POFR

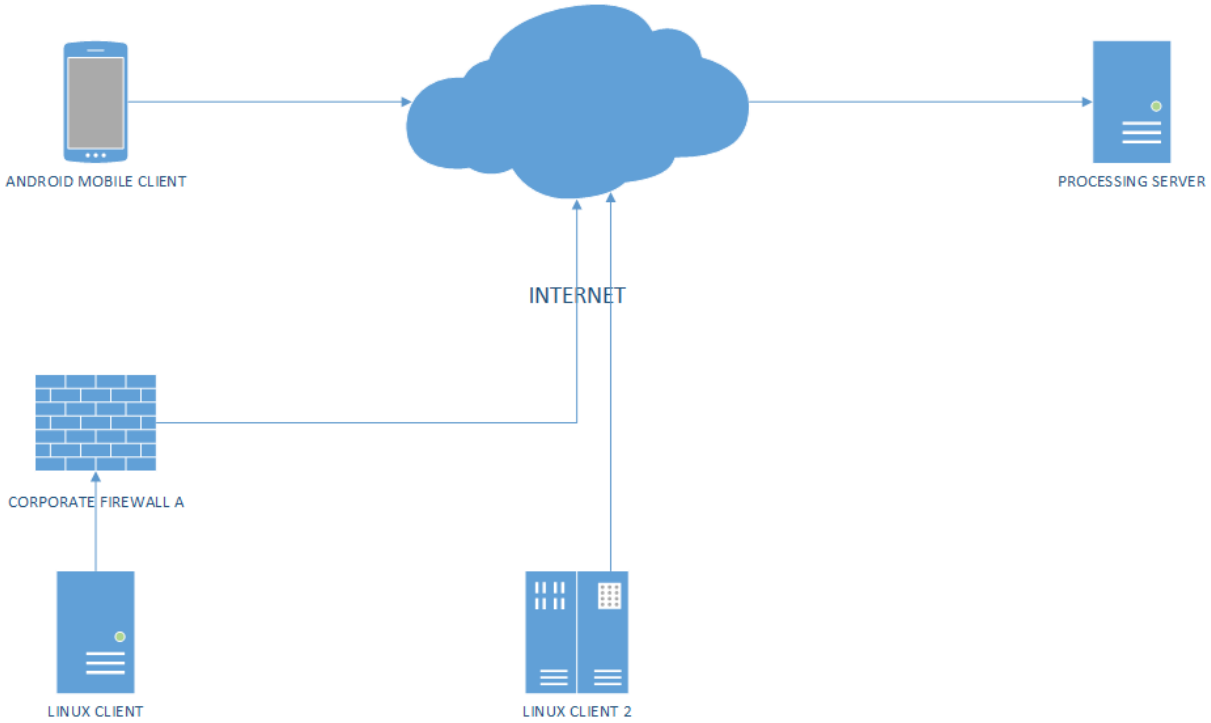
Penguin OS Forensic Recorder

Technical Operations and User Manual

A)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 misbehaving computer accounts, applications.
- Issues with DevOps frameworks at Operating System level
- 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.
- Comply to security/audit requirements.
- Study the collected network endpoint and process execution traffic over time to obtain OSINT related data (honeypots).



There are other open source projects that perform similar but not identical functionality to POFR. The closest alternative is the osquery project which can provide a detailed picture of the state of infrastructures, including some of the information mentioned in the above bullet points. However, in direct contrast to osquery, POFR is better suited to collect information in a time series fashion, accumulating data over a period of time, using less computational overhead and complexity.

POFR uses a client/server architecture. Clients are the systems to monitor and place the data on the server. The server parses, stores and acts as the gateway to analyze the data, by using an RDBMS layer. POFR was designed with simplicity and reliability in mind, dictated by the following principles:

- No kernel modules and/or agents exposing open network ports to obtain the data should run on client systems.
- The data collection should occur with minimal computational overhead on the client systems.
- Clients push all the data to the server using SHA message digests for data integrity in an encrypted manner by using the SSH protocol. Port 22 can traverse easily firewall/NAT schemes and the system should easily run on TCP/IP networks.
- No interference/reliance with system distro specific software modules is required on the client systems. Everything is provided by the source code.
- Data analysis and correlation is done exclusively on the RDBMS layer of the server. The SQL schema is simple enough (no complex index or foreign key dependencies) so that the data can be easily ingested in noSQL databases or other logging/visualization/search frameworks (Elasticsearch, Kibana).

B)Compatibility

B1) POFR clients

POFR clients have been tested with:

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

B2) POFR servers

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

C)Dependencies and preparation requirements

A compatible Linux distribution (see section B ‘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.

D)POFR Client Installation

To install the POFR client, follow the steps outlined below as the **root user**.

Choose a directory not accessible 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.

D1)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.

E)POFR Server installation

A server installation is a little bit more complex than that of a client. We recommend a Fedora 33/34 Workstation 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

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 4: Ensure that an sshd server is installed, is running and is not blocked by the local firewall:

A Fedora 33/34 installation normally includes by default an SSH server. If, for for some reason it does not, as user root, you can install it by typing:

dnf -y install openssh-server

You will then need to ensure that the server is systemd enabled and started properly:

systemctl enable sshd.service
systemctl start sshd.service

Check that the sshd service is running by confirming its status:

systemctl status sshd.service

The above command should give you a status similar to the one below, if sshd is installed, enabled and started properly:

```
● sshd.service - OpenSSH server daemon
     Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: disabled)
     Active: active (running) since Sun 2021-06-20 17:55:47 CEST; 32min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 883 (sshd)
      Tasks: 1 (limit: 9495)
     Memory: 7.6M
        CPU: 1.922s
     CGroup: /system.slice/sshd.service
             └─883 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
...
```

Finally, the last check is to ensure that the sshd service is not blocked by the local firewall and thus ensure your POFR clients will be able to contact the server. The default Fedora distro firewall config is controlled by firewalld-service which should be up and running (check with **systemctl status firewalld.service**). To check that the sshd.service is not blocked by firewalld.service, type:

firewall-cmd –list-all

A non blocking configuration would list ‘ssh’ in the ‘services:’ line:

```
FedoraWorkstation (active)
  target: default
```

icmp-block-inversion: no
interfaces: enp1s0
sources:
services: dhcpv6-client mdns samba-client ssh
ports: 1025-65535/udp 1025-65535/tcp
protocols:
…

If you do not see the ssh service listed in the services line, you can enable it by typing:

firewall-cmd --add-service=ssh –permanent

The above command should normally indicate ‘success’ if it is executed properly and at that point, you should re-run the command:

firewall-cmd –list-all

and verify that ‘ssh’ is included in the ‘services:’ line.

Step 5: Provisionally install your preferred terminal multiplexing utility (screen/tmux) and the ‘htop’ utility. These tools will come in handy to start/stop the server and check performance utilization on your multi-core server:

dnf -y install screen tmux htop

Step 6: 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 7: 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 8: 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 9: 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 10: 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.

F)POFR usage

After completing the client and server installation procedures (Sections D and F of this manual), using 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.

F1)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 methods 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 method is useful for interactive use on the client for testing our troubleshooting purposes. Once you are done with testing and troubleshooting, a more automated method 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

For instance, if you have installed the git repo under /root, this is what you enter:

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.

F2)Checking data reception on the POFR server

After completing one or more POFR client registrations (Section D1) and starting the monitoring process (Section F1), you can check that data are received properly on the server side. Starting on the client side (POFR/client directory), take a look at the contents of the ***response[ID].reg*** file which was created on the client as a result of the client registration process. For example, a POFR client might have a file looking like this:

responseab6fcbe3-3d1b-460d-96aa-11cde64b246a15610790.reg

If you take a look at the contents of the file, you see a single status line looking like the one below:

Status:GRANTED#71cc46a1b55e4a3f6b64af63d4cc5fc4#a7f1cb7c72f3bbcaea02105b358dc497e3e882519edb2c712ed704f43a54f613

The ‘GRANTED’ string indicates that the registration was granted by the POFR server. The second string “71cc46a1b55e4a3f6b64af63d4cc5fc4” is the userid of the account that the data is sent via SSH on the POFR server. The third is a unique registration ID for the POFR client employed by the POFR server to detect/avoid duplicate registrations .

You can ssh now on the POFR server, switch to the root user and navigate to the /*home*/71cc46a1b55e4a3f6b64af63d4cc5fc4 directory. If the client is pushing data to the server successfully, you should be able to do an ls -ltah and see an output like the one below:

-rw-r--r--. 1 71cc46a1b55e4a3f6b64af63d4cc5fc4 71cc46a1b55e4a3f6b64af63d4cc5fc4 1.5M Jun 20 19:24 1624209875762201#+0200#dcf647ebdf3b2018c745743f3c3d300c4eb08f4f501dcededd9850cb408ff4c7.tar
-rw-r--r--. 1 71cc46a1b55e4a3f6b64af63d4cc5fc4 71cc46a1b55e4a3f6b64af63d4cc5fc4 1.5M Jun 20 19:23 1624209830214172#+0200#6a2dd9da9be0cd93d9532e083a2a60c0b1405119a2bdc26b2aeaa611972caf06.tar
-rw-r--r--. 1 71cc46a1b55e4a3f6b64af63d4cc5fc4 71cc46a1b55e4a3f6b64af63d4cc5fc4 1.5M Jun 20 19:23 1624209784671618#+0200#fcb7968ab73de5c23b69c73a65f2cd2d032450d06469ab542f6dc7b2bb584021.tar
-rw-r--r--. 1 71cc46a1b55e4a3f6b64af63d4cc5fc4 71cc46a1b55e4a3f6b64af63d4cc5fc4 1.5M Jun 20 19:22 1624209739033015#+0200#5687b186918b71cb932a2d1d0530ee9c0cc37f5986fcc456a9df26a502bc3f65.tar
-rw-r--r--. 1 71cc46a1b55e4a3f6b64af63d4cc5fc4 71cc46a1b55e4a3f6b64af63d4cc5fc4 1.5M Jun 20 19:21 1624209693446555#+0200#82416a1edb42f15dd4130bf694613d5ccedbff04fb443335c8c1d7ec365bdc34.tar
…

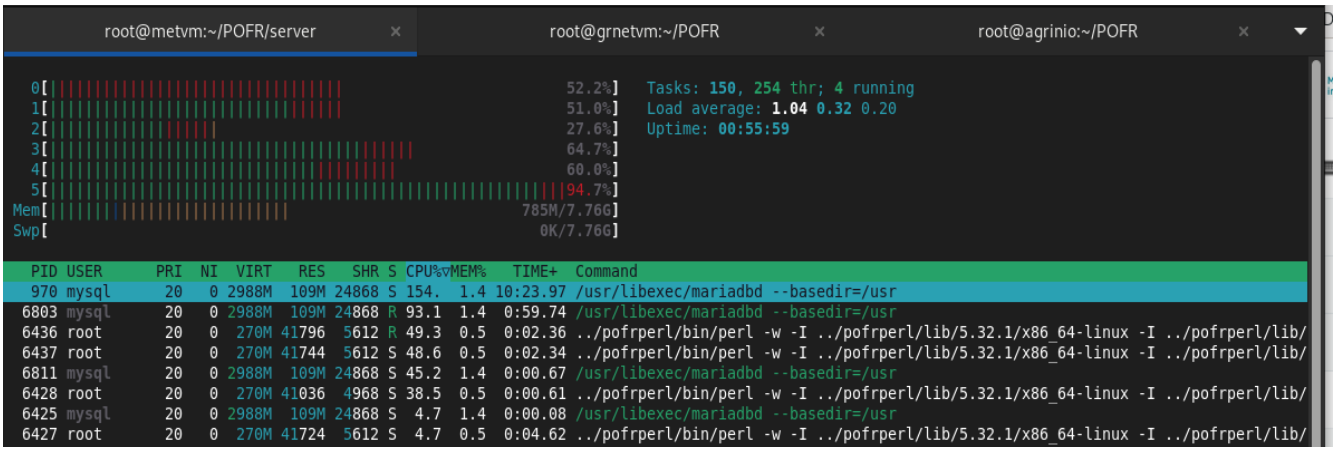
Each one of these tar files is a signed with a SHA message digest and time-stamped snapshot of the client data (look at the time stamps to see the frequency of reception) that was uploaded through the SSH channel. This indicates proper communication between the POFR client and server entities. If, after several minutes of starting the POFR client, you have no files on the POFR server, please refer to the troubleshooting section.

F3)Starting and automating the server data parsing

Once you have confirmed proper data reception on the server (Section F2), you will need to automate the process of parsing the received data. Prior doing that and if you have installed the server for the first time, it's worth starting the data parsing process manually, to check that everything is working. It's a good idea to grub a cup of your favorite beverage and let the client(s) send enough data to the server (say you leave the server collect data from active clients for a couple of hours). Then, as user 'root', change to the git cloned repo directory and in particular the 'server' subdirectory and then issue the following command:

./newdeltaparseproc.pl > run01.txt 2>&1

The above command will call the main data parsing script and will redirect standard and error output to the file 'run01.txt'. This file will contain various debugging messages that can inform you of what the various processing threads are doing. Once you issue the above command, a good test is to run '**top**' (or even better an '**htop**' that can also give you threaded process information).



If you see various processes POFR Perl interpreter processes and a busy MariaDB RDBMS process as displayed above, things should be progressing well. You could also like to verify that from the command line (again user 'root) with the command:

ps auxwww | grep pofr

If all is well, you should get multiple instances of the POFR Perl client, as shown below:

root 6427 2.5 0.5 276624 41744 pts/0 S 18:47 0:08 ../pofrperl/bin/perl -w -I ../pofrperl/lib/5.32.1/x86_64-linux -I ../pofrperl/lib/5.32.1 ./newdeltaparseproc.pl
root 6439 1.9 0.5 276896 41772 pts/0 S 18:47 0:06 ../pofrperl/bin/perl -w -I ../pofrperl/lib/5.32.1/x86_64-linux -I ../pofrperl/lib/5.32.1 ./newdeltaparseproc.pl
root 6440 0.0 0.4 275528 39428 pts/0 S 18:47 0:00 ../pofrperl/bin/perl -w -I ../pofrperl/lib/5.32.1/x86_64-linux -I ../pofrperl/lib/5.32.1 ./newdeltaparseproc.pl

The 'run01.txt' file should contain output similar to the one below:

parseproc.pl Error: Could not detect /dev/shm/luarmserver/net dir...Fresh boot? Creating it...
user 71cc46a1b55e4a3f6b64af63d4cc5fc4:size of myprocfiles is 3284 and of of threadflags is 0
Processing POFR server: metvm on IP: 192.168.122.244
Running on 6 server cores and having 1 active users.
Users are: 71cc46a1b55e4a3f6b64af63d4cc5fc4
Processing user 71cc46a1b55e4a3f6b64af63d4cc5fc4
…

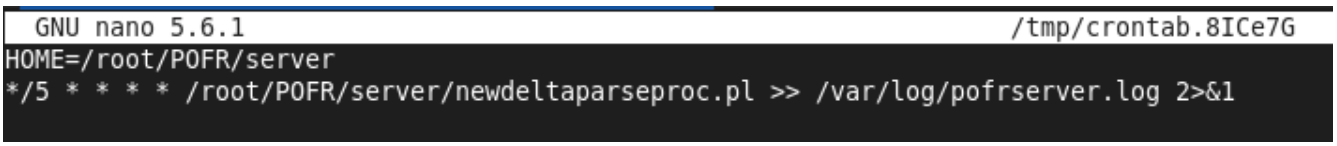
This indicates that the newdeltaparseproc.pl script is creating processing threads to parse the collected client data in parallel. If you do not see active processes as described above, then the output file should contain an error message to help locate the source of the problem. In such a case, you might need to consult again Section E of this manual.

If you have success in seeing active parsers, you can now proceed with the process of automating the execution of the newdeltaparseproc.pl script by means of creating (user root) a crontab entry with the following content:

HOME=[INSTPATH]/POFR/server
***/5 * * * * [INSTPATH]/POFR/server/newdeltaparseproc.pl >> /var/log/pofrserver.log 2>&1**

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

For example, if you installed the repo under /root, you write the following:



This should be calling the parser script every 5 minutes.