FDEunlock Documentation

Release 0.7.2

Robin Schneider

Sep 23, 2017

Contents

Contents:

CHAPTER 1

FDEunlock introduction

(GitLab CI) - (Travis CI) - -

FDEunlock – Check and unlock full disk encrypted systems via ssh

This script allows you to unlock full disk encrypted GNU/Linux systems via ssh after checking that the system has not been tampered with.

Usage example

Checkout the following example:

```
fdeunlock --host fde-server.example.org-initramfs
INFO, 2017-03-29 10:27:41,822: Host offline. Attempting to start using: virsh -c␣
↪qemu:///system start fde-server
Domain fde-server started

INFO, 2017-03-29 10:27:42,726: Start command returned with: 0
INFO, 2017-03-29 10:27:48,257: Host offline. Waiting ...
INFO, 2017-03-29 10:27:53,264: Ping result: 198.51.100.23 : [0], 84 bytes, 0.51 ms (0.
↪51 avg, 0% loss)
INFO, 2017-03-29 10:27:53,270: Running Network based checkers:␣
↪LinkLayerAddressChecker, UnauthenticatedLatencyChecker
INFO, 2017-03-29 10:27:53,273: Link layer address matches the trusted once.
INFO, 2017-03-29 10:27:53,283: ICMP ping round trip time: 0.5100 ms
INFO, 2017-03-29 10:27:53,283: Latency is within the boundaries.
INFO, 2017-03-29 10:27:54,296: SSH session to initramfs established.
INFO, 2017-03-29 10:27:54,296: Running SSH based checkers: ChecksumChecker,␣
↪AuthenticatedLatencyChecker
INFO, 2017-03-29 10:27:57,487: Checksums match the trusted once.
INFO, 2017-03-29 10:27:57,559: Latency to execute a command over SSH and get the␣
↪response back: 71.6000 ms
```

```
INFO, 2017-03-29 10:27:57,560: Trusted latency: 60.256694030762
INFO, 2017-03-29 10:27:57,560: Current latency: 71.61283493041992
Choose one of 'save', 'ignore' (for current run) or anything else to exit: save
INFO, 2017-03-29 10:28:02,739: All 4 checks passed.
INFO, 2017-03-29 10:28:02,820: Passing key for vda3_crypt to host fde-server.example.
↪org-initramfs.
INFO, 2017-03-29 10:28:05,140: Could not retrieve key for vdb3_crypt (host fde-server.
↪example.org-initramfs).
Please enter key for vdb3_crypt (or store it in a vault):
INFO, 2017-03-29 10:28:28,155: Passing key for vdb3_crypt to host fde-server.example.
↪org-initramfs.
INFO, 2017-03-29 10:28:43,322: System should be booting now.
```

The host `fde-server.example.org-initramfs` was defined in the ssh configuration `~/.ssh/config` and the key for `vda3_crypt` was provided in `/home/user/.config/fdeunlock/keys/fde-server.example.org-initramfs_vda3_crypt.key`. And last but not least, the start command was configured in `/home/user/.config/fdeunlock/config.cfg`.

Repositories

- GitLab (primary repo with issue tracker)
- GitHub (mirror)

Documentation

- Read the Docs

Authors

- Marcel McKinnon
- Robin Schneider

License

GNU Affero General Public License v3 (AGPL-3.0)

CHAPTER 2

Installation

Dependencies

FDEunlock makes use of a few Python and system packages. If you are using Debian (or a Debian based distribution), you can install them using the package manager prior to installing FDEunlock:

```
sudo apt install python3-paramiko python3-pexpect python3-appdirs openssh-client fping
```

Latest release

You can install FDEunlock by invoking the following commands:

```
gpg --recv-keys 'C505 B5C9 3B0D B3D3 38A1  B600 5FE9 2C12 EE88 E1F0'
mkdir --parent /tmp/fdeunlock && cd /tmp/fdeunlock
wget -r -nd -l 1 https://pypi.python.org/pypi/fdeunlock --accept-regex '^https://
↪(test)?pypi\.python\.org/packages/.*\.whl.*'
current_release="$(find . -type f -name '*.whl' | sort | tail -n 1)"
gpg --verify "${current_release}.asc" "${current_release}" && pip3 install --upgrade "
↪${current_release}"
```

Refer to Verifying PyPI and Conda Packages for more details. Note that this might pull down dependencies in an unauthenticated way! You might want to install the dependencies yourself beforehand.

Or if you feel lazy and agree that pip/issues/1035 should be fixed you can also install FDEunlock like this:

```
pip3 install fdeunlock
```

Development version

If you want to be more on the bleeding edge of FDEunlock development consider cloning the `git` repository and installing from it:

```
gpg --recv-keys 'EF96 BC32 AC57 CFC7 2DF0  1D8C 489A 4D5E C353 C98A'
git clone --recursive https://gitlab.com/ypid/fdeunlock.git
cd fdeunlock && git verify-commit HEAD
echo 'Check if the HEAD commit has a good signature and only proceed in that case!' &&
↪ read -r fnord
echo 'Then chose one of the commands below to install FDEunlock and its dependencies:'
pip3 install --upgrade .
pip3 install --user --editable .
./setup.py develop --user
./setup.py install --user
./setup.py install
```

hashdeep for ChecksumChecker

hashdeep is used as a statically linked binary for the *ChecksumChecker*. You should probably compile it yourself if you want to use the *ChecksumChecker* using the following instructions:

```
apt install dpkg-dev
apt-get build-dep md5deep
apt-get source md5deep
cd md5deep-XXX/
./configure
cd src
editor Makefile
make
strip hashdeep
```

At the editor step you will need to ensure that the **hashdeep** binary will be statically linked. Adding `-static-libgcc` to the compile options fixed it for ypid.

Then copy the binary to `${FDEUNLOCK_DATA_DIR}/bin/hashdeep_$(echo "$(uname -s)_$(uname -m)" | tr 'A-Z' 'a-z')`. Example `${FDEUNLOCK_DATA_DIR}/bin/hashdeep_linux_x86_64`. You can run FDEunlock without the binary in place and it will exit with an error telling you what binary it needed based on the detected platform of your remote system.

CHAPTER 3

Getting started

Server side setup

To check and unlock a FDE server you will need to have that host ready to accept SSH connections in the initramfs. For this, the Dropbear SSH server is recommended. Refer to `/usr/share/doc/cryptsetup/README.Debian.gz` for details. In case you use Ansible you might find debops-contrib.dropbear_initramfs interesting.

There are also lots of additional resources available on how to set this up:

- https://blog.tincho.org/posts/Setting_up_my_server:_re-installing_on_an_encripted_LVM/
- https://kiza.eu/journal/entry/697
- http://www.lug-hh.de/wp-content/uploads/kwi_cloudserver_01_fde_0.3.pdf
- https://www.reddit.com/r/linuxadmin/comments/3ot1xk/headless_server_with_fdeluks/

Note that FDEunlock makes use of the **`cryptroot-unlock`** script which is only available in the cryptsetup package of Debian stretch or newer. FDEunlock includes the script for now to make it work out-of-the-box with older Debian releases and potentially other GNU/Linux distributions.

FDEunlock has been successfully tested in the following configurations:

- Debian jessie, dropbear 2014.65-1: IPv4 only, IPv6 only, dual stack
- Debian jessie, dropbear-initramfs 2016.74-2: IPv4 only, IPv6 only, dual stack

Note that you currently might need to set the *address_family* for IPv6 only.

After you setup Dropbear you should write down the generated SSH host key fingerprints over your current, hopefully verified, session. To do this, issue the following commands and note their output for later comparison:

```
dropbearkey -y -f /etc/dropbear-initramfs/dropbear_rsa_host_key
dropbearkey -y -f /etc/dropbear-initramfs/dropbear_ecdsa_host_key
```

Defining a host in FDEunlock

FDEunlock assumes that you can ssh into a host you want to unlock using a simple **`ssh ${host}`**. Any SSH options should be placed into the SSH client configuration. Refer to the *ssh_config(5)* manpage for details.

The following example should get you started:

```
Host fde-server.example.org-initramfs
    HostName fde-server.example.org
    IdentityFile ~/.ssh/keys/root@fde-server.example.org-initramfs

Host *-initramfs
    User root
    UserKnownHostsFile ~/.ssh/known_hosts/initramfs
    IdentityFile ~/.ssh/keys/%r@%h
    ## %n would have been perfect instead of %h but this is not supported as of
    ## OpenSSH 7.4? They should have hacked harder.

    ## You might need to allow these for older versions of dropbear:
    # MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,hmac-sha1-96,
↪hmac-sha1
    # KexAlgorithms diffie-hellman-group-exchange-sha256,diffie-hellman-group14-sha1,
↪diffie-hellman-group-exchange-sha1
```

Configuration and data location

FDEunlock follows the XDG Base Directory Specification. The following places are used by FDEunlock:

`FDEUNLOCK_CONFIG_DIR` Used for configuration and keys (FileVault).

`FDEUNLOCK_DATA_DIR` Used for data of checkers.

Use the following to figure out where those paths are to be found on your platform:

```
FDEUNLOCK_CONFIG_DIR="$(python3 -c 'from appdirs import *; print(user_config_dir(
↪"fdeunlock"))')"
FDEUNLOCK_DATA_DIR="$(python3 -c 'from appdirs import *; print(user_data_dir(
↪"fdeunlock"))')"
```

Providing a key using the default FileVault

Place your key (either the passphrase or the keyfile) into `${FDEUNLOCK_CONFIG_DIR}/keys/${host}_${device_name}.key`.

`${device_name}` is either the plaintext device mapper target or the full ciphertext block device path with `/` replaced with `_`. Note that the later variant depends on your `/etc/crypttab` configuration.

Consider this example `/etc/crypttab` file:

```
sda4_crypt /dev/disk/by-partuuid/e1cd49d2-158b-11e7-99d8-00163e5e6c0f none luks
```

Where `sda4_crypt` is the plaintext device mapper target of your root filesystem. The following two `${device_name}` can be used here:

- `sda4_crypt`

- dev_disk_by-partuuid_e1cd49d2-158b-11e7-99d8-00163e5e6c0f

If both files exist the first one (more generic) is tested first and might need to be removed if it does not contain the correct password. The later, more explicit variant (using GPT partition UUIDs in this example) is generally preferred.

When you use a passphrase you will need to ensure that no newline is appended to the file (all common editors appended a newline automatically). One way to avoid the newline is to run the following command:

```
echo -n 'Please enter your passphrase: '; read -rs pw; echo -n "$pw" > "${key_file}";␣
↪unset pw
```

Alternatively, to generate a new passphrase you can run this command instead:

```
echo -n "$(pwgen -s 123 1)" > "${key_file}"
```

CHAPTER 4

CLI interface

Check and unlock full disk encrypted systems via ssh

```
usage: fdeunlock [-h] [-V] [-d] [-v] [-q] -H HOST [-n]
```

Named Arguments

-V, --version	show program's version number and exit
-d, --debug	Write debugging and higher to STDOUT|STDERR.
-v, --verbose	Write information and higher to STDOUT|STDERR.
-q, --quiet, --silent	Only write errors and higher to STDOUT|STDERR.
-H, --host	Hostname of the remove server
-n, --no-unlock	Don‘t enter passphrases. Start a interactive shell session after checking instead. Default: True

CHAPTER 5

Host checkers

Before the decryption key gets passed to the remote host, it must be ensured that the remote host is the authentic one and that it is in a trustworthy state.

Currently, the strongest sign that this is the case is the host key verification done by SSH which ensures that the remote host has access to it's private host key.

Additional to that, a few checkers are implemented which hook in at various stages of fdeunlock execution.

Network based checkers

`LinkLayerAddressChecker` Checks the network link layer address as observed by the machine running fdeunlock. A link layer address is only expected to be available if the remote machine and the machine fdeunlock is running on communicate over the same link layer network.

This check might be interesting because the link layer address is not stored on disk (at least not in clear text). An adversary having only access to the disks would have no easy way to get to know this address.

Note that the link layer address is typically capturable and spoofable in a local network.

`UnauthenticatedLatencyChecker` Checks the round trip time measured by **`fping`** based on one ICMP packet if it is within expected boundaries.

The default boundaries is 1.0 ms and can be configured using the `unauthenticated_latency_deviation` configuration option which is a float number representing the time deviation in ms.

The intention of this check together with the *AuthenticatedLatencyChecker* is to detect a variant of an evil maid attack where the host you think you are just unlocking is not the one you are actually unlocking. Such an attack might have different latency characteristics because even the most advanced adversary is still bound by the law of physics. For reference, the speed of light is 300 km/ms.

SSH based checkers

AuthenticatedLatencyChecker Measure the latency over SSH and check if it is within expected boundaries.

The default boundaries are 10.0 ms and can be configured using the `authenticated_latency_deviation` configuration option which is a float number representing the time deviation in ms.

Refer to *UnauthenticatedLatencyChecker* for the background.

ChecksumChecker Compute checksums for all files in the initramfs and compare the checksums to previously measured trusted once.

Note that if an reasonably funded adversary is in the position to tamper with your initramfs this check will probably not be able to catch them. All this check tries to do is to increase the cost of such an attach or even catch less skilled attackers.

There are many ways how this check can be fooled. For example, the check does not checksum the loaded/running kernel image nor the boot loader nor the system firmware.

The `additional_checksum_commands` configuration option can be used to specify additional commands for checksumming/verification. The output of those commands is included in the checksum file and compared with the previous measurement. Multiple commands can be given, separated by newline. Example:

```
[DEFAULT]
additional_checksum_commands =
  uname -a
  dmesg | egrep '(DMI:|Command line:|Booting paravirtualized kernel on bare␣
↪hardware)' | sed 's/^\[\s*[[:digit:].]\+\]\s*//;'
  cat /sys/devices/virtual/dmi/id/board_serial /sys/devices/virtual/dmi/id/
↪product_uuid
  ls /dev/disk/by-id/
  dd if=/dev/sda bs=512 count=1 | sha512sum -
```

The `diff_command` configuration option can be used to set another text diffing program than the default **`diff`** command. Comparison is run on your local machine. Note that the diffing program is exposed to untrusted input. The files path to the trusted and the currently untrusted checksum file are appended as the last two parameters to the given command, in the mentioned order (trusted first; untrusted second/last). Example:

```
[DEFAULT]
diff_command = git diff --no-index
```

Proper remote attestation (Trusted Computing) should be implemented. Feel free to add support for this to FDEunlock :-)

Ref: https://security.stackexchange.com/questions/46548/for-remotely-unlocking-luks-volumes-via-ssh-how-can-i-verify-integri

CHAPTER 6

Configuration

Optional configuration can be saved in *$FDEUNLOCK_CONFIG_DIR*/config.cfg or in a *.cfg file below *$FDEUNLOCK_CONFIG_DIR*/config.d.

The format is INI as handled by the configparser module. Refer to the module's documentation for details.

Each host has it's own section. Supported options inside sections:

address_family Specifies which address family to use when connecting. Refer to the *ssh_config(5)* manpage for details. Falls back to your ssh configuration.

start_command Command which is executed to start the remote host in case it is found offline.

The command can make use of the following tokens, which are expanded at runtime:

- %(originalhost)s, hostname as it was specified on the command-line.
- %(host)s, target hostname, after any substitution by ssh.
- %(ssh_port)s, SSH port.
- %(hostname)s, hostname without domain.
- %(domain)s, domain of the host.

start_command_shell Boolean determining if *config_start_command* is to be executed in the comforting environment of a shell – or not. Defaults to False.

exclude_checkers List of host checkers to exclude from running.

Multiple host checkers can be given, separated by newline. Example:

```
[fde-server.example.org-initramfs]
exclude_checkers =
  ChecksumChecker
```

codec_error_action Communication with remote systems is assumed to be UTF-8 encoded. How should UTF-8 decoding errors be handled? Set to replace to convert unknown bytes into the Unicode replacement character. Defaults to strict which will cause a UnicodeDecodeError to be thrown terminating execution immediately.

Host checkers might support additional configuration options. Refer to the *Host checkers* section for details.

CHAPTER 7

Related projects

Mandos

Mandos has very similar goals as FDEunlock but both address different uses cases. The key difference is the server/client model. With Mandos you have one or more Mandos servers providing keys to hosts. The hosts initiate the request for a key. They find the Mandos server either by configured IP address or using Avahi.

On the other hand, FDEunlock works the other way around. FDEunlock is started by the user to initiate a connection to the host. FDEunlock then checks the host and enters the keys it requests which are/where (previously) provided by the user for that host.

Also, on the implementation side there are a few differences:

	Mandos	FDEunlock
Transport security	TLS, GnuTLS, +optional: OpenVPN, ...	SSH, Dropbear, OpenSSH
Transport sec certs	OpenPGP keys with GnuTLS	OpenSSH host keys
Mode of operation	Hosts connect to any Mandos Server	FDEunlock connects to hosts
Complexity approx.	High. Python: ~3500 LOC; C: ~4000	Medium. Python: ~1000 LOC
Deployment	Server daemon	Standalone
Implemented in	Server: Python2; Client: C, Bash	FDEunlock: Python3
In Debian	Yes	No
Key encrypted	Yes, only decryptable by target	No, see *TODO list*
Anti Evil Maid	Not SOTA. Dead man switch using ICMP.	Not SOTA. Multiple checks.
Development status	Stable	Beta
License	GPL-3.0+	AGPL-3.0

Last changed 2017-03-29

Which to use really depends on your use case.

If you focus on end point/workstation security and don't put much trust in servers, which might not always be under your supervision then FDEunlock might work better for you because that is what it was build for (to use it on workstation of admins).

If you operate a big data center and want to have encrypted servers by default then Mandos should be your number one option.

Note that as both projects use Python to implement similar parts of their design, using/importing/combining/improving each other is possible but currently not done.

Plain SSH

If simplicity is key then not much will beat the default way for remote unlocking as documented by Debian. Either write the passphrase directly to `/lib/cryptsetup/passfifo` or run **`cryptroot-unlock`**.

```
ssh fde-server.example.org-initramfs "echo -ne 'fnord' > /lib/cryptsetup/passfifo"
```

antievilmaid

antievilmaid is a proper SOTA tamper detection tool for workstations using trusted boot.

tpmtotp

tpmtotp is a proper SOTA tamper detection tool which takes antievilmaid to the next level.

chkboot

chkboot is a non-SOTA Anti Evil Maid detection tool intended for workstations. It uses cryptographically strong checksums to measure the content of `/boot` BUT after the decryption key has already been entered/passed to the machine.

The functionally is similar to the *ChecksumChecker* of FDEunlock.

Others?

ypid is not aware of other similar projects. If you are, please get in touch.

CHAPTER 8

Design principles

- Standalone. No server running in the background. FDEunlock only runs (and hands out keys to servers) when you tell it to do so.
- Easy to setup up.
- The end point/workstation is ultimately* trusted.

  *There are plans to remove the "ultimately" part. Refer to *TODO list*.
- FDEunlock establishes the connection to the host you want to unlock. Makes it easier if your workstation is firewalled, SNATed or is otherwise not reachable from the Internet.
- Extensible checker design. Have a good idea for an additional check? Great! Just implement the interface of a `HostChecker` class and add it to the list of default checkers.

CHAPTER 9

fdeunlock package

Submodules

fdeunlock.checker module

Pre-unlock checkers

class fdeunlock.checker.**LinkLayerAddressChecker**(*unlocker*)
Bases: fdeunlock.checker.NetworkBasedChecker

Check network link layer address and compare it to previously observed trusted once.

check(***kwargs*)

update()
Update check cache while host is in normal operation.

class fdeunlock.checker.**UnauthenticatedLatencyChecker**(*unlocker*)
Bases: fdeunlock.checker.NetworkBasedChecker

Check the unauthenticated latency previously measured by fping if it is within expected boundaries.

check(***kwargs*)

update()

class fdeunlock.checker.**ChecksumChecker**(*unlocker*)
Bases: fdeunlock.checker.SshBasedChecker

Compute checksums for all files in the initramfs and compare the checksums to previously measured trusted once.

check(*shell=None*, ***kwargs*)

update()

class fdeunlock.checker.**AuthenticatedLatencyChecker**(*unlocker*)
Bases: fdeunlock.checker.SshBasedChecker

Measure the latency over SSH and check if it is within expected boundaries.

check(*shell=None*, ***kwargs*)

update()

fdeunlock.cli module

Command line interface of fdeunlock

fdeunlock.cli.**main**()

fdeunlock.fdeunlock module

Core of FDEunlock

class fdeunlock.fdeunlock.**FdeUnlock**(*vault*, *checkers=None*)

Bases: object

check_and_unlock(*host*, *unlock=True*)

run_checkers(*parent_class*, *shell=None*)

unlock(*init_shell*)

Get passphrase and unlock system.

fdeunlock.helpers module

fdeunlock helpers

fdeunlock.helpers.**ensure_permissions**(*path*, *mode*)

fdeunlock.helpers.**get_user_dir**(*dir_type*)

fdeunlock.helpers.**read_config**()

fdeunlock.helpers.**read_properties_config**()

fdeunlock.helpers.**read_ssh_config**()

fdeunlock.helpers.**write_properties_config**(*properties*)

fdeunlock.pxssh module

Simplified Pxssh

class fdeunlock.pxssh.**SimplifiedPxssh**(*timeout=30*, *maxread=2000*, *searchwindowsize=None*, *logfile=None*, *cwd=None*, *env=None*, *ignore_sighup=True*, *echo=True*, *options={}*, *encoding=None*, *codec_errors='strict'*)

Bases: pexpect.pxssh.pxssh

copy_to_remote(*local_file_path*, *remote_file_path*)

get_platform()
Return our platform name 'linux_x86_64'

Format based on PEP 425 Compatibility Tags (wheel/pep425tags.py).

login(*host*, *auto_prompt_reset=True*)
Radically simplified login without the 'New certificate – always accept it.' stuff.

run_command(*command*)
Run command and don't expect any additional output.

fdeunlock.vault module

Vault implementations

class fdeunlock.vault.**FileVault**
Bases: *fdeunlock.vault.Vault*

Simple, file based Vault implementation.

get_key(*host*, *device_name*)

class fdeunlock.vault.**Vault**
Bases: abc.ABC

Abstract Vault class.

get_key(*host*, *device_name*)

Module contents

Check and unlock full disk encrypted systems via ssh

CHAPTER 10

Contributing and issue reporting

You can contribute and report issues in the usual way as documented by GitHub. Unit tests can be run locally and are automatically run in CI. Acceptable contributions need to pass all of them.

If you found a security vulnerability that might put users at risk please send your report/patch to ypid@riseup.net. Please consider using OpenPGP to encrypt your email.

CHAPTER 11

Copyright

```
"""
Meta information about FDEunlock
"""

# https://stackoverflow.com/a/16084844
__version__ = '0.7.2'

__license__ = 'AGPL-3.0'
__author__ = 'Robin Schneider <ypid@riseup.net>'
__copyright__ = [
    'Copyright (C) 2013-2016 Marcel McKinnon <sdrfnord@gmx.de>',
    'Copyright (C) 2017 Robin Schneider <ypid@riseup.net>',
]


__all__ = ['__version__', '__license__', '__author__', '__copyright__']

# FDEunlock is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as
# published by the Free Software Foundation, version 3 of the
# License.
#
# FDEunlock is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

CHAPTER 12

Changelog

This project adheres to Semantic Versioning and human-readable changelog.

fdeunlock master - unreleased

fdeunlock v0.7.2 - 2017-09-23

Added

- Mention antievilmaid, tpmtotp and chkboot in *Related projects*. [ypid]
- Allow to exclude/disable checkers using *exclude_checkers*. [ypid]
- Make behavior for UTF-8 decoding errors configurable as *codec_error_action*. [ypid]

Fixed

- Fix support to control `cryptroot-unlock` in different configurations. [ypid]

fdeunlock v0.7.1 - 2017-04-01

Fixed

- FDEunlock tried to execute **`None`** as command and failed in case the `start_command` config option was not set. Problem was implicit type conversation from `None` to `str`. [ypid]

fdeunlock v0.7.0 - 2017-03-31

Added

- Add Python 3 support, documentation, CI, Python package, platform independences, unit testing. [ypid]
- Wrote checkers: `LinkLayerAddressChecker`, `UnauthenticatedLatencyChecker`, `AuthenticatedLatencyChecker`. Refer to *Host checkers* for details. [ypid]
- Add configurable *config_start_command* for starting an offline machine. [ypid]
- Add script to PyPI under the name fdeunlock. [ypid]
- Acquired CII Best Practices badge for FDEunlock. [ypid]
- Add IPv6 support. [ypid]

Changed

- Major rework and code quality improvements. [ypid]
- Rename from scout to FDEunlock (project name) and **`fdeunlock`** (package and command name). [ypid]
- Changed license from GPL-3.0+ to AGPL-3.0. [ypid]
- Taken over maintenance of the project. Refer to this issue for details. [ypid]

scout v0.6.0 - 2016-03-06

Added

- Support interactive mode to drop into a shell after verification. [sdrfnord]

scout v0.5.0 - 2015-01-13

Changed

- Use *`ssh_config(5)`* for ssh options instead custom configuration. [sdrfnord]
- Improved code quality. [sdrfnord]

scout v0.4.0 - 2014-06-30

Added

- Encryption passphrase can be read from a configuration file. [sdrfnord]

Changed

- Rewrite in Python using pexpect. [sdrfnord]

scout.bash v0.1.0 - 2013-10-06

Added

- Initial coding and design in Bash. Refer to Tauchfahrt mit Linux – Colocation Anti-Forensik (German). [husemann]

CHAPTER 13

TODO list

- Better way to retrieve credentials: Qubes OS; https://www.vaultproject.io/
- Isolation using a split-vm approach common in Qubes OS where the unlocking is done in a disposable vm which only gets the keys it needs potentially after the remote system has been verified or even in a different vm.
- Support remote attestation (Trusted Computing). Ref: http://trousers.sourceforge.net/faq.html#2.1
- Measure time between startup of the server and the availability of the ssh service ;) (If possible: If you are like me you also power on your server with a shell command ;))
- Option to run/update validation during normal operation.
- Support key encryption using GnuPG like Mandos has implemented it. Requires to add user separation to the initramfs because currently FDEunlock has full control over target and could just extract the decryption key.

Python Module Index

f

Index