PENETRATION TEST REPORT

for

Jigsaw LLC

V1.1
Amsterdam
March 11th, 2018

Document Properties

Client	Jigsaw LLC
Title	PENETRATION TEST REPORT
Targets	Outline client apps for macOS, Windows, iOS, and Android (outline-client) Outline server for Linux (outline-server) Outline server manager for macOS and Windows (outline-server) Dependency: shadowsocks-libev proxy (not including libev) Dependency: the tun2socks part of the badvpn dependency Dependency: the SSH key generation part of the NodeForge dependency
Version	1.1
Pentesters	Mahesh Saptarshi, Stefan Grönke
Authors	Stefan Grönke, Alex Arlt
Reviewed by	Marcus Bointon Alex Arlt
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	November 3rd, 2017	Stefan Grönke	Add general project information and draft report
0.2	November 8th, 2017	Stefan Grönke	Address review from Santiago Andrigo
0.3	November 9th, 2017	Alex Arlt	Reviewing
0.4	November 27th, 2017	Stefan Grönke	Verify previous findings and review updated source code
1.0	November 28th, 2017	Alex Arlt	Review and Finalization
1.1	March 11th, 2018	Stefan Grönke	Adapt final product name 'Outline'

Contact

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

Name	Melanie Rieback
Address	Overdiemerweg 28 1111 PP Diemen

	The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

This report summarizes the findings and non-findings of a code audit between September 4, 2017 and October 20, 2017 as well as a re-visit of previous remarks and issues by state of Nov 23rd, 2017.

Subject of the audit was a set of software projects named Outline for orchestrating Shadowsocks servers, share/invite users and have them connect from multiple platforms, including Mac OS, iOS, Windows, Linux and Android, to a secure VPN.

Client user interfaces are built with WebComponents using Polymer encapsulated in Cordova or Electron Apps. While the VPN servers and client connections are managed with code maintained by Outline, we did not explicitly test for general flaws in Shadowsocks or the effectiveness as resilient tool to circumvent network filters.

The code repositories referenced in the scope of work were checked for best practice violations, insufficient input data validation and vulnerabilities in Outline Client, Outline Server Manager and bundled the Outline Server.

1.2 Scope of work

- Outline client apps for macOS, Windows, iOS, and Android (outline-client)
- Outline server for Linux (outline-server)
- Outline server manager for macOS and Windows (outline-server)
- Dependency: shadowsocks-libev proxy (not including libev)
- Dependency: the tun2socks part of the badvpn dependency
- Dependency: the SSH key generation part of the NodeForge dependency

1.3 Project objectives

The objective of this audit was to grant users safe usage of Outline client applications and secure setup and management of Outline servers on DigitalOcean, other cloud providers, and when using their own infrastructure.

1.4 Timeline

The Security Audit took place between September 4, 2017 and October 20, 2017.

1.5 Results In A Nutshell

After manually tracing and analyzing execution paths of the target applications based on a snapshot from the repositories, ROS sought direct contact with the implementing engineers to discuss findings and leads early on. This collaboration allowed us to address findings on a general level instead of just suggesting to fix them individually.

To establish connections the reviewed applications fork subprocesses. Insufficient input data validation had the potential to become a serious security issue that could lead to client side code execution vulnerabilities through malicious URLs or manipulated DigitalOcean Droplets in general. This could occur for example when a server provider gets compromised and the clients read malicious data or when attackers manage to inject a crafted ss:// URL to a users clipboard. In addition to checks regarding the safety of executed sub-commands, ROS has spent special attention to data interfaces (clipboard, keyboard inputs, URL schema or scanned QR codes) by following the data consuming methods backwards to their data input sources. We found that the data entry points were strictly secured by end of the audit and found that the reviewed Outline code is safely executed with an array of arguments from the applications. Measures to secure development assets like Dockerfiles and other build scripts with the same principle of strict data validation as recommended were also applied before finishing the audit, so that we can assume that users and developers systems are protected against command injection from compromised server instances.

Apart from the analysis of remotely exploitable vulnerabilities, minor local issues in key handling and data processing functions affecting the development environment were reported and fixed in GitHub pull requests. The changes ensured that sensitive data is not printed in log files or stored in system-wide accessible locations.

The mentioned vulnerabilities were addressed immediately after verification with the engineers, so that we were able to focus on establishing detection and quality enforcement mechanisms in the form of automated build steps. The input data validation was moved into a separate TypeScript Library called ShadowsocksConfig that allows safe usage of untrusted user input by strictly validating the presented data. The library can be utilized in all JavaScript environments used by Outline applications: as a browser module, Node package, or TypeScript import.

Due to proper usage of Polymer data-binding there were no front-end vulnerabilities found during the audit. The absence of implementation flaws and configuration mistakes was manually verified during the ongoing development of the applications. Our remarks and recommendations have been addressed before finishing the report, so that we were not aware of any findings that require being mentioned in this document. The engineering team's efforts to keep up with best practices of hardening web applications and their native application wrappers (Cordova/Electron) are much appreciated by ROS and we are looking forward to the public release of Outline.

2 Review and Verification

With manually verifying the fixes and mitigations these final steps conclude the code security audit. To do so, ROS has approached the task with four strategies:

- **Input validation with ShadowsocksConfig**
  Both Outline Client and Outline Server Manager fork subprocesses with arguments from user-input using the Node.js argument-safe child_process.execFile() method. The called applications were covered by this audit as well, so that we could verify the safety of call arguments. The major source of input data relates to Shadowsocks configuration parameters and URLs, so that ROS recommended introducing an independent TypeScript library that establishes a central definition of input data formats for all parts of the application. Conception, implementation and testing infrastructure were developed in close cooperation between the security consultants and the Outline software engineers. In addition to our efforts to verify the safe processing of arguments, this library enforces strict validation.

- **Review of fixes and mitigation effectiveness**
  Vulnerabilities found in the first audit iteration were reported to engineering and explicitly verified to finalize this document. At the time of publication, there is no vulnerability in the implementation known to ROS that would harm the software users system integrity.

- **Code delta review since the initial audit**
  In addition to reviewing the absence of previous findings, code changed since the initial audit was reviewed for newly introduced flaws or best-practice violations. Where possible, ROS proposed automated testing to permanently enhance the overall code quality.

- **Hardened Cordova/Electron Permissions**
  ROS recommended to tighten the application permissions that limit access to the local system and allowed domains. We have discussed what the minimal required permissions are and were happy to see our recommendations were being applied to the repositories.

3 Future Work

- **Fuzzing of ShadowSocks configuration user input**
  We have integrated unit tests into the build chain of the Outline-client GitHub repository, which is the groundwork for testing and fuzzing user input. Values from the ShadowSocks configuration are used in various places for critical tasks where insufficient validation can lead to code execution. After establishing unit test coverage, the next step to constantly monitor the implementations performance and stability is to fuzz arbitrary user inputs while measuring the runtime, execution paths, and expected outputs.

- **Extend Automated Quality Assurance**
  Automated code quality analysis and testing can be utilized to prevent specific mistakes and flaws from appearing in the master branch. We have demonstrated unit test integration into GitHub statuses that are enforced before changes can be merged into the master branch. TypeScript and tools built around the compiler provide great features to base static code analysis on and enforce quality metrics.

- **Automatic Operating-System Updates**
  Running server operating systems can be updated frequently without requiring user interaction. Users of the Outline Server Manager app should be able to see the latest OS update date and system uptime, so that running software versions can be reviewed and manually verified. Having access to this information enables server administrators to take appropriate action in case of newly published vulnerabilities in the Operating System or it’s bundled software. An automated job that frequently updates the operating system is important to grant safe operation of the server, even if no administrator manually logs in to deal with it. Some operating systems (for example Debian 9 Stretch) support unattended updates out of the box, and Outline could provide an user interface to enable/disable such auto-updates.

- **Review changes since the audit**
  Code audits and penetration tests of software that is under development is just a single snapshot. Security must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain and package secure software.

4 Conclusion

The technology stack chosen for Outline is a good foundation to build reliable OS-independent user interfaces. The few interfaces with third party software (shadowsocks-libev and tun2socks) and user input were carefully checked for flaws.

Before introducing strict data-validation, attackers would have been able to inject code into the Outline clients by crafting malicious URLs or taking control of the Outline Server when users connect to manipulated servers.The vulnerabilities related to data validation did not appear in the frontend or server code itself but the interfaces with the operating system, where unvalidated input could lead to code execution.

By creating a TypeScript validation library that can be used in the browser and in server applications it became possible to validate untrusted input at the interface point rather than relying on sanitization at usage. The library was built with automated unit tests enforced by Travis CI, so that input data requirements are defined in one place.

With the completion of reviewing fixes and auditing code changes applied to the sources to the present day, we verified the absence of our previously raised security concerns. The Outline team showed exceptional interest in the application security and enabled ROS to explore automatable strategies to durably enhance the code quality and security.

Overall the Outline concept appears to be well-suited for enabling users to create, use and share their self-managed VPN servers. The reference implementation, using DigitalOcean Droplets as deployment target, is comfortable to use while following a technically simple and elegant approach. Our biggest concern was, supported by the findings during the audit, that insufficient input data validation could have lead to code execution on servers or clients. The measure of processing Outline configuration data exclusively with a library that strictly validates and parses untrusted input data addresses this potential issue in a central position, so that we evaluate Outline properly protected against malicious inputs. Although the software stacks utilized by the product have a certain complexity, the benefits from its abstractions oppose the risks. With appropriate care for the security of its dependencies and frequent releases, ROS feels comfortable to recommend Outline to users intending to dig encrypted tunnels between supported clients and managed server instances.

Users who suspect targeted attacks should take additional steps to verify the safety of dependencies included by the client applications. External libraries effectively mitigate basic web vulnerabilities, but also introduce complexity that, depending on the attackers endurance, might be a risk that should be considered.

Finally we want to emphasize that security is a continuous process; this code audit was 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.

Appendix 1 Testing team

Mahesh Saptarshi	Director, cyberSecurist Technologies. Mahesh is passionate about software security defences. He has performed a large number of pentests of enterprise, web and mobile applications. He has several US patents in the area of high availability and virtual machines technology.
Stefan Grönke	Stefan Grönke is a highly adaptable full-stack developer with a large focus on security and simplicity. He has 15 years of experience in (reverse) engineering, architecture and quality assurance. As he always enjoyed learning from and with open source code, he has contributed to a variety of projects.
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.