Zeek 3.0.0 — and beyond

Robin Sommer
robin@corelight.com
Just released: Zeek 3.0.0

```
s/Bro/Zeek/g
```

- `bro` -> `zeek`
- `broctl` -> `zeekctl`
- `bro-cut` -> `zeek-cut`
- `bro-pkg` -> `zkg`

- `/usr/local/bro` -> `/usr/local/zeek`
- `*.bro` -> `*.zeek`
- `bro_{init,done}` -> `zeek_{init_done}`
We got some new functionality, too

New analyzers for MQTT and NTP

Extended analyzers for DNS, RDP, SMB, and TLS

Support for decapsulating VXLAN tunnels

Support for logging in UTF-8

Language extensions:

- Iteration over tables through `for(key, value in t)...
- Vector slicing through `v[2:4]`
- Case-insensitive regular expressions: `/foo/i`
- Anonymous functions now capture their closures
- Efficient matching of a string against a large list of globs (paraglob)
New Release Schedule: Stability vs Features

3.0.0 is our first long-term stable release
   Support with critical fixes for one year (3.0.x)

Feature releases will be 3.x.0
   About every 4 months, plus bugfixes (3.x.y)

Next stable long-term stable release will be 4.0.0
   About one year after 3.0.0

We aim to provide backwards compatibility between subsequent stable release
   Typically, we will deprecate old functionality for one stable cycle
   Will discuss on mailing list in cases that’s not possible
Alright, what’s on the radar for 3.1.0?
Process Supervision

Persist Zeek Supervisor Process

ZeekControl

Manager
Logger
Worker 1
Worker 2
Worker 3
Cluster State Sharing

We used to have `&synchronize` to shares tables across cluster nodes:

```
    global my_state[addr] of string &synchronized;
```

We now have Broker data stores, but their API remains cumbersome.

Goal: Get the best of both worlds (+ persistence) by mapping tables to a data store:

```
    global my_state[addr] of string
                         &backend=Broker::SQLITE;
```
I/O Loop Modernization

# zeek -i lo
listening on lo

^CReceived termination signal
0 packets received on interface lo, 0 dropped

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>23163</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>904m</td>
<td>336m</td>
<td>262m</td>
<td>R</td>
<td>17.0</td>
<td>1.1</td>
<td>0:12.80</td>
<td>zeek</td>
</tr>
<tr>
<td>3557</td>
<td>robin</td>
<td>20</td>
<td>0</td>
<td>50952</td>
<td>30m</td>
<td>2624</td>
<td>S</td>
<td>0.7</td>
<td>0.1</td>
<td>5:29.82</td>
<td>tmux</td>
</tr>
</tbody>
</table>
Performance Baselining

- Corelight-hosted testbed with traffic generator
- Cluster communication benchmark
Code Modernization

Move to standard containers

Switch to C++17

Apply clang-tidy (and perhaps clang-format)

Introduce automatic reference counting, maybe?
event bro_init() {
    local query = [
        $ev=host_process_events,
    $query="SELECT pid, path, cmdline, cwd, uid, gid, time, parent
            FROM process_events"
    ];

    osquery::subscribe(query);
}

event host_process_events(resultInfo: osquery::ResultInfo, pid: int, path: string, cmdline: string, cwd: string, uid: int, gid: int, start_time: int, parent: int) {

    print fmt("UID %d executed %s", uid, path);
}

https://github.com/zeek/osquery-{-extension, framework}
How to become involved

GitHub
Follow activity in https://github.com/zeek/zeek
File issues & PRs

Look for starter tickets
good first issue
Type: Project

Propose ideas, and ask questions, on the development mailing list [1]

Watch out for emerging developer’s manual
First piece: Style guide on coding conventions [2]

[1] https://mailman.icsi.berkeley.edu/mailman/listinfo/zeek-dev
Thanks!

Robin Sommer
robin@corelight.com