Android log.

Sharing a file



Android logs

```
D sendNotificationAddedBroadcast org.onionshare.android.nightly
I Auth cookie created
I New state from: f-Zipping t-Starting w-Stopped
D New state: Starting
I New state from: f-Zipped t-Starting w-Stopped
I Autoreload is disabled because the development mode is off.
I Application started in 0.004 seconds.
I Application started: io.ktor.server.application.Application@7659f8a
I New state from: f-Zipped t-Starting w-Started
I Responding at http://127.0.0.1:17638
I new state: started
I Starting hidden service...
I New state from: f-Zipped t-Starting w-Started
D New state: Starting
I Acquiring wake lock org.onionshare.android.nightly
I Starting check job
I New state from: f-Zipped t-Starting w-Started
I Waiting for Tor to start...
D New state: Starting
I new state: connecting
I NOTICE Opening Socks listener on 127.0.0.1:53054
I NOTICE Opened Socks listener connection (ready) on 127.0.0.1:53054
I OR connection LAUNCHED
I NOTICE Bootstrapped 5% (conn): Connecting to a relay
I New state from: f-Zipped t-Starting w-Started
D New state: Starting
```

This information can be useful for a malicious app on the device.

Note: The port number recorded in logs is not inherently sensitive information to disclose. Nonetheless, concealing this data from the logs would be a beneficial measure to enhance the difficulty for an attacker or malicious application to gain access to the webserver.

Impact:

Logging sensitive information in the Android log, is not a recommended practice as this information can be access by other application on the same device (in some scenarios). In this case, the information about port number and server address might be useful for a malicious application.

Recommendation:

- Sending sensitive information to the Logcat is generally discouraged due to security and privacy concerns. Logcat is a debugging tool that captures system logs, including app logs, and can be accessed by developers and potentially other parties with sufficient privileges. By sending sensitive information to Logcat, the app expands its attack surface, increasing the chances of successful exploitation by malicious actors. If an attacker (or malicious app) gains access to Logcat logs, they could potentially harvest valuable data, exploit vulnerabilities, or even launch targeted attacks against users.
- Given these risks, app developers should avoid logging any sensitive information. Instead, focus on logging relevant application events and error messages that assist in debugging without compromising user data.

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-005 — Non finding test cases

During the audit of the Android 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.

- Verifying the Android app components like activities, are not exported in a way that they lead to a security issue
- Verify if the webserver can be accessed from any other device with in the network, without using the local proxy app
- 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 Android device
- Test for unintentional information leakage
- Verify the process memory for any sensitive information
- Verify if it is possible to crash the app by performing a DOS attack on the web page
- Analysing the app runtime behaviour while sharing the file

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 1 Elevated, 2 Moderate and 1 Low-severity issues during this penetration test.

In conclusion, this pentest focused on the limited functionalities 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 four 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	Abhinav has 10+ years of experience in the penetration testing of web, mobile and infrastructure. He has received numerous accolades from multiple organisations for responsible disclosure of vulnerabilities. He is also known for providing trainings on web, mobile and infrastructure security.
Aidan Elphick-Miner	Aidan is a pentester with a broad range of skills. They started out their journey in their early teens by playing CTFs and hosting various services on a Linux server for their friends. Their IT security work is primarily focused on infrastructure penetration testing, social engineering, and OSINT. They have also worked on web and mobile application assessments, and will enthusiastically take on practically anything else you can throw at them. Aidan has also worked as an application developer, and a systems administrator for non-profits and other organizations.
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.

Front page image by Slava (<https://secure.flickr.com/photos/slava/496607907/>), "Mango HaX0ring",
Image styling by Patricia Piolon, <https://creativecommons.org/licenses/by-sa/2.0/legalcode>.