Bro-Osquery

Bro4Pro 2017
Today: Bro as **Network** Intrusion Detection / Monitoring System

- Information as seen on the wire

**Monitoring Problems:**

- Some information are available on the hosts only
  - E.g. Logged in user, network application name
- Encryption of network traffic
  - Limited to meta-data analysis

**Result:**

- Losing visibility on the network infrastructure
  - Dark spots in the network
Goal: Integrate Host Events into Bro
- Transparent handling of both event types, e.g. "new_tcp" and "new_process"

Extension: Host information
- Make host events available Bro
  - Additional host information complement network visibility
- Bro can control *which* events *when* to be emitted by hosts
  - Subscribe to changes (diff events) vs. Retrieve current status (snapshot events)
  - Group hosts and address them individually or collectively
What if I tell you that you can already have this visibility in your Bro deployment?

Wait, how does this work?
- **Host Monitor:** *Osquery*
  - Open source project by Facebook

- **Communication Library:** *Broker*

- **Bro:** Script *framework*
Writing log files about events received from hosts

Network Monitor

Host events

Host Monitor
Host Sensor: Osquery

- Operating system as a high-performance relational database
  - SQL tables represent abstract concepts

- Power of a complete SQL language and dozens of useful tables

```
osquery> SELECT uid, name FROM listening_ports l, processes p WHERE l.pid=p.pid;
```

- running processes
- listening ports
- logged in users
- password changes
- USB devices
- firewall exceptions
- ....

https://osquery.io/
### Osquery Tables

**processes**

All running processes on the host system.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>BIGINT_TYPE</td>
<td>Process (or thread) ID</td>
</tr>
<tr>
<td>name</td>
<td>TEXT_TYPE</td>
<td>The process path or shorthand argv[0]</td>
</tr>
<tr>
<td>path</td>
<td>TEXT_TYPE</td>
<td>Path to executed binary</td>
</tr>
<tr>
<td>cmdline</td>
<td>TEXT_TYPE</td>
<td>Complete argv</td>
</tr>
<tr>
<td>state</td>
<td>TEXT_TYPE</td>
<td>Process state</td>
</tr>
<tr>
<td>cwd</td>
<td>TEXT_TYPE</td>
<td>Process current working directory</td>
</tr>
<tr>
<td>root</td>
<td>TEXT_TYPE</td>
<td>Process virtual root directory</td>
</tr>
<tr>
<td>uid</td>
<td>BIGINT_TYPE</td>
<td>Unsigned user ID</td>
</tr>
</tbody>
</table>

**usb_devices**

USB devices that are actively plugged into the host system.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usb_address</td>
<td>INTEGER_TYPE</td>
<td>USB Device used address</td>
</tr>
<tr>
<td>usb_port</td>
<td>INTEGER_TYPE</td>
<td>USB Device used port</td>
</tr>
<tr>
<td>vendor</td>
<td>TEXT_TYPE</td>
<td>USB Device vendor string</td>
</tr>
<tr>
<td>vendor_id</td>
<td>TEXT_TYPE</td>
<td>Hex encoded USB Device vendor identifier</td>
</tr>
<tr>
<td>model</td>
<td>TEXT_TYPE</td>
<td>USB Device model string</td>
</tr>
<tr>
<td>model_id</td>
<td>TEXT_TYPE</td>
<td>Hex encoded USB Device model identifier</td>
</tr>
<tr>
<td>serial</td>
<td>TEXT_TYPE</td>
<td>USB Device serial connection</td>
</tr>
<tr>
<td>removable</td>
<td>INTEGER_TYPE</td>
<td>1 If USB device is removable else 0</td>
</tr>
<tr>
<td>start_time</td>
<td>BIGINT_TYPE</td>
<td>Process start in seconds since boot (non-sleeping)</td>
</tr>
<tr>
<td>parent</td>
<td>BIGINT_TYPE</td>
<td>Process parent's PID</td>
</tr>
<tr>
<td>pgroup</td>
<td>BIGINT_TYPE</td>
<td>Process group</td>
</tr>
<tr>
<td>threads</td>
<td>INTEGER_TYPE</td>
<td>Number of threads used by process</td>
</tr>
<tr>
<td>nice</td>
<td>INTEGER_TYPE</td>
<td>Process nice level (-20 to 20, default 0)</td>
</tr>
</tbody>
</table>

```
select * from processes where pid = 1
```
Osquery

- High-performance and low-footprint (distributed) host monitoring
  - To query the system in an abstract way
  - Independent of OS, software or hardware configuration

- Host monitoring **daemon**
  - Allows to schedule queries
  - Aggregates query results over time
  - Generates logs which indicate state changes in infrastructure

- Instrumentation framework for
  - Intrusion detection
  - Infrastructure reliability
  - Compliance monitoring

https://osquery.io/
Detecting processes and USB devices...
Is that all you can do?!?
Ask host about current status on demand

Network Monitor

Host Monitor
Idea: Extending conn.log by user and application identification

- Bro captures new TCP/UDP connections
  - You know host address and port (source)

- Query host for application name and user
  - For the process having opened the socket with respective port

```
SELECT p.name, u.username
FROM process_open_sockets s, processes p, users u
WHERE s.protocol = 'UDP' AND s.local_port = 68
      AND s.pid = p.pid AND p.uid = u.uid;
```
Osquery Extension: Bro-Osquery

**Schedule**

**Query 1:**
- query: Select * from users;
- added: true;
- interval: 60;

**Query 2:**
- query: Select * from crontab;
- added: true;
- interval: 10;

**Query**

**Result**

**Tables**

**Daemon**

**Thrift**

**Extension**

**Broker**

**Write**

**Config**

**Update**

**Ad-hoc**

**Execute**

**Logger**

**Report**

**Send**

**Thrift**

**Execute**

**Send**
Connecting Osquery and Bro

- Bro-osquery project consists of
  - Osquery extension (c++)
  - Osquery framework (bro script)
Osquery Framework
Deployment
Installation

- For description of installation steps see: 
  - [https://github.com/bro/bro-osquery/install](https://github.com/bro/bro-osquery/install)

- Osquery is a dependency to this project
  - Osquery is build with a custom tool chain
  - Bro-Osquery has to follow same tool chain
    - And also all other dependencies (broker, caf)

- Tool chain includes
  - clang, c++11, libstdc++ and several system libraries in `/usr/local/osquery`

- Easy method (will most probably not work on your system)
  - `./install_ubuntu_16_04.sh`
  - `./run.sh`
Configuration

Osquery Hosts

- Same configuration file for osquery and extension
  - /etc/osquery/osquery.conf

Bro Monitor

- Load the osquery framework
  - site/osquery/__load__.bro

- Write framework based scripts with
  - osquery::subscribe()
  - osquery::execute()
Status & Future Work

- **Problems**
  - Bro-Osquery: Common installation script for all platforms
  - Osquery: Event-based tables are not available

- **Design discussion**
  - Discard extension design and integrate into Osquery code base?

- **Next Steps:**
  - Bro-Osquery: Extend Bro osquery framework
    - E.g. easy collectively addressing of host groups (host management)
  - Incorporating your feedback
Bro-Osquery Summary

- Extends your visibility on the network by integrating host events
  - Run osquery daemon and bro-osquery extension on hosts
  - Load osquery framework in Bro

- Application scenarios
  - Data collection: writing host events to Bro log
  - Host misbehaving: alarm about non-compliant hosts
  - Correlate network and host events
    - Schedule: host events to detect system changes
    - Ad-hoc: retrieve host information about a specific network incident

https://github.com/bro/bro-osquery

or mail me: haas@informatik.uni-hamburg.de