Bro IDS

A Bro Walk-Through

Robin Sommer

International Computer Science Institute &
Lawrence Berkeley National Laboratory

robin@icsi.berkeley.edu
http://www.icir.org

RWTH Aachen - Dezember 2007

Doing the Walk-Through ...

- Going from simple to more complex setups
  - Start simple and then load & customize more scripts over time
  - Examine the log files which Bro generates
  - Take a look at some scripts to get an idea how to look for more information
- Tries to mimic the process of using Bro
  - Due to lack of documentation, one often has to examine the scripts
  - That's how *we* do it as well; nobody knows everything about all scripts
- Discussion requires some understanding of scripting
  - Try to explain what's needed, assuming some familiarity with similar languages
  - You're going to try this yourself in the lab

Installation

- Download current development version 1.3.2
  - `http://www.bro-ids.org/download.html`
  - See Wiki for documentation: `http://www.bro-ids.org/wiki`
- Compile and install
  - `./configure --prefix=<path> && make install`
- Files and directories
  - Bro executable: `$PREFIX/bin/bro`
  - Policy files (scripts): `$PREFIX/policy`

    Important policy files for reference: `bro.init`, `events.bif.bro`
- Environment
  - `export PATH=$PATH:$PREFIX/bin`
  - `export BROPATH=$PREFIX/policy:$PREFIX/policy/sigs`
  - If no DNS: `export BRO_DNS_FAKE=1`

Overview

1. Connection summaries
2. Notices and alarms
3. Weird activity
4. Protocol analyzers
5. Packet filter
6. Dynamic protocol detection
7. Protocol-independent analyzers
8. More customization
    - Filter Notices and alarms
    - Log rotation
    - Tuning time-outs

Connection Summaries

- One-line summaries for all TCP connections
- Most basic, yet also one of the most useful analyzers

```
> bro -r trace tcp    (Output in conn.log)
```

Demo

```
Time               Duration  Source          Destination
1144876596.658302  1.206521  192.150.186.169  62.26.220.2 \
  http  53052    80       tcp    874       1841      SF    X
  Serv  SrcPort  DstPort  Proto  SrcBytes  DstBytes  State  Dir
```

- Works also for UDP (`udp.bro`) and ICMP (`icmp.bro`)

Connection Summaries (2)

- Connection state

`SF`	`Normal establishment & termination`
`REJ`	`Connection attempt rejected`
`SO`	`Connection attempt seen, no reply.`
`RSTO`	`Established, originator aborted`
`...`	`...`

- Connection direction
  - Set `local_nets` (see `site.bro`) to a list local networks

`L`	`Locally initiated`
`X`	`Not locally initiated`

Demo

Notices & Alarms

- A *Notice* reports potentially interesting behavior.
  - A Notice is *not* an alarm but can turn into one
- Site-policy determines what to do with a Notice
  - Escalate into an alarm (default for most; we'll see later how to customize this)
  - Just log
  - Ignore
- A Notice carries meta-information as context
  - Notice type
  - Source information
  - Per-type attributes
- Scripts to record Notices and Alarms to disk

```
> bro -r trace notice alarm ...
```

Demo

Weird Activity ...

- Network traffic contains tons of "crud"
  - Activity which does not conform to standards but is *not* an attack
- NIDSs often cannot separate crud from actual attacks
- Bro's approach
  - Performs thorough consistency-checks
  - Reports non-conforming activity as "weird" (Notice of type `WeirdActivity`)
- Weirds can be pre-filtered with `weird.bro`
  - E.g., no Notice raised but logged into `weird.log`

Demo

Protocol Analyzers

- Perform protocol-specific analysis
  - Log activity
  - Check for protocol-specific attacks
- Bro ships with analyzers for many protocols
  - Including FTP, HTTP, POP3, IRC, SSL, DNS, SSH, NTP, Portmapper, SMB, etc.
- Example: FTP analyzer

```
> bro -r trace ftp.bro
> cat ftp.log
```

Demo

Protocol Analyzers (2)

- Almost all analyzers can be tuned to site-specifics
- Examine analyzer script to learn about options
  - `Export` section contains options which can be changed
  - Use `redef` to modify their values

Demo

- Example: Adapt sensitive file names in `ftp.bro`

- More protocol analyzers:
  - SMTP analyzer (`smtp.bro, imap.bro`)
  - HTTP analyzer (`http-request.bro`, `http-reply.bro`, `http-header.bro`, `http-body.bro`)

Demo

Behind the Scenes: Packet Filter

- Bro analyzes only packets required by scripts' analysis
  - Examines which scripts are loaded
  - Builds BPF packet filter dynamically (e.g., port 80 packets for HTTP)
- To see the packet filter, load `print-filter.bro`

```
> bro tcp ftp smtp print-filter
((port smtp) or (port ftp)) or (tcp[13] & 7 != 0)
```

- Packet filter can be changed by
  - Adding to `capture_filters` to *include* additional traffic (i.e. "or")
  - Adding to `restrict_filters` to *exclude* traffic (i.e., "and")
  - Fully overriding the filter via `-f` command line option
- *Likely the number-one pitfall is to forget that, by default, Bro completely skips parts of the traffic.*

Dynamic Protocol Detection (1)

- How does Bro know the analyzer for a connection?
- Default mechanism: Examine the ports

From `http.bro`:

```
global http_ports = {
   80/tcp, 81/tcp, 631/tcp, 1080/tcp, 3138/tcp,
   8000/tcp, 8080/tcp, 8888/tcp };

redef dpd_config += {
   [ANALYZER_HTTP] = [$ports = http_ports] };
```

- That's how any other NIDS does it as well

Dynamic Protocol Detection (2)

- Problem: Well-known ports are pretty unreliable
- Bro can analyze protocols independent of ports
  - "Dynamic Protocol Detection (DPD)"
  - Currently supports HTTP, IRC, SMTP, SSH, FTP, and POP3
  - Find potential uses with signatures and then *validates* by parsing
  - Signatures in `policy/sigs/dpd.sig`
- Activated by loading `dpd.bro`

```
> bro -r trace -f "tcp" http-request http-reply dpd
```

  - Important to adapt the packet filter!
- Example: Analyze HTTP session on port 22

Demo

DPD: Advanced Applications

- Turning off analyzers if it's not "their" protocol `dyn-disable.bro`

  Demo

  - Fundamental question: when to decide it's not "theirs"?
  - Analyzers report `ProtocolViolation` if they can't parse basic structure
  - Policy script can then decide whether to indeed disable analyzer
  - `dyn-disable.bro` takes simplest approach: turn off for every violation
  - Not on per default due to potential evasion opportunities

- Reporting protocols found on non-standard ports `detect-protocols.bro`

  Demo

  - Reports `ProtocolFound` and `ServerFound` notices
  - Further identify applications on top of HTTP (e.g., Gnutella, SOAP, Squid) `@load detect-protocols-http`
  - Easy to extend by adding more patterns

Protocol-independent Analyzers

- Bro also has several protocol-independent analyzers
- Example: Scan detector (`scan.bro`)
  - Reports `PortScan` and `AddressScan` Notices (and more)
- Example: Flood detector (`synflood.bro`)
  - Detects flooded hosts and excludes them temporarily from analysis

Demo

Customization: Filtering Notices

- Local policy determines which Notices are relevant
- Simple filtering: Assign an action per Notice type

```
@load notice-action-filters
redef notice_action_filters += {
   [RemoteWorm] = file_notice
   };
```

`file_notice`	Write only to notice.log
`ignore_notice`	Ignore completely
`notice_alarm_per_orig`	Alarm once per source
`tally_notice`	Count occurrences
`...`	....

notice-action-filters.bro

Customization: Filtering Notices (2)

- More flexible filtering: *function* determines the action

```
redef notice_policy += {
  [$pred(a: notice_info) =
   {
      # Do not report this notice for remote hosts.
      return a$note == ProtocolDetector::ServerFound
             && ! is_local_addr(a$src);
   },
   $result = NOTICE_FILE,
  ]
};
```

Thanks for your attention.

Robin Sommer

International Computer Science Institute &
Lawrence Berkeley National Laboratory

`robin@icsi.berkeley.edu`
`http://www.icir.org`