



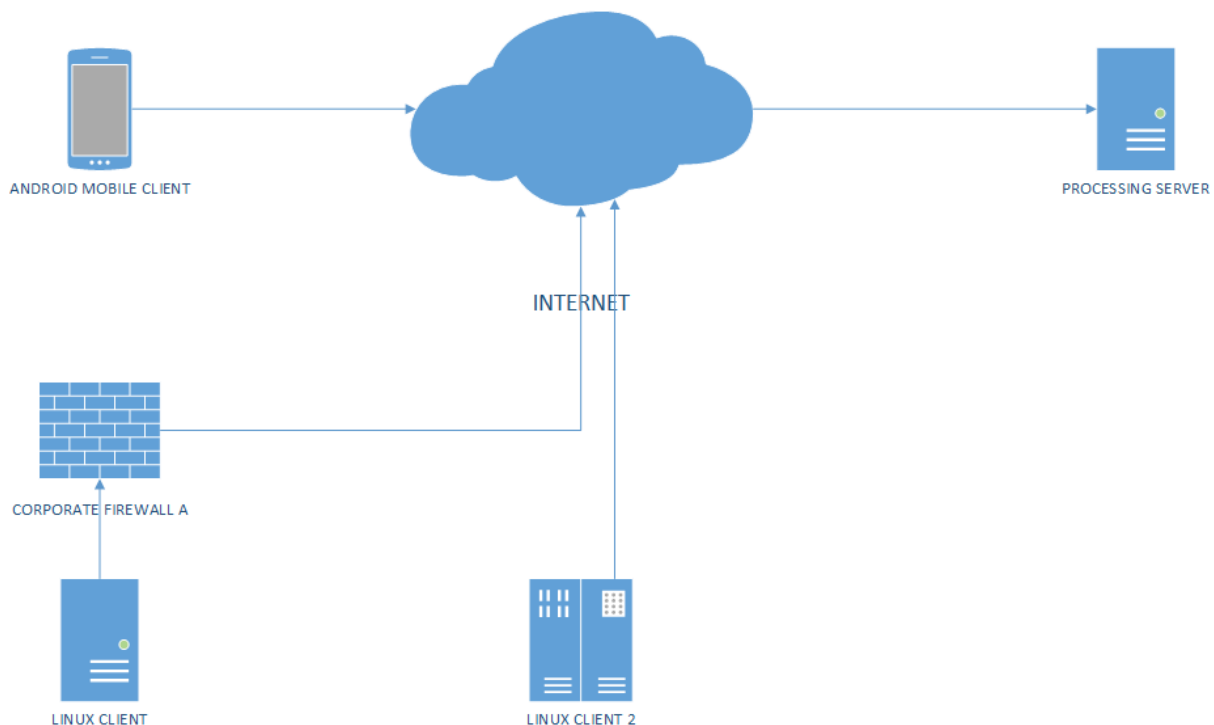
Technical Operations and User Manual



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 either a CentOS 7 or 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 ‘pofcreg.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.