Open Technology Fund

Security Audit Review

Prepared for:

Radio Free Asia's Open Technology Fund

Prepared by:

Tom Ritter – Principal Security Consultant, iSECpartners

Table of Contents

1 Executive Summary

Open Technology Fund (OTF) finances technologies that promote human rights and Internet freedom globally. As a funder of emerging technology, OTF is interested in keeping the quality of its investments high. One component of ensuring its projects meet the highest information security standards includes facilitating independent technology audits. Accordingly, OTF contracts information security auditors to assess the privacy and security limitations in OTF-funded projects and suggest remedial recommendations. These audits both mitigate risk inherent in funding cutting-edge technologies and strengthen the technical capacity of OTF supported projects and the broader human rights and Internet freedom technology community.

To derive the greatest possible value from these audits, OTF has sought to identify a process by which to evaluate technical security audit reports from the perspective of a funder, not a technologist. This document provides a framework for how an organization, such as a human rights funder or an NGO, can effectively and efficiently engage information security auditors, based on OTF's experience and findings.

Specifically, Section 2 introduces the Open Technology Fund, its security audit process, and defines terminology used through the document. Section 3 provides a methodology for working with information security auditors and assessing security audit reports. In Section 4 we identify areas of inadequacies, problems, or concerns we expect an auditor to evaluate, along with other criteria we expect security auditors to meet, such as recommendations for reducing security vulnerabilities in the future. Section 5 contains the results of OTF security audits, summing up individual reports, identifying best and worst practices, and common themes. The conclusion in Section 6 provides suggestions for interviewing and selecting an audit firm, and offers specific recommendations for OTF, although useful for other organizations, to improve its information security audit process. Finally, appendices and attachments include worksheets, checklists, and templates used by OTF that may assist other organizations with similar needs.

2 Introduction

Open Technology Fund (OTF) uses public funds to support projects that develop open and accessible technologies promoting human rights and open societies. Its funds are delivered as contracts promoting research, deployment, and implementation of applications promoting Internet freedom. A portion of these funds are also paid to security audit firms to conduct reviews of OTF-funded applications. The security audit firms review the applications to identify specific exploitable vulnerabilities, suggest fixes, and provide recommendations for preventing future flaws and improving the overall privacy and security of the application. This is often termed as “defense in depth”. Further, increasing the number of high quality information security audits provides sustenance to the growing field of human rights and technology.

This document reviews the breadth, scope, and coverage of security audits performed on OTF funded applications to date. Its goal is to identify the strengths and shortcomings in OTF’s current process. Further, it will provide recommendations to improve the breadth of coverage and to derive greater future value. iSEC Partners was chosen as a partner for this report due to its unique experience evaluating and contributing to Internet freedom technologies.[1]

This report presents a methodology to assess multiple audit firms producing multiple reports testing multiple applications and evaluates sixteen OTF-sponsored audits. Within are definitions of evaluation categories that auditors and funders should consider and commonalities observed between these categories and between audit firms. Finally, the report provides OTF with suggestions aimed to derive greater value from future audits.

Through the document, the following terminology is used:

- **Project** is used to refer to the team of individuals receiving funds from RFA
- **Project’s Application,** or just **application**, is used to refer to the specific application or tool funded by OTF. Projects may be working on multiple applications, not all of which are funded by OTF.
- An **Audit** is an evaluation or assessment of the security of an application. It identifies vulnerabilities and insecure design decisions and provides recommendations to fix the flaws and improve security.
- **Auditor** is used to denote the individual or organization performing the audit for a specific application.
- **Funder** is used to denote an organization that subsidizes security audits for an application. In this report it refers to OTF – but we aim to provide a methodology and recommendations that can be used by different funding entities.

[1] LibTech-Auditing-Cheatsheet - https://github.com/iSECPartners/LibTech-Auditing-Cheatsheet

A note on confidentiality and original audit reports

Final reports from auditors are often confidential. To fully explain their findings and help applications resolve any problems, reports will describe tools and methodologies used during the audit. These tools and methodologies are often trade secrets that make an auditor competitive. For this reason, the reports are not available for public review and considered confidential information available only to OTF and its consultants under confidentiality agreements. During the preparation of this report, auditors' final reports were made available to a single iSEC consultant operating under NDA forbidding disclosure or anti-competitive business practices.

iSEC Partners is a strategic digital security organization, performing application and system penetration testing and analysis for multiple platforms and environments. iSEC regularly provides Application Security and Network Security consulting practices for the most common and trusted components of the Internet ecosystem. Besides regularly performing security assessments of all types at iSEC, the author has extensive experience in OTF's field of work. He has written suggestions for auditing sensitive applications of these types and made them available via Creative Commons, tracks and comments on the development of many of OTF-funded projects and their peers, participates in the development of Internet security standards and protocols, and has conducted security research and presented at security conferences around the world.

3 Methodology

This section outlines the methodology used to evaluate sixteen audits commissioned by the Open Technology Fund (OTF). This methodology can be used to evaluate multiple security audits performed by multiple auditors that cover multiple applications. It aims to identify strengths, shortcomings, and provide recommendations for better coverage and deriving greater value in future technology audits. This methodology strives to be useful for funders with programs dependent on technology – in particular, technology where privacy and security standards are crucial.

An experienced auditor can look at an application and understand what the significant areas of concern are. These areas, or categories, of concern relate to the intended use of the technology, the programming language or pre-existing framework it is built with, and the underlying architecture of the application (such as web, mobile, etc.) While these *categories of concern* are used by an auditor for scoping and staffing, they are equally, if not more, valuable to the funder. They allow the funder to measure and evaluate an application's risk, think strategically when creating a portfolio, they can be used to guide budget allocation, and ultimately determine a project's success.

While outside the scope of this document, we would be remiss if we did not mention the value to the application developers themselves. It is often outside of the projects' means to consult with information security auditors whose typical clients are in the Fortune 500. With in-kind access to this tier of security auditors, not only are the applications safer, the developers behind the projects are smarter.

We start by identifying categories of concern from OTF funded applications (see Categories of Concern on page 11). This is not an exhaustive list of all possible categories, but we feel it adequately describes the types of applications that OTF has funded and is likely to fund in the future. After defining categories, we applied them to the projects' applications. Someone familiar with the basic technical architecture of the application can complete this process - whether by the funder, the project developers, or an external auditor. A list of the categories defined can be found in on page 21. All projects' applications have basic functions that should be tested for flaws (such as a messaging app sending a message to the wrong user) and this is assumed in all cases and excluded from our list.

With categories defined and applied, we determined the specific criteria we want to evaluate each audit with. We identified the following five areas, with some input from projects: Scope, General Recommendations, Issue Complexity and Issue Spread, Issue Remediation Recommendations, and Categories of Concern. Going forward, projects should be move involved defining these criteria, as ultimately they will be using the audit reports to improve their products. Below we attempt to describe these criteria with as little jargon and light technical explanation:

1. **Scope**: The scope of an audit is the major components that may be reviewed by the auditor. It often defines the sections of a website that should be tested, source code to review, or can specify that certain components should not be tested due to time or other constraints. An auditor should work with a project prior to the start of an audit to determine the scope. But beyond this essential step, after an audit is completed, the

report should explicitly identify which sections of the agreed-upon components were and were not tested and what types of testing were performed. It is common for auditors to include generic testing bullet points, which are often reused across clients to explain what services they offer. While they are not necessarily detrimental, coverage of each targeted component should be clarified or confirmed after the audit and included in the report.

While reviewing the audits sponsored by OTF, scope was categorized as follows:

- *Strong* - Scope statements explicitly identify the components covered or omitted, and the types of testing performed.
- *Haphazard* - Scope statements mention specific components covered or types of tests performed or omitted, but do not list items exhaustively.
- *Missing* - No direct statements regarding the scope of the assessment, or generic statements placed in all reports.

2. **General Recommendations:** A general recommendation is a suggestion for a project that will help to reduce the possibility of introducing flaws in the future and make an application more robust against any individual vulnerability. General recommendations are separate and in addition to recommendations given to fix individual vulnerabilities. When conducting a root-cause investigation or analysis of a flaw, general recommendations should be a few iterations up the chain. Often, general recommendations are considered broad and strategic while issue recommendations are considered specific and tactical.

   For the OTF-sponsored audits, reports were reviewed for the presence of general recommendations not tied to a single specific vulnerability or their non-presence.

   - *Good* – More than one useful to excellent general recommendation not tied to single specific issues
   - *Average* – At least one general recommendation not tied to single specific issues
   - *Missing* – No general recommendations

3. **Issue Complexity and Issue Spread**: A straightforward, if subjective, way to judge an audit is solely on the number of issues identified, their technical complexity, and the variation in their complexity. Although there is value in attempting this exercise, as the single metric in an evaluation it has a significant number of flaws. There are a number of ways this data can be interpreted, none of them correct in all circumstances. For example, a large number of technically unsophisticated findings does not imply the audit was unsophisticated, but can mean the auditors spent a large amount of time documenting easily identified issues. However, a consistent lack of sophisticated issues identified, across applications, may imply weakness in the auditing team. A small number of sophisticated findings does not mean an application is very secure, but could mean that the auditor did not thoroughly search for straightforward flaws, instead focusing on headline items - alternatively it could mean that there are not many straightforward flaws to be found. Although imperfect, the number of issues and their complexity should be evaluated.

In this report we used the categories and approximate criteria listed below. The target for an auditor is "Good".

- *Excellent* – Several very subtle and technical issues and/or a good range and number of issues identified
- *Good* – Strong technical capability demonstrated
- *Average* – Issues identified are generic and do not demonstrate logic flaws
- *Lacking* – Issues identified are technically unsophisticated and generic.

4. **Issue Remediation Recommendations**: For each issue identified, an auditor should explain it well enough for developers to understand the root cause behind it, and provide specific recommendations for the steps required to fix it. Ideally, they will also help the project prevent this type of flaw from being introduced in the future.

   Issue recommendations were judged based on two independent criteria. Ideally, an auditor will provide custom recommendations for all issues identified.
   - *Consistency* – The auditor provided useful recommendations for every issue identified, as opposed to omitting recommendations for some issues.
   - *Customization* – The auditor provided useful recommendations that were specific to the application as opposed to a generic recommendation relating to the type of issue identified.

5. **Categories of Concern:** Finally, a significant concern for a funder is how thorough the audit was, and if it covered all of the categories of concern identified for an application. This coverage is often the difference between feeling comfortable with an application and wanting to prioritize another audit.
   Auditors are evaluated on all categories of concern identified for an application separately. The target for an auditor is "Good", but it is common to receive "Average".
   - *Good* - One or more 'Excellent' or 'Good' issues (in terms of the issue complexity above) identified. If no issues are identified; the audit could alternatively include a strong assertion the category of concern was exhaustively tested
   - *Average* – One or multiple 'Average' or 'Lacking' issues (again, in terms of the issue complexity from above) identified. If no issues are identified; the audit could alternatively state that the category was tested for
   - *Poor* - No indication that the category was tested for

In conclusion, the methodology created in this document began by developing categories of concern and base criteria to evaluate an audit. To utilize this methodology as a tool, we applied the categories to Open Technology Fund's past and present projects. Next, we applied each criterion to individual audits completed to date: coverage of each category of concern, how clearly the scope was explained, if general recommendations were provided, the level of detail provided for fixing each vulnerability, and the overall complexity and breadth of issues identified. Finally, the data was entered into a matrix, with the following conclusions and suggestions drawn from its analysis.

4 Categories of Concern

The goal of these categories of concern is to identify types of testing that, when applied to an application, can be used to help funders scope audits and evaluate the value of audit reports. These categories relate to the application type, choice of implementation, intended use of the application, service administration, and other factors. They could also be used earlier on to classify an application type during the proposal process.

Multiple categories will apply to a single application. A funder, having received an audit report, should be able to move through the relevant categories and get an idea of what categories were covered, or alternatively, what categories were omitted. As mentioned before, all applications have some basic functionality that should be tested for flaws. While these flaws can overlap with an identified category (most commonly in a Web Application), often they will stand-alone unclassified.

The below categories have been identified based off OTF-funded projects. It is not wholly comprehensive, but adequately catalogues the projects OTF has worked with. Next, we attempt to introduce each category in non-technical terms and then move to provide a deeper technical context for each. Following the definition, we list suggested questions that funders should ask themselves while reviewing an audit report. As new projects are funded, it will be useful to re-view them to determine if additional categories and questions are needed.

- Browser Extension
- Unmanaged Language
- User Interface/Experience
- Anonymity
- Deployment Review
- Web Application
- Server Daemon
- Mobile Application (Clientside)
- Web API/Mobile Application (Serverside)
- Cryptographic Review
- Dependency Integration
- Desktop Client
- Forensics Review

4.1 Categories

4.1.1 Browser Extension

A browser extension extends the functionality of the web browser in some way, often by modi-fying existing web pages as you browse, or by packaging a separate application into the browser for easy access.

They are typically written in the JavaScript computer language, and share many types of vulnerabilities with web applications. Besides basic flaws like cross-site scripting[2], flaws that are more complicated are common, including improper use of cryptography and logic bypasses. However, browser extensions are afforded significantly more privileges than normal websites and have a larger attack surface.

A significant concern for browser extensions is how a 'normal' website (that is, one unprivileged on the broader Internet) can interact with the extension. Attacks specific to browser extensions include fingerprinting (if a website can determine if a user has the extension installed making it easier to identify or attack the user) and unintended activation (if a website can induce an extension to begin executing the extension unknown to the user). Extensions often access the user's local file system and the trusted browser user interface through menus, popups, or icons, and these APIs should also be reviewed for unintended file access, injection, and spoofing attacks. Finally, browsers generally provide capability control and opt-in sandboxing features to extensions, which should be used to limit the impact of a compromise as much as possible.

Note: A browser extension should not be confused with a browser "plugin", which is generally written in the C or C++ programing language. Extensions provide additional rendering facilities to the browser, handle specific file formats, or are invoked in <embed> or <object> tags. Flash, Silverlight, QuickTime, and Adobe Reader are examples of plugins. Popular browser extensions include AdBlock, StumbleUpon, and NoScript.

Pre-Audit Questions for a Project:

- Does your extension make use of any external components such as additional applications, or is it solely JavaScript based?

Pre-Engagement Questions for Auditors:

- Is testing the browser extension in every browser in scope, or only in a single browser?
- Have you reviewed or built browser extensions before? In which browsers?
- What types of issues, general and specific, do you look for when auditing browser extensions?

Post-Audit Review Criteria:

- Has the auditor identified issues in browser-specific extension APIs, or referred to their review?
- For projects with multiple extensions for different browsers, has the auditor demonstrated or specified they reviewed the browser-specific APIs in each?
- Has the auditor determined if the extension may be fingerprinted by a 'normal' website?
- Has the auditor specified sandboxing or hardening suggestions for Google Chrome[3], or identified the extension as being fully hardened?

[2] https://en.wikipedia.org/wiki/Cross-site_scripting, a type of attack that allows an attacker to execute malicious JavaScript to impersonate a user or steal credentials

[3] Chrome in particular provides several sandboxing options for extensions not present in other browsers.

4.1.2 Unmanaged Language

Applications are written in programming languages, and the choice of language can influence how likely there are to be flaws and vulnerabilities of different types. Certain languages, such as C and C++, are unmanaged languages, and while they can provide more flexibility and speed than others, they are also significantly more prone to extremely dangerous flaws. This is because they typically must keep track of difficult-to-manage components that are abstracted away in other programming languages.

In an unmanaged language (also called a memory unsafe language), all memory allocations, deletions, reads, and writes are targets for exploitation. If the application developers do not use safe best practices, an opportunity can be created for an attacker to compromise the user's computer. This is known as exploitation primitive. A single primitive can be enough to lead to arbitrary code execution and a complete compromise of the application along with the user's whole computer.

Source code review of an unmanaged language requires specific expertise, and is often augmented by writing fuzzers[4] specific to the application under review. Of particular concern is any unmanaged code that parses a network protocol or file format. Auditors can also make recommendations for hardening an application and making exploitation more difficult. These include building the application with all security-related options and using seccomp-based sandboxing[5] on Linux. Other related features, such as DEP/NX[6] and Address Space Layout Randomization[7], are basic enough that their omission should be considered a bug.

Pre-Audit Questions for a Project:

- Is your application written in a memory-unsafe language?
- Are any of your dependencies?

Pre-Engagement Questions for Auditors:

- Do you make use of any fuzzers, static, or dynamic analysis tools?

Post-Audit Review Criteria:

- Has the auditor suggested specific hardening options or techniques the project can make use of to make exploitation more difficult?
- Has the auditor explained in detail any runtime instrumentation they performed or provided specific fuzzers or fuzzer configurations?

4.1.3 User Interface / Experience

Most applications are intended to be used by non-technical individuals. The interface presented to the user, and the necessary actions to accomplish a certain task, represent opportunities

[4] https://en.wikipedia.org/wiki/Fuzz_testing, a type of testing that involves sending semi-random input to an application, to exercise strange conditions
[5] https://en.wikipedia.org/wiki/Seccomp, a mechanism to limit the ways an application can behave when it is being attacked
[6] https://en.wikipedia.org/wiki/Data_Execution_Prevention, a setting to make exploitation more difficult, preventing data from being executed as program code
[7] https://en.wikipedia.org/wiki/Address_space_layout_randomization, a setting to make exploitation more difficult, randomizing the location of program code when the application is running

for a user to inadvertently do something insecurely or other jeopardize themselves. A clear and unambiguous interface significantly increases secure use of an application.

Over the years, in papers such as “Why Johnny Can’t Encrypt” (1999), “Why Johnny Still Can’t Encrypt” (2006), “Why (Special Agent) Johnny (Still) Can’t Encrypt” (2011), and the common example of the ‘lock icon’ and certificate warning screens in browsers, we have seen empirical evidence that designing effective user interfaces for security and cryptographic tools is extraordinarily difficult, with few successes. While security auditors are not the most qualified to identify User Experience improvements, they are qualified to look for issues that may impact the use of the applications. These issues can include spoofing the user interface, attacks where an attacker impersonates a legitimate user, and unsafe defaults.

Separate from a security audit, examining an application by observing user interaction and using tools such as mouse and eye tracking can uncover significant design flaws or usability impairments.

Pre-Audit Questions for a Project:

- Have you performed any usability studies with your targeted user base?

Pre-Engagement Questions for Auditors:

- Do you have the capability to perform non-technical usability studies in addition to security audits?

Post-Audit Review Criteria:

- Has the auditor specified any issues that may cause users to use the application unsafely, such as dangerous default options, confusing dialog boxes, or impersonation attacks?

4.1.4 Anonymity

The classes of applications we concern ourselves with in this report often have stated or implicit goals to protect the user’s identity, hide use of the application, or prevent user tracking. If an application aims to provide a degree of anonymity or prevent user tracking, specific testing should be done.

The vast majority of audits are performed for corporate clients. Thus, the audited applications generally fall within a broad privacy policy that makes exceptions for sharing information with third parties for the purposes of providing a service to the user. However, for applications in the human rights and open society ecosystem, third parties are often seen as untrusted and user data should explicitly be protected from them. Design decisions or implementation flaws that lead to user tracking across connections or networks, inadvertent disclosure to third parties, and other full or partial de-anonymization issues are an important component of review that applies to most applications. While these concerns often overlap with other categories (such as detecting if a user has installed a browser extension), many times they do not, and must be considered by an auditor specifically.

Note: Because anonymity is extremely difficult to provide in Internet communications, it’s important for a project to have clearly documented the security and privacy issues they care about and the threats they hope to defend against. This process is called threat modeling.[8] Without such a document, an auditor is left to make guesses – while this is possible, it results in greater ambiguity in a review, and may result in omissions or issues the project does not feel are important.

Pre-Audit Questions for a Project:

- Do you have a threat model? Is it your intention to provide anonymity for users of your application, and if so, make them anonymous to whom (the server operator, users they communicate with?)

Pre-Engagement Questions for Auditors:

- Have you examined applications for privacy or anonymity concerns before?
- What types of issues do you look for when doing so?

Post-Audit Review Criteria:

- Has the auditor identified any threats to the anonymity?

4.1.5 Deployment Review

Certain applications contain a central server component that the project runs a copy of themselves for users to interact with. If the project team runs a copy of their software for general public use, it is critical that the auditor review the application's deployment in addition to testing the individual application for faults. This is traditionally referred to by auditors as a Network Penetration Test and potentially Configuration Review.

The review should cover all services running on the server(s) in question, as well as trusted communication partners of the server(s). It should look for issues such as misconfigurations, weak authentication protocols or passwords, outdated software, and unnecessary services. Additionally, beyond the scope of most normal network penetration tests, the review should consider the hosting provider (shared vs. virtualized vs. dedicated), DNS registrar, mail exchanges, and any other third parties that may have privileged access to the service.

In a white box audit, the review would cover the servers' configuration files, logging practices, network segmentation, and key storage. It may also cover the operational security practices of administrators and developers, use of two factor authentication, and private key protection.

Pre-Audit Questions for a Project:

- Do you run a copy of your application in a production environment for ordinary users?
- Are elements of your infrastructure (email, DNS) administered by third parties that preclude testing?

Pre-Audit Review Criteria:

- Is review of a test or production deployment in scope? Which elements?

Post-Audit Review Criteria:

- Has the auditor listed running services of concern, or noted that there were none?
- Has the auditor suggested hardening options for any server configurations or firewalls?

[8] https://en.wikipedia.org/wiki/Threat_model, briefly, the types of attacks an application aims to defend itself against, and the types of attacks it does not.

4.1.6 Web Application

A web application is accessed by users using a web browser, mobile phone, or tablet, and is delivered to the user from an Internet-accessible server. It generally consists of code that runs on a server (usually accessing a database) and HTML and JavaScript code that runs on the user's device.

A web application review begins with testing for standard vulnerabilities common in this class of application, including SQL Injection, cross-site scripting, Direct Object References, and others. However, a review must explicitly cover authentication, authorization and logic bypasses as well, because each application is significantly different in these regards. Bugs such as invalid state transitions and password reset circumvention are extremely damaging.

An auditor can make defense in depth recommendations tailored to the specific needs and deployment of an application. Common examples are recommending HTTPS Strict Transport Security[9] and hardening crossdomain.xml[10] files. More in-depth recommendations can include roadmaps for enabling and enforcing Content Security Policy[11], more defense-architected development patterns, and a framework for minimizing database permissions.

Additionally, for applications that process sensitive data or seek to protect the confidentiality of its visitors, a privacy-specific web application review may be warranted. These additional checks are generally based around reviewing what information is disclosed to third parties when a user visits the page, including information sent to analytics companies, certificate retrieval processes, and social media integration.

Pre-Audit Questions for a Project:

- Do you make use of any new web features, such as local storage, WebRTC, or web sockets?

Pre-Engagement Questions for Auditors:

- Will the scope include documenting third party resources and information leaks?

Post-Audit Review Criteria:

- Has the auditor identified logic flaws or only common web application vulnerabilities?
- Did the auditor provide recommendations for how developers should improve an application to provide defense in depth and make exploitation more difficult?
- Has the auditor noted any third party information disclosures (or stated there were none?)

4.1.7 Mobile Application (Client-side)

A mobile application is an application than runs phones or tablets and is usually delivered through an app store such as Google Play or the Apple App Store.

The client component of a mobile application review is the code running on the user's mobile device. Testing covers attacks that can be waged from the perspective of the network, a malicious application installed alongside the target app, and from a forensic analysis of the device.

[9] A setting to enforce use of SSL/TLS.
[10] A setting that governs what permissions Flash applets have when interacting with a domain.
[11] A setting that permits what content is allowed on a website, based on the origin and type.

Areas of investigation include system logging, local storage, certificate validation, and inter-process communication (IPC).

Pre-Audit Questions for a Project:

- Do you use certificate pinning, code obfuscation, or other mechanisms that will make auditing difficult?

Pre-Audit Questions:

- Will you be testing all mobile platforms the application runs on, or only a specific one?

Post-Audit Review Criteria:

- Has the auditor identified mobile-specific vulnerabilities or app-to-app attack vectors?

4.1.8 Server Daemon

A server daemon is an application that runs on an Internet-accessible computer that interacts with other applications on different computers. In general, this is a broad category that can include web applications and web APIs, but in the context of this report this category has a more narrow focus on more complicated or uncommon protocols.

A server daemon often allows an attacker to send arbitrary data to an application. The attacker has significant ability to transmit both valid and invalid data in an effort to confuse parsers or state engines, and is generally afforded tremendous access and information if they succeed. Accordingly, particular care must be given to developing Internet-exposed daemons.

Besides general security considerations such as logic bypasses, or implied by use of an Unmanaged Language, server daemons are at significant risk of Denial of Service. Denial of Service attacks increase maintenance costs, decrease the probability an application will actually see widespread adoption and use, and can result in people temporarily using insecure solutions. If an attacker is able to crash a service, without any indication of the cause of the crash being logged, the operator will have no choice but to enable full packet logging to try and detect the source of the crash – which brings its own set of risks and concerns.

Pre-Audit Questions for a Project:

- Does your server only present a standard interface (just as REST or SOAP) or do you make use of any custom protocols?
- Do you parse any file or network protocols, or have dependencies that do?

Pre-Engagement Questions for Auditors:

- What types of testing do you perform against server daemons?

Post-Audit Review Criteria:

- Has the auditor identified any Denial of Service conditions in server daemons?

4.1.9 Web API / Mobile Application (Server-side)

A Web API is an interface exposed on the Internet for other applications to interact with. Mobile Applications often interact with a Web API to fetch data from or save data to, and APIs are often used to provide access to multiple different types of applications (such as different Twitter clients).

Although similar to a server daemon, an API differs in that it is not responsible for parsing an underlying protocol and instead receives structured requests from a client. The API methods may be open and unauthenticated for use by a first-party application such as a mobile app, or may provide third-party apps integration to the first-party service (again, such as Twitter clients). A Web API should be tested for web application vulnerabilities such as authentication, authorization, SQL Injection, and Direct Object References. More complicated APIs may involve testing the implementation of OAUTH[12], a permissioning or capability model, or an app approval process.

Note: If an API is exposed for use by a Mobile App, it should not be considered 'mobile only', as anyone is able to learn about it and query it in unexpected ways. A mobile application review is not complete unless it also tests any Internet-facing APIs used.

Pre-Audit Questions for a Project:

- Do you have a permissioning or capability model, app review process, or OAUTH implementation that provides access to first-party data from third-party apps?

Pre-Engagement Questions for Auditors:

- What types of testing do you perform against APIs? Do you write sample applications to exercise them?

Post-Audit Review Criteria:

- Has the auditor specified clearly what scenarios were tested, and what types of testing were omitted? (In scenarios where the API restricts applications to certain permissions, it is uncommon to test every possible combination.)

4.1.10 Cryptographic Review

Cryptography uses delicate mathematical constructs to protect data while it is being transmitted or stored. A subtle and difficult-to-notice flaw can completely circumvent the security of an application – and most applications employ some level of cryptography.

The bare minimum review – confirming the correct application and configuration of SSL/TLS - is considered inherent to any audit. But if an application makes calls into cryptographic libraries, an audit must ensure it uses the correct parameters. Encryption should provide authentication, strong randomness should be used, and insecure mode selection should be avoided. Particular areas, such as overriding PKI certificate validation or any sort of uncommon or custom protocol, deserve rigorous review.

Pre-Audit Questions for a Project:

- Do you implement any cryptographic algorithms?
- Do you use a custom-designed protocol or data storage container?

Pre-Engagement Questions for Auditors:

- What types of cryptographic testing do you expect to conduct?

Post-Audit Review Criteria:

- Has the auditor mentioned testing specific well-known attacks on algorithm or protocol implementations, such as Bleichenbacher's attack or Padding Oracles[13]?

[12] A standard that describes how an application can be allowed to retrieve data from a server about a user.

[13] Bleichenbacher's attack and Padding Oracles are specific cryptographic attacks. A non-exhaustive list can be found at https://github.com/iSECPartners/LibTech-Auditing-Cheatsheet#appendix-b-cryptographic-attacks-cheat-sheet

4.1.11 Dependency Integration

All applications integrate some sort of dependency – they make use of code from other people, often run on a platform built by a different company, and rely on an Operating System developed by thousands of other people over decades. Just like an application being audited, these dependencies may contain flaws that allow an attacker to steal data, impersonate people, or take over a server.

Generally speaking, an audit will not include a review of standard libraries or operating system security measures (although these do lead to security incidents and should be covered under a Deployment Review). However, if an application integrates other third-party components, they should, at a minimum, be reviewed for published security vulnerabilities and insecure configuration or use. Depending on the application and use, it may be appropriate to specifically audit the third-party components in isolation.

Pre-Audit Questions for a Project:

- What are all of the dependencies you use in your application?
- Do any of them require specific versions or patches?
- Do you require specific configuration of other applications, and if so, what configuration settings?

Pre-Engagement Questions for Auditors:

- Will you be auditing the use of third-party components?
- What additional amount of time would be required to include auditing the third-party components themselves?

Post-Audit Review Criteria:

- Has the auditor identified any issues in third-party libraries?
- Has the auditor mentioned or demonstrated review of third-party library use or configuration?

4.1.12 Desktop Client

A desktop client is an application the user downloads and (usually) installs on their computer. It is a separate application from the web browser, and must be downloaded and installed on each computer it is to be used on.

A desktop client is often attacked through malicious files that are opened in the application: for example malicious Microsoft Word documents or video files. Applications that read complicated file formats are a concern. Other considerations are about making use of code signing, any permission structure available in the application's framework, and any update mechanism the application contains. Finally, just as with Unmanaged Languages, there are recommendations an auditor can make to provide defense in depth for desktop applications.

Pre-Audit Questions for a Project:

- Does a user have to download or install your application to use it?
- What Operating Systems does the application run on?
- Does the application contain an automatic update system that will download or install a new version automatically?

Pre-Engagement Questions for Auditors:

- Will you be auditing the desktop client on all supported Operating Systems, or only one?

Post-Audit Review Criteria:

- Has the auditor provided any defense in depth recommendations for the application, or explicitly mentioned that it makes use of all available security mechanisms?

4.1.13 Forensics Review

Computer Forensics is the examination of digital media with the aim of identifying, preserving, and analyzing data. A Forensics Review will look at data left on a device or accessible after it has left the owner's possession – this may be because it was stolen or seized by an adversary. The review will search for data such as private keys or contact lists that are not adequately protected, or other sensitive data that was written to application logs and accessible through forensic examination.

Certain tools may include a specific design goal to prevent forensics-based attacks from succeeding. Most commonly, this involves not logging or caching sensitive data to avoid forensic processes that target log files or deleted application files. However, other threat models include attackers who may capture process memory or even attackers who wish to learn if a user used a tool at some point in the past. In these situations, more specific testing is needed, and a thorough understanding and explanation of the limitations of the system should be given.

Pre-Audit Questions for a Project:

- Does your threat model include resisting a forensics examination? If so, under what conditions? (A 'stolen phone' scenario, an attacker who has access to the application while it is open, or others?)

Pre-Engagement Questions for Auditors:

- What threat model will you be testing under?

Post-Audit Review Criteria:

- Does the auditor decry that some threat model is impossible to counteract; or do they explain the difficulty of the situation, the best practice that can be used, and its limitations?

4.2 OTF Supported Projects

With categories of concern identified, we can apply them to Open Technology Fund projects. In some cases, OTF funds a specific component or application of a project, but we make an attempt to apply the categories to all major applications of all funded projects. This section lists OTF supported projects audited at the point of this report, the applications they work on, and the categories of concern relevant for that application.

4.2.1 Open Whisper Systems / https://whispersystems.org/

Open Whisper Systems makes mobile applications for Android and iPhone. Their major applications are listed individually.

RedPhone

RedPhone is an encrypted voice application that offers seamless upgrade to encrypted phone calls. The mobile application makes heavy use of cryptography and has a custom implementation of the ZRTP protocol. It also consists of a server component, including a custom proprietary daemon that is deployed on multiple nodes worldwide.

- User Interface / Experience
- Anonymity
- Deployment Review
- Mobile Application (Client-side)
- Web API / Mobile Application (Server-side)
- Server Daemon
- Cryptographic Review

TextSecure

TextSecure is an SMS application for mobile phones that encrypted messages sent between users who are both using the application. It contains no server component.

- User Interface / Experience
- Anonymity
- Mobile Application (Client-side)
- Cryptographic Review

4.2.2 Cryptocat / https://crypto.cat/

Cryptocat is an encrypted chat application, implemented as a JavaScript-based browser extension. It makes heavy use of cryptography, including a custom multi-party chat protocol. The project runs the primary Cryptocat service and makes use of the third-party application ejabberd[14]. Cryptocat is targeted specifically at non-technical users.

- Browser Extension
- User Interface / Experience
- Anonymity
- Deployment Review
- Web Application

[14] Explained in detail at https://github.com/cryptocat/cryptocat/wiki/Server-Deployment-Instructions

- Cryptographic Review
- Dependency Integration

4.2.3 JTor/Orchid / https://github.com/brl/JTor

JTor (being renamed Orchid) is an interoperable implementation of the Tor client protocol in Java. It is a code library, and is integrated in other applications; it cannot be used directly by a normal user.

- Anonymity
- Cryptographic Review
- Dependency Review

4.2.4 Mailvelope / http://www.mailvelope.com/

Mailvelope is a browser plugin to provide encrypted email in web-based email clients such as Gmail. It is written in JavaScript and makes heavy use of cryptography to implement Open-PGP-based email encryption and decryption. It also interacts directly with web sites such as Gmail to read an encrypted email, decrypt it, and display the decrypted content – all while preventing Gmail from access the decrypted message.

- Browser Extension
- User Interface / Experience
- Anonymity
- Web Application
- Cryptographic Review
- Dependency Integration

4.2.5 OONI / https://ooni.torproject.org/

OONI is a project that aims to track and measure network censorship worldwide. It consists of a server daemon that accepts connections and data from numerous known and unknown users. The client application, run by the users, detects network censorship and tampering, and transmits the results back to the server.

- Anonymity
- Deployment Review
- Server Daemon
- Web API / Mobile Application (Server-side)
- Desktop Client

4.2.6 SRLabs / https://srlabs.de/

SRLabs collects data about operating practices of mobile phone companies, and displays it in a web application. There is no capability to submit data nor are there user accounts or sensitive data at risk.

- Web Application

4.2.7 Commotion / https://commotionwireless.net/

Commotion is a project to provide mesh networking on consumer devices. The project customizes operating systems that run on inexpensive wireless routers and Android smartphones that perform mesh networking among the connected devices. This allows clients on the network to traverse multiple hops to reach a service or website available on the mesh or the Internet. It uses public protocols and applications to do the mesh routing and service advertisement and configures several applications within the operating system. It has a web application for configuration, as well as command line tools and a daemon used during setup.

- Unmanaged Language
- User Interface / Experience
- Anonymity
- Web Application
- Cryptographic Review
- Dependency Integration

4.2.8 GlobaLeaks / https://globaleaks.org/

GlobaLeaks is a web application that allows whistleblowers to security submit information. It consists of a web site and web API that collects anonymous reports from visitors, and makes them available to pre-approved users of the application. It has a heavy focus on strong anonymity for the submitters and a clear user interface that concisely explains risks.

- User Interface / Experience
- Anonymity
- Web Application
- Web API / Mobile Application (Server-side)
- Dependency Integration

4.2.9 LEAP / https://leap.se/

LEAP is a package of applications that provide services such as secure web browsing and email to a user. They integrate a number of large applications, include VPN and mail servers, and provide configurations for them. They have a desktop application for connecting to the VPN, a command line tool for the server, and a web application for management. They also have a desktop application that synchronizes a user's data between computers, making heavy use of cryptography and a custom protocol. They provide this suite of applications to service providers to run – they do not run any of the services themselves.

- User Interface / Experience
- Anonymity
- Web Application
- Server Daemon
- Web API
- Cryptographic Review
- Dependency Integration
- Desktop Client

4.2.10 Martus / https://www.martus.org/

Martus is a suite of tools produced by Benetech that are used to document and secure highly sensitive information, such as cases of human rights abuse. Its individual applications are listed.

Martus Server

The Martus Server is the server component that receives and stores data from both the Desktop and Android clients. Benetech will often run servers on behalf of Martus uers.

- Anonymity
- Server Daemon
- Web API / Mobile Application (Server-side)
- Cryptographic Review
- Deployment Review

Martus Client

The Martus Client is a desktop application that allows a user to read and write stories, and provide fine-grained permission to other Martus users.

- Anonymity
- User Interface / Experience
- Cryptographic Review
- Desktop Client
- Forensics Review

Martus Search Engine

The Martus Search Engine is a web application that allows access to bulletins that have been published publicly. Benetech often runs this service on behalf of users.

- User Interface / Experience
- Anonymity
- Deployment Review
- Web Application

Mobile Martus

Mobile Martus is an Android application that provides write-only access to a Martus Server. While it does interact with a server, we present it separately as the server also interacts with the desktop client.

- User Interface / Experience
- Anonymity
- Mobile Application (Clientside)
- Cryptographic Review
- Forensics Review

4.2.11 Guardian Project / https://guardianproject.info/

Guardian Project makes mobile applications for Android. Their major applications are listed individually.

Orbot

Orbot is a mobile implementation of Tor, and provides access to Tor to other mobile applications on the phone. It has a small user interface, and bundles a copy of Tor specifically modified for use on Android.

- User Interface / Experience
- Anonymity
- Mobile Application (Client-side)
- Cryptographic Review
- Dependency Integration
- Forensics Review

Orweb

Orweb is a privacy enhanced web browser for Android. It aims to have strong protections against tracking and disables unsafe features of the default browser.

- User Interface / Experience
- Anonymity
- Mobile Application (Client-side)
- Forensics Review

Gibberbot / ChatSecure

Gibberbot (being renamed ChatSecure) is a mobile application for Android that implements the OTR standard for encrypted chat conversations. It interoperates with other OTR implementations. It is only a mobile application; it requires you to connect it to an existing account on a chat provider.

- User Interface / Experience
- Anonymity
- Mobile Application (Client-side)
- Cryptographic Review
- Dependency Integration
- Forensics Review

ObscuraCam

ObscuraCam is a camera application for Android phones that performs metadata cleaning and uses facial recognition to blurs or censor faces.

- User Interface / Experience
- Anonymity
- Mobile Application (Client-side)
- Dependency Integration
- Forensics Review

Ostel

Ostel is a web application and server that allows users to sign up and create an account with which they can make encrypted phone calls. Guardian Project runs the primary deployment of the project.

- User Interface / Experience
- Anonymity
- Deployment Review
- Mobile Application (Client-side)
- Cryptographic Review
- Dependency Integration
- Forensics Review

5 Observations

Following the process of defining criteria, sixteen audits from three audit companies were reviewed. The results were entered into a matrix tagging each criteria and category according to the definitions specified in Methodology on page 8. The following observations were made.

5.1 Scope

Before an audit takes place, the funder, project, and auditor should agree on a scope of which components will be tested. In certain audits this scope was merely alluded to in the title, however, it appears scoping discussions were completed in all cases.

While communicating with projects for the preparation of this report, the projects expressed that they would like to see better explanations of what types of testing were performed for their application and what areas were covered. While most audit reports contained generic bullet points as to the types of testing performed, these did not satisfy project leaders as applying directly to their audits.

Most audits studied included application-specific statements about what components were tested. Some audits included examples of testing performed, but these examples did not seem to be comprehensive. A small minority included clear explanations of both components and types of testing performed.

5.2 Recommendations

General recommendations are aimed at reducing the possibility of introducing flaws in the future and improving defense in depth. Half the reports contain general recommendations for the project detailing architecture changes that can help avoid future problems, suggestions to clarify documentation or code for future audit and maintenance purposes, or other advice for projects. Half the reports do not include these recommendations.

Most reports identify specific items that could be categorized as 'Informational' or not directly exposing an exploitable condition. These items may lead to unintended behavior, introduce code complexity, or contradict documentation.

When reviewing the recommendations provided for individual issues, a significant number of reports do not include specific recommendations as it related to the application's code, but rather generic guidance on how to fix the class of vulnerability. Some other reports contain short recommendations that lack detail, while others contain considerable detail, references to source code, or multiple suggestions and alternatives.

5.3 Categories

Generally speaking, there were several commonalities in individual categories of concern, even across different auditors and applications. Several categories were consistently covered very well, including **Browser Extensions**, **Web Applications**, and **Mobile Applications (Client-side)**. All of these categories had sophisticated technical issues identified consistently and very rarely did an audit fail to indicate coverage. On the other end of the spectrum, several categories were found to be lacking. The audits rarely mentioned **Server Daemon**, **Anonymity,** or

Forensics Review. Four audits included strong anonymity issues, two audits included concerns relating to Denial of Service or daemon operation, and three audits identified specific forensics concerns; however, as a percentage, good coverage of any of these categories did not exceed one third of audits.

Scope played a large part in whether auditors assessed projects for concerns in **Deployment Review, User Interface/Experience, Web API/Mobile Application (Serverside),** and **Dependency Integration**. In no case (of four applications) did an auditor review the deployment of an application (mentioned in more detail in Specific Areas of Concern on page 29). One auditor identified concerns inside third-party dependencies and four other audits identified concerns relating to the use or configuration of dependencies; however, any indication of dependency coverage was missing in more than half of the audits. Not unexpectedly, no auditor performed user experience testing – five audits identified strong concerns relating to UI, but the remaining nine applying to the category made no mention of testing. APIs and the server-side component of mobile applications were excluded from scope six of the thirteen times they could have been reviewed, and when not excluded from scope only three audits had issues or indications it was reviewed at all.

The remaining category is **Cryptographic Review**. Custom protocols and use of low-level cryptographic APIs play a huge part in most OTF-funded applications, and cryptography is an extremely popular topic among auditors. About half of the audits present some evidence of cryptographic review, including issues relating to weak entropy and assertions of tests performed. Only one fifth of audits demonstrate sophisticated issues or include strong explanations of testing.

5.4 OTF Supported Projects

Seven projects received more than one audit, but no project received more than two. While two points of data do not support strong conclusions, some observations can be made.

GlobaLeaks is segmented between the GLClient[15] and GLBackend[16], the latter of which exposes an API. GlobaLeaks received two audits, one of which was a design review that did not review code; however, neither report made strong references to the API specifically. **Martus** contains a desktop client and server daemon; however, OTF has funded only the Mobile Martus Android client – the server was not reviewed.

In both audits **Whisper Systems** received, the RedPhone server deployment was considered out of scope (although the code was reviewed in one audit). The RedPhone client was in-scope for the other audit, and no security issues were identified. **Commotion Wireless** contains a significant number of dependencies, all of which were ruled out of scope for testing. Likewise, routing, Denial of Service, and user experience testing were also out of scope. In both audits **Cryptocat** received, the deployment was not tested, nor the server dependencies.

Finally, **Guardian Project** had three of its applications tested, but several application dependencies were not. The OSTel deployment and web application were not reviewed, and no general recommendations were given in either audit.

[15] https://github.com/globaleaks/GLClient
[16] https://github.com/globaleaks/GLBackend

6 Suggestions for the Future

A primary purpose of this report is to provide suggestions going forward for the Open Technology Fund and other funders to derive greater value from auditors, guide selection of future vendors, and identify gaps in coverage.

The summary of suggestions are:

- Create a pre-engagement checklist and/or worksheet for projects and auditors
- Ensure a technical scoping call before each audit
- Maintain and utilize Categories of Concern with both auditors and projects
- With the Categories of Concern, get explicit agreement between funder, projects, and auditor on scope pre-audit
- Ensure all auditors include general recommendations in the reports
- Ensure all issue recommendations span a greater scope than a single vulnerability identified
- Include application dependency, deployment, and user interface reviews in audits

6.1 Scope

The Categories of Concern were developed to aid funders in communicating with auditors and scoping security audits. A funder can provide these categories to an auditor along with their thoughts about which apply. By going through the descriptions, auditors can understand a funder's expectations and clearly communicate time and coverage estimates.

Through discussion with projects and review of audit reports, there is an unsatisfactory level of detail around what areas of a specific application were and were not reviewed, and the types of testing the auditor has performed. Generic testing bullet points are insufficient for this purpose. Some audits do include positive statements as to the areas and types of testing performed - going forward, requesting this specifically of auditors will hopefully fulfill the projects' requests.

It was noted that dependencies were omitted from some audits. It is unclear if there is a specific reason for this, but including them may help inform projects of poor quality applications they rely on. It is also worthwhile to inform projects that they should be active in the scoping discussion with an auditor and should prioritize types and areas of testing and other items of concern. Additional comments an auditor can provide, if asked, include informational findings and feedback on deploying an instance of the software or setting up a build environment.

6.2 Recommendations

Eight of sixteen audits did not contain any general recommendations for the projects. Although four audits were automated scanning of the applications' computer code, four manual audits also did not provide any suggestions. A primary outcome of a manual security audit is to make use of the auditor's expertise and breadth of exposure. Auditors should provide recommendations that will help a project reduce the possibility of vulnerabilities being introduced in the future, make the application more robust against a flaw becoming exploitable, and help the project remain forward-looking in an ever-changing landscape of Internet security. OTF should explicitly ask auditors to provide recommendations that span a greater scope than a single vulnerability identified.

Only six audits did not provide consistent and customized recommendations for individual issues. However, when speaking with projects, they appear to be split on the utility of them. For issues with subtleties surrounding the fix, a tailored recommendation is definitely useful. For other issues, at least two projects have indicated the auditor's time may be better spent elsewhere (although a third dissenting opinion was also registered). When scoping with the project, we suggest having the auditor and project team discuss how they would like to proceed, and to add this bullet point to a pre-engagement checklist.

6.3 Specific Areas of Concern

While reviewing the matrix of coverage for OTF-funded projects, certain deficiencies stood out.

Every project that runs a production deployment of their software for general usage did not have their deployment reviewed. This includes the primary **Cryptocat** server, the **Whisper Systems**' WhisperSwitches operated around the world for RedPhone use, **Guardian Project**'s OSTel deployment, and Benetech's **Martus** servers operated for users. As explained above, while flaws in the primary software can lead to exploitable conditions, it is often more common that networks and infrastructures are attacked through misconfigurations in alternate services running.

Additionally, most applications did not receive significant testing of critical dependencies. While many applications make use of common and popular dependencies such as GnuPG, other dependencies are less popular and warrant an examination that ranges from cursory to in-depth. These include dependencies used by **GlobaLeaks**[17], **Commotion**[18], **Cryptocat**[19], **Guardian Project**[20], and **Mailvelope**[21].

Nearly all applications expose a User Interface designed to be used by non-technical users. Some applications are geared more directly for these users (such as Whisper Systems, Cryptocat, and Mailvelope) while others are geared for the technical user who is not accustomed to building applications from scratch (such as Commotion). In all cases, however, there appear to be no usability studies or user experience testing. This testing can identify unexpected usability issues that significantly impact users' security and privacy that are unaccounted for in design or technical measures.

A select number of applications contain sophisticated protocols that may warrant further study from academic or engineering groups, as well as reviews of real-world deployment and performance characteristics. **Commotion** makes use of the OLSR protocol and the Serval service model. While designed in academic and industry settings, these protocols have not seen as

[17] https://raw.github.com/globaleaks/GLBackend/master/requirements.txt
[18] OLSRd, Serval, and significant others
[19] ejabberd, libyaml, libexpat, exmpp
[20] OTR4J, pgpdump, pure-python-otr, pyasn, and significant others
[21] Kendo UI and others

much review as other fundamental routing protocols. **Cryptocat** has been seeking outside help developing a multi-party OTR specification for some time[22].

6.4 Auditor Selection Criteria

For a funder seeking auditing firms, developing selection criteria can be a difficult task, particularly if they are unfamiliar with the security consulting marketplace. Consulting firms are used to bid processes; however, it is well within the rights of funders to ask for sample reports, references of past or current clients, and a technical scoping call between a project developer and auditor. Funders should make use of all of these tools before finalizing a contract.

The technical scoping call between a project developer and auditor should establish the auditor's technical competency. Auditors must be technically capable of performing an audit in the categories of concern for the application in question. Generally speaking, this should not present a problem, but for certain specialized areas such as browser extensions or unmanaged languages it is reasonable to ask for examples of work or talk through attack vectors. Additionally, cryptographic protocol review is an extreme specialty area that is difficult to staff. The ultimate goal is to achieve strong coverage of all categories of concern in an application, even if one vendor is unable to work on all components.

A sample report is a critical component to determining what an auditor can be expected to provide. It is within a funder's rights to ask for modifications to an auditor's report template to derive greater value. Examples of modifications OTF should request going forward precede this section - more explicit explanations of scope and explicit general recommendations being specific examples. Auditors must be willing to produce the outcomes requested by a funder. If specific or general recommendations are not supplied, or an auditor is unwilling to be explicit about the scope of testing, this reduces the value given to a project.

[22] https://github.com/cryptocat/mpOTR