SOSCleaner Documentation

Release 0.4.4

Jamie Duncan

Dec 20, 2019

Index

Soscleaner is an open-source tool to take an sosreport, or any arbitrary dataset, and sanely obfuscate potentially sensitive information so it can be shared with outside parties for support. An unaltered copy of the data is maintained by the user so data can be mapped and suggestions supplied by a support team can still be actionable, even without the sensitive information.

CHAPTER 1

Overview

1.1 Purpose

The goal of soscleaner's documentation is to provide not only insight into how the code works, but also the logic behind what is obfuscated and why.

1.2 Important Links

Soscleaner's build processes are all completely available online.

Source Code	Github
Issues	Github
Project Tracking	Github
RPMs	Fedora Copr
Python Packages	PyPi
Code Coverage	Coveraalls
CI/CD	Travis-CI

1.3 Obfuscated data types

1.3.1 Hostnames and Domainnames

Soscleaner obfuscates all domains specified by the -d option when executed, as well as the domain name of the given host in an sosreport. Subdomains for these are automatically obfuscated as unique objects.

1.3.2 IPv4 addresses

IPv4 addresses are obfuscated, with each network being assigned a unique obfuscated counterpart with the same subnet mask. More information can be found at Network Obfuscation

1.3.3 User-specified keywords

Users can specify a list of keywords at runtime to find and obfuscate.

1.3.4 System Usernames

Users can be supplied in a line-delimited file. The contents of the `last` file in an sosreport is also incorporated into this obfuscation.

1.3.5 MAC addresses

MAC addresses are randomized consistently throughout an entire sosreport or any dataset.

CHAPTER 2

Network Obfuscation

2.1 Network Obfuscation Overview

Beginning with version 0.3.0, soscleaner uses the ipaddr module to manage network objects and their obfuscation This will let the program be much more intelligent with how it obfuscates the data while being network away, etc.

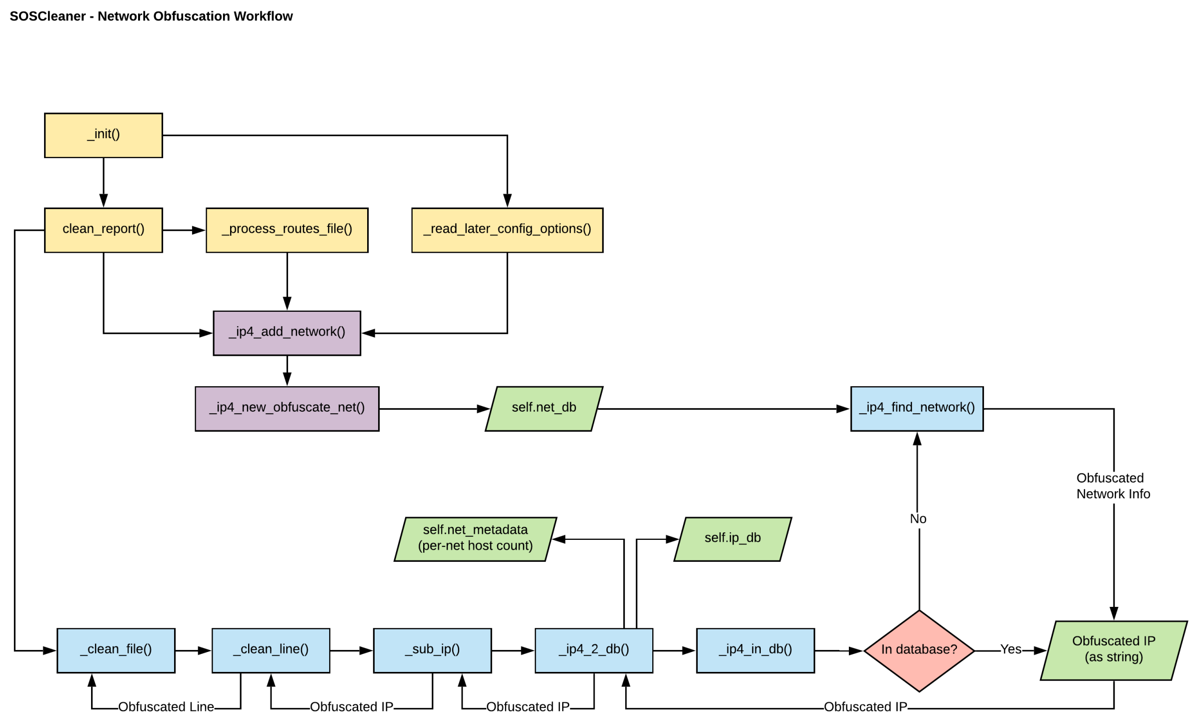

Fig. 1: Network Obfuscation Workflow Overview

2.1.1 Filing networking bugs

Please open networking obfuscation bugs using the network obfuscation bug template. This will ensure the proper labels are applied and we can move forward quickly with your issue.

2.2 IPv4 Network database

Each entry in `self.net_db` represents a network and its obfuscated value. `self.net_db` is a list of tuples. Each tuple has the following format:

```
(original_network, obfuscated_network)
```

For each entry in `self.net_db`, `x[0]` is the original network as an `ipaddr.IPv4Network` object and `x[1]` is the obfuscated network as an `ipaddr.IPv4Network` object.

2.3 IPv4 address database

Each entry in `self.ip_db` represents a found IP address and its obfuscated value as a key/value pair.

2.4 Obfuscating IPv4 addresses

When `self.clean_report is run`, it populates `self.net_db` with the networks found in an sosreports routing table as well as with any networks specified using the `-n` command line parameter.

Each time an IP is found in a file, it will be compared against the values in `self.net_db` to determine its parent network. The IP is then obfuscated sanely with fidelity to the subnet and relative network space. The obfuscated value for that IP address is then either retrieved from `self.ip_db`, or added to the database if it hasn't been obfuscated previously.

If an IP address is matched that doesn't exist in any other network, it will be obfuscated using an address from `self.default_net`. `self.default_net` is the first obfuscated network created when soscleaner is run.

Multicast obfuscation

Soscleaner doesn't obfuscate multicast addresses to other multicast address spaces because of the limitations without that IPv4 space. They are, however, obfuscated to a unique network so they can still be tracked and used for troubleshooting issues.

2.5 IPv4 metadata

`self.net_metadata` is a metadata dictionary for obfuscated networks. It tracks the number of allocated hosts in each network so the obfuscated networks can be iterated cleanly. Keys in `self.net_metadata` are set when networks are defined at the beginning of a soscleaner run.

2.5.1 `self.net_metadata` values

host_count Used to assign the next obfuscated IP address by tracking how many addresses on each network have been allocated.

The length of `self.net_metadata` is also used to determine how many obfuscated networks are in use.

2.6 IPv4 limitations and assumptions

If your dataset or sosreport contain subnets larger than a /8, you will break the math for creating obfuscating networks.

Why To calculate the next obfuscation subnet, I have no idea what the next subnet mask will be, and I don't want to get into crazy CIDR calculations.

How I take the default_net's first octet, increment it by the current existing obfuscated network count, and create a subnet with the corresponding subnet mask.

2.6.1 Example obfuscated network topology

An obfuscated network map could end up similar to:

CIDR	Network
128.0.0.0/8	`self.default_net`
129.0.0.0/24	obfuscated network 1
130.0.0.0/16	obfuscated network 2
131.0.0.0/30	obfuscated network 3
132.0.0.0/8	obfuscated network 4
133.0.0.0/32	obfuscated network 5

Essentially we're using up a lot of IP addresses to keep the math simple. The default network starts 1 above the loopback, so we don't have to account for that. We know there are corner cases here that could break the math. We have to hope common sense will prevail.

2.7 Network report

At the conclusion of a soscleaner run, the supplied network mappings are recorded in `self.report_dir/<SESSION_ID>-ip.csv`. If an SOSCleaner session fails to complete, this report isn't created.

..admonition:: Attention

This report only includes IPv4 data. IPv6 is (likely) coming in an upcoming release. The work for IPv6 obfuscation will happen under GitHub 7.

CHAPTER 3

Host and domain obfuscation

3.1 Host and domain obfuscation overview

SOSCleaner has completely re-written the host and domain obfuscation engine for the 0.4.0 release. In previous releases, all hostnames were obfuscated to *obfuscateddomain.com*. This could be confusing when troubleshooting issues across multiple domains.

3.1.1 Filing hostname bugs

Please open hostname obfuscation bugs using the hostname obfuscation bug template. This will ensure the proper labels are applied and we can move forward quickly with your issue.

3.2 Domain database

Domains that are obfuscation are maintained in `self.dn_db`, a dictionary, in `{'original_domain1': 'obfuscated_domain1',...}` format. Domains are obfuscated in addition to full hostnames because the domain in a configuration or in a log often makes a big difference in fixing or finding an issue.

3.2.1 Adding domains to the domain database

If obfuscating an sosreport, the FQDN of the report host is split between host and domain, and the domain is automatically added to `self.dn_db`.

Additional domains can be slated for obfuscation using the `-d` parameter on the command line. Multiple domains can be added by using multiple `-d` parameters, for example:

```
# soscleaner -d example.com -d foo.com -d someotherdomain.com mysosreport.tar.
xz
```

would add `example.com`, `foo.com`, and `someotherdomain.com` to `self.domains`.

3.2.2 Default domains

In addition to the host's domainname and any additional domains, soscleaner automatically adds `redhat.com` and `localhost.localdomain` to `self.dn_db`.

3.2.3 Processing domains

After the desired entries are added to `self.domains` using the above processes, `self._domains2db()` is called by, `self.clean_report()` to add all the entries to `self.dn_db` with their obfuscated counterparts.

3.2.4 Obfuscating subdomains

Each line in each file processed by soscleaner is processed by `self._clean_line()`, which calls `self._sub_hostname()`. This function uses a regular expression to match anything in the current line that is potentially a domain.

```
potential_hostnames = re.findall(r'\b[a-zA-Z0-9-\.]{1,200}\.[a-zA-Z]{1,63}\b', line)
```

The matches in `potential_hostnames` are validated againt the list of known domains using `self._validate_domainname()`. If the potential domain turns out to be a subdomain of a known domain, the newly matched subdomain is added to `self.dn_db` using `self._dn2db()`. For example, if `example.com` is a known domain, and a potential match is `apps.example.com`, `apps.example.com` will be added to the domain database and used for obfuscation going forward.

3.3 Hostname database

One of the primary functions of SOSCleaner is to obfuscate hostnames when they're found in a file beyond just the hostname of the server itself. To aid in troubleshooting, domain names are obfuscated separately. This is to keep the integrity of the data, even though the data is being obfuscated. Obfuscated hostnames are tracked in `self.hn_db`, a dictionary, using the `{'original_host1': 'obfuscated_host1',...}` format.

3.3.1 Default hostnames

If processing a sosreport the hostname of the sosreport host is added to `self.hn_db`.

3.3.2 Adding hostnames

When a hostname is found that is a member of a known domain in `self.dn_db`, it is obfuscated as `hostX.obfuscatedomainY.com`, with X being an incremented number equal to the current total of found hosts, `self.hostname_count`. Y is equal to the unique value assigned to the corresponding domain.

3.3.3 Host short name

There are many occurrences of the host-only part of the server's hostname in an sosreport and log files in general. These are obfuscated explicitly in `self._sub_hostname()`. When an soscleaner run is started, the host's hostname is stored as `self.hostname`. This is explicitly searched for in each line by soscleaner.

3.3.4 Short domains

There are a few short domain names that soscleaner obfuscates. By default, `localhost` and `localdomain` are added to `self.short_domains`, and are explicitly searched out and replaced in each line.

Short domains aren't editable

Currently there isn't a way to add additional entries to `self.short_domains`.

3.4 Hostname and Domainname reports

At the conclusion of a soscleaner run, the domain and hostname mappings are recorded in `self.report_dir/<SESSION_ID>-hostname.csv` and `self.report_dir/<SESSION_ID>-dn.csv`, respectively. If an SOSCleaner session fails to complete, these reports aren't created.

CHAPTER 4

MAC Address obfuscation

4.1 MAC address overview

MAC addresses are found in a line using the `re.compile(ur'(?:[0-9a-fA-F]:?){12}')` Python regular expression. For each match, a random valid MAC address is generated and saved in `self.mac_db` using the `{'mac_address': 'obfuscated_mac_address', ...}` format.

..admonition:: False Positives

> This is a new feature for the 0.4.0 release of soscleaner. Please report any issues you find regarding false-positives!

4.2 Filing MAC bugs

Please open MAC obfuscation bugs using the MAC obfuscation bug template. This will ensure the proper labels are applied and we can move forward quickly with your issue.

4.3 MAC address report

At the conclusion of a soscleaner run, the supplied MAC address mappings are recorded in `self.report_dir/<SESSION_ID>-mac.csv`. If an SOSCleaner session fails to complete, this report isn't created.

CHAPTER 5

Keyword obfuscation

SOSCleaner can take any arbitrary list of keywords and effectively obfuscate them in a sosreport or in a dataset. This can be extremely useful if you have particular key values, parameters from your IDP (Identity Provider). These are only matched against whole words.

5.1 Filing keyword bugs

Please open keyword obfuscation bugs using the keyword obfuscation bug template. This will ensure the proper labels are applied and we can move forward quickly with your issue.

5.2 How soscleaner handles keywords

The obfuscation engine for keywords is straightforward. Using the *-k* option on the command line supplies a line-delimited file of keywords. These keywords are then matched against whole words in every line of every file in an sosreport or dataset.

5.3 Keyword report

At the conclusion of a soscleaner run, the supplied keyword mappings are recorded in `self.report_dir/<SESSION_ID>-kw.csv`. If an SOSCleaner session fails to complete, this report isn't created.

CHAPTER 6

User obfuscation

6.1 User obfuscation overview

When obfuscating an sosreport, soscleaner uses the usernames in `sos_commands/last/lastlog` to populate `self.user_db`. This database is stored as a dictionary using a `{'user': 'obfuscated_user', ...}` format. `self.user_db` is populated using `self._process_users_file()` called early in `self.clean_report()`. Each line is passed into `self._sub_username()` in `self._clean_line()` as part of the obfuscation process.

6.1.1 What constitutes a username?

Usernames are anything in the `Username` column of `sos_commands/last/lastlog`:

```
Username         Port     From             Latest
root             pts/0    lnyce80te.elab.c Fri Feb 15 09:40:56 -0600 2019
bin                                        **Never logged in**
daemon                                     **Never logged in**
adm                                        **Never logged in**
lp                                         **Never logged in**
sync                                       **Never logged in**
shutdown                                   **Never logged in**
```

SOSCleaner does ignore a few common system users: `('reboot', 'shutdown', 'wtmp')`.

..admonition:: Adding usernames after soscleaner starts

Currently usernames can't be added to soscleaner after the run starts.

6.1.2 Filing user bugs

Please open user obfuscation bugs using the user obfuscation bug template. This will ensure the proper labels are applied and we can move forward quickly with your issue.

6.2 Username report

At the conclusion of a soscleaner run, the supplied username mappings are recorded in `self.report_dir/<SESSION_ID>-username.csv`. If an SOSCleaner session fails to complete, this report isn't created.

CHAPTER 7

Contributing

The easiest way to get and stay up to date is with by using the soscleaner-dev mailing list. While productions releases are announced on soscleaner-announce, most discussion happens on the dev mailing list.

7.1 Code contributions

Of course code contributions are welcome. Please follow the standard Github PR process.

7.2 Commit format

We've just started using gitchangelog to format git commits and generate a changelog. Please follow their example file like below:

```
Format:

ACTION: [AUDIENCE:] COMMIT_MSG [!TAG ...]

Description:
ACTION is one of 'chg', 'fix', 'new'

    Is WHAT the change is about.

    'chg' is for refactor, small improvement, cosmetic changes...
    'fix' is for bug fixes
    'new' is for new features, big improvement

AUDIENCE is optional and one of 'dev', 'usr', 'pkg', 'test', 'doc'

    Is WHO is concerned by the change.

    'dev'  is for developers (API changes, refactors...)
```

(continues on next page)

(continued from previous page)

```
    'usr'  is for final users (UI changes)
    'pkg'  is for packagers   (packaging changes)
    'test' is for testers     (test only related changes)
    'doc'  is for doc guys    (doc only changes)

COMMIT_MSG is ... well ... the commit message itself.

TAGs are additionnal adjective as 'refactor' 'minor' 'cosmetic'

    They are preceded with a '!' or a '@' (prefer the former, as the
    latter is wrongly interpreted in github.) Commonly used tags are:

    'refactor' is obviously for refactoring code only
    'minor' is for a very meaningless change (a typo, adding a comment)
    'cosmetic' is for cosmetic driven change (re-indentation, 80-col...)
    'wip' is for partial functionality but complete subfunctionality.

Example:

new: usr: support of bazaar implemented
chg: re-indentend some lines !cosmetic
new: dev: updated code to be compatible with last version of killer lib.
fix: pkg: updated year of licence coverage.
new: test: added a bunch of test around user usability of feature X.
fix: typo in spelling my name in comment. !minor

Please note that multi-line commit message are supported, and only the
first line will be considered as the "summary" of the commit message. So
tags, and other rules only applies to the summary.  The body of the commit
message will be displayed in the changelog without reformatting.
```

7.3 Testing

SOSCleaner uses a full suite of unit tests for automated testing during each build. The CI/CD platform for SOSCleaner is Travis-CI. Test code coverage is ~100% and tracked on Coveraalls.

The most important testing, however, is real world testing. So please, contribute in that way all you want. Some examples:

- Run soscleaner against sosreports with different plugins enabled and report back what isn't obfuscated. Report things that aren't obfuscated, plugins that increase run time significantly, or things that just don't look right to you.
- Run different datasets through soscleaner and report things that don't work correctly. Things like:
  - packet captures
  - dumps from various platforms like kubernetes
  - whatever else you can think of

7.4 Bugs and QA

Going hand in hand with Testing is reporting bugs and helping out with Quality Assurance. This is a *very* small open source project, but we do our best to test everything that we can think of. But if you have a use case that's not covered, file a bug! It's the only way SOSCleaner will improve.

7.5 Documentation

Docs for SOSCleaner are written using RestructuredText and hosted at Read The Docs. If you're interested in contributing, please docs admin.

CHAPTER 8

Using soscleaner

8.1 CLI quickstart

8.1.1 CLI help output

The default way to use SOSCleaner is using the command-line application of the same name.

```
Usage: soscleaner <OPTIONS> /path/to/sosreport

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -l LOGLEVEL, --log_level=LOGLEVEL
                        The Desired Log Level (default = INFO) Options are
                        DEBUG, INFO, WARNING, ERROR
  -d DOMAIN, --domain=DOMAIN
                        additional domain to obfuscate (optional). use a flag
                        for each additional domain
  -f FILES, --file=FILES
                        additional files to be analyzed in addition to or in
                        exception of sosreport
  -q, --quiet           disable output to STDOUT
  -k KEYWORD, --keyword=KEYWORD
                        additional keywords to obfuscate. use multiple times
                        for multiple keywords
  -K KEYWORDS_FILE, --keywords_file=KEYWORDS_FILE
                        line-delimited list of keywords to obfuscate
  -H HOSTNAMEPATH, --hostname-path=HOSTNAMEPATH
                        optional path to hostname file.
  -n NETWORK, --network=NETWORK
                        networks to be obfuscatedi (optional). by default it
                        looks through known routes to generate a list from a
                        sosreport
  -u USER, --user=USER  additional usernames to obfuscate in the sosreport or
```

(continues on next page)

(continued from previous page)

```
                        dataset - one user per flag
  -U USERS_FILE, --users-file=USERS_FILE
                        line-delimited list of users to obfuscate
  -o DIRECTORY, --output-dir=DIRECTORY
                        Directory to store soscleaner obfuscated sosreport or
                        dataset
  -m, --macs            disable MAC address obfuscation
```

8.2 Using a config file

If you find yourself having to use additional command line options a lot, you can create a config file at `/etc/soscleaner.conf` to handle these default values for you. Note: Please make sure the config file is own by root for both the UID & GID and that permission is set to READ & WRITE for the user ONLY (0600/-rw——-).

A sample config file:

```
[Default]
loglevel = debug  # the loglevel to run at, default is 'info'
root_domain = example.com  # domain to use for obfuscation
quiet = True # defaults to False, True suppresses output to stdout

[DomainConfig]
domains: example.com,foo.com,domain.com  # additional domains to obfuscate

[KeywordConfig]
keywords: foo,bar,some,other,words  # keywords to obfuscate
keyword_files: keywords.txt  # keyword files to obfuscate

[NetworkConfig]
networks: 172.16.0.0/16  # additional networks to obfuscate

[MacConfig]
obfuscate_macs = False  # True/False (defaults to True) - if False MAC obfuscation␣
↪will not occur
```

8.3 Using within a python prompt

SOSCleaner is a python library at its heart, and can be used in other applications as a library. The following sample is useful when testing SOSCleaner functionality from a python prompt, like when we're writing unit tests and other such incredibly fun activities.

```
from soscleaner import SOSCleaner
cleaner = SOSCleaner()
cleaner.loglevel = 'DEBUG'
cleaner.origin_path, cleaner.dir_path, cleaner.session, cleaner.logfile, cleaner.
↪uuid = cleaner._prep_environment()
cleaner._start_logging(cleaner.logfile)
```

Once the `cleaner` instance has been created, you can begin to populate the data structures. For example:

```
cleaner.hostname = 'somehost'
cleaner.domainname = 'example.com'
cleaner.domains.append('foo.com')
cleaner._domains2db()
```

CHAPTER 9

Annotated Source Code

CHAPTER 10

Proposals and Roadmap

10.1 Python 3

As comfortable as Python 2 is, it is EOL in early 2020. Python 0.4.x will maintain its Python 2 codebase. Python 0.5.0 will be released as early in 2020 as possible. This release will be 100% Python 3.

The current plan is maintain these as separate branches. There was talk of using tools like six. But the changes that come from moving to Py 3 involve a major refactoring of the network obfuscation module. Maintaining that in the same file as the Py2 code is seen as too cumbersome right now.

10.2 API

Currently each `soscleaner` run is an atomic process. If you run soscleaner against the same dataset twice, there's no assurance the same obfuscations will occur. It's probable, but not guaranteed. Additionally, if one dataset is obfuscated, there's no way to add to the dataset at a later time and have the same obfuscation database be used. This use case has been identified by multiple users.

In the 0.5 branch, we're planning work to add this capability. The most obvious path would be to add the capability to store the obfuscation database in something more useful than individual CSV files (a non-SQL database springs to mind). This would allow for querying and appending to existing datasets.

It would also allow to more easily front `soscleaner` workflows with a RESTful API. This feature would allow for a centralized, queryable database of obfuscated data for an enterprise.

10.3 Interacting with Kubernetes

If `soscleaner` was able to be accessed on-demand via an API, interacting with Kubernetes becomes a possibility. An initial thought is to extend `kubectl` with a helper script that would take the output of a supplied command and obfuscate it programmatically. For example:

```
$ kubectl obfuscate get pods -n default
```

The helper script would:

- get the output of `kubectl get pods -n default`
- pass it to `soscleaner` through its API.
- return the obfuscated `kubectl` output.

There's still some work to do here around session data, and how to know if it's a new obfuscation session or to append to an existing database. A sane default could be to obfuscate everything in a given namespace using the same obfuscation database. This is easily inferred from most `kubectl` commands or the kubernetes context.

The primary value for this enhancement would be in allowing output of `kubectl` (and variants like `oc`) to be consistently obfuscated during troubleshooting with outside parties. Currently there's not an automated tool to obfuscate command output from `kubectl` and other CLI based tools that provide interactive output to the user. Additionally, these tools are increasingly common. This would provide a good blueprint for `soscleaner` to fill this need.

CHAPTER 11

Licensing

Soscleaner is released under the GNU General Public License v2. You can find the full text at https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html#SEC1.