OpsLog Documentation

Release v1.1a May, 2019

Jacob Coburn

May 03, 2019

The opslog program is designed to allow operators to take detailed notes quickly and efficiently from the terminal without the need to open additional programs or enter VM's. Multiple operators can take notes on a single machine, without the need to logoff and logon to their own profiles. Each operator's notes, are stored in separate logs, and the current operator can be switched quickly without the need for restarting the terminal. Logs are stored in .csv format, allowing for easy import into any program that accepts it (excel for example).

Entries are added to the log by entering a command at the terminal from anywhere on the system. Entries can contain as little as simple note by the operator, or complete information about a command run on the network. Each log entry contains a timestamp and the operators name, in addition to information provided by the operator. Currently each log entry can hold the following:

- A pre-approved action number
- An ip address/range
- Command syntax
- Multiple flags used to tag entries
- Notes by the operator about the action taken

Using flags to tag log entries allows operators to later search through their logs easily and display entries of interest or entries regarding particular events.

Additionally, the program allows operators to log commands as they are run using the command syntax field. The command input here can, if the operator chooses, be run after logging, giving a timestamped record of when commands where run on the network.

Finally, operator logs can be exported in their .csv format, and multiple operator logs can be merged together to create an msl. Merged logs are automatically sorted properly, allowing an operator to combine their logs from multiple machines into one concise log.

CHAPTER 1

Overview on How to Install and setup this Script

1.1 Simple Installation

1. Ensure python3 is installed on the system
2.
   Copy the opslog.py file to your chosen folder and move to it
   - It is recommended not to move the program file after install
   - A folder dedicated to custom scripts is recommended
3. run `python opslog.py`
4. Follow the prompts to install the program

1.2 Install Location

opslog is installed to the /usr/lib/ops_log/ directory. The folder is created with root privileges and the sticky bit set. This allows the program to access it's configuration file and create operator logs regardless regardless of the user running it (root privileges are not required to run the program or create logs.)

Inside the ops_log/ folder are the config.ini file which the script uses for tracking the current operator, and a operator_logs/ folder which stores the .csv file logs for the operators. Like the main installation folder, the operator_logs folder is created by root, with sticky bit permissions to allow the program to update logs for operators even if they were initially created while another user was logged in.

Below is an sample directory listing of the install location:

```
/usr/lib/:
total 595256
....
drwsrwxrwx  3 root root     4096 Apr 29 13:16 ops_log
...

/usr/lib/ops_log/:
total 8
-rw-rw-rw- 1 root root   54 Apr 29 10:49 config.ini
drwxrwxr-x 4 root root  4096 Apr 29 15:35 html
drwsrwxrwx 2 root root 4096 Apr 29 10:50 operator_logs
```

```
-rwxrwxr-- 1 root root 15216 Apr 29 15:47 opslog.py

/usr/lib/ops_log/operator_logs:
total 4
-rw-r--r-- 1 assessor assessor 616 Apr 29 10:51 test_operator_ops_log.csv
-rw-r--r-- 1 assessor assessor 486 Apr 29 10:57 second_operator_ops_log.csv
```

1.3 Post Install

Once the initial installation is complete, the program can be run from anywhere on the system using the shortcut alias 'opslog'. This is due to an alias being created in the /etc/profile.d/ directory in the opslog_alias.sh file.

If for any reason this does not work and users are unable to use the 'opslog' shortcut after installation, you can manually create this alias by adding the line:

```
alias opslog='/usr/lib/ops_log/opslog.py'
```

in their .bashrc file found in their home directory.

A man page is also created for the program and can be access with the command:

```
man opslog
```

CHAPTER 2

Opslog Usage

This script is used to fill in operator notes automatically in .csv file format. You can use this functions to simply input timestamped notes using the -n option alone. Commands input with the -C option will be executed exactly as entered after logging. Be careful to use single quote marks around commands or notes if they contain anything that bash will try to interpret ($ or ! for example)

2.1 Basic Info

The basic usage and flags:

```
opslog.py [-h | -v | -o | -lo | -so operator] [-p #] [-i a.b.c.d/f]
             [-C 'Command' | -c 'Command'] [-n 'text']
             [-f Flag [Flag ...]] [--cat | -lf | -sf Flag [Flag ...]]
```

2.2 Log File Syntax

The log file for each operator is stored in .csv format; delimited by semicolons (;). The syntax is always the same:

```
date;operator name;flag;paa;ip address;command;executed;note
```

The eight fields are:

```
- Date:     The date and time the entry was made in UTC timezone
    - YYYY-MM-DD HH:MM:SS
- Operator: The operator who made the entry
- Flag:     Tags used in a log entry. These can be used later for searching or
catagorizing entries
- PAA:      The pre-approved action number. This is dependant on mission and crew
lead
- IP:       Any IP address involved with the entry.
- Command   The command syntax used.
- Executed  Field used only when Command field is present
    - 'Yes' if the command was executed after logging
    - 'No' if the command was not executed or failed to execute
- Note      The actual note entry to log.
```

2.3 Administration Arguments

The following arguments are mutually exclusive and either display program information or modify operator settings. If used, they will override any other flags and no log entry will be created.

The admin arguments are:

```
-h, --help            show this help message and exit
-v, --version         Show program version information
-o, --operator        Show the current operator
-lo                   List all operators
-so operator,
 --set-operator operator
                      Set the current operator
```

Most useful are the -o and -so arguments which are used to show/set the operator

2.4 Management Arguments

The following arguments are mutually exclusive and are used to export or merge operator logs.

The management arguments are:

```
--export FILENAME     Export the current log
--merge File1 File2   Merge multiple log files together into one
```

Note: The files can be given in absolute or relative path. If no path is specified the file will output to the current directory.

Note 2: The merge command can accept any number of log files. It will first check to ensure all supplied files are in the correct format, and then ask for the output log name before merging.

2.5 Output Arguments

The following arguments are mutually exclusive and display the current operator's log or selective information in it. If used, they will override any other arguments and no log entry will be created.

The output arguments are:

```
--cat                 Output the current log (can be piped to less/more,
                      head/tail)
-lf                   List all flags used in current operators log
-sf Flag [Flag ...]   Search the log entries for those tagged with Flag(s)
```

2.6 Logging Arguments

The following arguments are not mutually exclusive, with the exception of the -c and -C arguments, and are used to create a log entry in the current operators log. Any or all of the arguments may be used in any order.

The logging arguments are:

```
-p #                    The pre-approved action number
-i a.b.c.d/f            The target ip address/range
-C 'Command'            Command syntax to log before executing
-c 'Command'            Command syntax to log without executing
-n 'text'               Operator notes to include in the log entry
-f Flag [Flag ...]      Flag(s) used to tag the log entry
```

Note 1: When inputting command syntax and notes, use of single quote marks (') are recommended to prevent your shell from interpreting it before logging.

Note 1 Example:

```
>IP='1.2.3.4'
>opslog -c "ping $IP" -n "Testing connectivity to the $IP variable"
>opslog -c 'ping $IP' -n 'Testing connectivity to the $IP variable'
>opslog --cat

2019-04-29 18:59:24;argument_tests;;;;ping 1.2.3.4;no;Testing connectivity to the ip
1.2.3.4 variable
2019-04-29 18:59:42;argument_tests;;;;ping $IP;no;Testing connectivity to the $IP
variable
```

Note 2: Flags can be added with the -f option. Multiple flags may be used if space separated.

CHAPTER 3

Opslog Examples

3.1 Displaying and Changing the Current Operator

The current operator is stored in the programs configuration file and is referenced whenever log entries are made or the log is queried. You can find the current operator by using the `opslog -o` command.

Example:

```
> opslog -o
test_operator
```

Whenever the current operator is changed, the configuration file is updated to reflect the new operator. You can change the current operator using the `opslog -so` command.

Example:

```
> opslog -o
test_operator

> opslog -so new_operator
> opslog -o
new_operator
```

3.2 Creating Log Entries

Log entires are created by using any or all of the *Logging Arguments*. These can be as simple as a timestamped note using `opslog -n 'note'` command, or as complicated as a full entry using all six arguments.

Example 1:

```
> opslog -n 'This is a simple operator note'
> opslog --cat

2019-04-30 13:44:10;Example Operator;;;;;;This is a simple operator note
```

Example 2:

```
> opslog -c 'ping 1.2.3.4' -n 'This entry includes a command'
> opslog --cat
```

```
2019-04-30 13:45:53;Example Operator;;;;ping 1.2.3.4;no;This entry includes a command
```

Example 3:

```
> opslog -p 1 -i '127.0.0.1' -C 'ping -c 4 127.0.0.1' -f 'testing' -n 'This is a full
 note with command execution'
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.027 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.036 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.038 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 59ms
rtt min/avg/max/mdev = 0.027/0.034/0.038/0.007 ms
> opslog --cat

2019-04-30 13:49:14;Example Operator;testing;1;127.0.0.1;ping -c 4 127.0.0.1;yes;This
 is a full note with command execution.
```

Note 1: In all three examples. the `opslog --cat` command is executed to show the contents of the log.

Note 2: In example 2, the 7th field(executed) lists 'no' because the command syntax was entered with the `-c` option. This option only logs the command but does not attempt to execute it.

Note 3: In example 3, the 7th field(executed) lists 'yes' because the command syntax was entered with the `-C` option. This option creates the log entry and then attempts to execute the command exactly as entered. Example 3 also shows the results of the executed command.

3.3 Displaying and Searching the Log

Logs can be easily displayed using the `opslog --cat` command. The log displayed will always be the current operators log only. the output from this command can be piped into other commands as needed such as `head`, `less`, or `grep`.

Example 1:

```
> opslog -o
Example Operator
> opslog --cat

2019-04-30 14:00:03;Example Operator;;;;;;Sample Entry 1
2019-04-30 14:00:06;Example Operator;;;;;;Sample Entry 2
2019-04-30 14:00:31;Example Operator;mission;;;;;Sample Entry 3, with flag
2019-04-30 14:00:38;Example Operator;mission;;;;;Sample Entry 4, with flag
2019-04-30 14:00:49;Example Operator;opschecks;;;;;Sample Entry 5, with flag 2
2019-04-30 14:00:52;Example Operator;opschecks;;;;;Sample Entry 6, with flag 2
2019-04-30 14:01:14;Example Operator;example opschecks;;;;;Sample Entry 7, with 2
flag
2019-04-30 14:01:25;Example Operator;example mission;;;;;Sample Entry 8, with 2 flag
```

Example 2:

```
> opslog --cat | head -n4

2019-04-30 14:00:03;Example Operator;;;;;;Sample Entry 1
2019-04-30 14:00:06;Example Operator;;;;;;Sample Entry 2
```

```
2019-04-30 14:00:31;Example Operator;mission;;;;;Sample Entry 3, with flag
```

Although the logs can be searched by piping to grep, Flags provide a much more efficient way of tagging entries of particular interest. You can list out all the flags used in the current log using the `opslog -lf` command.

Example:

```
> opslog --lf

    Below are the flags being used in the current log

        Count      Flag            Entries
        -----      -----           -------
        3          opschecks       [5, 6, 7]
        3          mission         [3, 4, 8]
        2          example         [7, 8]
```

You can also search for and display log entries based on the flags the entry was tagged with using the `opslog -sf flag` command. The command can accept multiple flags in it's search.

Example 1:

```
> opslog -sf opschecks

2019-04-30 14:00:49;Example Operator;opschecks;;;;;Sample Entry 5, with flag 2
2019-04-30 14:00:52;Example Operator;opschecks;;;;;Sample Entry 6, with flag 2
2019-04-30 14:01:14;Example Operator;example opschecks;;;;;Sample Entry 7, with 2
flag
```

Example 2:

```
> opslog -sf example mission

2019-04-30 14:00:31;Example Operator;mission;;;;;Sample Entry 3, with flag
2019-04-30 14:00:38;Example Operator;mission;;;;;Sample Entry 4, with flag
2019-04-30 14:01:14;Example Operator;example opschecks;;;;;Sample Entry 7, with 2
flag
2019-04-30 14:01:25;Example Operator;example mission;;;;;Sample Entry 8, with 2 flag
```

3.4 Exporting and Merging Logs

Once the logs are complete, the can be exported by using the `opslog --export` command and specifying the export location. The location can use absolute or relative path, and will output to the current directory if only a filename is given

Example:

```
> ls -l ~/tmp/
total 0
> opslog --export ~/tmp/log.csv
Log file successfully exported
>ls -l ~/tmp/
total 4
-rw-r--r-- 1 assessor assessor 594 Apr 30 10:24 log.csv
> cat ~/tmp/log.csv
2019-04-30 14:00:03;Example Operator;;;;;;Sample Entry 1
```

```
2019-04-30 14:00:06;Example Operator;;;;;;Sample Entry 2
2019-04-30 14:00:31;Example Operator;mission;;;;;Sample Entry 3, with flag
2019-04-30 14:00:38;Example Operator;mission;;;;;Sample Entry 4, with flag
2019-04-30 14:00:49;Example Operator;opschecks;;;;;Sample Entry 5, with flag 2
2019-04-30 14:00:52;Example Operator;opschecks;;;;;Sample Entry 6, with flag 2
2019-04-30 14:01:14;Example Operator;example opschecks;;;;;Sample Entry 7, with 2
flag
2019-04-30 14:01:25;Example Operator;example mission;;;;;Sample Entry 8, with 2 flag
```

If for any reason multiple logs need to be combined, the `opslog --merge` command can do so. The command takes any number of files as arguments, checks these files to ensure they are properly formated log files, and merges them together into one log.

Example:

```
> ls -l
total 8
-rw-r--r-- 1 assessor assessor 138 Apr 30 10:29 merg1_log.csv
-rw-r--r-- 1 assessor assessor  92 Apr 30 10:30 merg2_log.csv
> cat merg1_log.csv
2019-04-30 15:28:32;merg1;;;;;;Sample entry 1
2019-04-30 15:28:41;merg1;;;;;;Sample entry 2
2019-04-30 15:29:19;merg1;;;;;;Sample entry 5
> cat merg2_log.csv
2019-04-30 15:28:55;merg2;;;;;;Sample entry 3
2019-04-30 15:29:03;merg2;;;;;;Sample entry 4
> opslog --merge merg1_log.csv merg2_log.csv
Checking files...
All files matches log format.
Enter destination filename: merged_log.csv
Merge Successful
> ls -l
total 12
-rw-r--r-- 1 assessor assessor 138 Apr 30 10:29 merg1_log.csv
-rw-r--r-- 1 assessor assessor  92 Apr 30 10:30 merg2_log.csv
-rw-r--r-- 1 assessor assessor 230 Apr 30 10:33 merged_log.csv
> cat merged_log.csv
2019-04-30 15:28:32;merg1;;;;;;Sample entry 1
2019-04-30 15:28:41;merg1;;;;;;Sample entry 2
2019-04-30 15:28:55;merg2;;;;;;Sample entry 3
2019-04-30 15:29:03;merg2;;;;;;Sample entry 4
2019-04-30 15:29:19;merg1;;;;;;Sample entry 5
```