Policy Tracing and Execution Analyzer

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Products or Functional Area	PCF
Applicable Platform(s)	SMI
Feature Default Setting	Enabled – Configuration required to disable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
First introduced.	2020.01.0

Feature Description

PCF comes with a set of utilities to actively monitor and trace policy execution. These utilities interact with the core Policy Server and the Mongo database to trigger and store traces for specific conditions.

Architecture

Cisco PCF comes with a trace pod to actively monitor and trace the policy execution. The utilities in this pod interact with the Policy Engine pods and the Mongo database pods to trigger and store traces for specific conditions.

The policy tracing and execution analyzer is a three-tier architecture:

- Tier 1—Command-line utilities to manage the policy trace generation and extract policy traces.
- Tier 2—Policy server creation of policy traces using triggers that are defined in Tier 1.
- Tier 3—Storage of the policy traces in a MongoDB.

How it Works

This section describes how this feature works.

Configuration Support for the Policy Traces

This section describes how you configure the policy traces.

Configuration support of the policy traces involves the following steps:

1. Setting Up the Trace Database
2. Configuring the Trace Microservice Pod
3. Executing the Tracing Scripts

Setting Up the Trace Database

This section describes how to configure the database and port where you want to store the traces.

1. Log in to Policy Builder.
2. From left pane, select your system and click the appropriate cluster.
3. From right pane, select the check box for **Trace Database**.

   The following table provides the parameter descriptions under **Trace Database** check box.

Table 3: Trace Database Parameters

Parameter	Description
Primary Database IP Address	The name of the Mongo database cluster that holds the trace information which allows debugging of specific sessions and subscribers based on the unique primary keys.

Parameter	Description
Secondary Database IP Address	The IP address of the database that provides fail over support for the primary database. This is the mirror of the database that is specified in the Primary IP Address field. Use this only for replication or replica pairs architecture. This field is present but deprecated to maintain the downward compatibility.
Database Port	Port number of the database that stores the trace data. Default value is 27017.

Configuring the Trace Microservice Pod

PCF hosts the tracing-specific commands on the trace microservice pod that is available under the /usr/local/bin directory.

To determine the trace pod, use the following configuration:

```
config
  kubectl -n pcf namespace [ get pods | grep trace ]
  end
```

Sample output of the command:

```
luser@for-cn-dev-10c-masterb92844ec32:~$ kubectl -n pcf get pods | grep trace
traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc      1/1     Running   0         40m
luser@for-cn-dev-10c-masterb92844ec32:~$
```

Executing the Tracing Scripts

Tracing logs assist you in backtracking the steps that you or the system has performed to accomplish a task. This information is useful when you want to conduct forensics of the unexpected outcomes.

PCF provides two scripts that let you obtain the tracing information:

- trace_ids.sh: Manages the rules for activating and deactivating traces within the system.
- trace.sh: Allows retrieval of the real-time and historical traces.

The execution of the tracing scripts involves the following steps:

1. Managing the Trace Rules
2. Managing the Trace Results

Managing the Trace Rules

The **trace_ids.sh** script fetches the real-time and historical traces. This script resides in `/usr/local/bin/` of the Tracing Pod that you have configured.

See Configuring the Trace Microservice Pod, on page 3 for procedure to set up a Pod.

The Execute the **trace_ids.sh** script with *-h* arguments produces a help text describing the capabilities of the script.

The **trace_ids.sh** script starts a selective trace and outputs it to a standard out.

1. To specify the audit ID tracing, use the following configuration:

   ```
   kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc

   -- trace_ids.sh -i specific id
   ```

2. To remove trace for specific audit ID, use the following configuration:

   ```
   kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc


   -- trace_ids.sh -r specific id
   ```

3. To remove trace for all IDs, use the following configuration:

   ```
   kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc
    -- trace_ids.sh -x
   ```

4. To list all the IDs under trace, use the following configuration:

   ```
   kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc
    -- trace_ids.sh -l
   ```

   Adding a specific audit ID for tracing requires running the command with the -i argument and passing in a specific ID. The Policy Server matches the incoming session with the ID provided and compares this against the following network session attributes:

   - Credential ID
   - Framed IPv6 Prefix
   - IMSI
   - MAC Address
   - MSISDN
   - User ID

   If an exact match is found, then the transactions are traced.

   Note Spaces and special characters are not supported in the audit IDs.

   - Removing a specific audit ID from active tracing requires specifying the -r argument with ID to remove.
   - Removing all IDs requires sending in the -x argument. This step purges all the IDs from the database.
   - Listing all IDs requires sending in the -l argument.

   Example output:

   ```
   kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc
    -- trace_ids.sh
   ```

```
-s mongo-admin-0 -p 27017 -t admin -d policy_trace -i 2001
```

Run the **trace_ids.sh** with *-h* arguments produces a help text describing the capabilities of the script as follows:

```
kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc -- trace_ids.sh
-h
/usr/local/bin/trace_ids.sh: option requires an argument -- h
usage:
/usr/local/bin/trace_ids.sh -i specific id
    /usr/local/bin/trace_ids.sh -r specific id
    /usr/local/bin/trace_ids.sh -x
    /usr/local/bin/trace_ids.sh -l
    /usr/local/bin/trace_ids.sh -s mongo service name
    /usr/local/bin/trace_ids.sh -p mongo service port
    /usr/local/bin/trace_ids.sh -t mongo replica set
    /usr/local/bin/trace_ids.sh -d mongo database name

This script starts a selectve trace and outputs it to standard out.
1. Add Specific Audit Id Tracing  /usr/local/bin/trace_ids.sh -i specific id
2. Remove Trace for Specific Audit Id  /usr/local/bin/trace_ids.sh -r specific id
3. Remove Trace for All Ids /usr/local/bin/trace_ids.sh -x
4. List All Ids under Trace /usr/local/bin/trace_ids.sh -l
5. K8 mongo service name -s (default: mongo-admin-0)
6. Mongo port -p (default: 27017)
7. Replica set name -t (default: admin)
8. Trace database name -d (default: policy_trace)
9. /usr/local/bin/trace_ids.sh -h displays this help
```

Managing the Trace Results

The **trace.sh** script that initiates selective trace process resides in `/usr/local/bin/` of the Tracing Pod that you have configured.

See Configuring the Trace Microservice Pod, on page 3 for procedure to set up a pod.

1. To specify the audit ID tracing, use the following configuration:

   ```
   kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc
   -- trace.sh -i  specific_id
   ```

   Specifying the *-i* argument for a specific ID causes a real-time policy trace to be generated while the script is running. You can redirect this to a specific output file using standard Linux commands.

2. To dump all traces for the specific audit ID, use the following configuration:

   ```
   kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc
   -- trace.sh -x specific_id
   ```

   Specifying the *-x* argument with a specific ID, dumps all historical traces for a given ID. You can redirect this to a specific output file using standard Linux commands.

3. To trace all, use the following configuration:

   ```
   kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc
    – trace.sh -a
   ```

   Specifying the *-a* argument causes all traces to output in the real-time policy trace while the script is running. You can redirect this to a specific output file using standard Linux commands.

4. To trace all the errors, use the following configuration:

```
kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc
 -- trace.sh -e
```

Specifying the *-e* argument causes all traces that are triggered by an error to output in real-time policy trace while the script is running. You can redirect this to a specific output file using standard Linux commands.

Example output:

```
kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc
 -- trace.sh -s mongo-admin-0
-p 27017 -t admin -d policy_trace -x 1234567890
```

5. Execute the **trace.sh** script with *-h* arguments to produce a help text describing the capabilities of the script as follows:

```
kubectl -n pcf exec -it traceid-pcf-pcf-engine-app-pcf-75b6dc6c4-hc7qc -- trace.sh -h
/usr/local/bin/trace.sh: option requires an argument -- h usage:
    /usr/local/bin/trace.sh -i specific_id
    /usr/local/bin/trace.sh -x specific_id
    /usr/local/bin/trace.sh -a
    /usr/local/bin/trace.sh -e
    /usr/local/bin/trace.sh -s mongo_service_name
    /usr/local/bin/trace.sh -p mongo_service_port
    /usr/local/bin/trace.sh -t mongo_replica_set
    /usr/local/bin/trace.sh -d mongo_database_name
    /usr/local/bin/trace.sh -h
This script starts a selectve trace and outputs it to standard out.
1. Specific Audit Id Tracing  /usr/local/bin/trace.sh -i specific_id
2. Dump All Traces for Specific Audit Id  /usr/local/bin/trace.sh -x specific_id
3. Trace All /usr/local/bin/trace.sh -a
4. Trace All Errors /usr/local/bin/trace.sh -e
5. K8 mongo service name -s (default: mongo-admin-0)
6. Mongo port -p (default: 27017)
7. Replica set name -t (default: admin)
8. Trace database name -d (default: policy_trace)
9. /usr/local/bin/trace.sh -h displays this help
```