iSECpartners
part of nccgroup

Freedom of the Press SecureDrop

Configuration Review & Application Penetration Test

Prepared for:

FREEDOM
OF THE PRESS
FOUNDATION

Prepared by:

Raphael Salas — iSEC Technical Lead

Valentin Leon — Senior Security Engineer

While precautions have been taken in the preparation of this document, iSEC Partners, Inc, the publisher, and the author(s) assume no responsibility for errors, omissions, or for damages resulting from the use of the information contained herein. Use of iSEC Partners services does not guarantee the security of a system, or that computer intrusions will not occur.

Document Change Log

Version	Date	Change
1.0	2015-06-05	Release for the Freedom of the Press Foundation.
1.1	2015-07-06	Public release.
1.2	2015-12-08	Unredact the OSSEC bug after conversations with the OSSEC team.

Table of Contents

1 Executive Summary 5
1.1 iSEC Risk Summary 6
1.2 Project Summary 7
1.3 Findings Summary 7
1.4 Recommendations Summary 8
2 Engagement Structure 10
2.1 Internal and External Teams 10
2.2 Project Goals and Scope 11
3 Detailed Findings 12
3.1 Classifications 12
3.2 Vulnerabilities 14
3.3 Detailed Vulnerability List 15
Appendices 25
A Additional Hardening Recommendations 25
A.1 Grsecurity 25
A.2 Tails Concerns 25
A.3 Airgap Hardening 26

1 Executive Summary

FREEDOM
OF THE PRESS
FOUNDATION

Application Summary	
Application Name	SecureDrop
Application Version	0.3.3
Application Type	Web Application
Platform	Python / Flask / Ubuntu 14.04.2 LTS / Linux 3.14.36 (grsec)

Engagement Summary	
Dates	May 24, 2015 – June 5, 2015
Consultants Engaged	2
Total Engagement Effort	4 person weeks
Engagement Type	Configuration Review & Application Penetration Test
Testing Methodology	White Box

Vulnerability Summary	
Total High severity issues	0
Total Medium severity issues	1
Total Low severity issues	2
Total Informational severity issues	7

Total vulnerabilities identified: 10

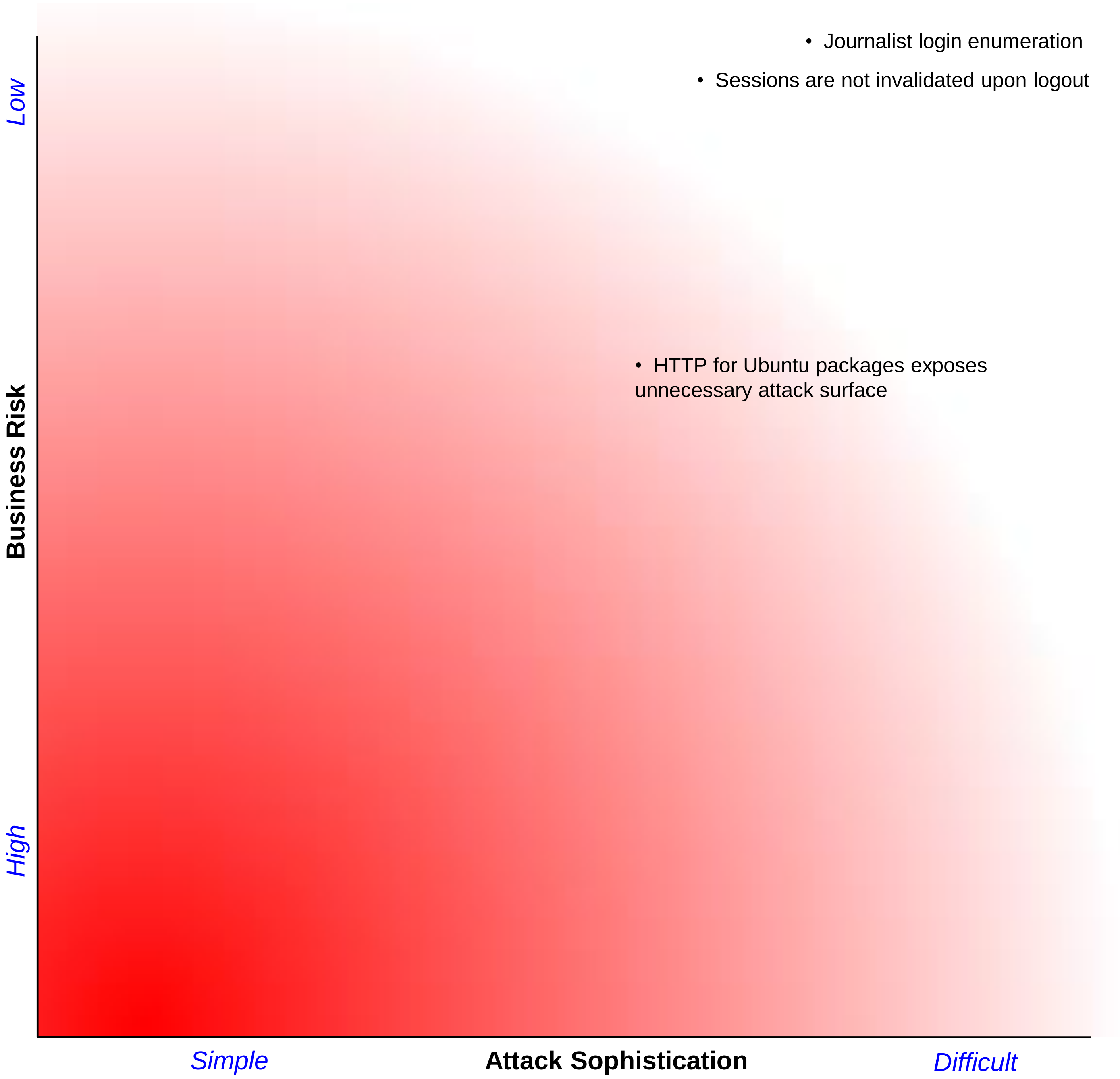
See section 3.1 on page 12 for descriptions of these classifications.

Category Breakdown:

Access Controls	0
Auditing and Logging	1 ■
Authentication	1 ■
Configuration	4 ■■■■
Cryptography	1 ■
Data Exposure	1 ■
Data Validation	0
Denial of Service	1 ■
Error Reporting	0
Patching	0
Session Management	0
Timing	1 ■

1.1 iSEC Risk Summary

The iSEC Partners Risk Summary chart evaluates vulnerabilities according to risk. The impact of the vulnerability increases towards the bottom of the chart. The sophistication required for an attacker to find and exploit the flaw decreases towards the left of the chart. The closer a vulnerability is to the chart origin, the greater the risk.



1.2 Project Summary

The Open Technology Fund (OTF) engaged iSEC Partners (iSEC) to perform a source-assisted security review of the Freedom of the Press' SecureDrop, a whistleblower submission system. The review was performed by two consultants during the weeks of May 25th and June 1st. An additional consultant assisted the review on a pro-bono basis the week of June 1st.

The goal of this engagement was to review the changes since the 0.3pre version following the previous year's security audit.[1] Particular items called out for review included the rewrite of the journalist authentication backend and hardening, including the transition from shell scripts to Ansible playbooks and the use of a Grsecurity Linux kernel.

Freedom of the Press (FPF) technical contacts met with the iSEC team in iSEC's New York office to discuss changes since the last review in the week before the engagement. Additionally, an FPF technical contact met with the iSEC team to assist in setting up virtual environments for testing. The FPF team made themselves available to answer questions, resolve issues, and engaging in ongoing discussions regarding recommendations and best practices, contributing greatly to the success of this engagement.

In line with the changes since 0.3pre, iSEC focused on traditional web application testing of the new journalist interface, which has an administrator interface and uses two-factor authentication for users. iSEC also performed a review of the environment in which SecureDrop runs on, focusing on covering the firewall configuration, a host review of the Grsecurity configuration, and other hardening procedures such as the SecureTempFile implementation that prevents plaintext submission data from being buffered to disk.

In accordance with SecureDrop's threat model,[2] attacks on the hardware on which SecureDrop runs on as well as attacks on the Tor network (e.g. side-channel/deanonymization attacks) were considered out of scope.

1.3 Findings Summary

SecureDrop's extensive hardening places it well above industry standards due to its emphasis on security and privacy. Overall, the newly introduced components do not add to the already limited attack surface, as the SecureDrop architecture remains the same. The addition of a Grsecurity-enabled kernel significantly raises the bar for attackers who manage to successfully exploit the application server via various memory corruption style attacks. Additionally, all interfaces except the one used by sources are only available over authenticated Tor hidden services.[3] These factors, combined with SecureDrop's existing defense in depth measures, make traditional forms of server compromise significantly difficult.

In the document interface, iSEC found that session identifiers are not invalidated after logout; this is a side effect of Flask's client-side sessions and not introduced by the SecureDrop code itself. However, the use of Tor Browser Bundle with an authenticated Tor hidden service to access this interface means it's highly unlikely for the necessary conditions to occur that would allow a session hijacking attack.

In this report, iSEC has included several findings affecting the new journalist authentication backend. Although this backend would be difficult to exploit, an attacker may be able to enumerate the journalist usernames as well as create a denial of service condition by submitting abnormally large passwords in

[1] https://freedom.press/blog/2015/03/announcing-securedrop-0.3
[2] https://github.com/freedomofpress/securedrop/blob/0.3.3/docs/threat_model.md
[3] The source interface is also available via a Tor hidden service; however, this interface is not an authenticated hidden service.

the login form. Some debug statements were also left in the production code, potentially leaking plaintext passwords to logs.

Throughout the SecureDrop environment, iSEC noted configuration patterns related to iptables rules which are likely to introduce unintended effects, and an unrelated instance of an iptables rule being unintentionally overridden. iSEC additionally noted several configuration decisions that do not present security risks themselves, but can aid in fingerprinting and discovery of SecureDrop-related resources.

Although unrelated to SecureDrop, during this assessment iSEC discovered that the OSSEC secure remote protocol, used for forwarding events, may be susceptible to cryptographic attacks due to static initialization vectors and lack of authenticated encryption. This does not seem to be a known or previously reported issue.

As the resiliency of the SecureDrop application against targeted attacks increases, gaps in the software ecosystem become more apparent. Particularly, the Tails live distribution, which is an important part of the security process for a SecureDrop deployment, becomes a more accessible and likely target for compromise. While Tails does perform some hardening and maintains control over software shipped, it does not employ the same security mechanisms which make SecureDrop difficult to exploit successfully. Namely, the lack of Grsecurity means that if a malicious source submitted a document which exploited the compression software during document decompression, or the default Tails application for the submitted document, this exploit – chained with a data ex-filtration technique or another infection spreading mechanism – could allow attacks outside the airgapped viewing station via infected media or compromise of submitted documents. Of course, this attack would require extensive development effort and knowledge, but is within the reach of a dedicated attacker. In these cases, Grsecurity on Tails can significantly increase the difficulty and reduce the reliability of these attacks.

1.4 Recommendations Summary

This summary provides a high-level overview of recommendations designed to increase the security posture of the application.

SecureDrop is on a good path regarding security posture; while there are several hardening recommendations listed, the majority are simple improvements that remediate one-time errors and maintain the integrity of future hardening by preventing regressions. iSEC recommends Freedom of the Press continues this hardening and explores additional tuning specific to the SecureDrop environment. For instance, additional non-standard Grsecurity options that apply to the application, such as those mentioned in Appendix A.1 on page 25 will significantly improve defense in depth.

A longer-term goal should be to extend this hardening across the SecureDrop ecosystem. Freedom of the Press should investigate hardening the secure viewing process, as possible ways of compromise would involve the submission of a malicious payload. Freedom of the Press relies on the security of Tails[4] for the journalist viewing station, and while this host is supposed to be air-gapped, iSEC noticed that Tails is lacking security features such as those introduced by Grsecurity, which is enabled on SecureDrop servers. iSEC recommends FPF and OTF to work with Tails in order to harden this system. For more information, see Appendix A.2 on page 25.

[4] https://tails.boum.org/

Short Term

Short term recommendations are meant to be relatively easy actions to execute, such as configuration changes or file deletions that resolve security vulnerabilities. These may also include more difficult actions that should be taken immediately to resolve high-risk vulnerabilities. This area is a summary of short term recommendations; additional recommendations can be found in the vulnerabilities section.

Throttle invalid login attempts. Throttle both valid and invalid journalist login attempts in the journalist web interface. Fix the iptables rules to restore the throttle on the SSH connections.

Implement a limit on the length of user supplied input. Ensure all user input is limited to context-appropriate maximum size limits. This is especially important in cases where a cryptographic function will be applied to that data.

Tighten AppArmor profiles. Ensure that AppArmor profiles do not give unnecessary capabilities, following the principle of least privilege.

Consider denying new USB connections in the monitoring and application servers. There is no need for these servers to have additional USB devices, and therefore new devices should be limited.

Long Term

Long term recommendations are more complex and systematic changes that should be taken to secure the system. These may include significant changes to the architecture or code and may therefore require in-depth planning, complex testing, significant development time, or changes to the user experience that require retraining.

Limit identification of SecureDrop servers. Provide the FPF repository as a Tor hidden service to minimize risks associated with plaintext HTTP as well as easily accessible fingerprinting vectors.

Consider modifying the session management to include a server-side component. An additional component can be added to the session management implementation which allows session invalidation when users log out. This prevents session identifiers from relying solely on cookie deletion. It also allows administrative functions, such as viewing how many sessions are currently open, and invalidation by either the administrator or the user.

Investigate additional hardening for the secure viewing process and continue hardening efforts. The current secure viewing process presents a more likely alternative for attack. Increasing the difficulty of such attacks will improve the overall security posture.

2 Engagement Structure

2.1 Internal and External Teams

The iSEC team has the following primary members:

- Raphael Salas — Security Engineer
  raphael@isecpartners.com
- Valentin Leon — Security Engineer
  valentin@isecpartners.com
- Tim Newsham — Security Engineer
  tim@isecpartners.com

The Freedom of the Press team has the following primary members:

- Garrett Robinson — Freedom of the Press
  garrett@freedom.press
- James Dolan — Freedom of the Press
  james@freedom.press
- Harlo Holmes — Freedom of the Press
  harlo@freedom.press
- Conor Schaefer — Freedom of the Press
  conor@freedom.press

2.2 Project Goals and Scope

The focus of this engagement was to identify vulnerabilities in the SecureDrop code base and interactions with the application stack, with particular attention to the changes introduced since the 0.3pre release. The secondary goal was to provide additional defense-in-depth recommendations to further remove identified gaps in the application ecosystem. This assessment was structured as "best effort" within the given time frame.

At a high level, the changes to the code from 0.3pre to 0.3.3 include:

- Web Application
  - A rewrite of the journalist interface, complete with 2-factor authentication
  - Implementation of a secure temporary file to encrypt uploads with an ephemeral AES key to prevent plaintext data from being stored on disk while buffering.
- Environment
  - Grsecurity Linux kernel is now default
  - Installation and hardening scripts are now done via Ansible

While reviewing the SecureDrop environment, iSEC focused on the following elements:

- OSSEC
- AppArmor Configuration
- Grsecurity Linux kernel configuration
- SSH Configuration
- Apache Configuration
- iptables Configuration

3 Detailed Findings

3.1 Classifications

The following section describes the classes, severities, and exploitation difficulty rating assigned to each issue that iSEC identified.

Vulnerability Classes	
Class	**Description**
Access Controls	Related to authorization of users, and assessment of rights
Auditing and Logging	Related to auditing of actions, or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices, or software
Cryptography	Related to mathematical protections for data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to the race conditions, locking, or order of operations

Severity Categories	
Severity	**Description**
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth
Undetermined	The extent of the risk was not determined during this engagement
Low	The risk is relatively small, or is not a risk the customer has indicated is important
Medium	Individual user's information is at risk, exploitation would be bad for client's reputation, of moderate financial impact, possible legal implications for client
High	Large numbers of users, very bad for client's reputation or serious legal implications.

Difficulty Levels

Difficulty	Description
Undetermined	The difficulty of exploit was not determined during this engagement
Low	Commonly exploited, public tools exist or can be scripted that exploit this flaw
Medium	Attackers must write an exploit, or need an in depth knowledge of a complex system
High	The attacker must have privileged insider access to the system, may need to know extremely complex technical details or must discover other weaknesses in order to exploit this issue

3.2 Vulnerabilities

The following table is a summary of vulnerabilities identified by iSEC. Subsequent pages of this report detail each of the vulnerabilities, along with short and long term remediation advice.

Vulnerability	Class	Severity
1. HTTP for Ubuntu packages results in unnecessary exposure	Data Exposure	Medium
2. Sessions are not invalidated upon logout	Authentication	Low
3. Journalist login enumeration	Timing	Low
4. SSH throttle ineffective	Configuration	Informational
5. Plaintext passwords potentially leaked	Auditing and Logging	Informational
6. IPTables default “ALLOW” policy is dangerous	Configuration	Informational
7. OSSEC remote protocol uses weak cryptography	Cryptography	Informational
8. Denial of Service on journalist login	Denial of Service	Informational
9. AppArmor profile grants unnecessary capabilities	Configuration	Informational
10. OSSEC alert fingerprinting	Configuration	Informational

3.3 Detailed Vulnerability List

1. HTTP for Ubuntu packages results in unnecessary exposure

Class: Data Exposure **Severity:** Medium **Difficulty:** High

Finding ID: iSEC-15FTC-8

Targets: The `sources.list` file in `/etc/apt/`.

Description: The APT package management system is configured to download Ubuntu and Tor packages over HTTP. While APT verifies the signatures of packages, so that any modification to a package will be noticed, using APT over HTTP allows for a passive network attacker to track which packages are being installed, and therefore fingerprint the server.

Secondly, using APT over HTTP allows for an active network attacker to impersonate an APT server which can be used to return an empty list of updates or selectively block certain updates, either of which would trigger a silent failure of the package update in SecureDrop servers.

Finally, HTTP exposes a great deal of unnecessary attack surface specific to APT's HTTP client. This is particularly harmful as it may allow manipulation of APT in unexpected ways. At worst, a vulnerability in APT's HTTP implementation may allow a network attacker to inject a payload in a response that may allow privileged code execution.

Exploit Scenario: An attacker able to passively sniff the network traffic of Ubuntu APT servers, leverages fingerprinting techniques to identify servers installing the exact set of packages specific to SecureDrop. The attacker is able to construct a list of all IP addresses running SecureDrop. By selectively blocking updates to a specific package that contains a vulnerability, or using an undisclosed vulnerability in APT's HTTP response handling, the attacker compromises SecureDrop instances.

Short Term Solution: Accept this risk. Currently, there are limited, if any, upstream Ubuntu repositories available over HTTPS.

Long Term Solution: Offer the `apt.freedom.press` repository as a Tor hidden service and mirror the Ubuntu and Tor repositories. This will minimize risks associated with HTTP traffic for APT.

2. Sessions are not invalidated upon logout

Class: Authentication **Severity:** Low **Difficulty:** High

Finding ID: iSEC-15FTC-1

Targets: The `logout`[5] function in the SecureDrop document interface.

Description: The document interface uses the session management mechanism provided by Flask. However, when users log out, while the cookie is removed from the web browser, the session identifier tied to the cookie is not invalidated. As a result, this session identifier, if compromised, can be reused to authenticate as the associated and previously logged out user. This is a side effect of Flask sessions, since they are client side sessions. In other words, session data is stored in a cookie signed by the server key to prevent tampering. Since the server does not manage the sessions, it does not have a mechanism to invalidate individual sessions and must rely on cookie deletion.

Exploit Scenario: Since the session identifier stays valid for an indefinite amount of time, an attacker exploits this opportunity to obtain this session via an unknown vulnerability. Even after a journalist logs out of SecureDrop when concluding their session, the attacker can set their cookie value to that of the session token they were able to capture to authenticate as the journalist. The attacker is now capable of sending messages to the source and viewing source metadata, as well as potentially creating more journalist accounts.

Short Term Solution: Accept this risk. Since the service is available only as an authenticated Tor hidden service, and typically viewed using a Tails environment, it is very unlikely that the session identifier would be leaked, as sessions are short-lived.

Long Term Solution: Consider partially managing user sessions on the server side, by adding a session component which allows the server to track whether a user has logged out in order to invalidate sessions. Additionally, consider using permanent sessions[6] with a short-lived lifetime value to allow inactivity-based expiration. The timeout can be reset by setting the `modified` property of the `session` object to `True` during the `before_request` handler.

[5] https://github.com/freedomofpress/securedrop/blob/0.3.3/securedrop/journalist.py#L141
[6] http://flask.pocoo.org/docs/0.10/api/#flask.Flask.permanent_session_lifetime

3. Journalist login enumeration

Class: Timing	**Severity:** Low	**Difficulty:** High

Finding ID: iSEC-15FTC-5

Targets: The `login` function in `journalist.py`.[7]

Description: The login process of the journalist interface implemented a mechanism to throttle logins in https://github.com/freedomofpress/securedrop/blob/0.3.3/securedrop/db.py#L40. The `throttle_login` function is called after the username is validated, resulting in only valid usernames being throttled. This can be used to enumerate valid logins in the journalist interface.

Exploit Scenario: An attacker who is able to reach the journalist interface is trying to guess valid credentials. The attacker reviews the code for SecureDrop and notices that only valid accounts are throttled. The attacker launches a dictionary attack against their victim and obtains a list of valid usernames.

Short Term Solution: Accept this risk. Since the login form is only available over an authenticated Tor Hidden Service, and 2-factor authentication is used, there is very limited value in enumerating user accounts.

Long Term Solution: Modify the throttle feature to keep track of all login attempts performed against the system. Throttle both valid and invalid accounts once the number of login attempts reaches the maximum number of attempts allowed within the defined time period.

[7] https://github.com/freedomofpress/securedrop/blob/0.3.3/securedrop/journalist.py#L114

4. SSH throttle ineffective

Class: Configuration **Severity:** Informational **Difficulty:** High

Finding ID: iSEC-15FTC-7

Targets: The iptables configuration in the `app_rules_v4` template of the `restrict_direct_access_app` Ansible role.[8]

Description: The iptables configuration of the monitoring and application servers limits the SSH connections to three new connections per minute. However, this limit is superseded by another rule[9] that accepts all outbound connections from the Tor process. The limiting rule is effectively bypassed by the first one and SSH connections are therefore not limited.

Short Term Solution: Fix the iptables rules to restore the throttle on the SSH connections. Simply reordering the rules to position the SSH throttle before the generic ACCEPT will prevent the bypass.

Long Term Solution: Implement regressions tests to ensure that security features such as the SSH throttle are not disabled or overridden whenever implementing new features. Run the regressions tests whenever releasing a new version of SecureDrop.

[8] https://github.com/freedomofpress/securedrop/blob/0.3.3/install_files/ansible-base/roles/restrict_direct_access_app/templates/app_rules_v4#L14

[9] https://github.com/freedomofpress/securedrop/blob/0.3.3/install_files/ansible-base/roles/restrict_direct_access_app/templates/app_rules_v4#L9

5. Plaintext passwords potentially leaked

Class: Auditing and Logging **Severity:** Informational **Difficulty:** High

Finding ID: iSEC-15FTC-2

Targets: The `_scrypt_hash` function in `db.py`.[10]

Description: The login process of the journalist interface uses scrypt to hash a user password before storing it in the database. This is done in a helper function called `_scrypt_hash`. This function has a debug clause enabled that prints the plaintext password into the standard output of the process. Because there are no checks about the type of environment in use (testing, staging or production), this means that this debug statement is enabled for production as well.

Logging plaintext passwords is considered a bad practice, as it may result in password disclosure in case of a compromise.

Note: The severity of this finding is informational because it does not seem that the application standard output is logged anywhere.

Short Term Solution: Remove the debug print statement from the code.

Long Term Solution: Develop regression tests to run before shipping a new version. These tests should check for any banned keywords or functions that should not be found in the code. Add `print` to this list, because it is not the standard way of logging messages in the web interfaces.

[10] https://github.com/freedomofpress/securedrop/blob/0.3.3/securedrop/db.py#L233

6. IPTables default "ALLOW" policy is dangerous

Class: Configuration **Severity:** Informational **Difficulty:** High

Finding ID: iSEC-15FTC-3

Targets: The iptables and ip6tables default policy rules for the INPUT, OUTPUT and FORWARD chains, configured in `/etc/network/iptables/*`. These rules are set by the `app_rules_v4` template in the `restrict_direct_access_app` Ansible role.[11]

Description: The "mon" and "app" servers use `iptables` to limit the network attack surface. This is a critical component of the servers' security. Although iSEC has not identified any flaws with the filter rules, the default policy for the INPUT, OUTPUT and FORWARD chains is set to ALLOW. This makes it much easier to make a unintentionally misconfigure the server and introduce future vulnerabilities.

- The IPv4 INPUT and OUTPUT chains already include rules to reject any packets that did not match earlier rules.
- The FORWARD chain allows all traffic, but IP forwarding is disabled on the servers using a `sysctl` setting.
- The IPv6 INPUT, OUTPUT and FORWARD chains allow all traffic, but IPv6 is disabled on the servers using a `sysctl`.

Short Term Solution: Set the default `iptables` and `ip6tables` policy of the INPUT, OUTPUT and FORWARD chains to DROP. This will increase the depth of defense and decrease the likelihood of introducing new flaws when making configuration changes.

Long Term Solution: Develop regression tests which ensure that the firewall rules behave as expected.

[11] https://github.com/freedomofpress/securedrop/blob/0.3.3/install_files/ansible-base/roles/restrict_direct_access_app/templates/app_rules_v4

7. OSSEC remote protocol uses weak cryptography

Class: Cryptography **Severity:** Informational **Difficulty:** High

Finding ID: iSEC-15FTC-4

Targets: The "secure" protocol accepted by the `ossec-remoted` daemon, as implemented in `remoted/secure.c`.[12]

Description: The "mon" server runs `ossec-remoted` to accept remote OSSEC log messages from the "app" server. This server supports several protocols, including the "secure" protocol, which runs over UDP port 1514. This protocol performs some minimal parsing of messages before decrypting the payload with a client-specific key using Blowfish in CBC mode. This implementation suffers from two weaknesses. First, the protocol uses a static initialization vector (IV) to encrypt and decrypt all messages.[13] Second, it does not authenticate the message. It does include an MD5 of the payload within the encrypted content,[14] but this is not a strong authenticator.

As a result of these flaws, an attacker can glean some information by looking at repeated blocks in the ciphertext. Since the IV is reused for each message, messages that start with the same plaintext will also start with the same ciphertext, therefore reducing the security of the block cipher to that of ECB mode. If an attacker can control plaintext, they can possibly use this to decrypt prior blocks.

The lack of ciphertext authentication makes it easier to tamper with messages. CBC mode encryption is particularly malleable, and allows portions of one message to be spliced together with portions of another message. The use of compression and a hash inside the plaintext message makes this less realistic in practice. However, an attacker can still alter portions of the message that are not covered by the hash, including padding. An attacker that can infer when messages are accepted or rejected might also be able to find a way to use the decryption process as an oracle that could allow incremental decryption of messages.

Short Term Solution: Accept this risk until the protocol can be updated.

Long Term Solution: File this bug upstream, as it does not seem to be a known security issue. To solve this problem, when using block cipher encryption modes that use an IV, generate a unique and random IV for each message and transmit it along with the ciphertext. Replace the use of Blowfish-CBC with an authenticated encryption (AE) cipher, such as AES-GCM which also benefits from hardware acceleration on some platforms. Authenticated Encryption protects from ciphertext tampering. For more information see the following whitepaper: https://www.nccgroup.trust/globalassets/newsroom/us/blog/documents/2013/introduction_to_authenticated_encryption.pdf

[12] https://github.com/ossec/ossec-hids/blob/v2.8.0/src/remoted/secure.c
[13] https://github.com/ossec/ossec-hids/blob/v2.8.0/src/os_crypto/blowfish/bf_op.c#L34
[14] https://github.com/ossec/ossec-hids/blob/v2.8.0/src/os_crypto/shared/msgs.c#L497

8. Denial of Service on journalist login

Class: Denial of Service **Severity:** Informational **Difficulty:** High

Finding ID: iSEC-15FTC-6

Targets: The `login` function in `journalist.py`[15] and `db.py`.[16]

Description: The application does not have an upper limit on the length of a user-supplied password, which allows for arbitrarily large strings to be passed to the password hashing function. Because passwords are hashed by default using `scrypt`,[17] extremely large passwords *e.g.* 2–4MB in size, can be used to consume large amounts of system resources, resulting in the site becoming unresponsive to other requests.

Short Term Solution: Implement an upper limit for password strings—128 characters should be more than sufficient. Enforce this limit before performing any further processing on user-supplied passwords.

Long Term Solution: Before performing any compression, decompression or hashing operation of any type, ensure that user-supplied data meets basic sanity parameters in terms of size and length.

[15] https://github.com/freedomofpress/securedrop/blob/0.3.3/securedrop/journalist.py#L106

[16] https://github.com/freedomofpress/securedrop/blob/0.3.3/securedrop/db.py#L345

[17] https://github.com/freedomofpress/securedrop/blob/0.3.3/securedrop/db.py#L240

9. AppArmor profile grants unnecessary capabilities

Class: Configuration **Severity:** Informational **Difficulty:** High

Finding ID: iSEC-15FTC-9

Targets: The custom Apache AppArmor profile used for SecureDrop.[18]

Description: SecureDrop applies a custom AppArmor profile for its Apache server. This profile restricts the directories the Apache process can access as well as Linux capabilities granted to the Apache process. The `net_bind_service` is necessary for binding to privileged ports. While the `kill` and `ptrace` capabilities seem to be needed, the `dac_override` capability is not. The `dac_override` capabilities allows a process to ignore discretionary access controls and access files it would not otherwise be able to. Since most of the files necessary for the application (such as the SecureDrop source code) are either owned by Apache or world-readable, it does not need to override access controls and thus should be removed.

Short Term Solution: Remove the `dac_override` capability from the AppArmor profile.

Long Term Solution: When developing custom AppArmor profiles, perform testing on AppArmor profiles to ensure that they require the minimum set of privileges necessary.

[18] https://github.com/freedomofpress/securedrop/blob/0.3.3/install_files/ansible-base/usr.sbin.apache2#L11

10. OSSEC alert fingerprinting

Class: Configuration **Severity:** Informational **Difficulty:** High

Finding ID: iSEC-15FTC-10

Targets: OSSEC email alerts sent by the monitor server.

Description: OSSEC sends various alerts to notify administrators of important events occurring in the SecureDrop environment. While the messages are GPG encrypted, the headers are in plain text and visible over the network. Some of these headers may disclose information indicating the presence of a SecureDrop monitoring server due to specific information contained in the headers.

```
Received: from monitor.securedrop
X-Google-Original-From: ossec@monitor.securedrop
Message-ID: <XXXX.XXX@monitor.securedrop>
```

Listing 1: Example headers indicating SecureDrop deployment

While these headers cannot be removed, as they are added by SMTP relays, the information which ends up within these headers can be modified. To prevent trivial fingerprinting of services, these should be changed to less distinguishable values.

Short Term Solution: Scrub information from these headers by using less conspicuous values. Recommend that a codename or alias is used for OSSEC alert information (as opposed to something like "monitor.securedrop") to prevent unknowingly advertising the use of SecureDrop.

Long Term Solution: Investigate other ways by which SecureDrop deployments may be fingerprinted, such as length/frequency of encrypted messages, and develop mechanisms to increase the difficulty of reliably detecting deployments.

Appendices

A Additional Hardening Recommendations

A.1 Grsecurity

As the SecureDrop servers do not have a need for additional USB devices, they should be limited to only those that are necessary. This can be done via the `kernel.grsecurity.deny_new_usb` sysctl interface before the Grsecurity lock is applied. Optionally, consider enabling the `GRKERNSEC_DENYUSB_FORCE` kernel configuration flag to only enable devices present at boot time.

A.2 Tails Concerns

Tails is a Linux distribution which aims to preserve the privacy and anonymity of its users. It is designed to run from live media, and allows users to create persistent volumes which use full-disk encryption via LUKS. It includes several applications pre-configured with security and privacy in mind, for instance, all applications are configured to route their traffic through Tor. It also bundles cryptographic applications such as GPG, and IM clients with OTR, to facilitate end-to-end encrypted communications. Tails also includes tools for securely deleting files from media, and even will automatically wipe memory during shutdown as a means of limiting forensic recovery.

In the context of SecureDrop, Tails is a key ingredient that eases administrator and journalist access to the journalist interface anonymously, and end-to-end encrypted. A central part of SecureDrop's architecture lies in the use of an airgrapped machine used exclusively for viewing documents in a safe way. Journalists will transfer encrypted documents from their correspondence with sources to the viewing station, where the private keys necessary for decryption reside. Journalists can then re-encrypt documents for use in editing.

Note that SecureDrop, due to its relatively simple functionality, and minimal processing of actual submissions, exposes a very limited attack surface to an attacker. However, this is less true of the secure viewing station that uses Tails, where most of the processing of files submitted by anonymous sources will occur. Because the media uploaded by a source can take many forms (e.g. images, PDFs, videos, and office documents), this leaves Tails with a wide attack surface exposed. For example, SecureDrop's use of Grsec and strict monitoring controls makes undetected compromise significantly more difficult. The security posture of SecureDrop keeps evolving much faster than Tails can keep up with, and as a result becomes a more feasible target for attack. Tails does not have the security protections offered by Grsec, and all applications run under the same context (e.g. no method for app containerization), though it does use AppArmor,[19] which makes the latter less of an issue. It should be noted that while Tails does not possess these important security features, they are being currently investigated[20].[21]

Due to these considerations, Freedom of the Press should be aware that while SecureDrop may continue improving, its secure viewing workflow becomes a more lucrative target. Freedom of the Press and OTF should work with Tails to reduce that gap and slowly remediate these weaknesses over time.

[19] https://tails.boum.org/contribute/design/application_isolation/

[20] https://tails.boum.org/blueprint/Mandatory_Access_Control/

[21] https://tails.boum.org/blueprint/Linux_containers/

A.3 Airgap Hardening

The main risk that lies in the secure viewing station is malware escaping the airgap via transfer media. Currently, since the secure viewing station is typically set up with a persistent volume, this seems like a realistic threat, as malware can persist in the permanent volume, and spread via media used to transport documents. To avoid malware persisting in the secure viewing station, consider mounting the permanent volume read-only after its initial setup.

The use of transfer media in an airgap is a difficult problem, as most modern storage media come with features prone to abuse. This is particularly true for USB, as it allows for many kinds of functionality and peripherals other than storage. What looks like a USB drive may in fact be any USB peripheral, such as a keyboard or a WiFi card. Other media, such as SD cards, are exclusively storage devices, provided they are in SD, SDHC, or SDXC formats, but not SDIO, which is comparable to USB. Optical storage media such as CD-R and DVD have the advantage of being exclusively read only, which lowers the risk of malware infection. There are also specialized USB devices which can block writes to USB storage in order to enforce read-only access; these are typically used by forensic examiners. However, they are not currently an ideal solution due to pricing. If this is a desired property, optical media should be preferred until the market price of these devices becomes more accessible to smaller operators of SecureDrop. With these considerations in mind Freedom of the Press should document these advantages and disadvantages of different transfer media so that administrators of SecureDrop can make conscious decisions about how to deal with transfer media for the airgapped viewing station.