Events Schema v4

April 9, 2009

Contents

Introduction

This document describes the format of events exported by Riverbed Cascade Profiler. This version of the schema applies to Profiler versions 7.5.1 and later.

When an event is first generated by the Profiler, a new entry called the “start” row is inserted into the `events.internal_export_table` table. This has a `start_time` field with a UNIX epoch timestamp corresponding to when the event activity began on the network. The event is written to the table shortly after. So the start_time does not correspond exactly to the time at which the first entry is written to the table. The `end_time` is set to null in this entry.

When the event expires, another entry called the “end” row with the same event identifier (eid) is entered into the export table. This row has the `end_time` set. The `end_time` represents the end of the event activity in the network and does not correspond exactly to when the event is written to the export table. This is because event expiry is determined based on the lack of event activity for a timeout period.

Each entry has a unique `entry_id` field. Note that `entry_id` is unavailable in prior versions (v3, Profiler versions 7.4, 7.5). The `entry_id` provides a way to query for expired and ongoing events periodically, without skipping any events. This method is described in the section titled “Using entry_id to poll for events”.

The end row’s contents may differ from the start row since it records the state of the event on expiration. Each field records the most recent value or an updated list.

There are three views into the internal export table.

1. `events.export_csv_view` provides data in CSV format
2. `events.export_xml_view` provides data in XML format
3. `events.export` is backward (Profiler version 7.4, 7.5) compatible schema

The schema version is stored in `events.export_version`.

```
mazu=# select * from events.export_version;

 major | minor
-------+-------
     4 |     0
(1 row)
```

The `events.export_types` table lists the integer type and name of each exported event. Profiler versions prior to 8.2 export the following event types.

```
mazu=# select * from events.export_types;

type |         name
------+-----------------------
    0 | DOS/Bandwidth Surge
    1 | Worm
    2 | Host Scan
    3 | Port Scan
    4 | Suspicious Connection
    5 | New Host
    9 | New Server Port
   11 | Rule Based Event
(8 rows)
```

Profiler versions 8.2 and above export the following event types.

```
mazu=# select * from events.export_types;
 type |           name
------+-------------------------
    0 | DOS/Bandwidth Surge
    1 | Worm
    2 | Host Scan
    3 | Port Scan
    4 | Suspicious Connection
    5 | New Host
    9 | New Server Port
   11 | Rule Based Event
   17 | Application Availability
   18 | Link Congestion
   19 | Link Outage
   20 | Application Performance
(12 rows)
```

Events are periodically deleted from core database tables when the total number of events in the system reaches a limit (currently 10000). Whenever an event is removed from the core tables, it is also removed from the export tables.

Schema

events.internal_export_table

Use this table if you need a combination of CSV and XML fields

For events that are specified by rules, the `event_description` provides the name of the rule along with the event type name as `TYPE_NAME, "RULE_NAME"` in CSV; for others it is simply `TYPE_NAME`.

The `recorded_count` fields give the size of the corresponding CSV list.

```
CREATE TABLE events.internal_export_table (
       entry_id                 INT PRIMARY KEY
                                DEFAULT nextval('events.export_entry_seq'),
       eid                      INT,
       event_description        TEXT,
       type                     INT CHECK (type >= 0),
       severity                 INT CHECK (severity >= 0 AND severity < 101),
       alert_level      INT CHECK (alert_level >= 0 AND alert_level < 4),
       src_actual_count         INT,
       src_recorded_count       INT,
       src_ip_csv               TEXT,
       src_mac_csv              TEXT,   -- count same as ip
       dst_actual_count         INT,
       dst_recorded_count       INT,
       dst_ip_csv               TEXT,
       dst_mac_csv              TEXT,   -- count same as ip
       srcs_xml                 TEXT,
       dsts_xml                 TEXT,
       hosts_xml                TEXT,
       src_port_actual_count    INT,
       src_port_recorded_count  INT,
       src_port_csv             TEXT,
       dst_port_actual_count    INT,
       dst_port_recorded_count  INT,
       dst_port_csv             TEXT,
       srcports_xml             TEXT,
       dstports_xml             TEXT,
       ports_xml                TEXT,
       attributes_xml           TEXT,
       start_time               INT NOT NULL,
       end_time                 INT,
       email_sent               BOOLEAN,
       trap_sent                BOOLEAN,
       notifications_xml        TEXT
);
```

events.export_csv_view

```
mazu=# \d events.export_csv_view
         View "events.export_csv_view"
         Column          |  Type   | Modifiers
-------------------------+---------+-----------
 entry_id                | integer |
 eid                     | integer |
 event_description       | text    |
 type                    | integer |
 severity                | integer |
 alert_level             | integer |
 src_actual_count        | integer |
 src_recorded_count      | integer |
 src_ip_csv              | text    |
 dst_actual_count        | integer |
 dst_recorded_count      | integer |
 dst_ip_csv              | text    |
 src_mac_csv             | text    |
 dst_mac_csv             | text    |
 src_port_actual_count   | integer |
 src_port_recorded_count | integer |
 src_port_csv            | text    |
 dst_port_actual_count   | integer |
 dst_port_recorded_count | integer |
 dst_port_csv            | text    |
 start_time              | integer |
 end_time                | integer |
 email_sent              | boolean |
 trap_sent               | boolean |
```

events.export_xml_view

```
mazu=# \d events.export_xml_view
      View "events.export_xml_view"
      Column       |  Type   | Modifiers
-------------------+---------+-----------
 entry_id          | integer |
 eid               | integer |
 event_description | text    |
 type              | integer |
 severity          | integer |
 alert_level       | integer |
 srcs_xml          | text    |
 dsts_xml          | text    |
 hosts_xml         | text    |
 srcports_xml      | text    |
 dstports_xml      | text    |
 ports_xml         | text    |
 attributes_xml    | text    |
 start_time        | integer |
 end_time          | integer |
 notifications_xml | text    |
```

events.export (for backward compatibility only)

```
mazu=# \d events.export
        View "events.export"
    Column     |  Type   | Modifiers
---------------+---------+-----------
 eid           | integer |
 type          | integer |
 hosts         | text    |
 ports         | text    |
 attributes    | text    |
 start_time    | integer |
 end_time      | integer |
 severity      | integer |
 notifications | text    |
 alert_level   | integer |
```

CSV Format

The format is consistent with RFC 4180 (http://www.rfc-editor.org/rfc/rfc4180.txt).

Examples:

```
event_description: Rule Based Event,"any traffic"
IP addresses: 1.6.0.5,1.6.0.4,1.1.0.1
Ports: tcp/25(smtp),tcp/444(snpp),tcp/443(https),tcp/1290
Mac: 00:00:01:06:00:05,00:00:01:06:00:04,00:00:01:01:00:01
```

Comments:

1. Ports may or may not include a name within parenthesis.
2. A comma may be immediately followed by another when an entry is null
3. Use the `recorded_count` fields to determine the number of entries in a CSV list
4. The default maximum in the CSV list is 32.

XML Format for Hosts

Example:

```
<hosts recorded_count='1' actual_count='1'>
  <host ip='173.16.254.1' mac='00:00:00:00:00:00'/>
</hosts>
```

The `recorded count` is the number of hosts recorded in the export table, while `actual_count` is the total number of hosts that are involved in an event. The default maximum for `recorded_count` is 32.

The first host listed is either the ATTACKER or VICTIM depending on the type of the event. In a Worm event, all the listed hosts are infected. The first host is the first infected among the listed hosts.

DOS/Bandwidth Surge	VICTIM
Worm	ATTACKER
Host Scan	ATTACKER
Port Scan	ATTACKER
Suspicious Connection	ATTACKER
New Host	-
New Server Port	-
Rule Based Event	-

XML Format for Ports

Example:

```
<protoports recorded_count='1' actual_count='1'>
 <protoport protocol='6' port='888'/>
</protoports>
```

XML Format for Notifications

Example:

```
<notifications>
 <notification type='email_sent' value='t'/>
 <notification type='trap_sent' value='t'/>
</notifications>
```

Currently, there are only two notification types that are recorded in the database: Email and SNMP traps.

XML Format for Attributes

Attributes are used to show event-specific details. Currently, only Rule Based Event and New Server Port event types provide the corresponding Rule identifier associated with them, as their attribute.

```
<attributes><attr key="rule_id" value="1"/></attributes>
```

Additional Notes on the XML Format

1. You may want to prefix `<?xml version="1.0"?>` to every XML field in your SELECT query to provide valid input to an XML parser.

2. XML stored in hosts, ports, notifications or attributes does not have spaces or tabs between each element.

For example:

```
mazu=# select eid, hosts from events.export where eid = 1 limit 1;
 eid |                                             hosts
-----+------------------------------------------------------------------------
---------------------------------------
   1  |  <hosts  recorded_count='2'  actual_count='2'><host  ip='22.1.31.212'
mac=''/><host ip='11.0.0.1' mac=''/></hosts>
```

Using entry_id to poll for events

1. The entry_id is incremented for each row in the export table and differs between the 'start' and 'end' row of the same event.

2. The connector records the max. entry_id `E` in the previous run, and periodically issues a query to retrieve all expired and ongoing events as follows:

```
select eid, start_time, end_time from events.export_csv_view where end_time is not
null and entry_id > E

   UNION

select eid, start_time, end_time from events.export_csv_view where entry_id > E
and eid NOT IN (select eid from events.export_csv_view where end_time is not null
and entry_id  > E);
```

3. The query lists event ids for events that have either (a) expired since the last entry_id E or, (b) begun since the last entry_id E and not yet expired. The two parts of the UNION in the query are (a) and (b) respectively. The results include all events that have newly begun since `E` and are either ongoing or have expired. It also includes events that were known to have begun in a prior poll, and have now expired.

4. This means, the connector may receive 2 records for a single event if the event started in one poll period and expired in a subsequent period. If an event starts and expires within the same poll period, it will receive only the expiry record.

5. entry_id is available only in schema version v4. A connector using entry_id cannot be used with export schema version v3 (Profiler versions 7.4, 7.5).