POFR

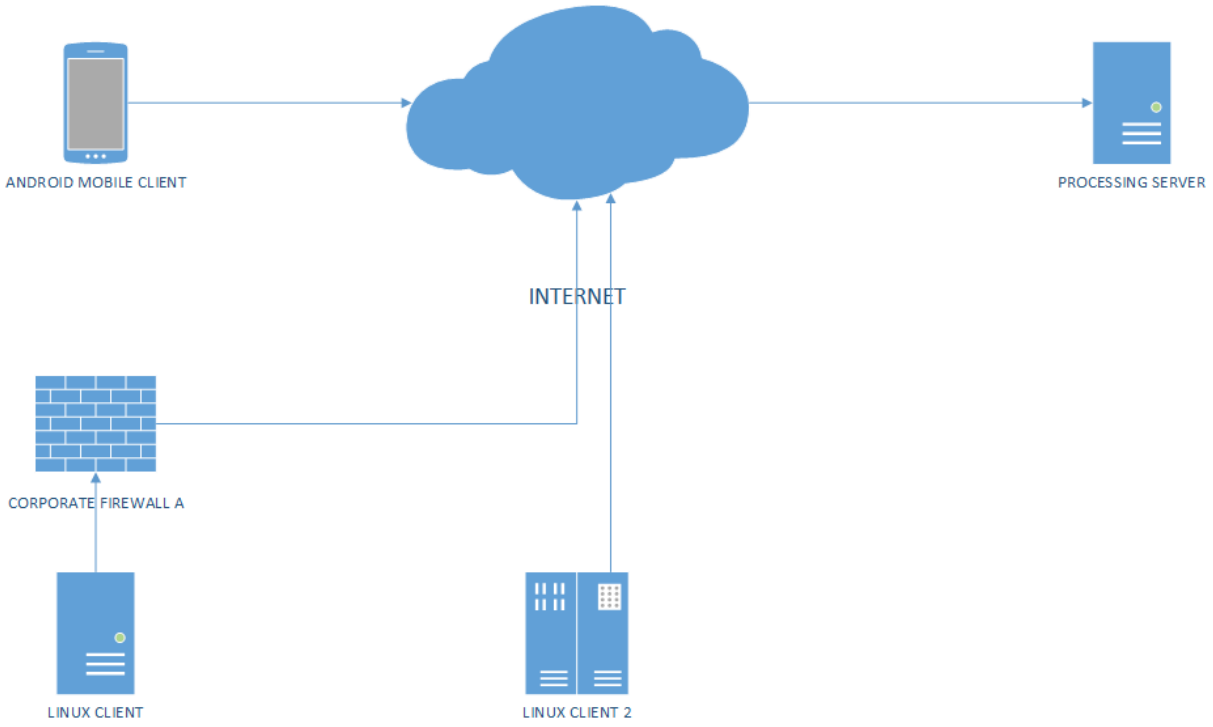
Penguin OS Forensic Recorder

Technical Operations and User Manual

Introduction

The Penguin OS Flight Recorder collects process execution, file access and network/socket endpoint data from the Linux Operating System. Like an aircraft flight recorded (or black box), its main purpose is to reliably reproduce/replay OS level events that concern process execution, file access and network endpoint creation from each of the monitored Linux clients. IT experts (security analysts, system administrators, devops engineers can then use the collected information to:

- Examine/interrelate in great detail processes, file access and network endpoint events in a monitored system.
- Detect computer account involvement misbehaving apps and/or malware.
- Conduct post mortem evidence after security compromises with the evidence stowed away from the monitored system.
- Conduct incident threat response exercises and see their effect on Linux systems.



POFR uses a client/server architecture. Clients are the systems to monitor and they push data to a server via the SSH protocol. The server parses the data and updates a Relational

Database that is used to store and present the data for further analysis. Emphasis is given on minimizing the computational overhead and implementation complexity of the monitoring process. No kernel modules, no agents exposing open network ports to obtain the data and no interference/reliance with system distro specific modules are required on the client systems. The idea is to easily install the POFR within minutes, register it to a server and start pushing data to the server.

Compatibility

POFR clients have been tested with:

- CentOS/RHEL/ALMALinux versions 7 and 8
- Fedora 32/33/34
- Recent versions of Ubuntu.

For a POFR server, we recommend a Fedora 33/34 distro OR the sample KVM and Docker images provided.

Dependencies and preparation requirements

A compatible Linux distribution (see section Compatibility). Everything needed by the client and server components is provided, including an up-to-date PERL distribution with all the modules installed. No need to install system wide PERL modules or many packages for the software to function properly. However, you need to do some planning:

- The POFR clients can reach port 22 (SSH) of the POFR server (directly or via NAT). No special hardware requirements, most of the computational resource intensive parts are performed on the POFR server (see below). Even on busy systems, no more than a single core and 400 Megs of RAM are required to run the client software.
- The IP address, FQDN and SSH keys of the POFR server need to remain the same throughout the monitoring session.
- For the POFR server, one needs to ensure adequate hardware requirements: POFR is resource hungry on the server, so the more resources you have, the better. A 32 core, 24 Gb RAM and a few Tb of disk space on /home and /var filesystems should be sufficient to monitor 8-10 busy devices. You need a separate server instance to simultaneously monitor more devices at an adequate speed, so you need to partition the load to more servers/VM instances. For a single server instance use the following figures as a resource allocation guide:

- Disk space: 180-250 Mbytes per hour of continuous monitoring
- RAM: 3-4 Gigs per client
- (v)CPU cores: 4-8 cores per client

We strongly recommend to run both client and software with **SELinux enabled**. The software has been engineered to function with or without SELinux enabled, but you should certainly not encounter problems if SELinux is enabled.

POFR Client Installation

To install the POFR client, follow the procedure outlined below as the **root user**. Choose a directory not reachable by ordinary system users and clone the git repo:

git clone https://github.com/gmagklaras/POFR.git

Change into the cloned git repo directory and unpack the proper POFR PERL distribution based on what sort of distribution you run. For example, if you run a Fedora Linux client, type the following:

cd POFR
tar xvfz pofrperlfedorax86_64.tar.gz

When the untaring of the compressed tarball completes, you should have a ‘pofrperl’ directory. This is the directory that contains the specially made PERL distribution that the client runs with all the necessary modules pre-installed. As an option, if you would like to save space, you can delete the rest of the compressed pofrperl tarballs. That’s all, the software for the POFR client is now installed.

POFR Client Registration

POFR client registration is a process that contains many checks and looks a bit cumbersome. However, the checks are necessary to ensure that we send the data to the right server. You do not want your client data to end up in the wrong server (or worse to a server that impersonates the real server). This section will walk you through that process and explain it step by step.

If you know the IP address and the registration password for your POFR server, change to the ‘client’ directory and invoke the ‘pofrcreg.pl’ client registration script. For example, if your POFR server is called mypofrserver.domain.com

cd client
./pofrcreg.pl --server mypofrserver.domain.com

The registration client will send a request to the POFR server via the SSH protocol. All requests are sent to a designated userid of the POFR server (by default pofrsreg). The client registration should respond by informing you of the server it will try to send the request to and it will ask you permission to proceed:

Hostname is : mypofrserver.domain.com
Client IP is : 192.168.18.24
33373436-3933-5a43-3332-303941394a371728960233
pofrcreg: OK. Connecting to the specified POFR server: mypofrserver.domain,com to send our registration request.
scp -p ./request33373436-3933-5a43-3332-303941394a371728960233.luarm pofrsreg@mypofrserver.domain.com:~/
Proceed [y/N]:

If you are certain that this is the POFR server you wish to connect to, respond with a ‘y’ to the proceed prompt.

If this is the very first time you register this client to the server, the SSH protocol will ask you to verify that the SSH key fingerprint of the POFR server is the proper one:

The authenticity of host 'mypofrserver.domain.com (192.168.18.24)' can't be established.
ECDSA key fingerprint is SHA256:EMjv4RMWrk6+vartverX6d8SuiJElKi8IXMl4tqIX+rCs.
ECDSA key fingerprint is MD5:e3:03:e8:ab:37:37:2f:29:48:e9:b6:9c:b9:43:67:eb.
Are you sure you want to continue connecting (yes/no)?

Again, if you are certain that this is the proper server, type ‘yes’. The very next thing is a prompt to type the registration password:

pofrsreg@mypofrserver.domain.com's password:

Type the password for the default registration user ‘pofrsreg’ and once you hit ‘Enter’, if you entered the password correctly, you should get a verification by the client:

pofrcreg: OK. Request sent successfully to server mypofrserver.domain.com

.pofrcreg.pl: Waiting for the POFR server mypofrserver.domain.com to respond on our request...

At this stage, the POFR client has successfully sent a **registration request** to the server. The sent request has to be acted upon on the server side. If you have control of the POFR server, you need to switch to your server SSH session and execute the **server/pofrsreg.pl** script (refer to the POFR Server Installation and Administration Section) to complete the registration. If you do not have control of the POFR server, you need to follow the instructions of your POFR server admin.

The POFR client will wait a bit and then ask you for the registration password again, this time to download the **registration response** credentials from the server:

scp -pr pofrsreg@anaximander.steelcyber.com:~/response33373436-3933-5a43-3332-303941394a371729070943.reg ./

Instruct the registration process to proceed in the process of downloading the server registration request:

Proceed [y/N]:y
pofrsreg@anaximander.steelcyber.com's password:

If all goes well, you should observe two things:

- The response from the client that all completed well:
  ##
  #pofrcreg:STATUS:OK.Client33373436-3933-5a43-3332-30394139 #
  #was registered at the POFR server: mypofrserver.domain.com. #
  ##
- Under the ‘client’ directory, you should see four files amongst the client code:
  - **.lcaf.dat** : File containing authentication credentials for the server
  - **response[NUMERICCLIENTID].reg:** A unique client ID response returned from the server containing registration status of the server.
  - luarm.rsa: A file containing the private RSA key of the POFR client. This is NOT the root/system wide RSA key but keys for the POFR data transmission process.
  - luarm-rsa.pub: A file containing the public RSA keys of the POFR client. Again, this is NOT the root/system wide public RSA key but keys for the POFR data transmission process.

If you reached this stage, congratulations, you have registered your client and you should be ready to begin data transmission. If not, it is best to delete the 4 files mentioned before of the client directory and start again.

POFR Server installation

A server installation is a little bit more complex than that of a client. We recommend a Fedora 33/34 distro for best results, although the procedure outlined below in steps could be adapted for any Linux distro.

As the POFR server runs performance intensive workloads, it is recommended to run it on a dedicated server, physical or virtual (see Section '***Dependencies and preparation requirements***' of this document).

Step 1: Start as the root user and choose a directory not reachable by ordinary system users and clone the git repo:

git clone https://github.com/gmagklaras/POFR.git

Step 2: Change into the cloned git repo directory and unpack the proper POFR PERL for a Fedora distro as shown below:

cd POFR
tar xvfz pofrperlfedorax86_64.tar.gz

Step 3: Install 'scponly' and a MariaDB RDBMS server. POFR has been tested with the default MariaDB version of an actively maintained Fedora distro (10.4.x for Fedora 33 and 10.5.x for Fedora 34):

dnf install -y scponly mariadb mariadb-server
systemctl enable mariadb.service
systemctl start mariadb.service

Step 4: Once the above packages are installed, check that the MariaDB server has started properly with the following command:

systemctl status mariadb.service

and if all is well, you should get a response like the one below:

Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
Active: active (running) since Fri 2021-06-04 16:08:58 CEST; 36s ago
….
Status: "Taking your SQL requests now..."
Tasks: 17 (limit: 4664)
Memory: 69.5M
CPU: 453ms
CGroup: /system.slice/mariadb.service
└─17349 /usr/libexec/mariadbd --basedir=/usr

Step 5: Continuing using the root account, secure your fresh MariaDB installation by running the following script:

mysql_secure_installation

The script will ask your input on a number of choices, outlined below. The current password on a fresh installation is blank (press Enter):

Enter current password for root (enter for none):
OK, successfully used password, moving on…

You should switch to unix_socket authentication:

Switch to unix_socket authentication [Y/n] Y
Enabled successfully!
Reloading privilege tables..
... Success!

Choose a secure root password for the MariaDB server and set this password up by entering it twice:

Change the root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!

Remove the anonymous users from your fresh installation:

Remove anonymous users? [Y/n] Y
... Success!

Disallow remote root login to maintain the security of your installation:

Disallow root login remotely? [Y/n] Y
... Success!

Remove the test database and access to it:

Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Finally, please reload the MariaDB privilege tables to ensure that the security changes you made are applied to the running instance of the MariaDB server:

Reload privilege tables now? [Y/n] Y
... Success!

Cleaning up...

Step 6: At this point, check that you can login to the RDBMS with your newly chosen MariaDB root password:

mysql -u root -p

If you have changed properly, you should have no problem getting into the MariaDB SQL prompt:

Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 10.5.10-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>

While you are there, see if you have permission to create a database by typing the following at the prompt:

MariaDB [(none)]> create database lhlt;

If you get a:

Query OK, 1 row affected (0.001 sec)

you have proved that the basic permissions are set for MariaDB. Exit the MariaDB prompt by typing ‘quit’.

Step 7: Now navigate to the ‘server’ directory of the POFR repo

cd server

and create the schema for the host lookup table (type your chosen MariaDB root password at the prompt and press Enter):

cat LHLT.sql | mysql -u root lhlt -p

If you do not get any error/warning messages, you have probably created the lookup table properly. You can verify that from the SQL prompt with this:

```
MariaDB [(none)]> describe lhlt.lhltable;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| hostid  | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| uuid    | varchar(36) | NO   |     | NULL    |                |
| cid     | varchar(59) | NO   |     | NULL    |                |
| ciduser | varchar(32) | NO   |     | NULL    |                |
| lastip  | varchar(35) | YES  |     | NULL    |                |
| ryear   | smallint(6) | NO   |     | NULL    |                |
| rmonth  | tinyint(4)  | NO   |     | NULL    |                |
| rday    | tinyint(4)  | NO   |     | NULL    |                |
| rmin    | tinyint(4)  | NO   |     | NULL    |                |
| rhour   | tinyint(4)  | NO   |     | NULL    |                |
| rsec    | tinyint(4)  | NO   |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
11 rows in set (0.004 sec)
```

If you get something like the above output, you are all set. Type ‘quit’ to exit back to the OS shell.

Step 8: Create the authentication database file (.adb.dat). You should be under the server folder and yhe easiest way to do this is the following:

echo "root,lhlt,MY_MARIA_DB_ROOT_PASS,localhost" > .adb.dat

Important: Replace the string **MY_MARIA_DB_ROOT_PASS** with whatever root password you chose when you secured your MariaDB instance (Step 5).

Step 9: An equally important next task is to create the POFR registration user ‘pofrsreg’. This user will be used to register POFR clients in the server. You should create this user locally on the POFR server with the following command (as user root):

useradd -d /home/pofrsreg -s /bin/scponly pofrsreg

Create a secure password for that user by typing and verifying it after issuing this command:

passwd pofrsreg

Take good care of this password. You will be quoting/using it for every client registration (refer to Section ‘***POFR Client registration***’ of this document).

This concludes the installation of the POFR server.

POFR usage

After completing the client and server installation procedures, the process of daily usage of POFR involves starting and stopping the client, as well as logging into the server to browse/search the collected data. This section will also demonstrate basic troubleshooting steps to ensure proper operation of the POFR engine.

Starting and stopping the POFR client

POFR chooses a simple/minimal way to start the client outside the systemd framework. The systemd framework is excellent for starting and stopping services, the decision for not using it for POFR is a design choice. A growing number of Linux malware attacks

target the systems via systemd. In addition, if one wishes to forensically audit aspects of systemd's operation, it is best to not insert code/dependencies that can affect its performance/state. For these reasons and while it is feasible to make a system service file, it is recommended not to use systemd for the purposes of starting/stopping the POFR client. Use instead the scripts and procedure outlined below.

As the 'root' user, fire up your favourite terminal multiplexer (screen/tmux/other) and execute the startup script. For example:

screen -R client

and then in the screen session, change to the POFR/client directory location and issue something like the following:

./startclient.pl >> /var/log/pofrclient.log 2>&1

At that point and if you have SSH connectivity between the client and the server, all the processes to collect and periodically send the data to the server. You could then detach from the 'screen' session. You could also inspect the log file you specified in the startup commands (/var/log/pofrclient) to see the status messages for the client.

If at any time, you wish to stop the client, you could navigate to the POFR/client directory and call 'stopclient.pl' script:

./stopclient.pl

The above command will stop all the necessary processes.

Should you wish to restart the client process you could re-attach to the screen session (screen -r client) and reissue the 'startclient.pl' script command before you detach again.

The above process is useful for interactive use on the client for testing our troubleshooting purposes. Once you are are done with testing and troubleshooting, a more automated approach that does not involve a terminal multiplexer, you could just use a cronjob as the root account by adding to the root account's crontab the following lines (crontab -e):

HOME=[INSTPATH]/POFR/client
***/5 * * * * [INSTPATH]/POFR/client/startclient.pl >> /var/log/pofrclient.log 2>&1**

where **[INSTPATH]**=the PATH of the directory under which you have installed POFR

This would ensure that the services start automatically without terminal multiplexers and remain up for as long as the client OS is running AND has connectivity with the POFR server.