SOFWERX

SecureDrop Security Analysis

FINAL REPORT

limitless innovation. no compromise.

Prepared for: Pablo Breuer
Innovation Officer

SOFWERX
1925 E 2nd Avenue, Suite 102
Tampa, FL 33605

November 2, 2018

Version:	Final
Prepared for:	Sofwerx
Date:	November 2, 2018

Table of Contents

Executive Summary

SofWerx engaged Leviathan Security in August of 2018 to perform a time-bound security review of Freedom of the Press Foundation's SecureDrop project. The assessment officially kicked off September 25, 2018 and was completed on October 5, 2018.

Our objectives were to review the SecureDrop web application encryption standards, and deployment and configuration recommendations for vulnerabilities that could lead to compromise, especially compromise of the confidentiality of sources. Testing was informed by the SofWerx planned deployment model as well as documentation and source code from the SecureDrop repository. We also reviewed issues reported by the community via GitHub and triaged them.

Observations

While we did not find any critical- or high-severity issues in SecureDrop's implementation and design, we did identify a number of areas that could be improved. Testing revealed an edge case that could prevent a journalist from receiving messages because the system reports message receipt before the message is actually sent. Another concern is that even though the SecureDrop system uses modern cryptographic standards to secure data, the cryptographic libraries are written in Python, which has no mechanism for removing secrets from memory after use. This issue applies only to those secrets not handled by GnuPG.

We also identified two aspects of SecureDrop documentation that are lacking. First, SecureDrop documentation needs to be updated to explain what to do to protect the identity of sources if the system is compromised. In addition, the OS software used by SecureDrop is reaching end-of-life. This issue is being tracked in GitHub, but the documentation does not cover how to update grsecurity or the custom kernel.

Finally, USB drives should not be used to transmit data either to or from an air-gapped system.

In summary, our review uncovered only medium- and low-severity findings.

Recommendations

Short-term recommendations:

- Provide documentation for dealing with a system compromise. This should include what to look for as well as what to do.
- Rewrite cryptography libraries in C so that secrets can be scrubbed from memory.
- Do not use USB drives on the Secure Viewing Station; only use CD-Rs.

Long-term recommendations:

- Roll out custom pip mirrors for secure deployment that contain only vetted packages, or add checksums for dependency packages and guard against installation of packages that do not have a checksum.

- Remediate remaining Git issues to address defense-in-depth.
- Upgrade to the newest LTS release of Ubuntu.

Vulnerability Classification

Impact	When we find a vulnerability, we assign it one of five categories of severity, essentially describing the potential impact if an attacker were to exploit it: Informational only – We found a condition that doesn't present a current threat, but it could create one in the future if certain changes are made. You'll probably want to fix it. Low – The vulnerability might allow an attacker to gain information that could be combined with other vulnerabilities to carry out further attacks. It may also allow an attacker to bypass auditing. However, it doesn't allow direct access to data or resources. Medium – The vulnerability may allow access to systems or servers. It may also allow access to confidential or sensitive data or a disruption in availability resulting in damage to reputation. No actual access to data was obtained. High – The vulnerability allows access directly to systems or servers. Confidentiality and integrity of data may be impacted, availability may be disrupted. Damage to reputation is likely. Critical – This is a high-impact vulnerability that may imminently allow an attacker to disrupt functionality, disclose data, resulting in significant reputational damage.
Skill Level to Exploit	When we find a vulnerability, we also assess how skilled an attacker must be to exploit it: Simple – Only a minimal understanding of the underlying technology is required. Tools and techniques for exploiting it can be easily found on the Internet. Moderate – An attacker must have a working knowledge of the technology and may also require the unwitting cooperation of a victim or target to carry out an attack. Advanced – Near-complete and superior understanding of the technology involved is required. Direct interaction with the victim or target may also be required.

Impact Rating (Weight)	Skill Level to Exploit Rating (Weight)		
	Advanced (1)	Moderate (2)	Simple (3)
Critical (4)	4	8	12
High (3)	3	6	9
Medium (2)	2	4	6
Low (1)	1	2	3

Severity	
Critical	10-12
High	7-9
Medium	4-6
Low	1-3

Vulnerability Index

Newly reported issues:

SEVERITY	TITLE	COMPONENT	ID
Low	Unhandled Exception on Some Inputs When Generating Sanitized Filename	Web Application	83860
Low	Add Documentation for Dealing with a Compromised System	System Configuration and Deployment	83889
Low	Python Has no SecureString Class	Encryption and Privacy	83857
Low	Unhandled Exception If Journalist Calls Reply When User Has No Key	Web Application	83859
Low	User Prompted That Message Is Received Before It Completes	Web Application	83858
Info	Limit Use of Flash Drives	System Configuration and Deployment	83870

Previously reported issues:

SEVERITY	TITLE	COMPONENT	REFERENCE
Medium	Login for Journalists Not Throttled	Web Application	https://github.com/freedomofpress/securedrop/issues/3566
Medium	Sessions Do Not Expire If Admin Changes Journalist Password	Web Application	https://github.com/freedomofpress/securedrop/issues/2300
Medium	Sources Should Disappear Over Time	Encryption & Privacy	https://github.com/freedomofpress/securedrop/issues/2068
Medium	SecureDrop Application GPG Should Have Expiration	Encryption & Privacy	https://github.com/freedomofpress/securedrop/issues/1139
Medium	Python Package Download Process Needs to Be Hardened	System Configuration & Deployment	https://github.com/freedomofpress/securedrop/issues/1617
Medium	Expiration Dates Not Added to Apt Repos	System Configuration & Deployment	https://github.com/freedomofpress/securedrop/issues/2436
Medium	Reconfigure SSH	System Configuration & Deployment	https://github.com/freedomofpress/securedrop/issues/1161
Low	Kernel Needs to Be Hardened (e.g., Modules Should Be Pruned)	System Configuration & Deployment	https://github.com/freedomofpress/securedrop/issues/2726
Low	Initial Submissions Stored in /tmp	Web Application	https://github.com/freedomofpress/securedrop/issues/3067
Low	SVG QR Codes Require Dropping Safety Settings	Web Application	https://github.com/freedomofpress/securedrop/issues/1574
Low	Remove JQuery	Web Application	https://github.com/freedomofpress/securedrop/issues/1233
Low	Improve Session Security	Web Application	https://github.com/freedomofpress/securedrop/issues/204
Low	Uploading Large Files Could De-anonymize Users	Encryption & Privacy	https://github.com/freedomofpress/securedrop/issues/986
Low	Remove Timestamps	Encryption & Privacy	https://github.com/freedomofpress/securedrop/issues/822

SEVERITY	TITLE	COMPONENT	REFERENCE
Low	HTML Pages Might Be Fingerprintable	Encryption & Privacy	https://github.com/freedomofpress/securedrop/issues/2566
Low	Sign OSSEC Alerts	System Configuration & Deployment	https://github.com/freedomofpress/securedrop/issues/966
Low	Explain Risks of Transferring Data by USB	System Configuration & Deployment	https://github.com/freedomofpress/securedrop/issues/3598

Observations & Analysis

Our analysis of the provided design documents is organized by topical area. For each area, we consider how the proposed system supports the security of said system with respect to the design goals.

Web Application

The web application allows a user to interact with SecureDrop, which is composed of three main interfaces: The Source Interface lets a user submit to journalists; the Journalist Interface lets a user download or delete submitted information as well as reply to sources; and the Admin interface manages Journalist users.

Threat Analysis

The web component contains the portions of the codebase that a user will interact with. This means that it needs to be free of vulnerabilities common to web applications, such as XSS, SQL injection, and command injection. Compromising the source or the journalist is equally damaging: Sources are assumed to need anonymity to protect themselves from reprisals due to their use of the system, and impersonation of the journalist could lead to opportunities for an attacker to suppress information or expose the source. Other threats involve authentication and authorization. Session management, access controls, and secrets management should all be handled using security best practices to protect the information being exchanged and the bidirectional trust relationship between sources and journalists.

Methodology

The web application is written using the Flask framework in Python. We verified that protections built into the framework are enabled and working. These include output sanitization with Jinja templating, CSRF tokens, and form and parameter validation.

We verified that areas of the application that process user input have appropriate error handling so that the application will not crash due to users' input of invalid data. Instances of insufficient error handling leading to denial of service are identified as findings later in this section. We also verified that the web application validates user inputs and does not simply assume they are well-formed.

Finally, we validated that authentication and authorization are appropriately handled. This included ensuring that secrets are not hardcoded and are not displayed to any user, administrators are the only users allowed to modify existing journalists, two-factor authentication is appropriately configured, and sessions are handled with security best practices.

Observed Trends

Security was clearly considered when writing SecureDrop. Frameworks and libraries utilized were configured to properly prevent common web vulnerabilities such as XSS, CSRF, and SQL injection. The issues we identified both via our own code review and via existing issues on GitHub fell mostly into two separate categories.

The issues identified via code review mostly describe unhandled exceptions and logic issues in the code. None of these are severe, but they may allow an attacker to impact the availability of the system for journalists or for sources. Additionally, one of the issues we identified describes a situation wherein a user is notified that their submission was received before it is added to the journalist's interface. As a result, a transient error could cause a message to be dropped even though the source believes it to have been uploaded, causing confusion, mistrust of the system, and squandering of the source's willingness to share.

Most of the GitHub issues for the web interface related to session management. Logins for a journalist are not throttled, meaning that someone could attempt to brute force the login. Further, upon changing a journalist's password from the admin portal, the existing session would not be invalidated from the database. Finally, the session management logic pertaining to sources may be subverted to leak some information about them.

Existing GitHub issues:

Title	Reference	Severity
LOGIN FOR JOURNALISTS NOT THROTTLED	https://github.com/freedomofpress/securedrop/issues/3566	**Medium**
SESSIONS DO NOT EXPIRE IF ADMIN CHANGES JOURNALIST PASSWORD	https://github.com/freedomofpress/securedrop/issues/2300	**Medium**
INITIAL SUBMISSIONS STORED IN /TMP	https://github.com/freedomofpress/securedrop/issues/3067	**Low**
SVG QR CODES REQUIRE DROPPING SAFETY SETTINGS	https://github.com/freedomofpress/securedrop/issues/1574	**Low**
REMOVE JQUERY	https://github.com/freedomofpress/securedrop/issues/1233	**Low**
IMPROVE SESSION SECURITY	https://github.com/freedomofpress/securedrop/issues/204	**Low**

Vulnerabilities

UNHANDLED EXCEPTION ON SOME INPUTS WHEN GENERATING SANITIZED FILENAME

ID	83860
Component	Web Application
Severity	**Low**
Impact / Skill Level	Low/Simple
Reference	n/a
Location	securedrop/store.py:L118

Observation

When a Source submits a file, it is possible for the filename sanitizer to return an empty string. This will throw an exception that is not handled due to being unable to create the file. This could DoS the application from the privilege of a Source.

When testing the gzip function with an empty string, it raised an exception that will be uncaught in this scenario.

securedrop/store.py:L118

```
from werkzeug.utils import secure_filename
[...]
L118
def save_file_submission(self, filesystem_id, count, journalist_filename,
                             filename, stream):
        sanitized_filename = secure_filename(filename)
[...]
        with SecureTemporaryFile("/tmp") as stf:  # nosec
            with gzip.GzipFile(filename=sanitized_filename,
                               mode='wb', fileobj=stf, mtime=0) as gzf:
```

Recommendation

Check the return value to ensure that it is not an empty string.

UNHANDLED EXCEPTION IF JOURNALIST CALLS REPLY WHEN USER HAS NO KEY

ID	83859
Component	Web Application
Severity	**Low**
Impact / Skill Level	Low/Advanced
Reference	n/a
Location	securedrop/journalist_app/main.py:L105 securedrop/crypto_util.py:L213

Observation

An unhandled CryptoException will occur if the journalist calls Reply() before the Source has a generated key. This could be used to DoS the application; however, it would require a custom request from the privilege level of a journalist.

securedrop/journalist_app/main.py:L105

```
        current_app.crypto_util.encrypt(
            form.message.data,
            [current_app.crypto_util.getkey(g.filesystem_id),
             config.JOURNALIST_KEY],
            output=current_app.storage.path(g.filesystem_id, filename),
        )
```

```
securedrop/crypto_util.py:L213
        out = self.gpg.encrypt(plaintext,
                               *fingerprints,
                               output=output,
                               always_trust=True,
                               armor=False)

        if out.ok:
            return out.data
        else:
            raise CryptoException(out.stderr)
```

Recommendation

Ensure that the user has a key before making this call, or add an exception handler.

USER PROMPTED THAT MESSAGE IS RECEIVED BEFORE IT COMPLETES

ID	83858
Component	Web Application
Severity	**Low**
Impact / Skill Level	Low/Advanced
Reference	n/a
Location	securedrop/source_app/main.py:L166

Observation

After a Source submits a message or file, they are prompted that the message was successfully received. However, although the server has received it, the submission is never inserted in the database, and the Source has not been marked as non-pending before the receive message occurs. This could cause a journalist to miss a message that a Source believes was successfully sent.
securedrop/source_app/main.py:L166

```
html_contents = gettext('Thanks! We received your message.')
[...]
for fname in fnames:
    submission = Submission(g.source, fname)
    db.session.add(submission)
if g.source.pending:
    g.source.pending = False
```

Recommendation

Either place this message further into the function or create a script that will regularly ensure no files exist on the system that have not been added into the database.

Encryption and Privacy

The main promise behind SecureDrop is that it can provide a way to anonymously and securely upload files to a journalist. In order to maintain user trust, the application needs to provide strong privacy guarantees for sources who submit, and data needs to be encrypted at rest in case a journalist's machine is compromised.

Threat Analysis

Threats in this component fall into two areas: breaking the encryption of data, and leaking information about the source who uploads information to a journalist. Breaking the encryption of the data could involve either man-in-the-middle attacks on the data transfer or poor encryption on files that are stored on the journalist's machine at rest. This would allow an attacker to be able to read sensitive data that is shared. Given the nature of the content that is stored on this machine, leaking any of it would be extremely severe. De-anonymizing the source is a risk as there are potential legal repercussions for uploading information to this service. If a source experienced repercussions as a result of using SecureDrop, it would make people wary of using the service again.

Methodology

The methodology here was to trace the dataflow in the source code for all user tainted information. The Source only ever interacts with the source interface in the web application. This means we can analyze every request through the routes implemented.

Encryption involves ensuring that best practices are used, as well as relying on a paranoid threat model for all phases of the data lifecycle. The first step is to ensure that modern settings are used in cryptographic algorithms. Additionally, keys for the encryption should not be easy to find, or else there is little reason to encrypt. Finally, there should be no way to trick the code into decrypting the information for an unprivileged user.

Observed Trends

Our review showed the encryption in the application is used appropriately and is well-implemented; we did not identify any trivial attacks. In the appendix, we have included a flowchart depicting the data transfer between the journalist and a source and the relevant parts of the cryptosystem design. Appropriate key sizes and encryption algorithms are used.

Notwithstanding this, because SecureDrop is written in Python, the tenancy of secrets and sensitive data in memory is a problem; the language runtime does not provide a means to scrub them. A forensic search of the journalist's machine or direct access to memory by an attacker could reveal those secrets or sensitive data, including encryption passwords. SecureDrop partially treats this risk by recommending a reboot of the computer running it every 24 hours, which limits the effect of a single retrospective search but not of a persistent compromise. If the machine were seized and forensically searched after it had been used following a reboot, at least some information would leak in spite of the reboot recommendation. The

leaked information would include the passphrase for the source's key, valid codenames for sources, partial contents of leaked files, and the AES key used to decrypt files present in /tmp. The codename could be used to decrypt journalist replies.

Beyond this weakness to forensic searches, we note a number of potential ways to de-anonymize a source. Traffic analysis remains a problem, in that a party interested in identifying the use of SecureDrop could check for large file uploads or fingerprint the web pages that render the source interface if they had control over the source's network connection. In a related issue, SecureDrop uses timestamps that could be used to correlate SecureDrop activity with other events in an investigation.

Although SecureDrop does not currently implement a GPG key expiration and rotation scheme (as described in GitHub issue #1139), we suggest that doing so would limit the scope of impact in the event that a key is revealed to an attacker. Although previously captured data cannot be protected from attackers who later steal the key, a key rotation scheme would limit the number of documents that could be so decrypted and would prevent the key from compromising documents indefinitely into the future. Keys could be disclosed accidentally by journalists, and sources may not rotate the encryption keys they use in this eventuality unless the rotation happens at regular intervals.

Existing GitHub vulnerabilities:

TITLE	REFERENCE	SEVERITY
SOURCES SHOULD DISAPPEAR OVER TIME	https://github.com/freedomofpress/securedrop/issues/2068	Medium
SECUREDROP APPLICATION GPG SHOULD HAVE EXPIRATION	https://github.com/freedomofpress/securedrop/issues/1139	Medium
UPLOADING LARGE FILES COULD DE-ANONYMIZE USERS	https://github.com/freedomofpress/securedrop/issues/986	Low
REMOVE TIMESTAMPS	https://github.com/freedomofpress/securedrop/issues/822	Low
HTML PAGES MIGHT BE FINGERPRINTABLE	https://github.com/freedomofpress/securedrop/issues/2566	Low

Vulnerabilities

PYTHON HAS NO SECURESTRING CLASS

ID	83857
Component	Encryption and Privacy
Severity	**Low**
Impact / Skill Level	Low/Advanced
Reference	https://www.sjoerdlangkemper.nl/2016/06/09/clearing-memory-in-python/
Location	N/A

Observation

When handling sensitive strings, such as the codename or passphrase for keys, Python stores them in string objects. These are immutable in Python, which means that overwriting them is impossible. This makes it easier for an attacker to leak sensitive information if they're able to read memory.

Recommendation

Rewrite crypto_utils and other potentially sensitive functions in a non-Python language that contains lower level memory handling. Some instances will be impossible to fully fix, such as the receiving of requests over the network. However, minimizing the number of instances of the string in memory will be useful.

System Configuration and Deployment

The SecureDrop project provides documentation and installation scripts for deployment. Following the default guidelines should produce a system with minimal attack surface and appropriate segmentation that is resistant to compromise.

The underlying platform hosting SecureDrop applications should be kept updated to ensure that publicly known or fixed security vulnerabilities in dependencies are promptly mitigated. Appropriate access controls should be in place to limit the impact of an attacker successfully exploiting the system. Documentation should be in place for maintaining the system and dealing with compromises.

Threat Analysis

SecureDrop's deployment is critical to the anonymity of sources. An attacker could leverage a poor configuration to gain entry, escalate privileges, or evade detection; any of these could contribute to the compromise of a source and reduce confidence in SecureDrop.

If vulnerable network services are left exposed after installation, or if insecure update channels are used, an attacker could compromise the system. In addition, insecure update channels could give an attacker a foothold on the system. That foothold could be used to disclose confidential information from the system, particularly if that information is not protected by data privilege segmentation at the operating system level.

Finally, administrators must also have the means to detect a compromise and know exactly what to do in case the system or encryption keys are compromised.

Methodology

We approached configuration and deployment by reviewing system design and documentation, and reading code related to installation and maintenance. We reviewed the overall architecture for design decisions that could impact security, including software choices and methods for transferring data. We reviewed deployment guidelines in the documentation to make sure that administrators have enough information to deploy a secure instance of SecureDrop. We reviewed Ansible playbooks, OSSEC rules, and AppArmor profiles for anything that exposes SecureDrop to unnecessary risk.

Observed Trends

We found that if SecureDrop is deployed according to the documentation, using the described installation scripts and configuration, the deployment will be secure. SecureDrop is built on top of Ubuntu 14.04 LTS, which will only be supported through 2019. Further, SecureDrop developers provided a grsecurity hardened version of the Ubuntu 14.4.5 kernel, which mitigates several exploitation vectors. It is imperative that SecureDrop be updated with the Linux kernel and grsecurity patch set; this is a maintainability issue because grsecurity no longer makes timely updates available to the open-source community as of early 2017. Because the LTS release of Ubuntu in use is about to become end-of-life, SecureDrop will have to be tested against a newer version (e.g., 18.04), and this upgrade process will likely have to be repeated

again in 2023. Existing installations will have to install the update. This is being tracked in GitHub issue #3204, but that issue does not discuss grsecurity.

If applications are exploited, Docker and AppArmor limit an attacker's ability to further compromise the system by restricting file system access for Tor, Apache, and web applications. This control prevents an attacker who can compromise one of those services from gaining a persistent compromise of the anonymity and confidentiality of SecureDrop via means such as privilege escalation, though such a compromise would still reduce the security of the system.

The documentation provides a detailed guide for setting up a pfSense hardware firewall with limited internet and intranet connectivity. OSSEC is also used to help administrators monitor the integrity of the system. This will help administrators know when to invoke incident response if OSSEC detects system tampering.

We recommend adding documentation describing appropriate incident response plans in the event of a compromise. Administrators must be made aware of when new keys need to be generated in order to protect the identity of sources.

The documentation currently recommends using USB drives to transfer data from the journalist workstation to the Secure Viewing Station. We suggest this recommendation be changed to use CD-R media, since USB drives can provide for a bidirectional data flow, and malicious software could use them as a vector to exfiltrate data from the Secure Viewing Station in violation of the Bell-LaPadula Confidentiality Model; a CD-R would prevent a write operation disclosing encrypted data down to a lower integrity level than the Secure Viewing Station.

Existing GitHub vulnerabilities:

TITLE	REFERENCE	SEVERITY
PYTHON PACKAGE DOWNLOAD PROCESS NEEDS TO BE HARDENED	https://github.com/freedomofpress/securedrop/issues/1617	Medium
RECONFIGURE SSH	https://github.com/freedomofpress/securedrop/issues/1161	Medium
EXPIRATION DATES NOT ADDED TO APT REPOS	https://github.com/freedomofpress/securedrop/issues/2436	Medium
KERNEL NEEDS TO BE HARDENED (E.G. MODULES SHOULD BE PRUNED)	https://github.com/freedomofpress/securedrop/issues/2726	Low
SIGN OSSEC ALERTS	https://github.com/freedomofpress/securedrop/issues/966	Low

We investigated GitHub issue #3286 ("apt is over HTTP instead of HTTPS") and found that it should not be a problem because apt verifies package digests from a signed manifest; this security control has been widely discussed by the Linux community in the past. GitHub issue #1617, however, does have some validity; although the requirements.txt file used to pull in Python packages specifies acceptable SHA256

digests, we suspect it does not have digests for the listed packages' dependencies, and so the dependency code is effectively untrusted.

Vulnerabilities

ADD DOCUMENTATION FOR DEALING WITH A COMPROMISED SYSTEM

ID	83889
Component	System Configuration and Deployment
Severity	**Low**
Impact / Skill Level	Low/Advanced
Reference	n/a
Location	docs.securedrop.org

Observation

SecureDrop may contain unknown vulnerabilities. Documentation should provide explicit remediation guidelines regarding what to do if it is suspected that the system has been compromised.

Recommendation

We recommend completely wiping all SecureDrop systems and generating new keys for all parties. New documents should not be generated with potentially leaked encryption keys. Old documents, including those found on backup drives, should be re-encrypted with a new key.

LIMIT USE OF FLASH DRIVES

ID	83870
Component	System Configuration and Deployment
Severity	**Info**
Impact / Skill Level	Informational/Advanced
Reference	https://github.com/freedomofpress/securedrop/issues/3598
Location	docs.securedrop.org

Observation

The Secure Viewing Station's hardware should be physically isolated from any other network. Using USB drives violates this standard and introduces unnecessary attack vectors.

SecureDrop documentation recommends using USB flash drives or CDs to transmit data to and from the Secure Viewing Station. Malicious USB drive firmware could be used to exfiltrate data. This poses an unnecessary amount of risk to Sources since exfiltrated data could be used to identify them.

Recommendation

Only use CD-Rs to move data to the Secure Viewing Station. If using flash drives is necessary, use a high-quality write blocker to prevent exfiltration.

Future Work & Recommendations

Develop Guidelines for Dealing with Compromise

SecureDrop's documentation should clearly state how to deal with a compromise in order to minimize the impact on sources. Having procedures in place will signal to sources that their anonymity is important.

The documentation would require research to correctly identify when a system should be considered compromised. The documentation should minimize false positives and downtime while ensuring that the threat has been appropriately mitigated.

Time Estimate: 10 Days

Task	Days to Fix
Research	4
Writing: Identifying Compromise	3
Writing: Dealing with Compromise	3

Rewrite Cryptographic Libraries

Rewriting the cryptographic libraries would provide greater privacy of secrets. All relevant functions need to be rewritten and tested thoroughly.

Time Estimate: 18 Days

Task	Days to Fix
Rewrite crypto_util Library (1 class, 13 functions)	8
Write Tests	4
Test Modifications	4
Document Changes	2

Fix AuthN/AuthZ Issues

There are several GitHub issues involving throttling logins, expiring sessions on password reset, and preventing sensitive information from being leaked on the source session.

Time Estimate: 20 Days

Task	Days to Fix
Write Fixes	10
Write Regression Tests	4
Test Modifications	4
Document Changes	2

Fix Denial of Service Bug

These are small fixes that should not take long to implement. They require time to write and test each commit.

Time Estimate: 8 Days

Task	Days to Fix
Write Fixes	2
Write Regression Tests	2
Test Modifications	2
Document Changes	2

Total Estimate: Maximum of 56 Days of Effort

Appendix A – Technical Services

Leviathan's Technical Services group brings deep technical knowledge to your security needs. Our portfolio of services includes software and hardware evaluation, penetration testing, red team testing, incident response, and reverse engineering. Our goal is to provide your organization with the security expertise necessary to realize your goals.

SOFTWARE EVALUATION We provide assessments of application, system, and mobile code, drawing on our employees' decades of experience in developing and securing a wide variety of applications. Our work includes design and architecture reviews, data flow and threat modeling, and code analysis with targeted fuzzing to find exploitable issues.

HARDWARE EVALUATION We evaluate new hardware devices ranging from novel microprocessor designs, to embedded systems, to mobile devices, to consumer-facing end products, to core networking equipment that powers Internet backbones.

PENETRATION & RED TEAM TESTING We perform high-end penetration tests that mimic the work of sophisticated attackers. We follow a formal penetration testing methodology that emphasizes repeatable, actionable results that give your team a sense of the overall security posture of your organization.

SOURCE CODE-ASSISTED SECURITY EVALUATIONS We conduct security evaluations and penetration tests based on our code-assisted methodology, allowing us to find deeper vulnerabilities, logic flaws, and fuzzing targets than a black-box test would reveal. This gives your team a stronger assurance that the significant security-impacting flaws have been found and corrected.

INCIDENT RESPONSE & FORENSICS We respond to security incidents for our customers, including forensics, malware analysis, root cause analysis, and recommendations for how to prevent similar incidents in the future.

REVERSE ENGINEERING We assist clients with reverse engineering efforts not associated with malware or incident response. We also provide expertise in investigations and litigation by acting as experts in cases of suspected intellectual property theft.

Appendix B – Risk and Advisory Services

Leviathan's Retained Services group is a supplement to an organization's security and risk management capability. We offer a pragmatic information security approach that respects our clients' appetites for security process and program work. We provide access to industry leading experts with a broad set of security and risk management skills, which gives our clients the ability to have deep technical knowledge, security leadership, and incident response capabilities when they are needed.

INFORMATION SECURITY STRATEGY DEVELOPMENT We partner with boards, directors, and senior executives to shape your enterprise's overall approach to meeting information security requirements consistently across an entire organization.

ENTERPRISE RISK ASSESSMENT We develop an information asset-centric view of an organization's risk that provides insight to your organization's Enterprise Risk Management capability. This service can be leveraged with annual updates, to account for your organization's changing operations, needs, and priorities.

PRIVACY & SECURITY PROGRAM EVALUATION We evaluate your organization's existing security program to give you information on compliance with external standards, such as ISO 27000 series, NIST CSF, HIPAA, or PCI-DSS among others. This is often most useful before a compliance event or audit and helps to drive the next phase of growth for your Security and Risk Management programs.

VENDOR RISK ASSESSMENT We assess the risk that prospective vendors bring to your organization. Our assessment framework is compatible with legislative, regulatory, and industry requirements, and helps you to make informed decisions about which vendors to hire, and when to reassess them to ensure your ongoing supply chain security.

NATIONAL & INTERNATIONAL SECURITY POLICY In 2014, we launched a public policy research and analysis service that examines the business implications of privacy and security laws and regulations worldwide. We provide an independent view of macro-scale issues related to the impact of globalization on information assets.

M&A/INVESTMENT SECURITY DUE DILIGENCE We evaluate the cybersecurity risk associated with a prospective investment or acquisition and find critical security issues before they derail a deal.

LAW FIRM SECURITY SERVICES We work with law firms as advisors, to address security incidents and proactively work to protect client confidences, defend privileged information, and ensure that conflicts do not compromise client positions. We also work in partnership with law firms to respond to their clients' security needs, including in the role of office and testifying expert witnesses.

SAAS AND CLOUD INITIATIVE EVALUATION We give objective reviews of the realistic threats your organization faces both by moving to cloud solutions and by using non-cloud infrastructure. Our employees have written or contributed to many of the major industry standards around cloud security, which allows their expertise to inform your decision-making processes.