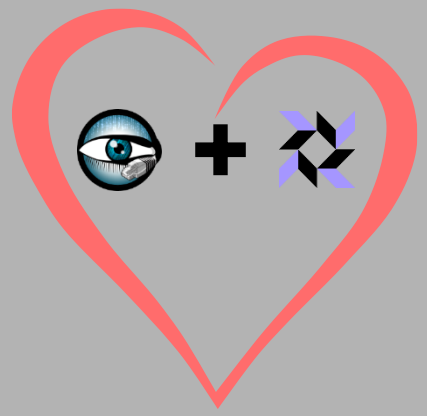
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

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

Bro-Osquery



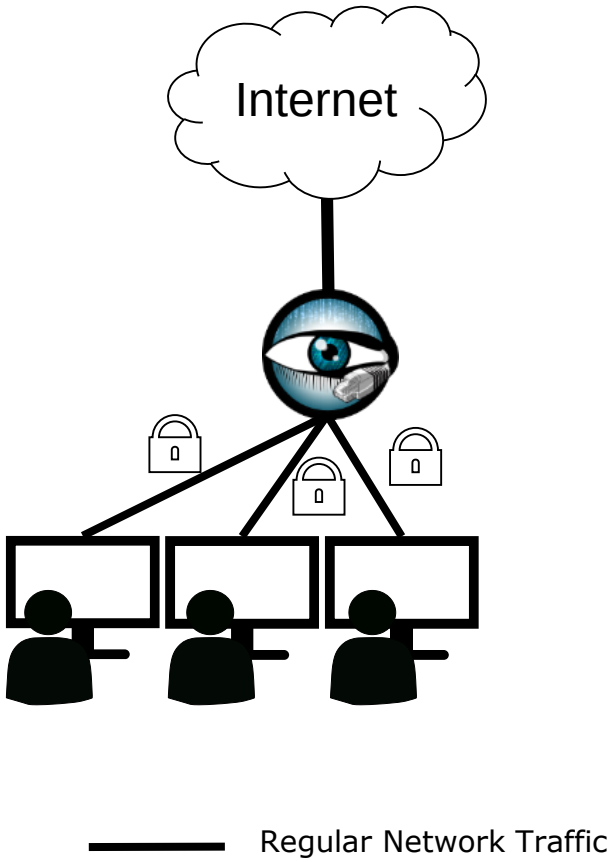
Let Bro know about the hosts it monitors

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

Osquery Host Monitor
https://osquery.io/

Motivation

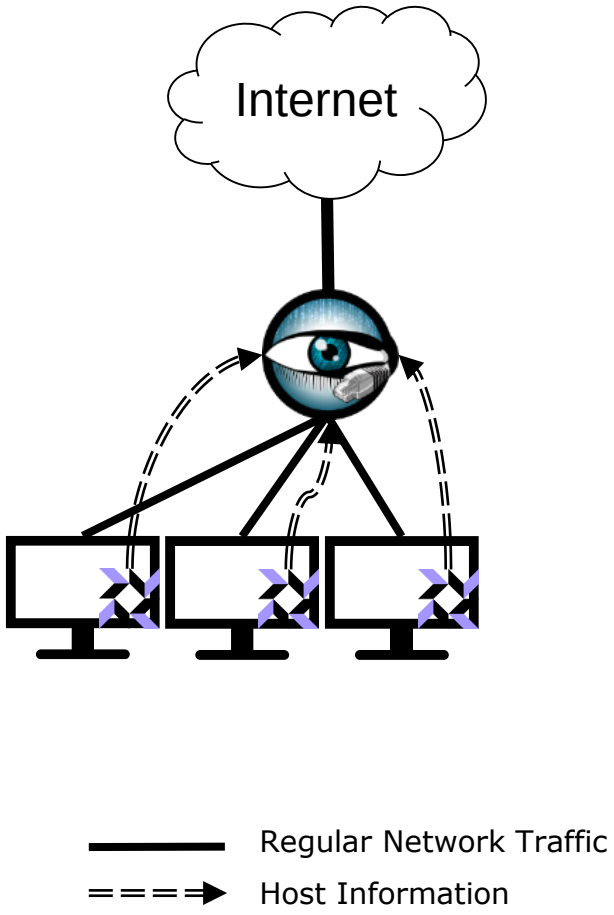
- Today: Bro as ***Network*** Intrusion Detection / Monitoring System
  - Information as seen on the wire
- Monitoring Problems:
  - Some information are available on the hosts only
    - E.g. Logged in user, network application name
  - Encryption of network traffic
    - Limited to meta-data analysis
- Result:
  - Losing visibility on the network infrastructure
    - Dark spots in the network
- Solution: Combination of host and network monitoring
  - More context about network communications
  - More context about communicating applications



Bro-Osquery in a nutshell

- Two types of data sources in your network
  - Network Monitor: Bro
  - Host Monitor: Osquery

- Bro as central analysis platform
  - Monitors network communication
  - Receives data from Osquery hosts
  - Enables correlation of host and network data
    - Which app/user is responsible for specific communication?
  - Detection of (attack) scenarios with knowledge from hosts and network
    - Tracking execution of downloaded files
    - Detecting SSH-Chain
    - Identifying users responsible for data exfiltration



facebook/osquery

Performant endpoint visibility

Osquery in a nutshell

- Open source endpoint monitoring tool by facebook
- Operating system as a high-performance relational database
  - SQL tables represent abstract concepts

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

- Power of a complete SQL language and dozens of useful tables (about 200)

processes

All running processes on the host system.

Improve this Description on Github

COLUMN	TYPE	DESCRIPTION
pid	BIGINT	Process (or thread) ID
name	TEXT	The process path or shorthand argv[0]
path	TEXT	Path to executed binary
cmdline	TEXT	Complete argv
state	TEXT	Process state
cwd	TEXT	Process current working directory
root	TEXT	Process virtual root directory
uid	BIGINT	Unsigned user ID
gid	BIGINT	Unsigned group ID

usb_devices

USB devices that are actively plugged into the host system.

Improve this Description on Github

COLUMN	TYPE	DESCRIPTION
usb_address	INTEGER	USB Device used address
usb_port	INTEGER	USB Device used port
vendor	TEXT	USB Device vendor string
vendor_id	TEXT	Hex encoded USB Device vendor identifier
version	TEXT	USB Device version number
model	TEXT	USB Device model string
model_id	TEXT	Hex encoded USB Device model identifier
serial	TEXT	USB Device serial connection
class	TEXT	USB Device class

patches

Lists all the patches applied. Note: This does not include patches applied via MSI c
Update (e.g. Service Packs).

Improve this Description on Github

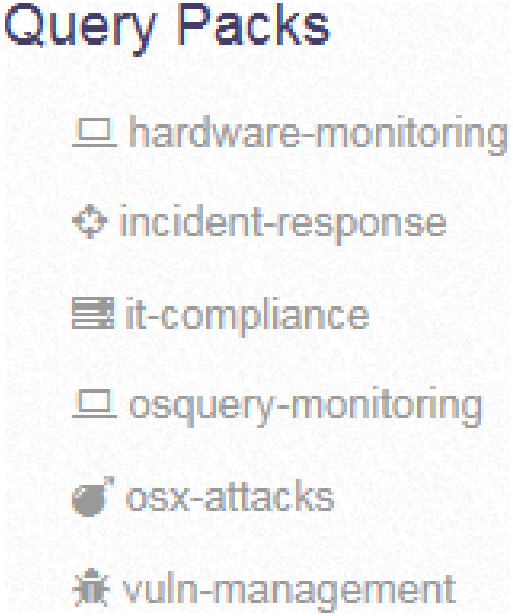
COLUMN	TYPE	DESCRIPTION
csname	TEXT	The name of the host the patch is installed on.
hotfix_id	TEXT	The KB ID of the patch.
caption	TEXT	Short description of the patch.
description	TEXT	Fuller description of the patch.
fix_comments	TEXT	Additional comments about the patch.
installed_by	TEXT	The system context in which the patch as installed.
install_date	TEXT	Indicates when the patch was installed. Lack of a valu not installed.
installed_on	TEXT	The date when the patch was installed.

Osquery in a nutshell (2)

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

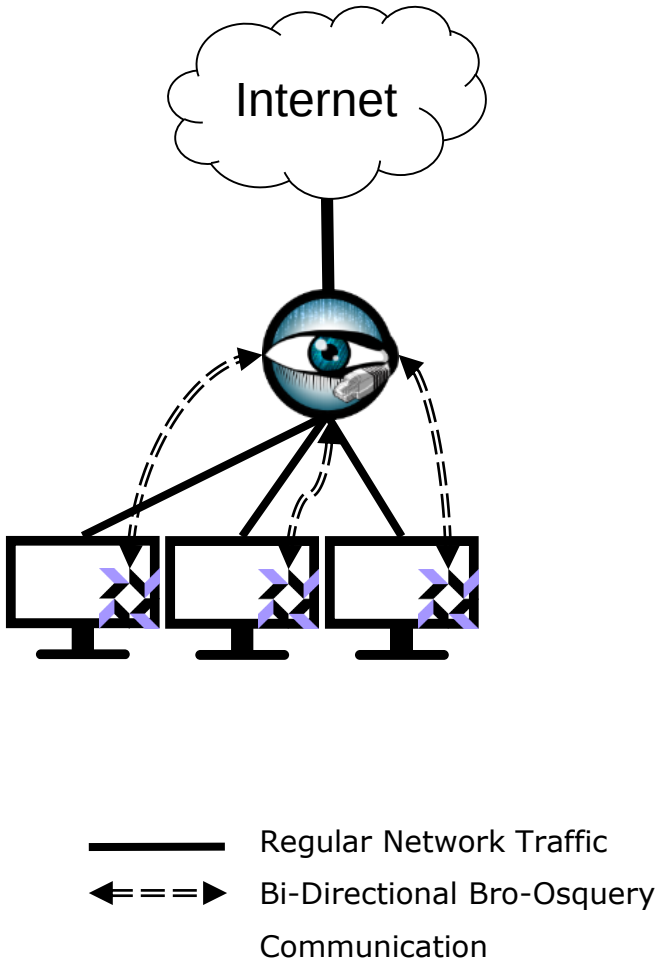
- Host monitoring **daemon/agent**
  - Allows to schedule queries to be executed regularly
  - Aggregates query results over time
  - Generates logs which indicate state changes in infrastructure

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



Features of Bro-Osquery

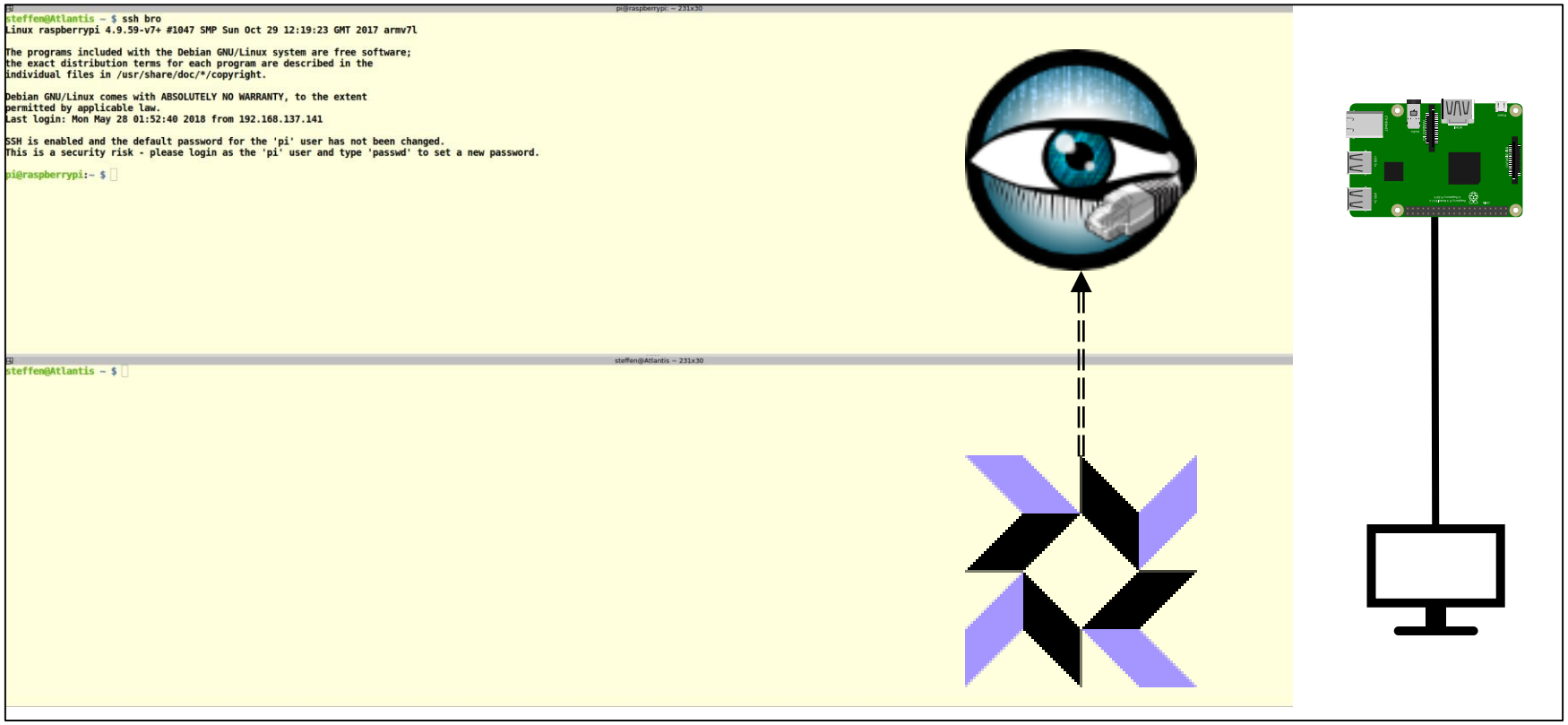
- Controlling Osquery schedule and receiving results with Bro
  - Central control instance for querying groups of Osquery hosts
    - Maintaining query schedule of hosts at runtime
    - Ability to execute one-time queries
  - Results are natively fed back and are available in Bro script
- Logging query results
  - Central logging of structured data as Bro log files
  - Extending network sessions with users/applications
- Detection of sophisticated scenarios
  - Ability to write Bro scripts with access to full host and network data
  - Event-based detection in real-time extensible by custom scripts
- Large-scale deployments
  - Load distribution using proxies and/or multiple Bros



Demo: Logging of SQL Queries

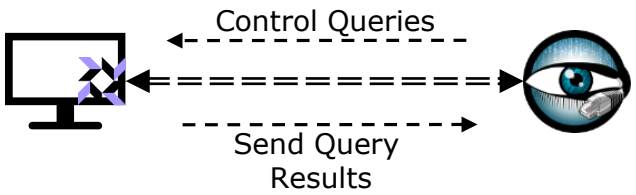
- Controlling and logging the query results for all connected Osquery hosts

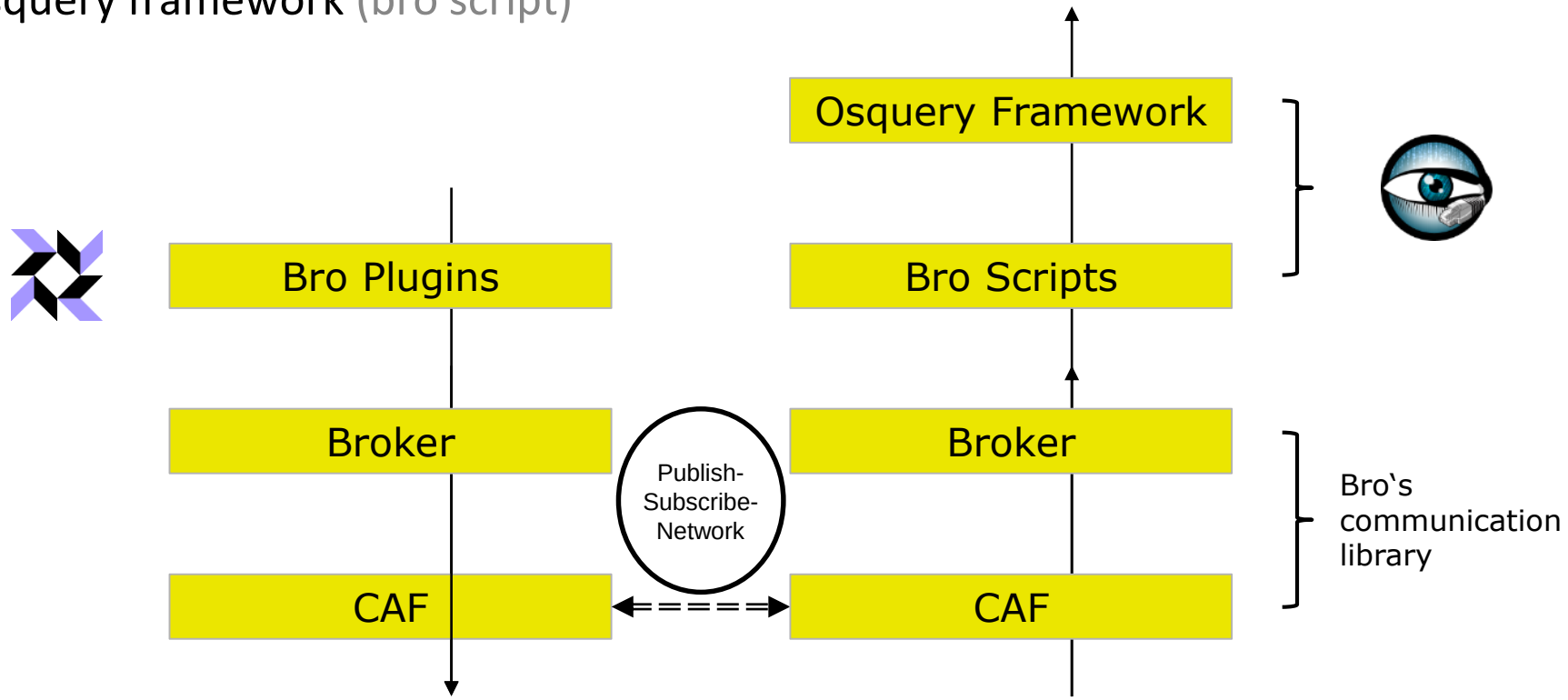
```
event bro_init()
{
  Log::create_stream(LOG, [$columns=Info, $path="osq-processes"]);

  local query = [$ev=host_processes,
    $query="SELECT pid,name,path,cmdline,cwd,root,uid,gid,on_disk,start_time,parent,pgroup FROM processes"];
  osquery::subscribe(query);
}
```



Network Stack in Bro-Osquery

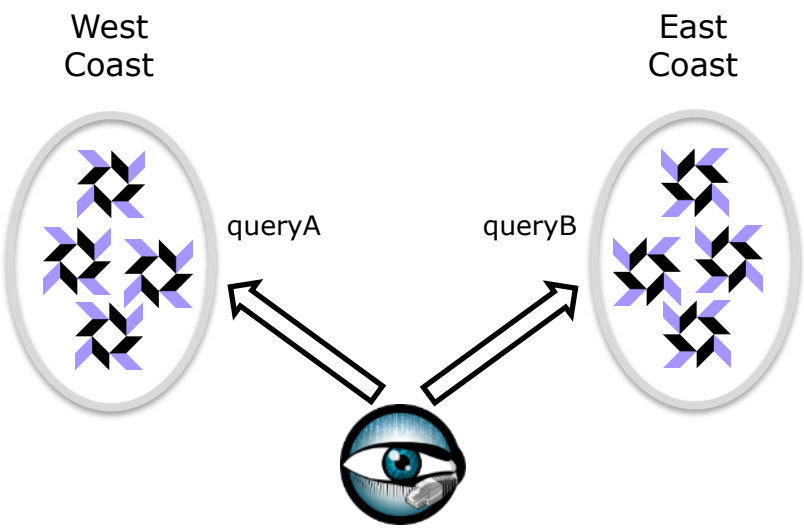
- Extensions to the existing open-source tools
  - In Osquery:
    - Bro plugins including communication library (c++)
  - In Bro:
    - Osquery framework (bro script)





Using the Osquery Framework

- Organization of Osquery hosts
  - Hosts are organized in groups (non-disjoint)
    - Statically by configuration
    - Dynamically based on IP subnets
  - Groups can be addressed by SQL queries
  - Default group contains all Osquery hosts



- Communication with Osquery hosts
  - API for organizing groups (IP subnet -> group name)

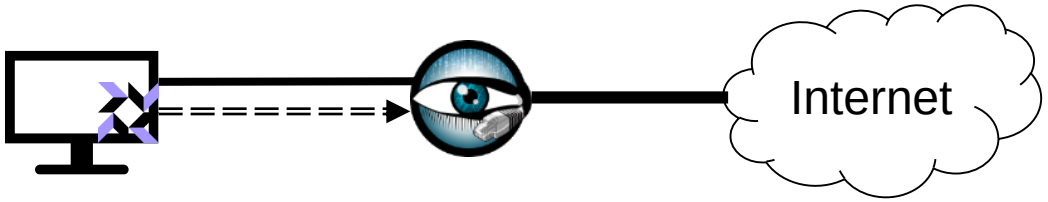
```
global set_host_group: function(range: subnet, group: string);
```

  - API for subscribing queries (query result -> topic name)

```
global subscribe: function(q: Query, host: string &default="", group: string &default="");
```

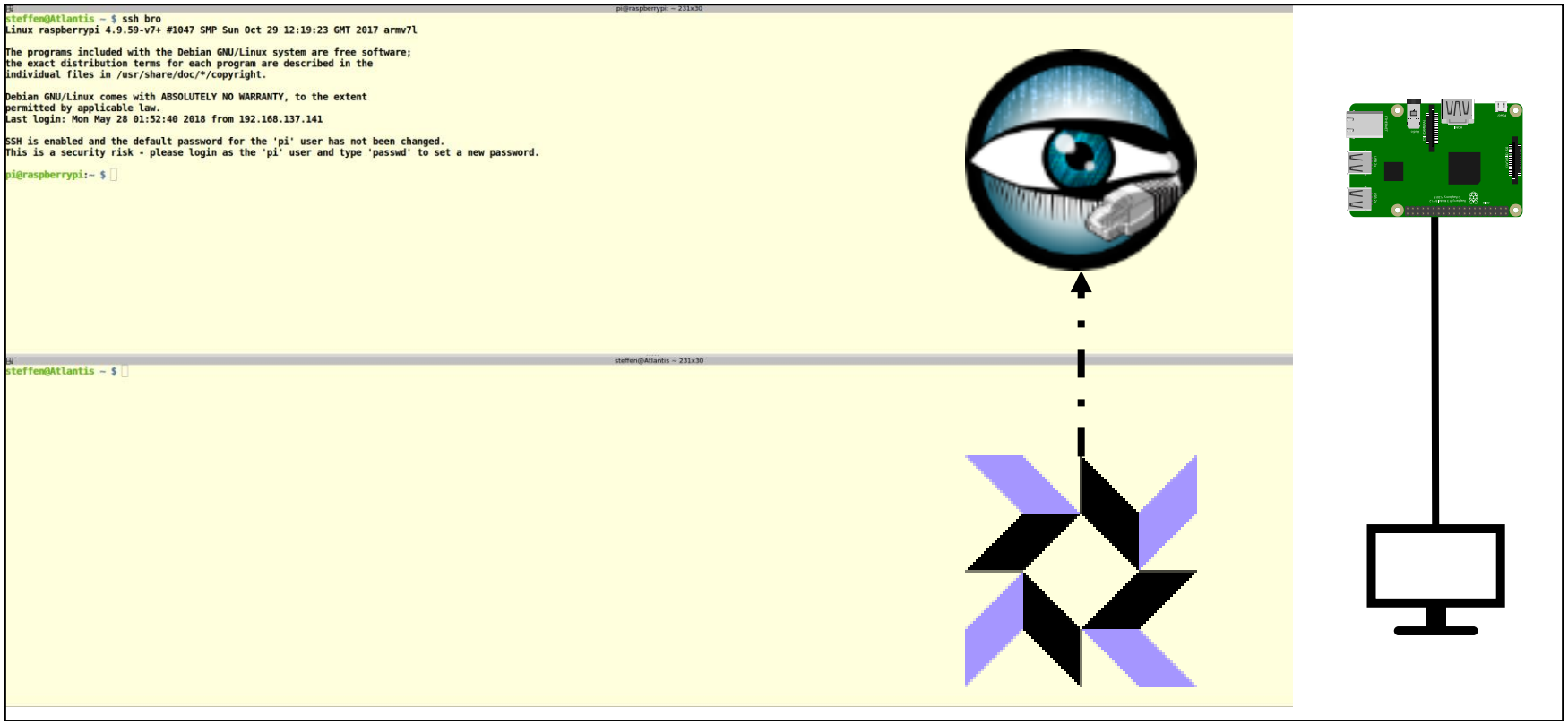
  - API for executing one-time queries (query result -> topic name)

```
global execute: function(q: Query, host: string &default="", group: string &default="");
```

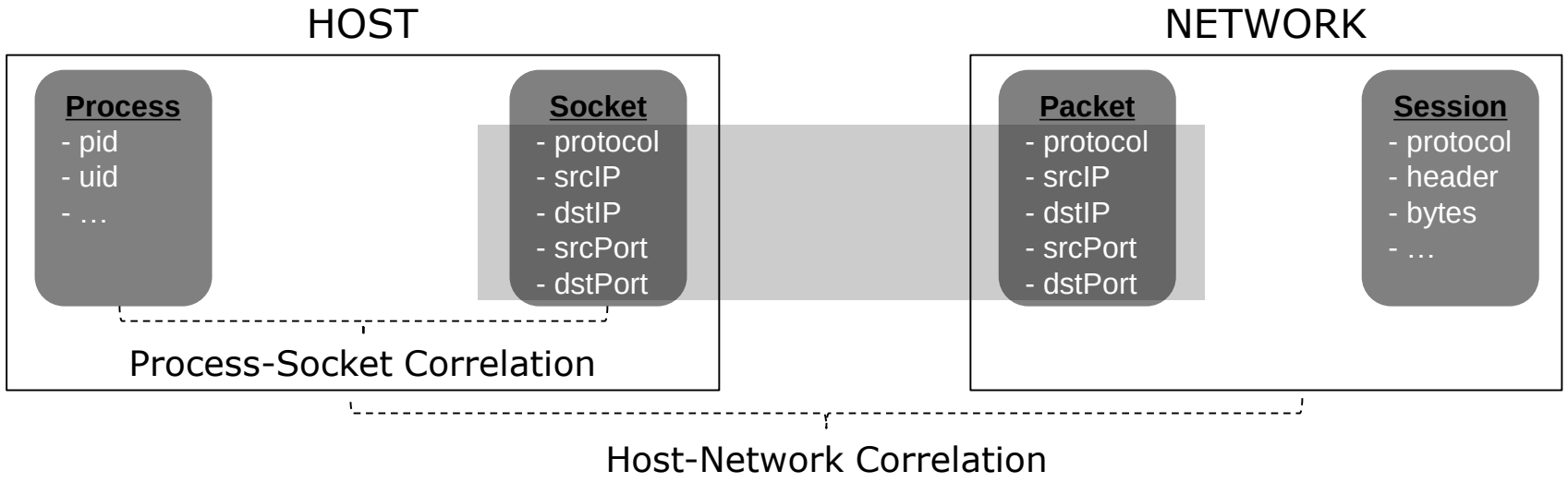


Demo: Host-Network Correlation

- Tie username and process to TCP connections



Process-Socket Correlation



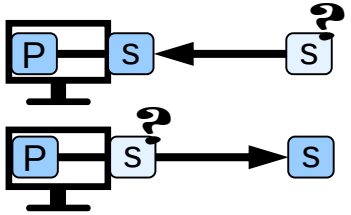
- Process-Socket Correlation based on audit
  - Processes: Event-based table "process_events"
  - Socket: Event-based table "socket_events"
    - Incomplete five-tuple socket
    - Two possible socket actions: "bind" and "connect"

action	protocol	local_addr	local_port	remote_addr	remote_port	process ID
connect	✗	✗	✗	<remote_addr>	<remote_port>	<pid>
bind	✗	<local_addr>	<local_port>	✗	✗	<pid>

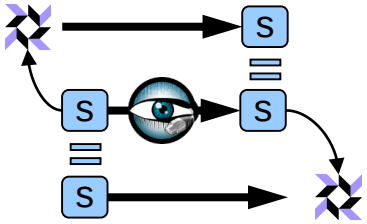
Host-Network Correlation

- Process-Socket Correlation
  - Merging of process/socket events based on common process ID
  - Process-Socket data of each host
    - Socket binds on local IPs and ports
    - Socket connects to remote IPs and ports

P = Program
S = (IP:Port)



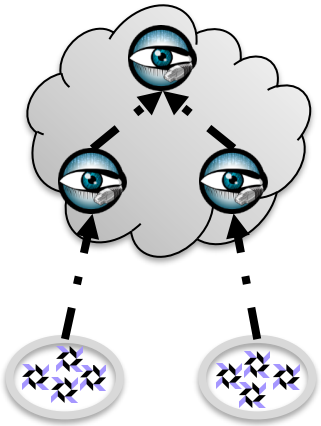
- Host-Network Correlation for specific network connection
  - Matching the five-tuples that identify
    - Sockets on hosts
    - Connections in the network



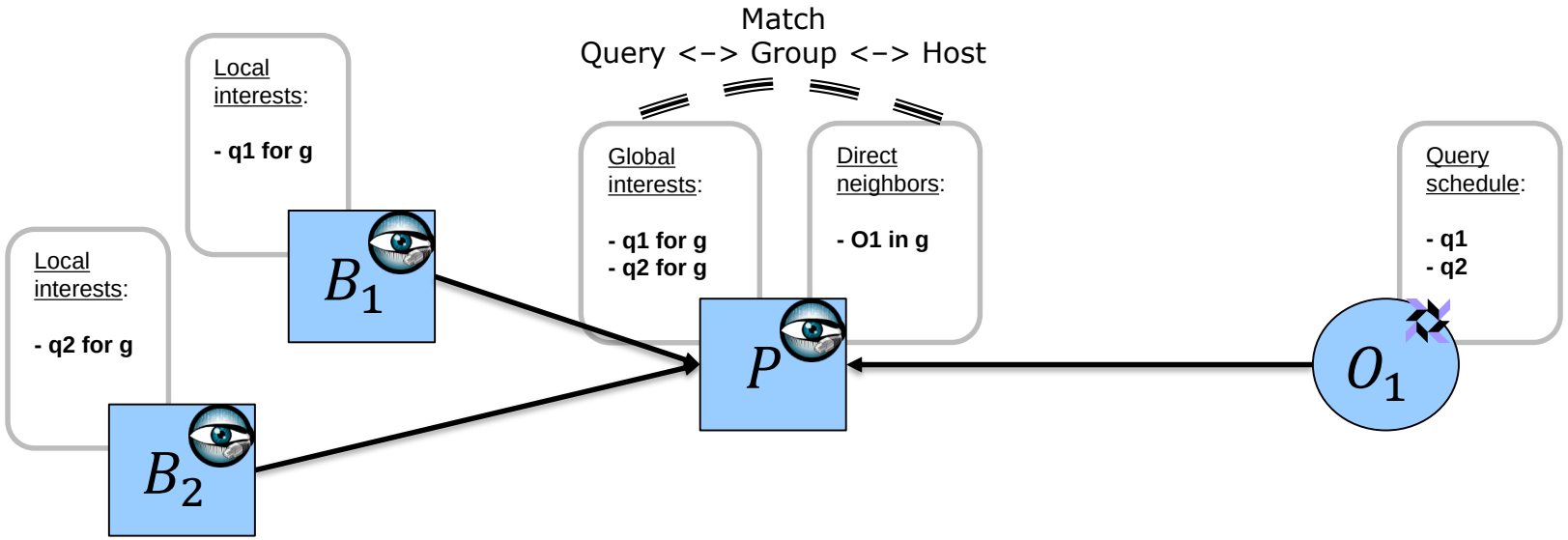
- Host-Network Correlation with Process-Socket Correlation based on audit
  - Identify hosts for source and destination IP of the connection
  - Search the Process-Socket data of the two hosts for specific network connection
    - Source host: Match remote address (IP+Port) only
    - Destination host: Match local address (IP+Port) only

Large-Scale Deployments

- Load distribution through proxies and multiple Bros
  - Backbone consists out of Bros and proxies
  - Queries of interest pushed to backbone edges
  - Osquery hosts connect to an edge Bro/proxy



- Distribution of interests



Project Status of Bro-Osquery

- [Osquery] Complete view on processes
  - Using event-based table to capture short-lived processes
  - Table contains only “execve” syscalls
  - Network communication probably by asynchronous threads
    - Created by “fork”/”clone” syscall

- [Bro] Script packets
  - Adapting scripts to Bro Package Manager
  - For better usability

- [Bro-Osquery] SSL between Osquery and Bro
  - Configurable SSL Certificates/Passwords
  - Authentication?

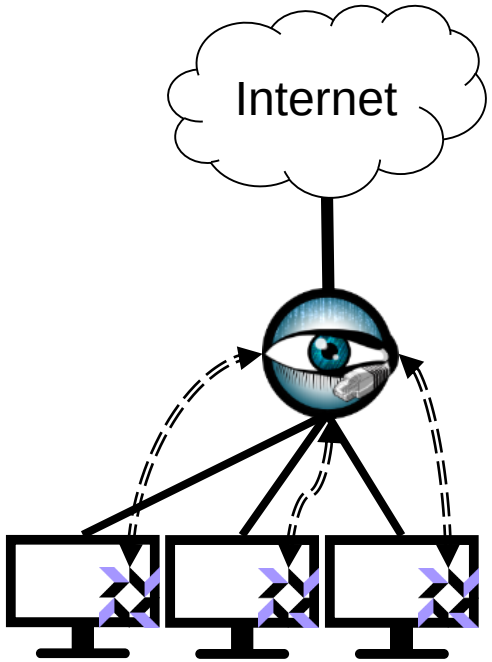
- [Bro-Osquery] Large-scale testbed
  - Are you interested in running Bro-Osquery?

How to run Bro-Osquery?

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

- Install Osquery-featured Bro
  - Build from source for required development features
  - Install the osquery framework as Bro scripts
  - Use existing/custom Bro scripts to query Osquery hosts

- Install Bro-featured Osquery
  - Build from fork until Bro is officially supported
  - Optionally: Set up as service and write configuration file



Questions?