Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Steffen Haas
Department of Computer Science
IT Security and Security Management (ISS)

Bro-Osquery

Bro Network Monitor
https://www.bro.org

Osquery
https://osquery.io/

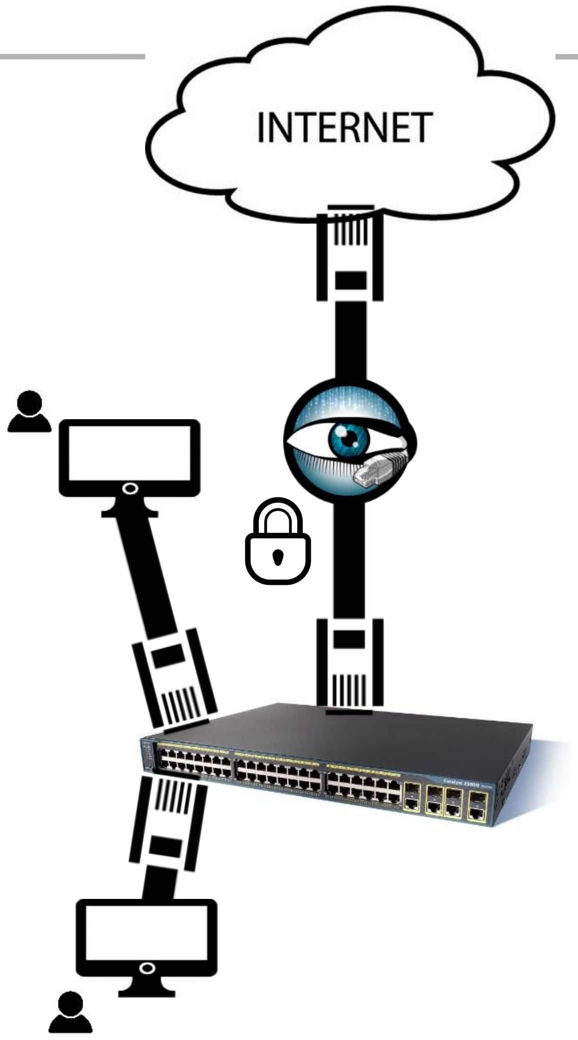
Bro4Pro 2017

Motivation

- Today: Bro as ***Network*** Intrusion Detection / Monitoring System
  - Information as seen on the wire

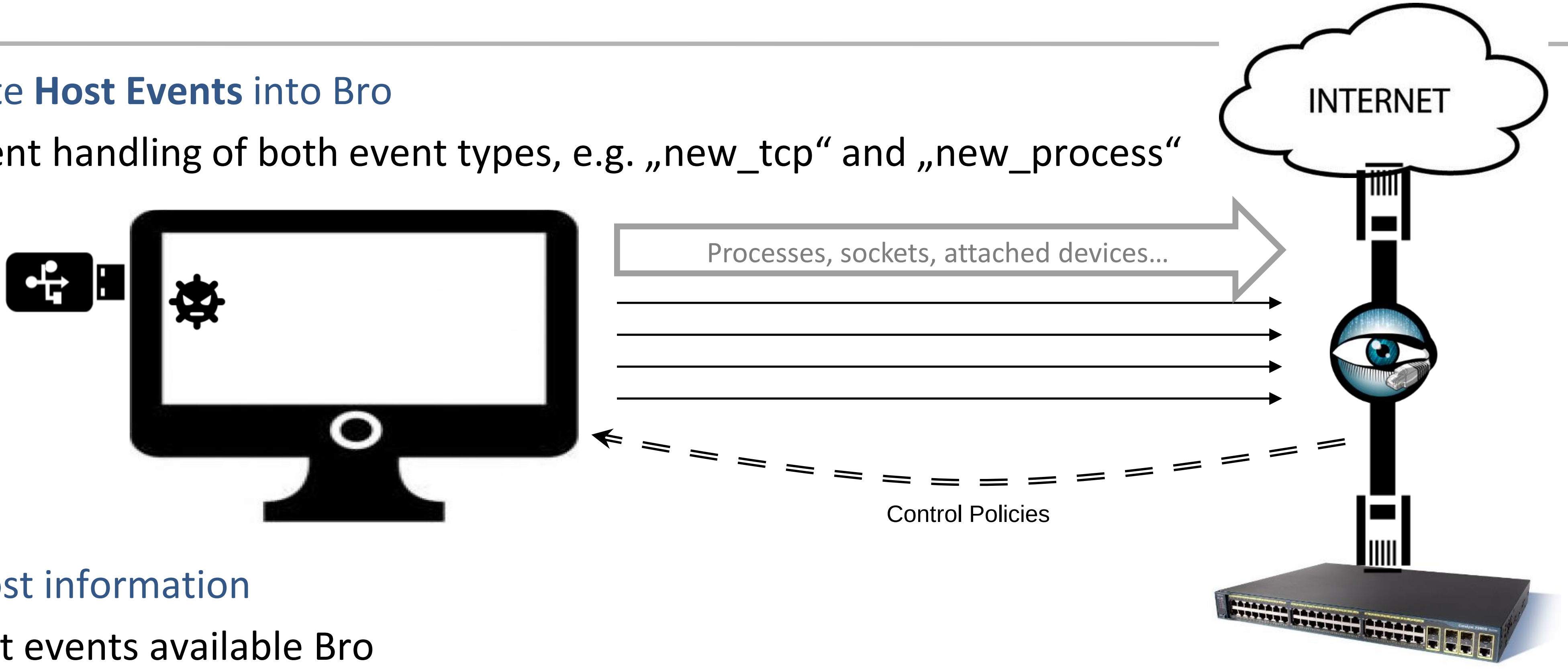
- Monitoring Problems:
  - Some information are available on the hosts only
    - E.g. Logged in user, network application name
  - Encryption of network traffic
    - Limited to meta-data analysis

- Result:
  - Losing visibility on the network infrastructure
    - Dark spots in the network



Requirements

- Goal: Integrate **Host Events** into Bro
  - Transparent handling of both event types, e.g. „new_tcp“ and „new_process“



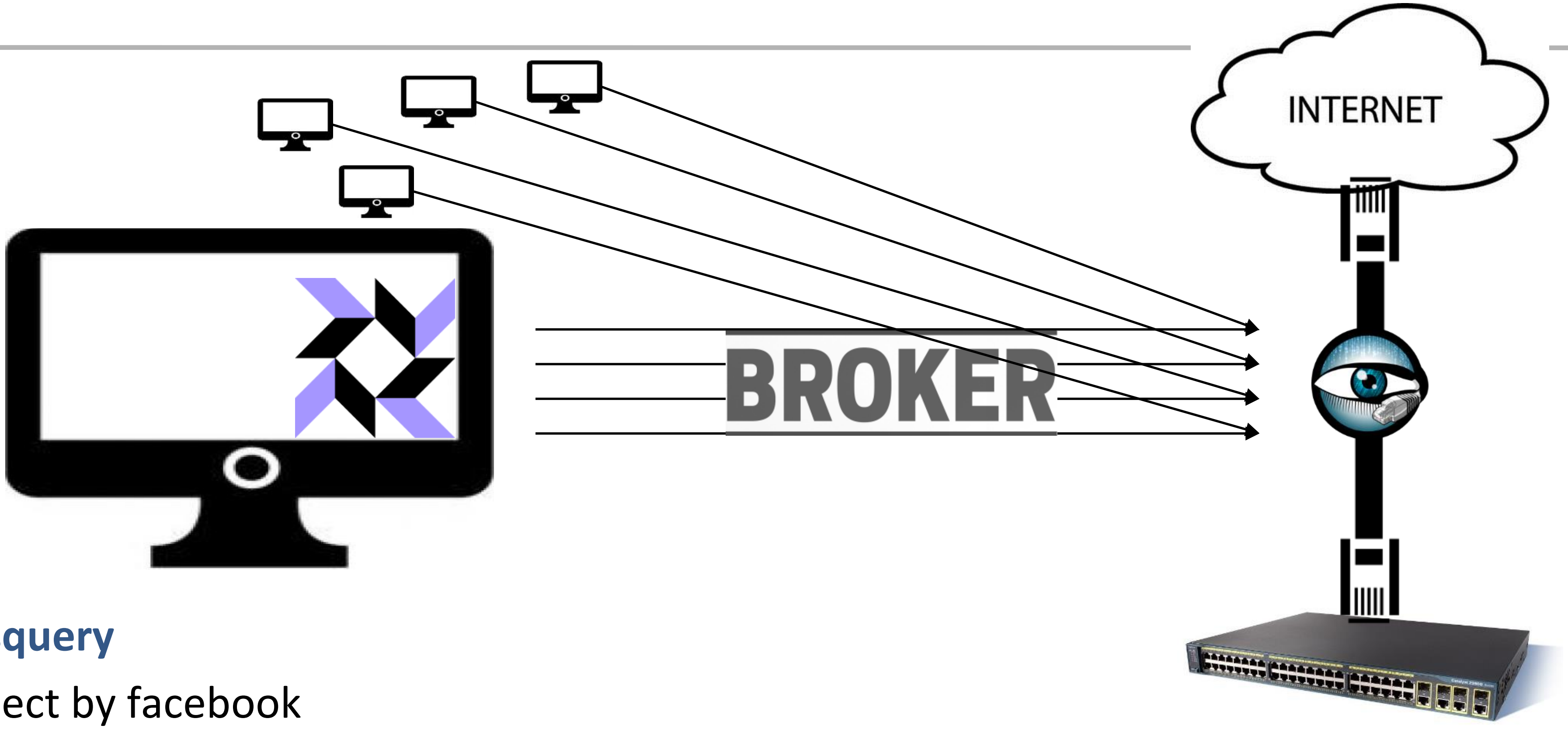
- Extension: Host information
  - Make host events available Bro
    - Additional host information complement network visibility
  - Bro can control *which* events *when* to be emitted by hosts
    - Subscribe to changes (diff events) vs. Retrieve current status (snapshot events)
    - Group hosts and address them individually or collectively

Demo

What if I tell you that you can already have this visibility in your Bro deployment?

Wait, how does this work?

Solution

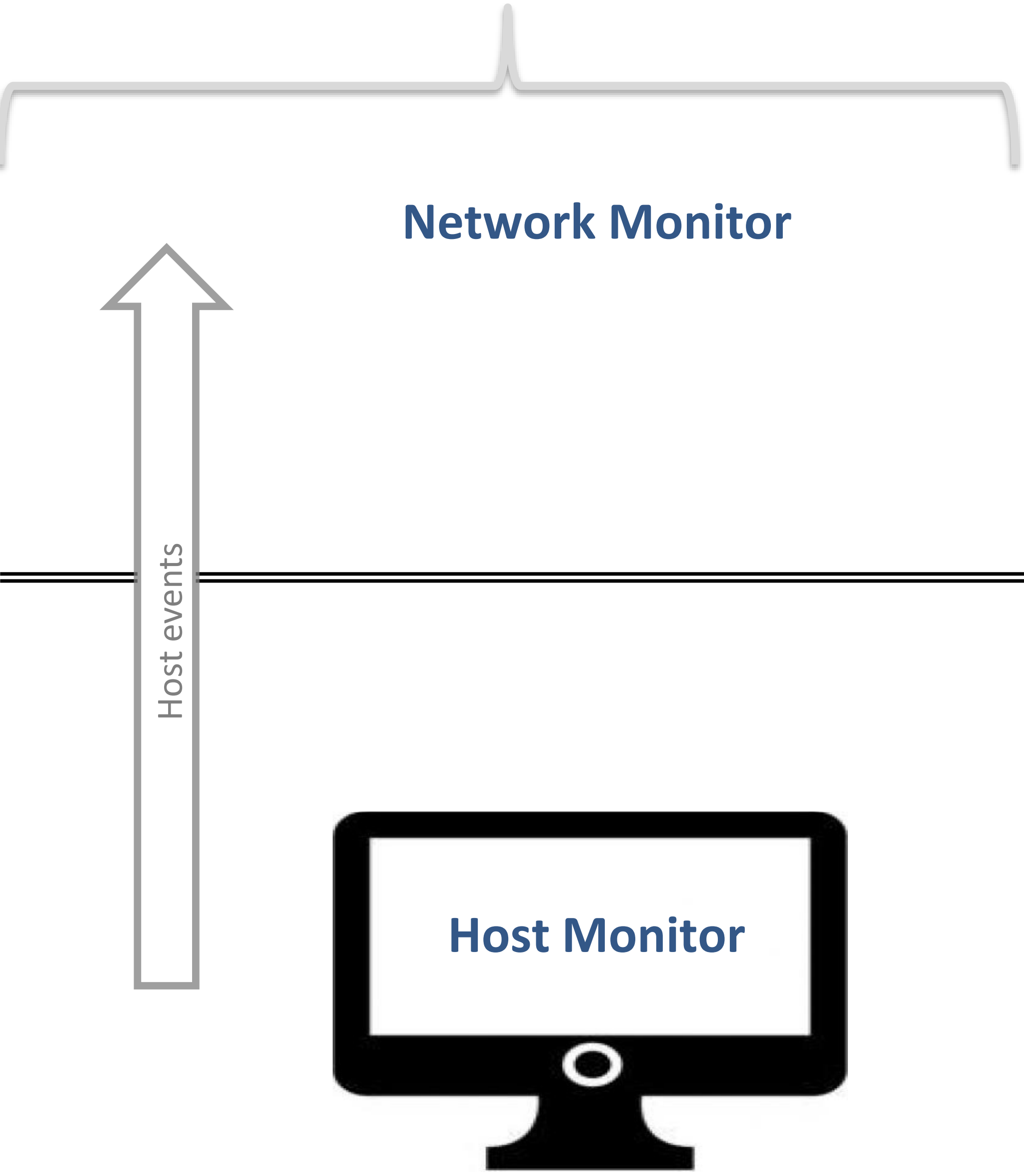


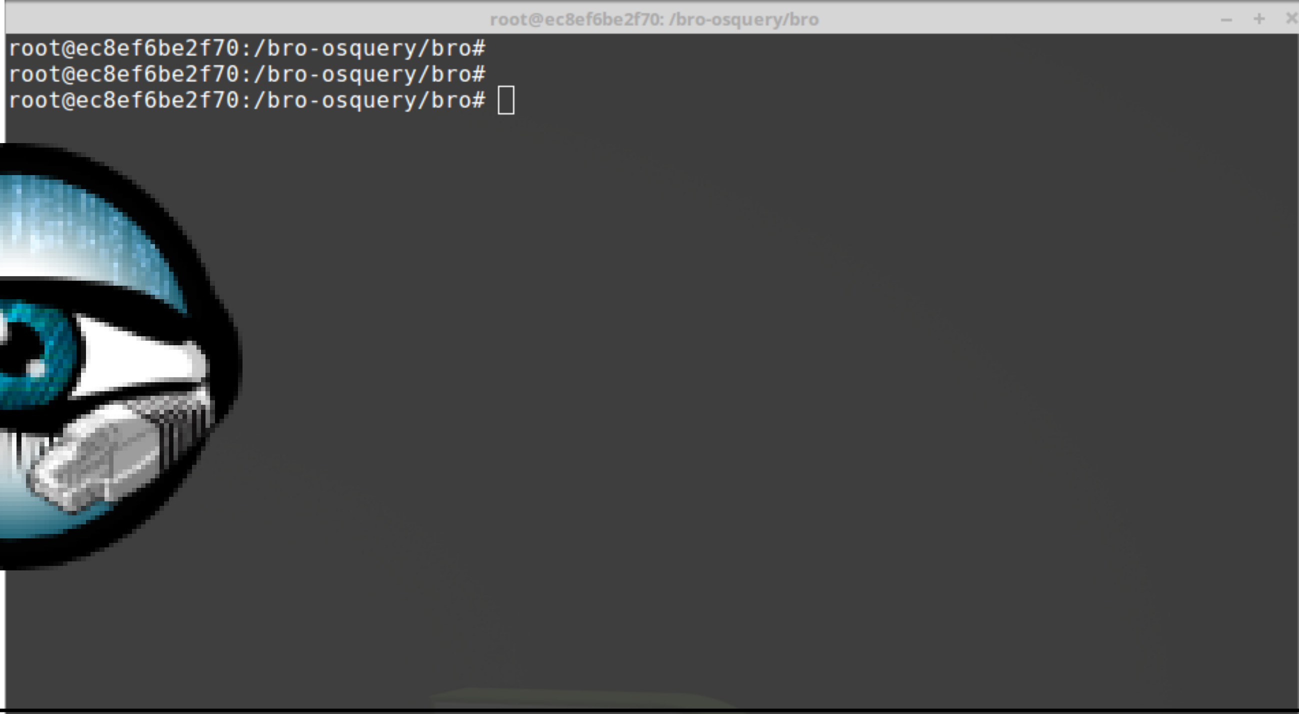
- Host Monitor: **Osquery**
  – Open source project by facebook

- Communication Library: **Broker**

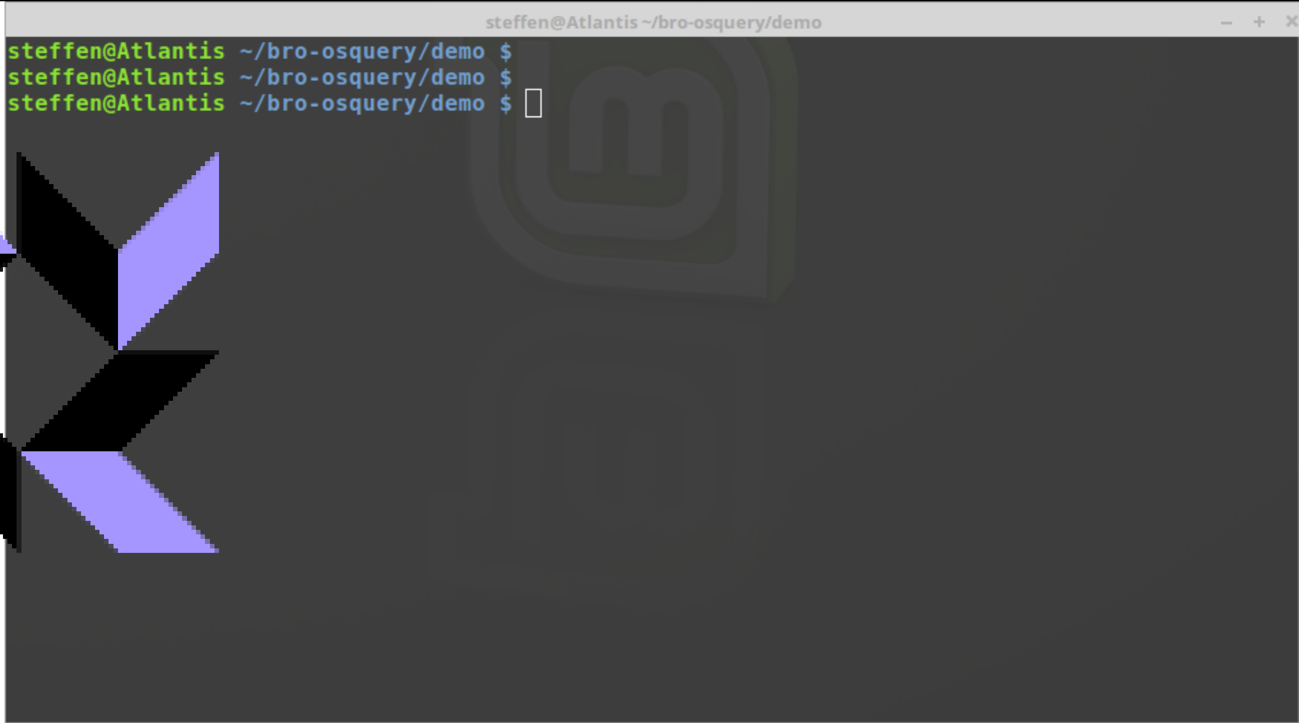
- Bro: Script **framework**

Writing log files about events received from hosts





```
root@ec8ef6be2f70:/bro-osquery/bro#
root@ec8ef6be2f70:/bro-osquery/bro#
root@ec8ef6be2f70:/bro-osquery/bro#
```



```
steffen@Atlantis ~/bro-osquery/demo $
steffen@Atlantis ~/bro-osquery/demo $
steffen@Atlantis ~/bro-osquery/demo $
```

Host Sensor: Osquery

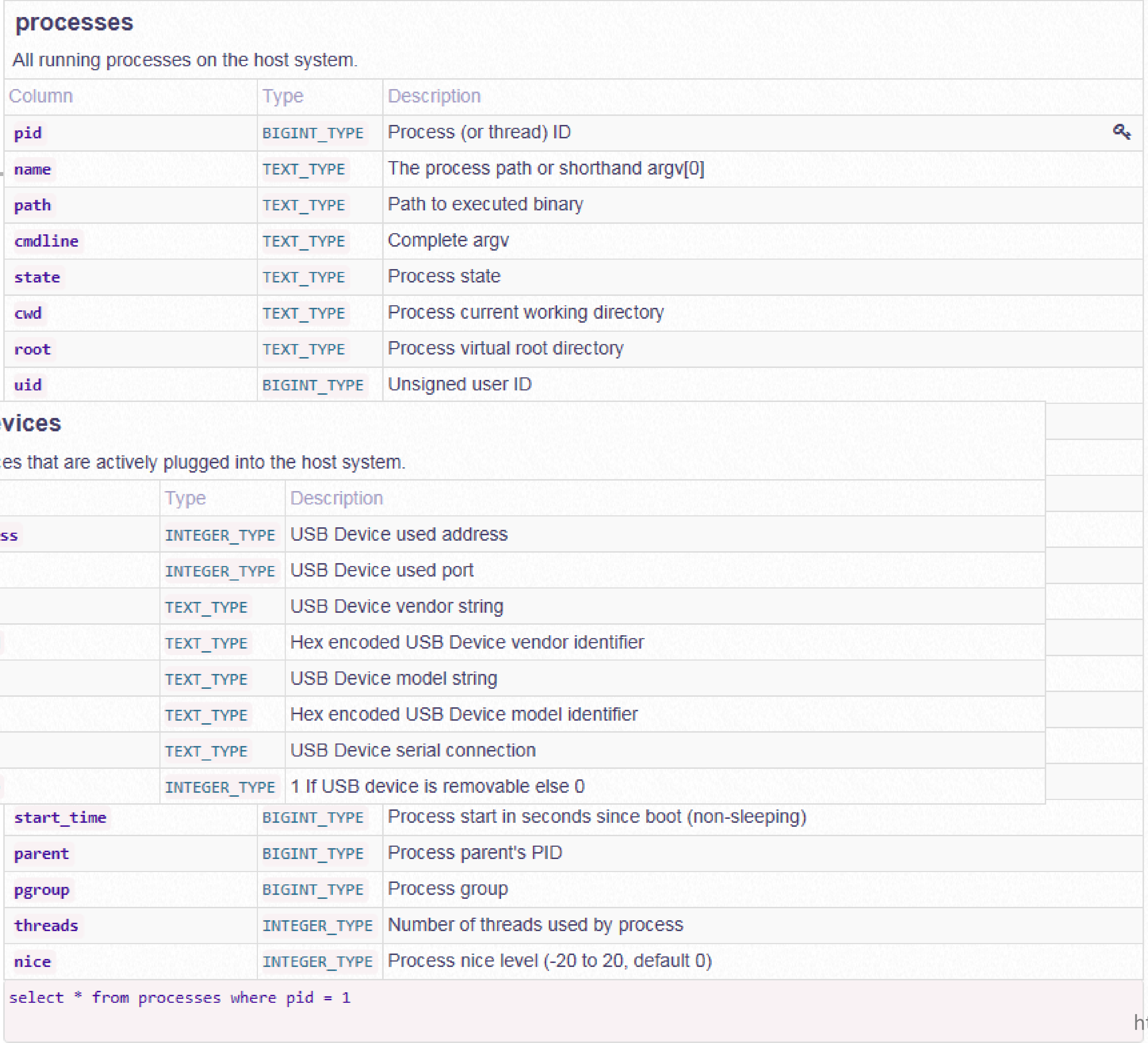
- Operating system as a high-performance relational database
  - SQL tables represent abstract concepts

- 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
- ….

Osquery Tables

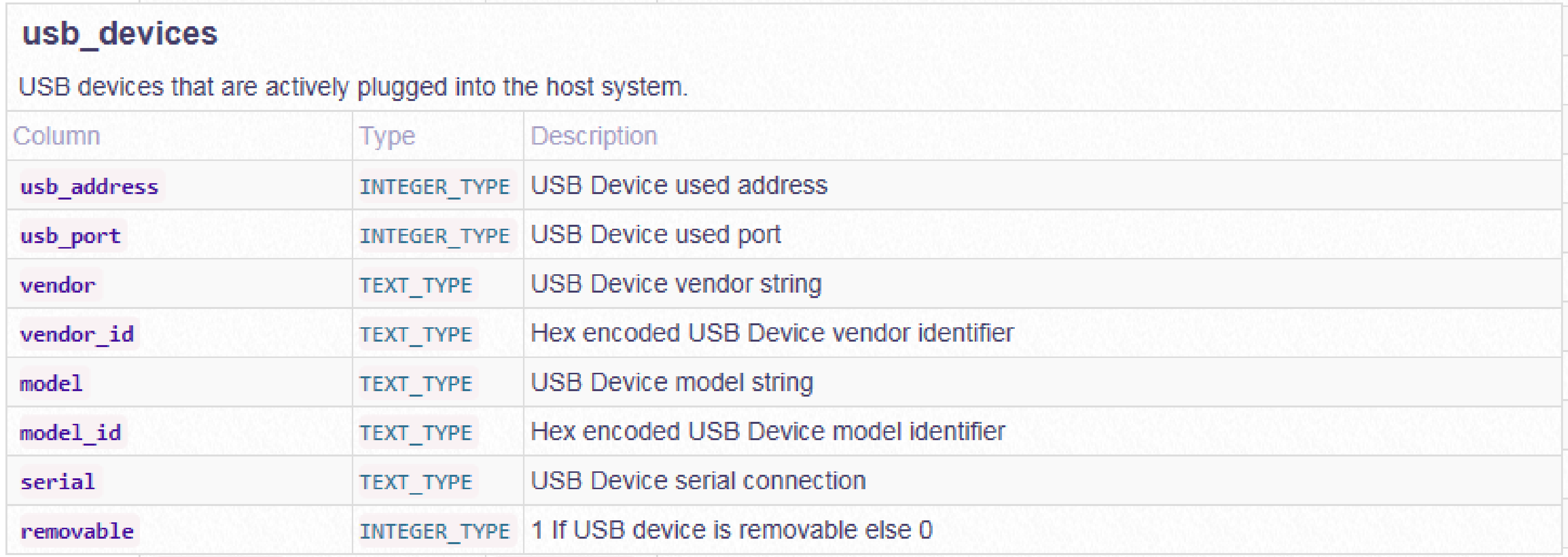


processes

All running processes on the host system.

Column	Type	Description
pid	BIGINT_TYPE	Process (or thread) ID
name	TEXT_TYPE	The process path or shorthand argv[0]
path	TEXT_TYPE	Path to executed binary
cmdline	TEXT_TYPE	Complete argv
state	TEXT_TYPE	Process state
cwd	TEXT_TYPE	Process current working directory
root	TEXT_TYPE	Process virtual root directory
uid	BIGINT_TYPE	Unsigned user ID
start_time	BIGINT_TYPE	Process start in seconds since boot (non-sleeping)
parent	BIGINT_TYPE	Process parent's PID
pgroup	BIGINT_TYPE	Process group
threads	INTEGER_TYPE	Number of threads used by process
nice	INTEGER_TYPE	Process nice level (-20 to 20, default 0)

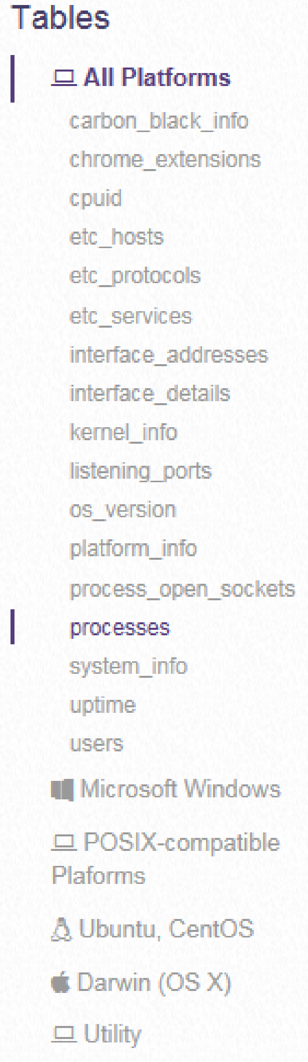
```
select * from processes where pid = 1
```



usb_devices

USB devices that are actively plugged into the host system.

Column	Type	Description
usb_address	INTEGER_TYPE	USB Device used address
usb_port	INTEGER_TYPE	USB Device used port
vendor	TEXT_TYPE	USB Device vendor string
vendor_id	TEXT_TYPE	Hex encoded USB Device vendor identifier
model	TEXT_TYPE	USB Device model string
model_id	TEXT_TYPE	Hex encoded USB Device model identifier
serial	TEXT_TYPE	USB Device serial connection
removable	INTEGER_TYPE	1 If USB device is removable else 0



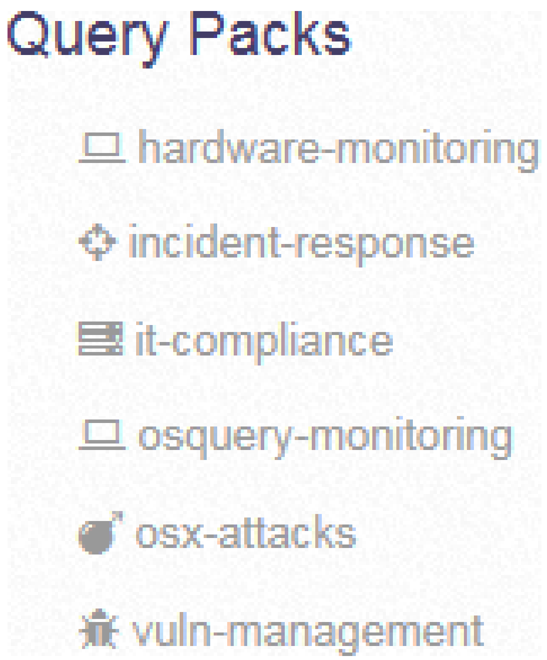
Tables

- All Platforms
  - carbon_black_info
  - chrome_extensions
  - cpuid
  - etc_hosts
  - etc_protocols
  - etc_services
  - interface_addresses
  - interface_details
  - kernel_info
  - listening_ports
  - os_version
  - platform_info
  - process_open_sockets
  - processes
  - system_info
  - uptime
  - users
- Microsoft Windows
- POSIX-compatible Plaforms
- Ubuntu, CentOS
- Darwin (OS X)
- Utility

https://osquery.io/

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


Query Packs

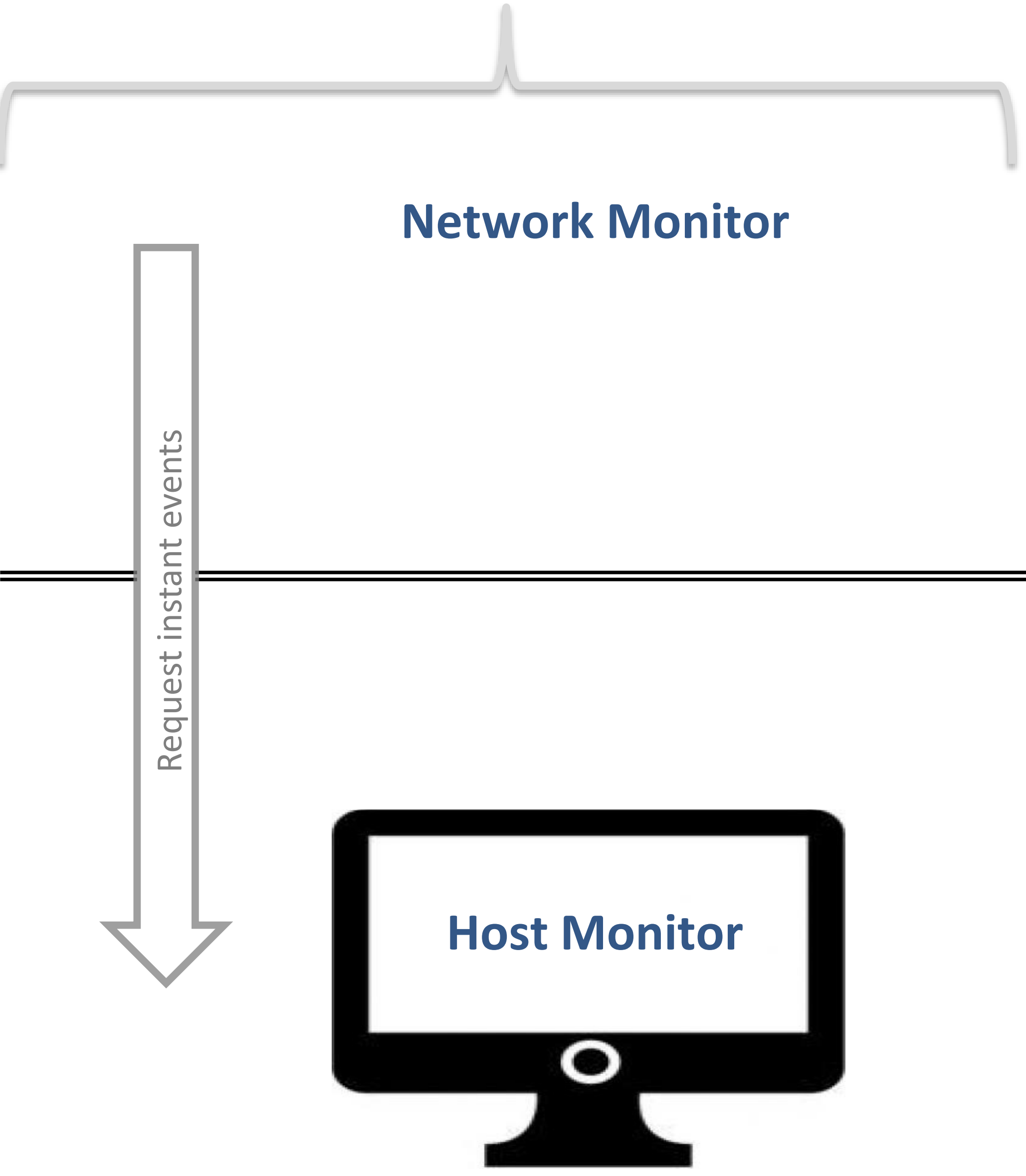
hardware-monitoring
incident-response
it-compliance
osquery-monitoring
osx-attacks
vuln-management

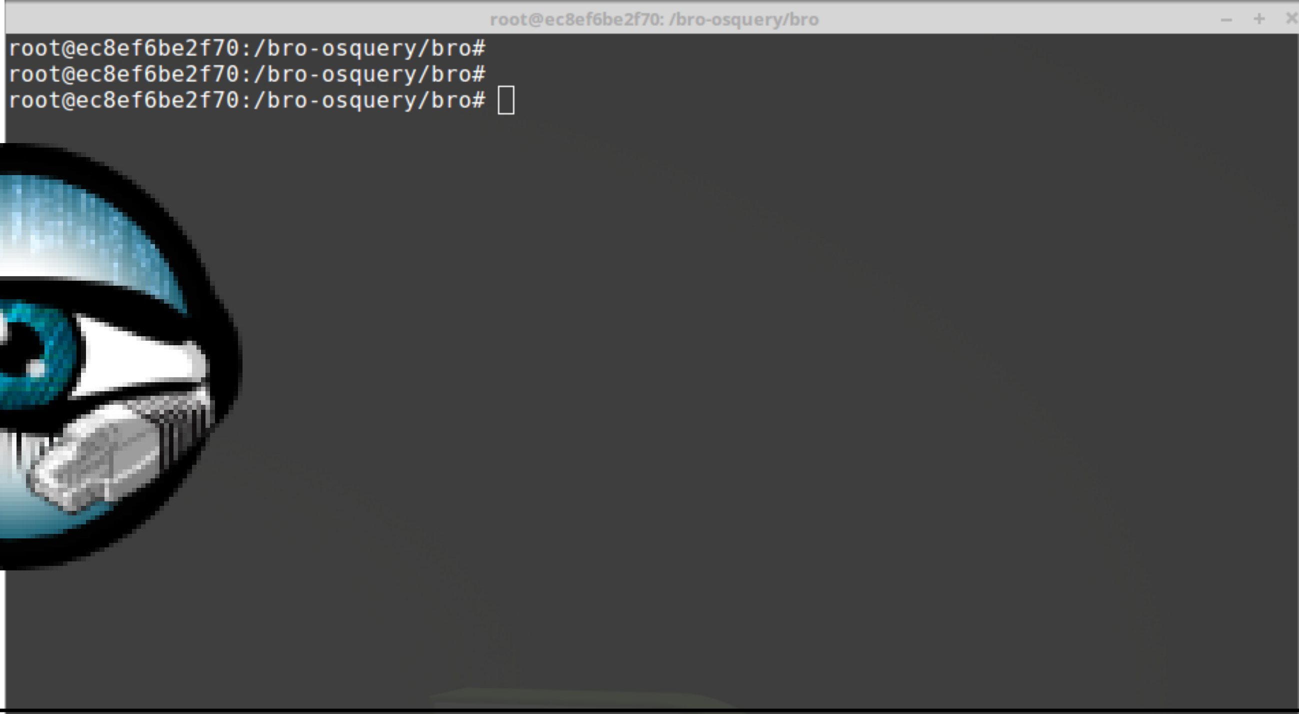
https://osquery.io/

Demo

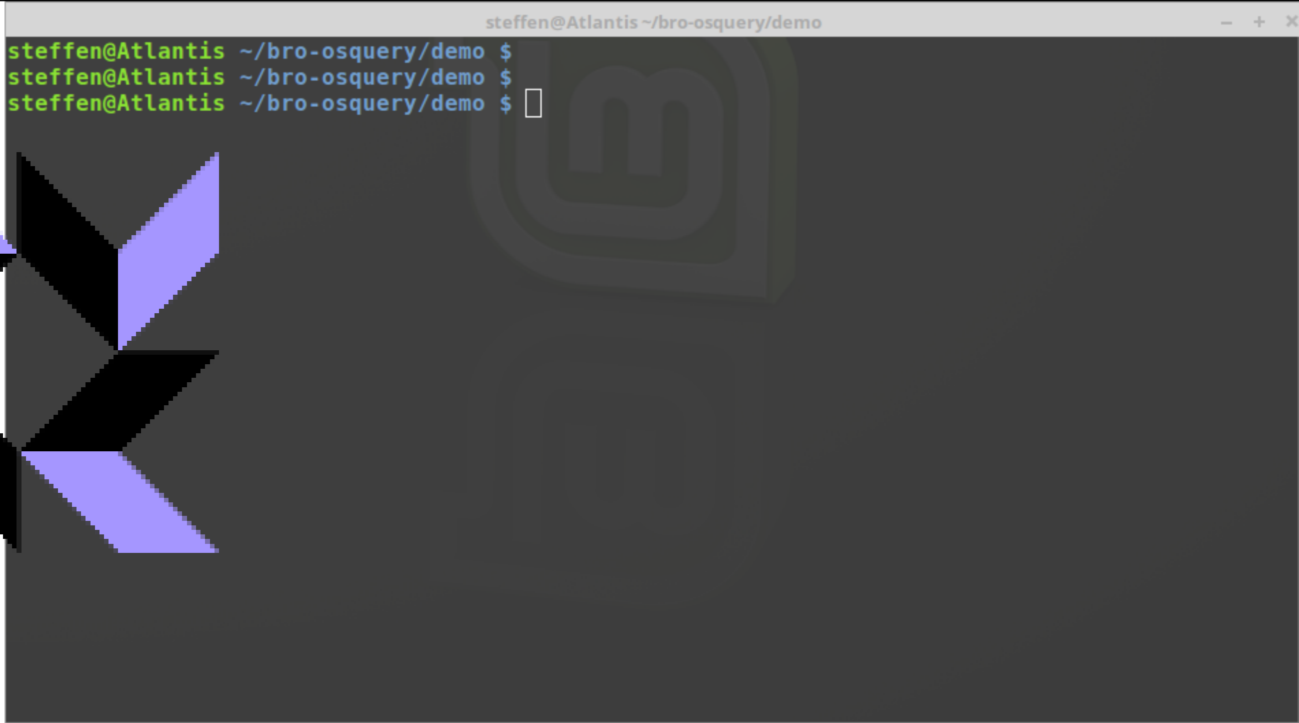
Detecting processes and USB devices…
Is that all you can do?!?

Ask host about current status on demand




root@ec8ef6be2f70: /bro-osquery/bro

```
root@ec8ef6be2f70:/bro-osquery/bro#
root@ec8ef6be2f70:/bro-osquery/bro#
root@ec8ef6be2f70:/bro-osquery/bro#
```


steffen@Atlantis ~/bro-osquery/demo

```
steffen@Atlantis ~/bro-osquery/demo $
steffen@Atlantis ~/bro-osquery/demo $
steffen@Atlantis ~/bro-osquery/demo $
```

Idea: Extending conn.log by user and application identification

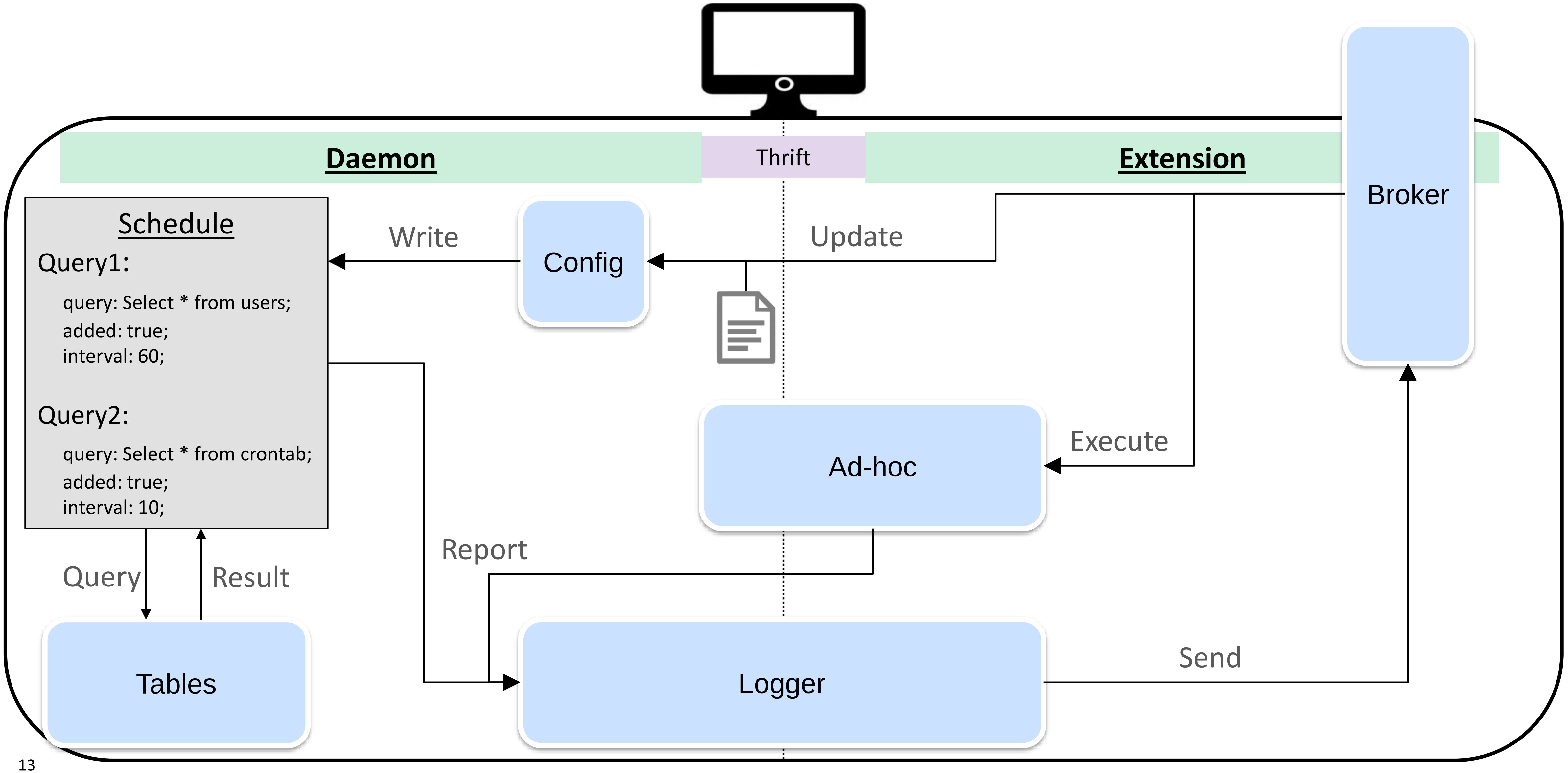
- Bro captures new TCP/UDP connections
  - You know host address and port (source)

```
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;
```

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

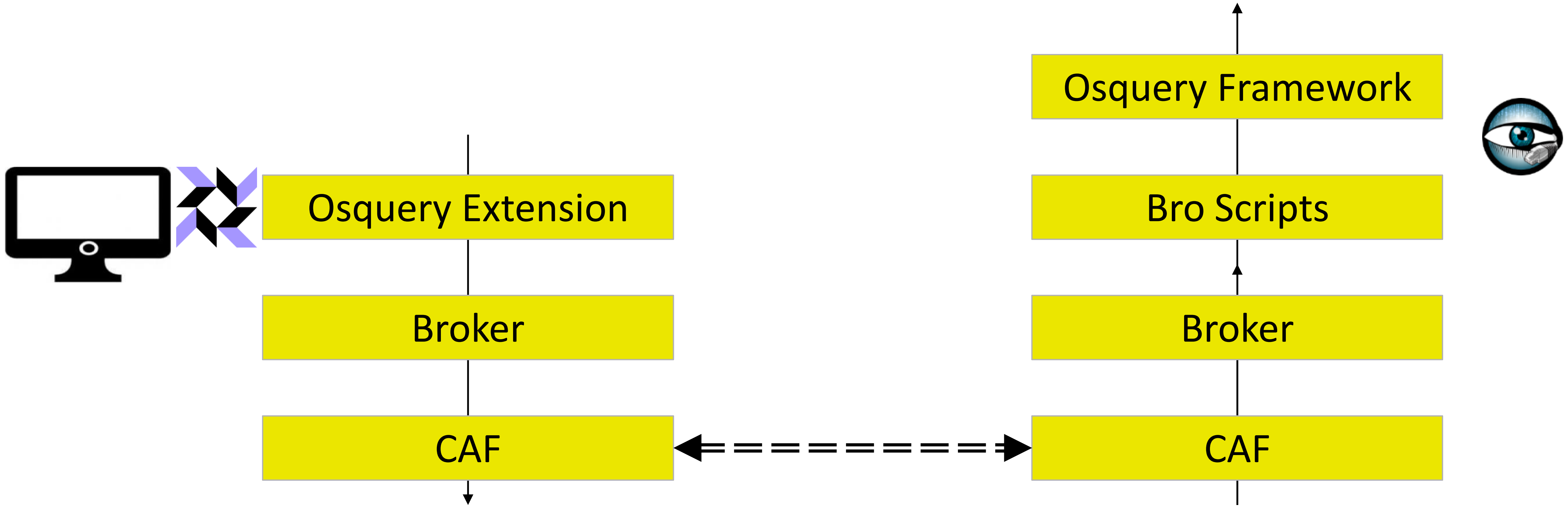
```
#fields  ts                 uid          id.orig_h      id.orig_p  id.resp_h      [...]  application  user
#types   time               string       addr           port addr  addr           [...]  string       string

         1258531221.486539  arKYeMETxOg  192.168.1.102  68         192.168.1.1    [...]  appA         alice
         1258531680.237254  nQcgTWjvg4c  192.168.1.103  37         192.168.1.255  [...]  appB         alice
         1258531693.816224  j4u32Pc5bif  192.168.1.102  37         192.168.1.255  [...]  conficker    bob
         1258531635.800933  k6kgXLOoSKl  192.168.1.103  138        192.168.1.255  [...]  firefox      eve
         1258531693.825212  TefuqmnG4bh  192.168.1.102  138        192.168.1.255  [...]  appC         bob
         1258531803.872834  5OKnovw6xl4  192.168.1.104  137        192.168.1.255  [...]  appD         alice
         1258531747.077012  FrJExwHcSal  192.168.1.104  138        192.168.1.255  [...]  appA         trudy
         1258531924.321413  3PKsZ2Uye21  192.168.1.103  68         192.168.1.1    [...]  appE         eve
         [...]
```

Osquery Extension: Bro-Osquery



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

- 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()

```
{
  // Bro-Osquery option to configure the extension
  "bro": {
    // The IP and port of the Bro endpoint.
    "bro_ip": "",
    //"bro_port": "9999",

    // The predefined unique ID of osquery host.
    // If this is not set at startup, a hardware dependent ID is derived from
    // the host's MAC addresses. Therefore, ID is persistent until interface
    // changes.
    //"uid": "",

    // The predefined groups of osquery host.
    // Groups can be assigned at runtime via broker messages.
    "groups": {
    //     "group1": "eu/de/",
    //     "group2": "uhh/cs/iss"
    }
  },

  // Configure the daemon below:
  "options": {
    // Select the osquery config plugin.
    "config_plugin": "filesystem",

    // Select the osquery logging plugin.
    "logger_plugin": "bro",

    // Enable debug or verbose debug output when logging.
    "verbose": "true"
  },

  // Define a schedule of queries:
  "schedule": {
  },

  // Decorators are normal queries that append data to every query.
  "decorators": {
  },

  "packs": {
  }
}
```

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 <u>your</u> 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