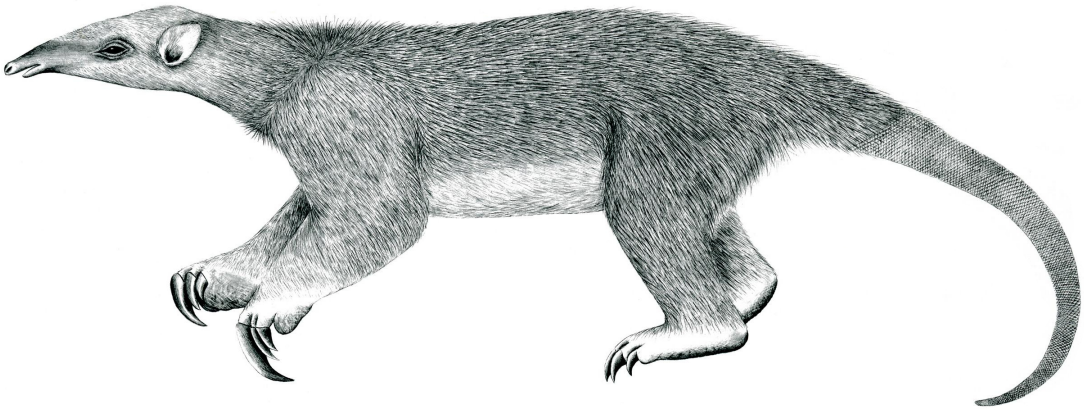
Tranalyzer2

psqlSink

PostgreSQL



Tranalyzer Development Team

Contents

1 psqlSink

1.1 Description

The psqlSink plugin outputs flows to a PostgreSQL database.

1.2 Dependencies

1.2.1 External Libraries

This plugin depends on the **libpq** library.

Ubuntu:	`sudo apt-get install`	`libpq-dev`
Arch:	`sudo pacman -S`	`postgresql-libs`
Gentoo:	`sudo emerge`	`postgresql`
openSUSE:	`sudo zypper install`	`postgresql-devel`
Red Hat/Fedora[1]**:**	`sudo dnf install`	`libpq-devel`
macOS[2]**:**	`brew install`	`postgresql`

1.2.2 Core Configuration

This plugin requires the following core configuration:

- *$T2HOME/tranalyzer2/src/tranalyzer.h*:
  - `BLOCK_BUF=0`

1.3 Database Initialization

The psqlSink plugin requires a PostgreSQL server running on `PSQL_HOST`, e.g., `127.0.0.1` on port `PSQL_PORT`, e.g., 5432. In addition, a user with name `PSQL_USER`, e.g., `postgres`, and password `PSQL_PASS`, e.g., `postgres` **MUST** exist and be allowed to create databases. This can be achieved with the following commands:

Linux: `$ sudo -u postgres psql` **macOS:** `$ psql postgres`

```
postgres=# CREATE ROLE postgres WITH LOGIN PASSWORD 'postgres';
CREATE ROLE
postgres=# ALTER ROLE postgres CREATEDB;
ALTER ROLE
```

1.4 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
`PSQL_OVERWRITE_DB`	2	0: abort if DB already exists

[1] If the `dnf` command could not be found, try with `yum` instead
[2] Brew is a packet manager for macOS that can be found here: `https://brew.sh`

Name	Default	Description
PSQL_OVERWRITE_TABLE	2	1: overwrite DB if it already exists 2: reuse DB if it already exists 0: abort if table already exists 1: overwrite table if it already exists 2: append to table if it already exists
PSQL_TRANSACTION_NFLOWS	40000	0: one transaction > 0: one transaction every *n* flows
PSQL_QRY_LEN	32768	Max length for query
PSQL_HOST	"127.0.0.1"	Address of the database
PSQL_PORT	5432	Port of the database
PSQL_USER	"postgres"	Username to connect to DB
PSQL_PASS	"postgres"	Password to connect to DB
PSQL_DBNAME	"tranalyzer"	Name of the database
PSQL_TABLE_NAME	"flow"	Name of the table
PSQL_SELECT	0	Only insert specific fields into the DB
PSQL_SELECT_FILE	"psql-columns.txt"	Filename of the field selector (one column name per line)

1.4.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (ENVCNTRL>0):

- PSQL_HOST
- PSQL_PORT
- PSQL_USER
- PSQL_PASS
- PSQL_DBNAME
- PSQL_TABLE_NAME
- PSQL_SELECT_FILE (require PSQL_SELECT=1)

1.5 Insertion of Selected Fields Only

When PSQL_SELECT=1, the columns to insert into the DB can be customized with the help of PSQL_SELECT_FILE. The filename defaults to psql-columns.txt in the user plugin folder, e.g., *~/.tranalyzer/plugins*. The format of the file is simply one field name per line with lines starting with a '#' being ignored. For example, to only insert source and destination addresses and ports, create the following file:

```
# Lines starting with a '#' are ignored and can be used to add comments
srcIP
srcPort
dstIP
dstPort
```

1.6 Post-Processing

The following queries can be used to analyze bitfields in PostgreSQL:

- Select all A flows:
```
SELECT to_hex("flowStat"::bigint), *
FROM flow
WHERE ("flowStat"::bigint & 1) = 0::bigint
```
- Select all IPv4 flows:
```
SELECT *
FROM flow
WHERE ("flowStat"::bigint & x'4000'::bigint) != 0::bigint
```
- Select all IPv6 flows:
```
SELECT to_hex("flowStat"::bigint), *
FROM flow
WHERE ("flowStat"::bigint & x'8000'::bigint) != 0::bigint
```

1.7 Example

```
# Run Tranalyzer
$ t2 -r file.pcap

# Connect to the PostgreSQL database
$ psql -U postgres -d tranalyzer

# Number of flows
tranalyzer=# SELECT COUNT(*) FROM flow;

# 10 first srcIP/dstIP pairs
tranalyzer=# SELECT "srcIP", "dstIP" FROM flow LIMIT 10;

# All flows from 1.2.3.4 to 1.2.3.5
tranalyzer=# SELECT * FROM flow WHERE "srcIP" = '1.2.3.4' AND "dstIP" = '1.2.3.5';
```

For examples of more complex queries, have a look in `$T2HOME/scripts/t2fm/psql/`.

1.7.1 Clean up an existing database

```
# Connect to the PostgreSQL database
$ psql -U postgres

# Drop the database
tranalyzer=# DROP DATABASE tranalyzer;
```