Chronicle Forwarder FAQ

Short link: go/forwarder-faq
Created date: 2018-12-17
Last updated date: 2022-04-03

About this document

This is a living document that people should add more questions and answers regarding matters related to forwarder. This document consists of three main sections: Docker based forwarder FAQ, Windows based forwarder FAQ, and Other forwarder related FAQ.

Please feel free to enrich or update this document so that it will be useful for all of us. Thank you for your contribution in advance! :)

Docker Based Forwarder FAQ

Forwarder Logs

Linux:

Run the following command on the forwarder host: **sudo docker logs cfps > logs.out > logs.err**

Windows:

The log files are under this dir : **C:\Windows\Temp**

Error Contacting Notary Server

Problem:

Customer fails to pull docker image from Google Container Registry (GCR). The docker pull command fails as follows:

$ docker pull gcr.io/chronicle-container/cf_production_stable
Using default tag: latest
Error: error contacting notary server: denied: Token exchange failed for project 'gcr.io:chronicle-container'. Please enable or contact project owners to enable the Google Container Registry API in Cloud Console at https://console.cloud.google.com/apis/api/containerregistry.googleapis.com/overview?project=gcr.io:chronicle-container before performing this operation.

Resolution:

GCR does not use Notary to deliver trusted container image. Customer should disable Docker engine to contact Notary while pulling image from GCR. To do so, customer should unset the DOCKER_CONTENT_TRUST environment variable.

$ env | grep -i docker
DOCKER_CONTENT_TRUST=1

$ unset DOCKER_CONTENT_TRUST
$ env | grep -i docker
// DOCKER_CONTENT_TRUST should not show up.

Files within the Container were Disappearing

Problem:

The customer would run the forwarder within the container and the first time it was run, things worked. The second and subsequent times did not work and (when looking inside the container), files were missing.

Resolution:

This issue has been seen exactly once on a CentOS system. As a workaround, a special container was created using an experimental "--squash" flag and provided to the customer. Please contact the CI team if this problem is seen again.

Unable to Authenticate

Problem:

The chronicle forwarder showed "deadline exceeded" but did not transfer any data. A tcpdump on the system showed a connection being made with google.

Resolution:

Validate the configuration file. In this case, the customer had modified the configuration file to remove "\n" from the "private_key" line in the configuration file. This caused authentication to silently fail.

Additional Notes:

The reason the customer removed the "\n" from the file was because they are using Ansible for file distribution and Ansible was having trouble with the "\n".

Open /opt/chronicle/external/*.conf: no such file or directory

Problem:
When starting the forward in the container, the container restarts and the message:

F0130 17:55:09.205078 204 main_linux.go:118] open /opt/chronicle/external/*.conf: no such file or directory

(or something similar related to *.conf: no such file shows up in the logs.

Resolution:

`open /opt/chronicle/external/*.conf: no such file or directory`

means that the forwarder (inside the container) can't find the *.conf files. This generally means one of the following things:

1) The *.conf files are not in the host's ~/config directory
2) The host's ~/config directory hasn't been mapped into the container
3) SELinux is blocking access

For (1), the customer should check the ~/config directory and put the *.conf files there if they are not (where ~/config is a subdirectory in the home directory for the user which is starting the docker container).

For (2), the mapping of the ~/config directory into the container is done as part of the "docker run" command via the -v option, specifically: "-v ~/config:/opt/chronicle/external". Validate what command and options the customer is using to start up the docker container. The recommended start for containers is the following script:

```
#!/bin/bash
docker stop cfps
docker rm cfps
docker pull gcr.io/chronicle-container/cf_production_stable
docker run \
  --detach \
  --name cfps \
  --restart=always \
  --log-opt max-size=100m \
  --log-opt max-file=10 \
  --net=host \
  -v ~/config:/opt/chronicle/external \
  gcr.io/chronicle-container/cf_production_stable
```

For (3), SELinux tends to be on by default for Redhat and CentOS versions of linux. It can be enabled in other versions as well. To determine if SELinux is in use, one can use the `sestatus` or `getenforce` command to determine the status of SELinux. If these commands are not found, that generally means SELinux is not enabled.

To temporarily disable SELinux, one can put it in "permissive" mode so that there is notification of normally disallowed behavior via `setenforce Permissive`. If this fixes the problem, then one can create an SELinux policy to allow docker to access the config directory.

Alternatively, newer versions of Docker allow for a "z" option to be added to a directory mapping which automatically enables the appropriate SELinux policy. See https://docs.docker.com/storage/bind-mounts/ and the section titled "Configure the selinux label" for more information. An example startup command with the "z" option:

```
#!/bin/bash
docker stop cfps
docker rm cfps
docker pull gcr.io/chronicle-container/cf_production_stable
docker run \
  --detach \
  --name cfps \
  --restart=always \
  --log-opt max-size=100m \
  --log-opt max-file=10 \
  --net=host \
  -v ~/config:/opt/chronicle/external:z \
  gcr.io/chronicle-container/cf_production_stable
```

Starting docker shows an error related to unknown parameter "log-opt"

Problem:
When starting the docker container, docker exits with an error related to "unknown parameter log-opt". This appears to happen with docker version 1.13.

Resolution:
At the moment, we don't know why that error happens.

It's good to validate the version of docker (docker --version) If it is older than 18.XX, see if the customer is willing to upgrade to a more recent version.

Looking at the 1.13 documentation:

https://docs.docker.com/v1.13/engine/admin/logging/overview/#/json-file

it indicates that the argument "should" work (famous last words :) and the documentation even gives an example run command showing its use.

Something to validate is how they are doing logging. If they run the following command, what do they see?

$ docker inspect -f '{{.HostConfig.LogConfig.Type}}' cfps

Also, what is the contents of their /etc/docker directory. Specifically, do they have a file "/etc/docker/daemon.json" and if so, what is the contents of that file?

$ cat /etc/docker/daemon.json

Workaround:
The container (and forwarder) can run without the "--log-opts max-size=100m" and "--log-opts max-file=10" parameters. However, the risk is that the docker logs can grow without bound. Our normal forwarder output isn't overly large, so it would take a while for the logs to get "big" (for some value of big). The --log-opts parameters were intended to limit the size of the log files to 1Gbytes to help make sure we couldn't harm the host machine by filling up the disk accidentally with logging output.

Log line size limits

- syslog has 64KB limit per logline.
- For the rest of the Forwarder collector types, each line can be as large as the batch size allowed (which is 1MB) : the limitation is a maximum 1MB payload size. This means that each log file size should not exceed 1MB. Also if you are sending compressed data, than the uncompressed data size should not exceed the 1MB.

Windows Based Forwarder FAQ

Incorrect Function Error

Problem:

The Chronicle Forwarder Service fails to start and you see the following error message in the Windows Event Viewer:

The Chronicle forwarder service terminated with the following service-specific error: Incorrect function.

Resolution:

This is likely to be caused by a misconfigured input. Verify that inputs in the Forwarder configuration file are properly configured. In particular, if a 'pcap' input is configured, verify the the interface name. See the Chronicle Forwarder Guide for details.

Passing extra arguments to Chronicle Forwarder Service

Problem:

Chronicle forwarder is installed as window services. How to pass extra arguments to Chronicle forwarder service?

Resolution:

Go to command prompt and enter the following command:

$ sc start chronicle_forwader <arguments>
For example:

$ sc start chronicle_forwarder -v 3

Forwarder not starting

Problem:

The forwarder doesn't start when Windows reboots

Resolution:

There are three "stages" that are needed for the forwarder to send us logs (under windows):

1) Windows needs to start the forwarder
2) The forwarder needs to see the network and be able to communicate with chronicle
3) The forwarder sends logs to chronicle.

If (3) isn't happening, that implies we're stuck in either (1) or (2). To sanity check where we're stuck:

a) Check for the temporary logs that the forwarder creates in \windows\temp directory.

The forwarder should create a temporary file each time the forwarder is started. If there is a temporary file, that implies we are stuck in (2). Look at the temporary file to try to determine why the forwarder is unable to communicate with Chronicle.

If there is no temporary file, that implies we are stuck in (1). One would need to look at the windows event logs to understand why the forwarder isn't running (did windows attempt to start it? If windows did, did the forwarder error out, etc).

Other Forwarder Related FAQ

Deadline Exceeded

Problem:

When running the forwarder, the initial stats message is not sent and the logs show "Deadline Exceeded".

Resolution:

Scenario #1: The customer's firewall settings are blocking the Forwarder. Validate any firewall settings both local to the machine as well as in routers. Basically, work with the customer to make sure the path from the machine running the forwarder to the internet is clear for the host (malachiteingestion-pa.googleapis.com) and port (443) that we use.

Full URL: malachiteingestion-pa.googleapis.com:443

Notes:
One way to validate this is (or is not) a firewall issue is to run tcpdump on the machine running the forwarder and validate that a connection is successful with googleapis.com.

There is also a "--test" option which can be given to the forwarder which causes the forwarder to try to contact google.com and indicates success or failure and then exits.

Scenario #2: The host running the Forwarder has a skewed clock (in the future or past). This causes the Oauth portion to fail. Correct the clock and/or use NTP.

No data being received by the forwarder using syslog

Problem:

When running the forwarder, the forwarder can send the initial stats message to us, but it never gets any syslog data and hence, never sends syslog data to us..

Resolution:

Firewall was blocking data *even though tcpdump was showing data getting to the machine.*

Make sure the firewall isn't dropping tcp/udp packets to the syslog port. In the case of CentOS, it uses firewalld and one can use firewall-cmd to inspect the state of the firewall (and turn it off). Use:

- `firewall-cmd --state` to see the state of the firewall
- `firewall-cmd --get-active-zones` to see what is currently active
- `systemctl stop firewalld` to stop the firewall
- `systemctl disable firewalld` to disable the firewall from starting at boot time (note: doesn't stop the currently running firewall).

The firewall-cmd tool can also be used to poke holes in the firewall if desired. For example (the `--permanent` argument makes the change persistent):

- `firewall-cmd --permanent--zone=trusted --change-interface=eth0` to put the eth0 interface into the trusted zone (which should allow all packets).
- `firewall-cmd --permanent --zone=trusted --add-port=10514/tcp` to put the 10514 port into the trusted zone (which should allow all TCP packets to port 10514)

Notes:
Tcpdump gets its data before iptable firewall rules on CentOS (and likely other linux systems as well). This means that if there is a firewall on the machine, one can see data flowing via tcpdump but the forwarder still doesn't see data.

- **Data flowing in tcpdump is NOT an indication of whether there is a firewall blocking data or not.**

One can also use the `iptables` (and `ip6tables`) command to see if there are firewall rules regardless of higher layer tools which are used to enable/disable things. Firewall rules tend to be associated with the default or raw tables, but it doesn't hurt to look at all of them

```
sudo iptables -L
sudo iptables -t raw -L
```

```
sudo iptables -t mangle -L
sudo iptables -t nat -L
```

How do you validate the config file?

Problem:

The config files are in yaml format and it is very picky. A single space character either missing or extra would cause the forwarder not to run.

Resolution:

You can test the config file using --check_config flag. For example:
chronicle_forwarder --check_config -config <configfile>

Tips and Tricks

Debugging Splunk Problems

http://go/malachite-g3doc/ci/forwarder#splunk

Splunk Login Using Curl

Forwarder needs to authenticate with Splunk before it can start querying Splunk. To verify that forwarder will be able to authenticate with Splunk on the box which it runs on, use Curl to login and Splunk should return a session key if login succeeds.
Login succeeds:

```
# Add -k to the curl command.
# I remove -k from the following command to get rid of the following linter
# error:
https://g3doc.corp.google.com/ops/security/skippy/g3doc/rooles/downloaders_nossl.md?cl
=head
# I can't figure out a way to get rid of the linter error.
# If you find a way to do so, please include -k directly in the Curl command.

$ curl https://localhost:8089/services/auth/login?output_mode=json -d username=admin
-d password=changeme
{"sessionKey":"XrFfcozRecXBY_77D5nj_Alwoll7yrFQUWojt8nEpSDPlYjo6AVZaf8zrs1Mkii0p9jR4gr
W3aIlk6wYP2kUPBFGmFWpBobjEJQe2olprtBjPutrsH7"}
```

Login fails:

```
# Add -k to the curl command.
# I remove -k from the following command to get rid of the following linter
# error:
https://g3doc.corp.google.com/ops/security/skippy/g3doc/rooles/downloaders_nossl.md?cl
=head
# I can't figure out a way to get rid of the linter error.
# If you find a way to do so, please include -k directly in the Curl command.

$ curl https://localhost:8089/services/auth/login?output_mode=json -d username=admin
-d password=invalid
{"messages":[{"type":"WARN","text":"Login failed"}]}
```

Search against Splunk using Curl

Please substitute with Splunk endpoint, search index, start time, and end time of your interest accordingly.

Method 1: Using username and password

```
# Add -k to the curl command.
# I remove -k from the following command to get rid of the following linter
# error:
https://g3doc.corp.google.com/ops/security/skippy/g3doc/rooles/downloaders_nossl.md?cl
=head
# I can't figure out a way to get rid of the linter error.
# If you find a way to do so, please include -k directly in the Curl command.


curl -u admin:changeme https://localhost:8089/services/search/jobs?output_mode=json \
  -d search="search index=fake | table _time, _raw" \
  -d index_earliest="1524850000" \
  -d index_latest="1524855000" \
  -d exec_mode="oneshot" \
  -d count="0"
```

Method 2: Using session key

Please refer to Splunk Login Using Curl for instruction on how to retrieve session key.

```
# Assume session key is "replace-this-session-key"

# Add -k to the curl command.
# I remove -k from the following command to get rid of the following linter
# error:
https://g3doc.corp.google.com/ops/security/skippy/g3doc/rooles/downloaders_nossl.md?cl
=head
# I can't figure out a way to get rid of the linter error.
# If you find a way to do so, please include -k directly in the Curl command.
```

```
curl https://localhost:8089/services/search/jobs?output_mode=json \
  -H "Authorization: Splunk replace-this-session-key" \
  -d search="search index=fake | table _time, _raw" \
  -d index_earliest="1524850000" \
  -d index_latest="1524855000" \
  -d exec_mode="oneshot" \
  -d count="0"
```

Using netcat to determine if a firewall is blocking data

To help verify whether or not there is a firewall, netcat + localhost is your friend. Use netcat (`sudo apt-get install netcat` on a debian based system) to send and receive data via localhost to verify things are working. Localhost tends to not be blocked by firewall rules, so if things work with localhost, but don't work with an external system, it's probably a firewall issue.

Example commands (command line netcat is nc):

- Send data to a syslog listener at port 10514 via tcp
  - `nc localhost 10514`
- Send data to a syslog listener at port 10514 via udp
  - `nc -u localhost 10514`
- Receive data (aka be a syslog listener) at port 10514 via tcp
  - `nc -l 10514`
- Receive data (aka be a syslog listener) at port 10514 via tcp
  - `nc -u -l 10514`

Example of using nc to look for a firewall that is listening for syslog messages at port 10514:
Run the forwarder

- Use nc to send data via tcp and see if the data shows up as a packet for sending in the forwarder logs.
- Use nc to send data via udp and see if the data shows up as a packet for sending in the forwarder logs.

Changing the verbosity level of forwarders running in a container

One can optionally set the forwarder logging level to -v 4 in order to see the actual data being received by adding a forwarder_extra_args file into the ~/config directory which would contain a single line "-v 4 -logtostderr" (without the quotes):

```
$ more ~/config/forwarder_extra_args
-v 4 -logtostderr
```

To reset the verbosity back to the default, either remove the file or rename it.

NOTE: The `forwarder_extra_args` file can be used to pass any forwarder arguments to the forwarder inside the container.

Accessing the inside of the container from a host system

The containers are using a standard Debian distribution (specifically, debian-slim) which allows the forwarder to be compiled and run in a known environment. To access that environment, on the HOST machine, run:

```
$ docker exec -it cfps /bin/bash
```

If one wants to start the container without starting the forwarder (for example, to try out the netcat commands mentioned above), on the HOST machine run:

```
$ docker stop cfps
$ docker rm cfps
$ docker run \
  --name cfps \
  --log-opt max-size=100m \
  --log-opt max-file=10 \
  --net=host \
  -v ~/config:/opt/chronicle/external \
  --entrypoint=/bin/bash \
  -it gcr.io/chronicle-container/cf_production_stable
```

In both cases a `/bin/bash` shell will be started and the prompt on the terminal starting the command will change and become a root prompt (#) that is running inside the container. Commands can now be run from that terminal which are executing inside the container. For example:

```
$ docker exec -it cfiu  /bin/bash
root@css2:/opt/chronicle# ls
bin  chronicle_bin  chronicle_startup.sh  container.version  data
dry  external  internal_unstable.tgz  internal_unstable.tgz.sig
root@css2:/opt/chronicle#
```

Adding packages to a container for testing

The container is running a standard Debian distribution which means the usual Debian package manager works for installing/removing packages within the container.

1. Get access to the inside of the container as described above.
2. Use `apt-get` to install the desired packages.

Packages which are commonly useful for debugging network issues:

```
# apt-get install net-tools tcpdump netcat
```

If one is running inside a container and tries to install these packages and the install fails, there is likely an issue with a firewall somewhere blocking access to the debian repositories.

Resetting a container after it's been modified for testing

Container images are persistent between `docker stop` and `docker start` commands. To cause docker to re-pull the container (which will remove any internal modifications made during testing), stop the container and then remove the instance. The next docker run command will pull a fresh container. For example, assume we have a container running...

1. We go into the container and make changes
    a. `docker exec -it cfps /bin/bash`
        i. `apt-get install net-tools tcpdump netcat`
        ii. ... other commands inside the container ...
        iii. `exit`
2. Back on the host we do:
    a. `docker stop cfps`
    b. `docker start cfps`
3. Our changes are still inside the container.
4. Now let's make our changes go away.
    a. `docker stop cfps`
    b. `docker rm cfps`
    c.
```
docker run \
    --detach \
    --name cfps \
    --restart=always \
    --log-opt max-size=100m \
    --log-opt max-file=10 \
    --net=host \
    -v ~/config:/opt/chronicle/external \
    gcr.io/chronicle-container/cf_production_stable
```
5. We are now running a fresh container and our changes from above are gone.

Forwarder Spanner Queries

https://g3doc.corp.google.com/googlex/security/malachite/g3doc/infrastructure/forwarder/troubleshooting.md?cl=head

Useful docker commands

- View logs :
  - docker logs <container-name>
- View logs at runtime :
  - docker logs -f <container-name> (like "tail -f")
- To find out container's config info like networking, volume binds :
  - docker inspect <container-name>
- To run a command inside a container :
  - docker exec -it <container-name> <command-to-run-inside-container>
- To get a shell running inside the container :
  - docker exec -it <container-name> bash
- When Forwarder is failing to come up, use the entrypoint option to launch into a shell after starting up a container. For eg:
  - docker run --name cfps --log-opt max-size=100m --log-opt max-file=10 --net=host -v ~/config:/opt/chronicle/external --entrypoint=/bin/bash -it gcr.io/chronicle-container/cf_production_stable

IP addr ranges to whitelist(firewall)

The IP address range for the API calls used by Chronicle Forwarder and Carbon Black Forwarder and API based ingestion will be the same as that in use by Google and its services. An example of getting the current whole range of addresses - https://support.google.com/a/answer/60764?hl=en

Parameters control the Splunk query frequency

These are the parameters that impact Splunk query frequency:

- **Minimum_window_size**:
    - default value - 10s
    - As the name implies, this governs the minimum time range passed to the Splunk query.
    - In the steady state(i.e we are caught up and not lagging behind) then this will equate to how frequently we hit Splunk server to fetch the data. So this can be used to tune if the requirement is to change how frequently Splunk server is queried when our Forwarder is in steady state(ie live and caught up). What this also means is that when we are lagging behind, then we will hit Splunk server as often as we can(depending on the round trip time of the Splunk api call).
- **Maximum_window_size:**
    - default value is 30s
    - As the name implies, this governs the maximum time range passed to the Splunk query.
    - Change this(equal to or greater than) when changing the min parameter. This can also be used for tuning if we want to fetch more data per query in the cases where we are lagging behind. Lagging case can happen if a Splunk query call is taking longer than our **maximum_window_size**
    - One thing to note is that we never overlap the time ranges when querying Splunk - its just that the time range queried is always between min and max window parameters

- **LiveDataOffset:**
    - Default is 60 sec and can be set to some arbitrary value to go back in time.

- Used to fetch data from the past. Note that this parameter is not intended for this purpose by design but that it can used to fetch past data as a hack.
- Note that the side effect of this is that the data fetched will always be **LiveDataOffset** number of seconds behind the live data. Some workarounds for this is to configure multiple connectors with different values set for this parameter

CB Event Forwarder

- Guide to set up the Forwarder that is developed by Carbon Black to send data from their products to Chronicle: https://docs.google.com/document/d/1qE60DtKYBI0mFOaj0Bvfdd8w-WMHIghmUr4XBTXli1k/edit#heading=h.mrpy38xa0p8s

How to check if Forwarder is running

We could check on our end for incoming Forwarder heartbeats.

Couple ways to check are:

- Customer metrics dashboards have log type called CHRONICLE_HEARTBEAT : for eg: https://partnerdash.google.com/partnerdash/d/chronicle-data#a=1130448308&p=dashboard-bc8ed7a0-2571-45c6-a456-4d5e198f68f6
- Spanner : RawEventBatches table in Event database (this has the stats event that Forwarder sends once every 5mins)

L7 proxy for ingestion/forwarder

https://docs.google.com/document/d/1xnlMHWBdpnQZp6NNqx23kiMGYS93dJ718uLdv61YsWs/edit?hl=en&forcehl=1&resourcekey=0-SGwCIjI7_EDYvJJZWSOdKQ#heading=h.5tgw0mseeqa3

Playbooks based on past Customer cases

go/chronicle-forwarder-playbooks