Using Bro to Detect Network Intruders: Experiences and Status

Vern Paxson

Network Research Group
Lawrence Berkeley National Laboratory
University of California, Berkeley

vern@ee.lbl.gov

RAID '98
September 15, 1998

Design goals & constraints:

High-speed, large volume monitoring (FDDI).

No packet filter drops.

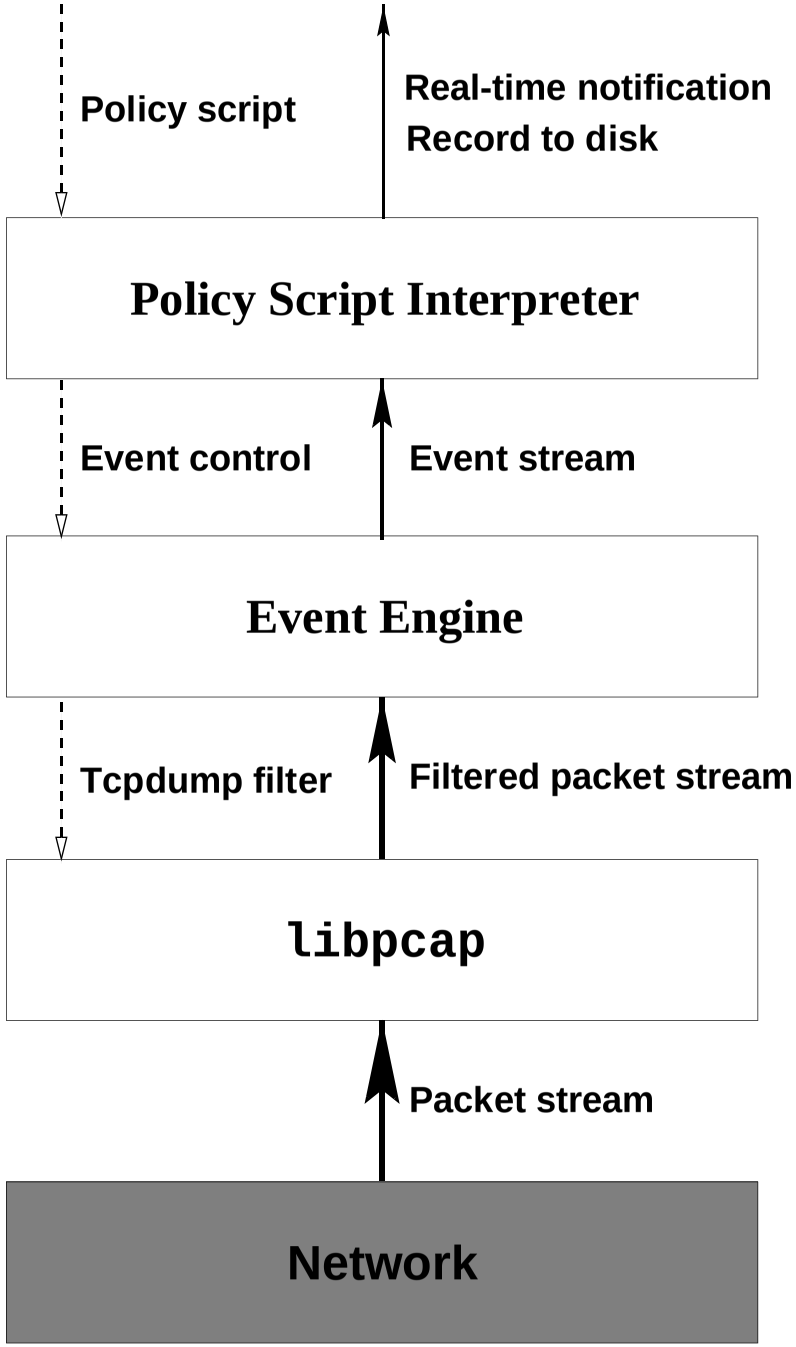
Real-time notification.

Mechanism separate from policy.

Extensible.

Avoid simple mistakes $\Rightarrow$ specialized policy language.

The monitor will be attacked.


Policy script
Real-time notification
Record to disk
Policy Script Interpreter
Event control
Event stream
Event Engine
Tcpdump filter
Filtered packet stream
libpcap
Packet stream
Network

The `Bro` language:

Strongly typed $\Rightarrow$ catch errors at compile time.

Usual arithmetic (including `count`, `bool`).

Strings are counted rather than NUL-terminated:

```
    USER nice\0USER root
```

`time`, `interval` types (and constants, e.g., "`30 min`").

`port`: TCP or UDP port (e.g., "`http`", or "`80/tcp`").

`addr`: IP address (e.g., "128.3.254.23"; `lbl.gov` [*]).

The `Bro` language, con't:

`record` types for a collection of values.

`table` types for associative arrays:

(e.g., `[src_addr, dst_addr, serv] in RPC_okay`).

Easy to construct large tables:

```
const allowed_services: set[addr, port] = {
    [ftp_serv, [ftp, smtp, auth, 20/tcp]],
    [nntp.lbl.gov, nntp] };
```

Built-in functions: `fmt`, `edit`, `mask_addr`, `network_time`, `getenv`, `set_record_packets`, `parse_ftp_port`, etc.

`Bro` **statements:**

The usual control constructs (except no loops), plus `log`.

`event` generates an event — looks just like a function call.

Events are queued FIFO, drained after each new packet.

Event handlers look just like function definitions w/ `function` replaced by `event`.

Attacks on the monitor:

Overload: make it drop packets, sneak in.

Defense: leave doubt as to monitor's capabilities; shed load.

Crash: make it fault, or run out of resources.

Defense: watchdog timer & backup tracing; resource management hooks.

Subterfuge: fool the monitor.

E.g., send innocuous data with insufficient TTL to reach destination, retransmit attack data with sufficient TTL.

Subterfuge attacks, con't:

Defense?

In part, exercise great care when developing IDS, attending to all possible exceptional events. Development of Bro attempted to do this.

But perhaps hopeless, due to basic ambiguities.

See the excellent paper “Vulnerabilities in Network Intrusion Detection Software,” Ptacek/Newsham:

http://www.securenetworks.com/papers/IDS.PS

Application analysis:

For all TCP connections:

start time, duration, service, addresses, sizes

port, address scanning

Finger: `finger_request` event w/ request string.

Portmapper: *set/unset/getport/dump/callit*

XDR on top of UDP & TCP.

Attempts and request/replies.

Application analysis, con't:

FTP: `ftp_request`, `ftp_reply`.

Script parses `PORT`, `PASV` to identify subsequent conn.

Telnet:

- `telnet_logged_in`, `telnet_failure`
- `authentication_accepted`
- `authentication_rejected`
- `activating_encryption`, `telnet_confused`

Heuristic.

HTTP: `http_request`, `http_stats` (fledgling).

Status:

25,000 lines C++. 1,900 lines of policy script.

Digital Unix, FreeBSD, IRIX, Linux, SunOS, Solaris.

Freely available with sample policy scripts.

(but no documentation :-( ).

Currently: 400 MHz Pentium II's w/ plenty of disk & memory.

25 Mbps-busy-hour-utilization FDDI ring no problem.

But: filter discards a whole lot.

Can sustain $>$ 1,250 filtered packets/sec (peak 5,000/sec).

Status, con't:

In operation since April, 1996.

800,000,000 packets per day.

60 MB connections logs per day.

30 real-time notifications per day (large variance).

Break-ins detected: not uncommon.

Scans detected: every day.

6,000 email messages.

100+ CERT/CIAC incident reports.

Crud seen on a DMZ:

TCP connections reused prematurely.

Storms of 10,000+ FIN or RST packets.

Private addresses leaking out.

SYN packets with URG set.

Legitimate tiny fragments.

Fragments with DF set.

Overlapping fragments.

TCPs that retransmit different (valid) data than the first time.

TCPs that ack data that was never sent.

Additional monitoring:

Five monitor machines on DMZ:

- Bro except for HTTP
- Bro HTTP (planned)
- Spot tracing
- Telnet analysis for full FDDI ring (20,000+/day)
- Record everything.

Record everything: stripped disks $\Rightarrow$ single 69 GB partition.

Can copy full FDDI stream to disk w/o drops.

For now, records all LBNL interactive traffic.

Soon: LAN monitoring.

Near-term future directions:

More application modules.

Regular expression matching.

Automatic demuxing of “hot” streams.

Generic spot-tracing tool.

Reactive firewall.

Longer-term future directions:

More application modules.

Distributing across multiple CPUs/hosts.

Compiling and optimizing scripts.

BPF speed hacks.

Wish list has 100+ items.

Further info:

Minimalist Bro home page: http://www-nrg.ee.lbl.gov/bro-info.html

Includes link to Bro USENIX paper.

Current Bro 0.4 alpha release:

ftp://ftp.ee.lbl.gov/.vp-bro-0.4-alpha.tar.gz

BSD copyright (but please don't redistribute alpha releases)

Bro mailing list: bro@lbl.gov

Send mail to bro-request@lbl.gov, subject “subscribe”

Acknowledgments:

Many thanks to DEC-WRL (Jeff Mogul in particular) for ERP grant of monitoring hardware.

And to Craig Leres, Mark Rosenberg, Van Jacobson, Stu Loken and Dave Stevens.