Bro User Manual

version 0.9, 12-1-2004, **DRAFT**

Vern Paxson, Jim Rothfuss, Brian Tierney
Contact: vern@icir.org
http://www.bro-ids.org/

This the User Manual for Bro version 0.9.

For further information about this notice, contact:

Vern Paxson email: vern@icir.org

Table of Contents

Bro User Manual

1 Overview of Bro

1.1 What is Bro?

Bro is a Unix-based Network Intrusion Detection System (IDS). Bro monitors network traffic and detects intrusion attempts based on the traffic characteristics and content. Bro detects intrusions by passing network traffic through rules describing events that are deemed troublesome. These rules might describe activities (e.g., certain hosts connecting to certain services), what activities are worth alarming (e.g., attempts to a given number of different hosts constitutes a "scan"), or signatures describing known attacks or access to known vulnerabilities. If Bro detects something of interest, it can be instructed to either issue a log entry or initiate the execution of an operating system command (such as sending email, or creating a router entry to block an address).

Bro targets high-speed (Gbit/second), high-volume intrusion detection. By judiciously leveraging packet filtering techniques, Bro is able to achieve the performance necessary to do so while running on commercially available PC hardware, and thus can serve as a cost effective means of monitoring a site's Internet connection.

1.2 Bro features and benefits

- **Network Based**

  Bro is a network-based IDS. It collects, filters, and analyzes traffic that passes through a specific network location. A single Bro monitor, strategically placed at a key network junction, can be used to monitor all incoming and outgoing traffic for the entire site. Bro does not use or require installation of client software on each individual, networked computer.

- **Custom Scripting Language**

  Bro policy scripts are programs written in the Bro language. They contain the "rules" that describe what sorts of activities are deemed troublesome. They analyze the network activity and initiate actions based on the analysis. Although the Bro language takes some time and effort to learn, once mastered, the Bro user can write or modify Bro policies to detect and notify or alarm on virtually any type of network activity.

- **Pre-written Policy Scripts**

  Bro comes with a rich set of policy scripts designed to detect the most common Internet attacks while limiting the number of false positives, i.e., alarms that confuse uninteresting activity with the important attack activity. These supplied policy scripts will run "out of the box" and do not require knowledge of the Bro language or policy script mechanics.

- **Powerful Signature Matching Facility**

  Bro policies incorporate a signature matching facility that looks for specific traffic content. For Bro, these signatures are expressed as regular expressions, rather than fixed strings. Bro adds a great deal of power to its signature-matching capability because of its rich language. This allows Bro to not only examine the network content, but to understand the context of the signature, greatly reducing the number of false positives. Bro

comes with a set of high-value signatures, selected for their high detection and low false positive characteristics, as well as policy scripts that perform more detailed analysis.

- **Network Traffic Analysis**

  Bro not only looks for signatures, but also analyzes network protocols, connections, transactions, data volumes, and many other network characteristics. It has powerful facilities for storing information about past activity and incorporating it into analyses of new activity.

- **Detection Followed by Action**

  Bro policy scripts can generate output files recording the activity seen on the network (including normal, non-attack activity). They can also send alarms to event logs, including the operating system *syslog* facility. In addition, scripts can execute programs, which can, in turn, send e-mail messages, page the on-call staff, automatically terminate existing connections, or, with appropriate additional software, insert access control blocks into a router's access control list. With Bro's ability to execute programs at the operating system level, the actions that Bro can initiate are only limited by the computer and network capabilities that support Bro.

- Snort **Compatibility Support**

  The Bro distribution includes a tool, snort2bro, which converts Snort signatures into Bro signatures. Along with translating the format of the signatures, snort2bro also incorporates a large number of enhancements to the standard set of Snort signatures to take advantage of Bro's additional contextual power and reduce false positives.

1.3 Getting more Information

- **Reference manual**

  An extensive reference manual is provided detailing the Bro Policy Language

- **FAQ**

  Several Frequently Asked Questions are outlined in the Bro FAQ. If you have a question not already covered in the FAQ, send it to us and we'll add it.

- **E-mail list**

  Send questions on any Bro subject to bro@bro-ids.org The list is frequented by all of the Bro developers.

  You can subscribe by going to the website:
  `http://mailman.icsi.berkeley.edu/mailman/listinfo/bro`,
  or by placing the following command in either the subject or the body of a message addressed to bro-request@icsi.berkeley.edu.

  ```
  subscribe [password] [digest-option] [address=<address>]
  ```

  A password must be given to unsubscribe or change your options. Once subscribed to the list, you'll be reminded of your password periodically.

The "digest-option" may be either: "nodigest" or "digest" (no quotes!). If you wish to subscribe an address other than the address you use to send this request from, you may specify "address=<email address>" (no brackets around the email address, no quotes!)

- **Website**

  The official Bro website is located at: http://www.bro-ids.org. It contains all of the above documentation and more.

2 Requirements

2.1 Network Tap

Bro requires a network tap to give it access to live network traffic. The tap needs to be full-speed for the link being monitored and must provide copies of both directions of the link, or you need to two taps, one in each direction.

Normally the network tap for Bro should be placed behind an external firewall and on the DMZ (the portion of the network under the control of the organization but outside of the internal firewall), as shown in the figure below. Some organizations might prefer to install the network tap outside the firewall in order to detect all scans or attacks. Placing Bro outside the firewall will allow the organization to better understand attacks, but will produce a more notifications and alarms. Another option is to place Bro inside the internal firewall, allowing it to detect internal hosts with viruses or worms. In addition to the connection to the network tap, a separate network connection is recommended for management of Bro and access to log files.

For more information on taps and tap placement see the Netoptics White paper titled *Deploying Network Taps with Intrusion Detection Systems* (http://www.netoptics.com/products/pdf/Taps-and-IDSs.pdf).

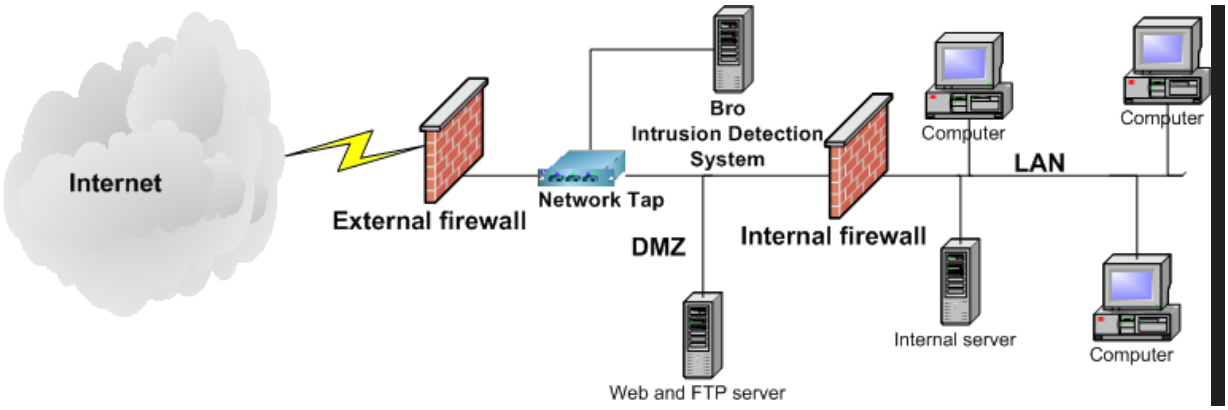

Figure 2.1: Typical location for network tap and Bro system

2.2 Hardware and Software Requirements

Bro requires no custom hardware, and runs on low-cost commodity PC-style systems. However, the Bro monitoring host must examine every packet into and out of your site, so depending on your site's network traffic, you may need a fairly high-end machine. If you are trying to monitor a link with a large number of connections, we recommend using a second system for report generation, and run only Bro on the packet-capture host.

Item	Requirements

Processor

Note: these are **rough** estimates. Much depends on the number of connections/second, the types of traffic on your network (e.g., HTTP, FTP, email, etc.), and you can trade off depth of analysis (especially, which protocols are analyzed) for processing load. (See the Performance chapter of the Bro User Guide for more information.)
1 GHz CPU for 100 Mbps monitoring with average packet rate <= 5,000 packets/second
2 GHz CPU for 1 Gbps monitoring with <= 10,000 packets/second
3 GHz CPU for 1 Gbps monitoring with <= 20,000 packets/second
4 GHz CPU for 1 Gbps monitoring with <= 50,000 packets/second

Operating System

Recommended: FreeBSD (`http://www.freebsd.org/`). Bro works with many Unix systems, including Linux and Solaris, but has been primarily tuned for FreeBSD. We currently recommend using FreeBSD version 4.10 for Bro. If your site has a large number of packets or connections per second you should look at the section on Section 9.1 [Hardware and OS Tuning], page 46. FreeBSD 5.x should work, but is not quite as fast as 4.10. For sites with very high traffic loads and capturing traffic on two interfaces, contact us for a FreeBSD 4.x kernel patch to do *BPF bonding*, which allows merging the two directions of a network link into a single interface as seen by Bro. While Bro can instead merge the two interfaces at user-level, this costs some performance.

Memory

512 MB suffices for small networks (say 200 hosts connected via a 100 Mbps link). For larger networks, 1 GB RAM will be required, with 2-3 GB is recommended.

Hard disk

10 GB minimum, 50 GB or more for log files recommended.

User privileges

superuser to install Bro, with Bro then running as user *bro*.

Network Interfaces

3 interfaces are recommended: 2 for packet capture (1 for each direction), and 1 for host management. Capture interfaces should be identical. For some network taps, both directions of the link are captured using the same interface, and the separate host management interface, while prudent, is not required.

Other Software

- Perl version 5.6 or higher (`http://www.perl.org`) (for report generation)
- libpcap version 0.7.2 or higher (`http://www.tcpdump.org`)
Note: Some version of FreeBSD come with older versions of libpcap. Bro recommends newer versions of these tools for performance reasons.

3 Installation and Configuration

3.1 Download

Download Bro from: http://www.bro-ids.org/

You can unpack the distribution anywhere except into the directory you plan to install into. To untar the file, type:

```
tar xzf bro-pub-0.9-current.tar.gz
```

3.2 Install

You'll need to collect the following information before beginning the installation.

- localnets: a list of local subnets for your network. Bro needs to know which networks are "internal" and which are "external".
- interface names: the names of the capture interfaces in your host (e.g. sk0 or en1). Use `ifconfig -a` to get the list of all network interfaces on your Bro host.

If you want to use Bro's periodic email report feature, you'll also need:

- email list: a list of email addresses to send the reports to.
- PGP keys: if you want to encrypt all email reports, the location of the GPG keyring of all recipients.

Bro is easy to install. Log in as `root`, and type:

```
./configure
```

By default Bro is installed in `/usr/local/bro`. This location is referred to in the rest of the manual as `$BROHOME`. To install Bro in a location other than '`/usr/local/bro`', use:

```
./configure --prefix=/path/to/bro
```

By default Bro uses the version of libpcap that is installed on the system. If your system version older than version 0.7.2, you can run configure Bro with –enable-shippedpcap to use the version of libpcap that comes packaged with Bro. For example:

```
./configure --enable-shippedpcap
```

Then type:

```
make
make install
```

or

```
make install-brolite
```

Use *make install* to install all the Bro binaries and policy script files. Use *make install-brolite* to also run the configuration script (described in the next section) and install all the configuration files and `cron` jobs. *make install* can be run as any user, but *make install-brolite* requires you to be root.

To update an existing Bro installation with new binaries and standard policy files, instead of `"make install"` do a `"make update"`. This will preserve all your local customizations.

Then add `$BROHOME/bin` and `$BROHOME/scripts` to your $PATH to use Bro's utilities and scripts.

Also note that this documentation is installed in `$BROHOME/docs` as both HTML and PDF versions.

3.3 Bro Configuration

The *Bro-Lite* configuration script can be used to automatically configure (or reconfigure) Bro for you. It checks your system's BPF settings, creates a "bro" user account, installs a script to start Bro at boot time, installs the report generation package, and installs a number of `cron` jobs to checkpoint Bro every night, run periodic reports, and manage log files.

To run this configuration script type:

```
bro_config
```

This script creates the file '`$BROHOME/etc/bro.cfg`'. `bro_config` will ask a number of simple questions. Note that the full functionality of this script is only supported under FreeBSD. Some additional configuration may need to be done by hand under Linux.

Sample output of `bro_config`, along with explanation, is shown below:

```
Running Bro Configuration Utility
Checking interfaces ....  Done.
Reading /usr/local/bro/etc/bro.cfg.example for defaults.
```

> The `bro_config` script looks first at ./bro.cfg, then /usr/local/bro/etc, for default values to use below.

```
Bro Log archive location [/usr/local/bro/archive]
```

> This is the directory where log file archives are kept. If you expect the log files to be very large, it is recommended to put these in a separate disk partition.

```
User id to install and run Bro under [bro]
```

> `bro_config` will create a new user account with this username if the user does not exist.

```
Interface names to listen on. [en1,en2]
```

> `bro_config` looks for all network interfaces and does a short test to determine which interfaces see the most traffic, and selects these interfaces as the default.

```
Site name for reports (i.e. LBNL, FOO.COM, BAZ.ORG) []
Starting Report Time [0600]
Report interval (in hours) [24]
Email addresses for reports [bro@localhost]
```

> Daily reports will be created. Enter the site name you want to appear at the top and in the subject of all email reports. The "start time" and "interval" define the window of network activity that the daily report will cover, starting at "Starting Report Time" and lasting through "Report interval". The start time should be entered using 24hr clock notation. For example: 12:30AM = 0030, 2PM = 1400

```
Do you want to encrypt the email reports (Y/N) [N]
Y
```

If you want the email reports encrypted, you will need to set up GPG (`http://www.gnupg.org`) and create a GPG keyring containing the public keys of all email recipients. Instructions for this are in Section 3.5 [Encrypted Reports], page 10. *Note: PGP keys are compatible with GPG, but the Bro supplied scripts require GPG, not PGP.*

```
Running script to determine your local subnets ...
Your Local subnets [198.129.224.1/32]
```

Bro needs to know a list of your local subnets. `bro_config` runs a tool that attempts to discover this automatically. You should always verify the results of this tool. The format is a list of subnet/significant bits of address. For example: 131.243.0.0/16, 198.128.0.0/18, 198.129.224.1/32 This information will be stored in the file `$BROHOME/site/local.site.bro`

```
Saving settings to file: /usr/local/bro/etc/bro.cfg
Bro configuration finished.
To change these values, you can rerun bro_config at any time.
```

Indicates that the script finished successfully.

For site monitoring very high traffic rates on Gigabit Ethernet, there is some additional system tuning that should be done. See the Chapter 9 [Performance Tuning], page 46 section for more details.

To reconfigure Bro, run:

```
BRHOME/scripts/bro_config
```

This will update your '`/usr/local/bro/etc/bro.cfg`' file. You can also edit this file using your favorite editor if you prefer.

For other site customizations, you can edit the file $BROHOME/site/brohost.bro. For example, to tell bro to not look at traffic for host 198.162.44.66, add:

```
redef restrict_filters += {
  ["ignore host 198.162.44.66 "] =
    "not host 198.162.44.66"
};
```

More details are available in the section on Chapter 7 [Customizing Bro], page 35.

3.4 OS Configuration

This section contains information on critical OS tuning items. More detailed tuning information can be found in the section on Chapter 9 [Performance Tuning], page 46.

FreeBSD Configuration

The standard FreeBSD kernel imposes a per-process limit of 512 MB of memory. This is not enough for most Bro installations.

To check your current limit type:

```
limits -H
```

Unfortunately the only way to increase this limit in FreeBSD 4.x is to reconfigure and rebuild the kernel. In FreeBSD 5.x it is much easier. Just increase `kern.maxdsiz` in '`/boot/defaults/loader.conf`' and reboot. For example:

```
kern.maxdsiz="2G"
```

and look at the `datasize` setting, which should be the same as your amount of RAM. If this is not true, see section Section 9.1 [Hardware and OS Tuning], page 46 for information on fixing this.

For FreeBSD 5.3+, BPF devices are no longer created using MAKEDEV, but rather are created on demand. This is configured automatically by running '`make install-brolite`', or you can figure it by hand by adding the following to `/etc/rc.local`

```
devfs ruleset 15
devfs rule add 15 path 'bpf*' mode 660 user bro
```

Linux Configuration

You may want increase these for a high traffic environment.

not done: need to get recommended values for these:

```
/proc/sys/net/core/rmem_default (IP-Stack socket receive queue)
/proc/sys/net/core/rmem_max     (similar to rmem_default)
/proc/sys/net/core/netdev_max_backlog (queue between driver and socket)
```

3.5 Encrypted Reports

Bro can use GPG (http://www.gnupg.org/) to encrypt the reports that it sends. To have Bro encrypt your reports you must have said "yes" to the bro_config question to encrypt your reports. Then each email recipient much generate a public/private key pair, and their public key must be installed on the Bro machine in the home directory of the user running the Bro process.

To create a key-pair:

```
gpg --gen-key
```

To export the public key:

```
gpg --armor --output mykey.gpg --export myemail@address.com
```

Then login to the machine running Bro and import the list of public keys:

```
gpg --import mykey.gpg
```

Then you must to make the list of keys "trusted" so that they can be used to encrypt the email reports. To do this, you must edit the key to add "ultimate" trust to the key.

```
gpg --edit-key myemail@address.com

pub  1024D/4A872E40  created: 2001-02-05 expires: never      trust: -/f
sub  3072g/B72DD7FE  created: 2001-02-05 expires: never
(1). Some R. User <myemaill@address.com>

Command> trust
pub  1024D/4A872E40  created: 2001-02-05 expires: never      trust: -/f
sub  3072g/B72DD7FE  created: 2001-02-05 expires: never
(1). Some R. User <myemail@address.com>
```

```
Please decide how far you trust this user to correctly
verify other users' keys (by looking at passports,
checking fingerprints from different sources...)?

 1 = Don't know
 2 = I do NOT trust
 3 = I trust marginally
 4 = I trust fully
 5 = I trust ultimately
 m = back to the main menu

Your decision? 5
Do you really want to set this key to ultimate trust? yes

pub  1024D/4A872E40  created: 2001-02-05 expires: never      trust: u/u
sub  3072g/B72DD7FE  created: 2001-02-05 expires: never
(1). Some R. User <myemail@address.com>

Command> quit
```

For more information on GPG see http://www.gnupg.org/

3.6 Generating Reports on a Separate Host

Warning: this section assumes a reasonably high level of Unix system administration skills!

If your site has lots of traffic, lots of connections, or if Bro is using on average more than around 40% of your CPU, you'll want to use a second host for generating reports.

To do this, on the Bro host, run bro_config, and say "N" to all report generation questions. Then install Bro on the second host using the following:

```
./configure
make
make install-reports
```

Then follow the instructions in Section 3.3 [Bro Configuration], page 8 for setting up report generation.

You'll also need to set up a method to copy files from the Bro host to the report generation host. One way to do this is using `rsync`, and the Bro script `push_logs.sh` does this for you. For example, you can set up a cron job like this on the Bro host:

```
1 1 * * * (push_logs.sh /usr/local/bro/etc/bro.cfg host:/home/bro) >> /tmp/bro-push.lo
```

To make sure your `rsync` command has time to transfer all log files before your report generation script is run, the `push_logs.sh` script is designed to be used with the scripts `frontend-site-report.sh` and `frontend-mail-report.sh` on the frontend host. These `frontend` scripts wait for a file with a particular name to exist before running. It is also important to use the `nice` command to help ensure the network copy does not unduly divert processing away from Bro.

You may want to `rsync` the log files over a secure ssh connection. To do this, you need to first generate a ssh key pair on the Bro capture host with no passphrase:

```
ssh-keygen -t rsa -C "batch key" -f ./batch.key
```

Put this in user `bro`'s .ssh/config file, also on the Bro capture host

```
Host recvhost brohost.foo.com
IdentityFile ~/.ssh/batch.key
```

On the frontend host where the log files will be processed, add batch.pub to the authorized_keys file

```
cat batch.key.pub >> authorized_keys
```

Then create a cron entry on the Bro capture host

```
1 1 * * * nice -n 20 rsync -e 'ssh' -azv \
  /usr/local/bro/logs host:/home/bro
```

4 Running Bro

4.1 Starting Bro Daemon

Bro is automatically started at boot time via the `bro.rc` script (located in '`$BROHOME/etc`' and '`/usr/local/etc/rc.d`' on FreeBSD, or '`/etc/init.d`' on Linux).

To run this script by hand, type:

```
bro.rc start
```

or

```
bro.rc checkpoint
```

or

```
bro.rc stop
```

Use `checkpoint` to restart a running Bro, loading a new policy file.

Note that under Linux, Bro must be run as the 'root' user. Linux must have root privilages to capture packets.

4.2 Running Bro from the command line

If you use `bash` for your shell, you do something like this to start Bro by hand:

```
cd /usr/local/bro
. etc/bro.cfg
./bro -i eth1 -i eth2 myhost.mysite.org.bro
```

The '. etc/bro.cfg' should set your $BROHOME and $BROPATH correctly to find all of the needed the files.

Files are loaded is the following order: Bro is invoked with a start file (in the above myhost.mysite.org.bro). In that file (which is in $BROHOME/site) there should be a couple of lines like this at the top:

```
---------------- myhost.mysite.org.bro ----------------------------
@prefixes = local
@load site      # file generated by the network script for dynamic config
                   # of the local network subnets.

# Make any changes to policy starting here
....
-------------- end  ---------------------------------------
```

The '@load site' will load the local.site.bro file from $BROHOME/site. If you are making changes, you should make them in 'myhost.mysite.bro' file.

Bro can also be run on `tcpdump -w` files instead of on live traffic. To do this, you must set a `BROPATH` enviroment variable to point at your set of policy scripts. For example (in csh):

```
setenv BROHOME /usr/local/bro
setenv BROPATH $BROHOME/site:$BROHOME/policy
bro -r dumpfile brohost
```

More information on Bro run-time flags and environment variables is available in the Reference Manual.

4.3 Bro Cron Scripts

Installing *brolite* automatically creates the following `cron` jobs, which are run on at the specified intervals.

- `site-report.pl`: generates a text report of all alarms and notifications
- `mail_reports.sh`:emails the reports generated by `site-report.pl` to the list of addresses specified in the file `$BROHOME/etc/bro.cfg`

These scripts can also all be run by hand at any time. Be sure your $BROHOME environment variable is set first.

As Bro log files can get large quickly, it is important to ensure that the Bro disk does not fill up. Bro includes some simple scripts to help manage disk space. Most sites will want to customize these for their own requirements, and integrate them into their backup system to make sure files are not removed before they are archived.

- `check_disk.sh`: send email if disk space is too low
- `bro_log_compress.sh`: remove/compress old log files

These scripts can be customized by editing their settings in `$BROHOME/etc/bro.cfg`. The settings are as follows:

- `check_disk.sh`:
  - `diskspace_pct`: when disk is >= this percent full, send email (default = 85%)
  - `diskspace_watcher`: list of email addresses to send mail to
- `bro_log_compress.sh`:
  - `Days2deletion`: remove files more than this many days old (default = 60)
  - `Days2compression`: compress files more than this many days old (default = 30)

5 Bro Output

This section explains the contents of the various output files and reports that Bro creates.

5.1 Rotating Log Files

There are a number of ways to control the rotation of Bro's log files. Here are some examples:

```
# if using one log, must open append (default = truncate )
@load log-append

@load rotate-logs

# rotate at midnight
redef log_rotate_base_time = "0:00";

# do not rotate on shutdown  (default = T)
redef RotateLogs::rotate_on_shutdown = F;

# rotate frequency
redef log_rotate_interval = 24 hr;
```

5.2 Alarm File

5.2.1 Alarm File Format

The alarm.log is a 'tagged' format that is fairly self descriptive. This is an example from the alarm.log file:

```
   t=1000057981.940712 no=AddressScan na=NOTICE_ALARM_ALWAYS sa=10.1.1.1 sp=2222/tcp da=10.
) tag=@42
```

Where the tags indicate the following:

- t: time
- no: notice
- na: notice action
- sa: source address
- sp: source port
- da: destination address
- dp: destination port
- msg: message (in this case a host has scanned 20 hosts)
- tag: identifier to match this to lines in notice.log and conn.log:

The alarm file format is designed to be easy to parse and interpret by programs, not humans.

5.2.2 Sample Alarm File Contents

NOTE: the examples in this section still use the old log format. This needs to be updated

Bro generates a number of types of alarms, such as suspicious connection attempts directed at your systems (address scanning), port scans, and attempts to gain access to user accounts. We describe some of these in more detail here.

Vulnerability scans directed against systems

The first category of suspicious connection attempts that Bro identifies and reports is vulnerability scans directed against systems. Instead of burdening you with every vulnerability scan (no matter how tiny) against systems that occurs, it reports only scans that occur at or above its threshold in terms of a specified size, such as the number of vulnerability scan attempts per second. The following entry indicates that IP address 66.243.211.244 has scanned 10000 of your systems on tcp port 445, the port used by newer Windows systems such as Windows 2000, XP and Server 2003 for share access:

```
Nov 16 06:30:49 AddressScan 66.243.211.244 has scanned 10000 hosts (445/tcp)
```

Important note: If the source of a scan is an IP address within your own network, the probability that this system is infected or has been taken over by an attacker is very high.

At the bottom portion of each Bro report summaries of scan activity also will appear for your convenience (see the bottom part of Figure 1). Scan summaries look like the following:

```
Nov 16  8 01:30:11 ScanSummary 194.201.88.100 scanned a total of 145 hosts
```

Port scanning

Port scanning means that a system has targeted one system, connecting to one port, then another, until it has scanned a whole range of ports. In the following example a system with an IP address of 218.204.91.85 has scanned 50 ports of a system named diblys.dhcp within an internal network:

```
Nov 16 06:30:50 PortScan 218.204.91.85 has scanned 50 ports of siblys.dhcp
```

An address dropped entry is likely to appear shortly after a port scan is reported. In the following entry, the system with an IP address of 218.204.91.85 was systematically probing low ports, that is, ports in the range from 0 to 1023, on sibyls.dhcp:

```
Nov 16 06:30:52 AddressDropped low port trolling 218.204.91.85 403/tcp
```

Connection attempts

Bro finds attacks against user accounts, such as password guessing attempts. When it does, it reports them as follows:

(Need example here)

Weird events

Bro labels unusual or exceptional events as *weird* Weird events include a wide range of events, including malformed connections, attacker's attempts to confuse or evade being detected, malfunctioning hardware, or misconfigured systems or network devices such as routers. Some weird events could be the result of an attack; others are just anomalies. The better you know your own systems and networks, the more likely you will be to correctly determine whether or not weird events compromise an attack. Weird events are divided into three categories:

1. Weird connections that are not formed in accordance with protocol conventions such as the way the tcp protocol should work

1. Weird flows in which data is being sent between two systems, but no specific connection between them can be identified, and

1. Weird network behavior that cannot be associated with any particular system.

The following entry shows a weird connection in which packets are being sent between systems in what appears to be an ongoing ftp session, but Bro has not identified any initial connection attempt (i.e., there is an *ack above a hole*):

```
Nov 16 06:30:58 WeirdActivity window/50406 > klecker.debian.org/ftp ack above a hole
```

The next entry (below) shows another weird traffic pattern in which eqvaadvip1.doubleclick.net has sent a flood of FIN packets, packets that tell the other system in a connection to terminate the connection, to bcig-8 within the network that Bro protects.

```
Nov 16 06:32:09 WeirdActivity bcig-8/2044 > eqvaadvip1.doubleclick.net/http: FIN_storm
```

Here's another one — in this case a system with an IP address of 222.64.93.208 has sent a flood of packets that have both the SYN and the ACK flags set, something that should normally happen only once in a TCP session. `nsx` is the destination system.

```
Nov 16 06:30:58 WeirdActivity 222.64.93.208/1115 > nsx/dns: repeated_SYN_with_ack
```

Packets sent over networks are often broken up (or *fragmented*) for various reasons. Fragmented packets are not necessarily a sign of an attack, but large fragments can indicate suspicious activity such as attempts to cause denial of service. In the following example Bro has identified a very large packet fragment sent by p508c7fc5.dip.t-dialin.net to an internal system with an IP address of 131.243.3.162:

```
Nov 16 06:30:50 WeirdActivity p508c7fc5.dip.t-dialin.net -> 131.243.3.162: excessively_large_fragment
```

Sometimes attackers attempt to evade being detected by sending malformed packets to the system they are attacking. The receiving system cannot process them, so it informs the sending (attacking system) accordingly, asking it to resend them. The sending system may now send packets that constitute an attack. Some intrusion detection systems (but not Bro) ignore what is resent because in theory it is unnecessary to reanalyze what has already been sent. Bro detects this kind of retransmission inconsistency (*rexmit inconsistency*) and reports it. The following example shows that there has been retransmission inconsistency in packets sent by a system at IP address 131.243.223.212 to origin2.microsoft.com:

```
Nov 16 06:30:59 RetransmissionInconsistency 131.243.223.212/2270 > origin2.microsoft.com/http
rexmit inconsistency (HTTP/1.1 200 OK^M^JDate: Tue, 16 Nov 2004
```

5.3 Connection Summary File

See the Connection Summary section in the reference manual for a description of the `conn.log` files.

5.4 Analyzer-specific Files

A number of analyzers such as HTTP and FTP generate their own log files. These files are fairly self explanatory.

coming soon: sample http.log, ftp.log, etc

5.5 Tracefiles

Bro can be configured to output captured packets that look to be part of suspicious sessions. These files are in `tcpdump` format.

5.6 Bro Summary Reports

NOTE: The Bro report facility is still under development. This section may be out of date.

Bro reports are generated using the `site-report.pl` script, which is typically run as a nightly `cron` job.

A daily report is created that covers three sets of information:

- Incident information
- Operational status of Bro
- General network traffic information

The reports will be mailed to the email addresses specified during Bro installation. These email addresses can be changed by re-running the `bro_config` script or by editing `$BROHOME/etc/bro.cfg` directly.

The report is divided into three parts, the summary, incidents, and scans. The summary includes a rollup of incident information, Bro operational statistics, and network information. The incidents section has details for each Bro alarm. The scans section gives details about scans that Bro detected.

5.6.1 Parts of a Report

Header

The header gives some basic information about the report.

> **Site name** is determined by the "Site name for reports" that was given during the installation and configuration process.
>
> **Start time and interval** of the report are also entered during the configuration process.

See Section 3.3 [Bro Configuration], page 8.

Summary

This section give a numeric summary of the events that have happened in the reporting period.

> **Incidents** shows the number of incidents that are recorded in the report period. An incident is any occurrence that is deemed worth investigating. An incident is formed by the triggering of one or more alarms.
>
> **Scanning Hosts** are the number of specific IP addresses that have been detected scanning either into or out from the site.
>
> > **A scan can be a:**
> >
> > - **port scan:** scanning several ports of a single host.
> > - **network scan:** scanning several hosts for open ports.
> > - **signature scan:** attacking multiple hosts with a specific vulnerability attack (signature).

- **targeted attack:** launching multiple signatures against a single host.
- **password scan:** attempts to guess passwords on telnet terminals.

A successful scan is when:

- the bytes sent by a single probe of a scan against a host or several hosts are more than three deviations away from the standard deviation of the rest of scan. In essence, where the bytes transferred on one connection is different than the rest of the scan other connections involved in the scan.
- a separate connection back to the attacker host is detected from the local network.
- the number of bytes sent back from the targeted victim host to the offender during a scan connection exceeds 20480.

Signature Summary shows the total number of alarms triggered by signatures during the report period and the number of those that are unique. These numbers do not include alarms triggered by embedded Bro rules. See Section 6.3 [Understand What Triggered the Alarm(s)], page 29.

Signature Distributions

This is a list of all signatures that were triggered during the report period.
NOTE: This section does not include alarms triggered by embedded Bro rules. See Section 6.3 [Understand What Triggered the Alarm(s)], page 29.

Count is the number of times the signature was seen.

Unique Sources is the number of unique ip addresses that used the specific signature as an attack.

Unique Dests is the number of unique ip addresses that were attacked by the particular signature.

Unique Pairs are the number of unique source/dest ip address pairs where the source used the signature to attack the destination.

Incidents

Legend: This is the legend for reading the *connections* portions of the each incident. It is shown once on each report at the top of the *Incidents* section.

Incident number: Each incident listed in the Bro report is assigned a unique, sequential, identification number prefixed with the organization identifier. This number is unique for all incidents, not just to the daily reports.

Remote and Local hosts: The Remote and Local hosts are identified by both ip address and hostname. The local hosts are those that are in local subnets as determined during Bro configuration. It is important to note that *remote*

host does not infer *attack host*. Attacks can come from local hosts (indicating an inside hacker or a compromised host).

Alarms: The network event(s) that Bro detects and identifies as possible attacks. There are two general types of alarms, those triggered by signatures and those triggered by Bro rules. See Section 6.3 [Understand What Triggered the Alarm(s)], page 29, for more information about the differences. All alarms will include the date/time of the attack, the direction of the attack, and the ports involved. A `SensitiveSignature` will include the signature code and payload to help evaluate what triggered the alarm. Embedded Bro rules will include the payload and a session number which can be used for further investigation in the logs. See Section 6.5 [Examine HTTP FTP or SMTP Sessions], page 32.

Connections: A list of the first 25 connections after the first alarm is triggered that are attempted between the attacking and victim host. This tabulation of connections can be used to see if connections were accepted by the victim host, the amount of bytes transferred in both directions, the timing between the connections, and the ports involved.

Scans

This is a summary of the ip addresses involved in successful scans, the type of scans, and the attacks used by the scanners.

Connection Log Summary

This section gives a overview of the most prominent connections that have occurred during the report period, as shown by way of five tables.

Site-wide connection statistics: The number of successful and unsuccessful connections and the ratio between the two.

Top 20 Sources: Hosts that have initiated the most connections.

Top 20 Destinations: Hosts that have accepted connections.

Top 20 Local E-mail Servers: The most active E-mail servers.

Top 20 Services: The services, as determined by port number, that have been involved in connections.

Byte Transfer Pairs

This section gives a summary of the ip address address pairs that have transferred the most bytes during the report period.

5.6.2 Annotated Example Report:

```
Site Report for ORG_NAME
from 2004/11/03 00:00:00 to 2004/11/04 00:00:00
generated on Sat Nov 13 12:02:48 2004
```

annotation: *ORG_NAME will normally be replaced with "Site name for reports" that was given during the installation and configuration process.*

```
======================================================================
Summary
```

```
=========================================================================
```

annotation: *Since this report is simple and only includes two incidents, the summary is rather uninteresting. A glance at this summary would reveal a rather "slow" day (for which you should be thankful).*

```
  Incidents               2
  Scanning Hosts
    Successful            8
    Unsuccessful          15
  Signature Summary
    Total signatures          2
    Unique signatures         2
    Unique sources            2
    Unique destinations       2
    Unique source/dest pairs  1
```

annotation: *Since the same to ip addresses were involved in both signature attacks, there is only one unique source/dest pair.*

```
=========================================================================
Signature Distributions
=========================================================================
                                        Unique      Unique    Unique
  Signature ID               Count      Sources     Dests      Pairs
  ------------------------  --------  ---------  ---------  --------
  bro-687-5                 1          1          1          1
  bro-144-3                 1          1          1          1
=========================================================================
Incident Details
=========================================================================
```

annotation: *The following legend appears once in every report at the top of the "Incidents" section*

```
     # legend for connection type #
                  ------------------------------
         C Connection Status
           # number corresponds to alarm triggered by the connection
           * successful connection, otherwise unsuccessful.
         I Initiatator of Connection
           > connection initiated by remote host
           < connection initiated by local host
-------------------------------------------------------------------------
Incident      ORG_NAME-000004524
-------------------------------
```

annotation: *The host domain name "org_name.org" will normally be replaced by the local domain name. The IP addresses in this example have been synthesized from an imaginary range outside of the octal range. (We realize these ip addresses cannot exist). In this example the ip ranges 124.333.0.0/24 and 132.257.0.0/24 are considered the local subnets.*

```
Remote Host: 84.136.338.21   p54877614.dip.hacker.net
 Local Host: 124.333.183.162 pooroljoe.dhcp.org_name.org
```

annotation: *This attacker was successful in using an SQL attack and then downloaded a "tool" using TFTP. Both of these were detected and created the following alarms.*

```
Alarm: SensitiveSignature
  1    bro-687-5: MS-SQL xp_cmdshell - program execution
       7/29 12:43:31                  84.136.338.21 -> 124.333.183.62
                                            566/tcp -> 1433/tcp
       signature code:
       signature bro-687-5 {
         ip-proto == tcp
         dst-port == 1433
         event "MS-SQL xp_cmdshell - program execution"
         tcp-state established,originator
         payload /.*[xX]\x00[pP]\x00_\x00[cC]\x00[mM]\x00[dD]\x00[sS]
         \x00[hH]\x00[eE]\x00[lL]\x00[lL]\x00/
       }
       payload: xp_cmdshell 'echo.> c:\\temp\\bcp.cmd'

Alarm: SensitiveSignature
  2    bro-1444-3: TFTP Get
       7/29 12:43:31                  84.136.338.21 -> 124.333.183.62
                                           2318/upd -> 69/udp
       signature code:
       signature bro-1444-3 {
           ip-proto == udp
           dst-port == 69
           event "TFTP Get"
           payload /\x00\x01/
           }
       payload: Runtime.exe
```

annotation: *Looking at the "C" column below, the alarms are signified by "1" and "2", both occuring at 12:43:31. Since the attacks take place within one second, this is probably an automated attack. The remote host continues to connect to the victim host, using a different port each time to avoid detection. The large transfers from the local host to the remote host, subsequent to the alarmed attacks, signifies that the attack is probably successful.*

```
Connections (only first 25 after first alarm are listed)
-----------
                 time     byte   remote        local   byte
 date   time   duration transfer  port  C   I   port transfer  protocol
----- -------- -------- -------- ------ ------ ----- -------- ----------
07/29 12:43:31        ?     566 b  4634 1   >  1433      467 b tcp/MSSQL
07/29 12:43:31        0         ?  2318 2  <     69       20 b udp/tftp
07/29 12:43:32    265.7       4 b  4638 *  <   2318      3.0kb udp
```

```
07/29 12:48:56        ?         ?  4640     >  2362          ? tcp
07/29 12:50:05        ?    11.4kb  4639 *  <   3333       8.6kb tcp
07/29 12:53:00        0         ?  4684 *   >  2362          ? tcp
07/29 12:53:07        ?         ?  4685 *   >  2362          ? tcp
07/29 12:53:59        ?         ?  4689 *   >  2362          ? tcp
07/29 12:54:14      6.1         0  4693 *  <   2380     94.2kb tcp
07/29 12:54:21       .5      50 b  4694     >  2381          0 tcp
07/29 12:54:23       .7         ?  4695    <   2382          0 tcp
07/29 12:54:25       .5      51 b  4696 *   >  2383          0 tcp
07/29 12:54:27       .5      61 b  4697 *   >  2384          0 tcp
07/29 12:54:28       .7      39 b  4698     >  2385          0 tcp
07/29 12:54:31       .5      41 b  4699 *   >  2386          0 tcp
07/29 12:54:33      1.2    4.9 kb  4700     >  2387          0 tcp
07/29 12:54:35     12.8  195.0 kb  4701 *  <   2388          0 tcp
07/29 12:54:53       .2         ?  4703    <   2390          0 tcp
07/29 12:54:54       .5      37 b  4704     >  2391          0 tcp
07/29 12:54:56      3.4      23 b  4705 *   >  2392          0 tcp
07/29 12:55:04     21.4  308.7 kb  4706     >  2393          0 tcp
07/29 12:55:27     50.7         ?  4707     >  2394          ? tcp
07/29 12:59:23        ?         ?  4775     >  1433          ? tcp
07/29 12:59:25        ?         ?  4774 *   >  3333          ? tcp
```

annotation: *The next Incident demonstrates alarms triggered by embedded rules, rather than signatures.*

```
------------------------------------------------------------------------
Incident      ORG_NAME-000004525
-------------------------------
Remote Host:  80.143.378.186     p508FB2BA.dip.t-dialin.net
 Local Host:  128.333.181.191      lemonade.lbl.gov
```

annotation: *Since these alarms are triggered in the HTTP protocol, the actual trigger rules are found in the file* '`bro/policy/http.bro`'.

```
Alarm: HTTP_SensitiveURI
       11/13 11:36:05               80.143.378.186 -> 128.333.181.191
                                          1560/tcp -> 80/tcp
       session: %4672
       payload: GET http://cn.edit.vip.cnb.yahoo.com/config/login?.redir
                _from=PROFILES

Alarm: HTTP_SensitiveURI
       11/13 11:53:54               80.143.378.186 -> 128.333.181.191
                                          2434/tcp -> 80/tcp
       session:%7386
       payload: GET http://l10.login.scd.yahoo.com/config/login?.redir_f
                rom=PROFILES?&
```

annotation: *In the connections shown below, all connections are from the remote host to the local host, with no successful connections back. Also the payload above is seeking yahoo.com. Hence the likelihood is that this is not an attack.*

```
Connections (only first 25 after alarm are listed)
-----------
                 time     byte   remote        local   byte
 date   time   duration transfer  port  C   I   port transfer  protocol
----- -------- -------- -------- ------ ------ ----- -------- ----------
11/13 11:36:05 1.109227      297   1560 *    >    80     1531       http
11/13 11:36:06        ?        ?   1560      >    80        ?       http
11/13 11:41:51 0.843209      301   3175 *    >    80     1533       http
11/13 11:41:52        ?        ?   3175      >    80        ?       http
11/13 11:47:37 2.562365      281   4701 *    >    80     1382       http
11/13 11:47:39        ?        ?   4701      >    80        ?       http
11/13 11:53:53 0.694131      293   2434 *    >    80     1529       http
11/13 11:53:54        ?        ?   2434      >    80        ?       http
11/13 11:59:23 0.685181      301   3975 *    >    80     1529       http
11/13 11:59:23        ?        ?   3975      >    80        ?       http
11/13 12:04:53 1.054925      289   1700 *    >    80     1527       http
11/13 12:04:54        ?        ?   1700      >    80        ?       http
11/13 12:11:56 2.579652      283   3442 *    >    80     1523       http
11/13 12:11:59        ?        ?   3442      >    80        ?       http
11/13 12:18:08 1.046188      289   1083 *    >    80     1531       http
11/13 13:14:42        ?        ?   3282      >    80        ?       http
11/13 13:16:46        ?        ?   4802      >    80        ?       http
11/13 13:19:04 1.731771        0   2764 *    >    80        0       http
11/13 13:19:07        ?        ?   2764      >    80        ?       http
11/13 13:20:42 0.994114      289   4142 *    >    80     1527       http
11/13 13:20:43        ?        ?   4142      >    80        ?       http
11/13 13:22:37 1.122448      292   1732 *    >    80     1523       http
11/13 13:22:38        ?        ?   1732      >    80        ?       http
11/13 13:24:40 1.042112      289   3179 *    >    80     1531       http
11/13 13:24:41        ?        ?   3179      >    80        ?       http


======================================================================
Scans (only first 100 shown)
======================================================================
```

annotation: *The scans show below are considered "successful". Four interesting scans shown below are the ones originating from the 124.333 and 132.257 domains, since they are local domains. These should be investigated. The attack against 132.257.85.96 might also be investigated further. With each report, a review of the attacks will give an understanding of what types of scans are becoming "popular".*

```
Scanning IP      Victim IP       Attack
132.257.70.234   multiple        bro-1344-5
132.257.52.64    multiple        bro-1367-5
63.251.3.51      multiple        bro-2570-6
124.333.181.191  multiple        bro-1599-7
```

```
210.313.36.53    132.257.85.96   >1000 port scan
211.300.24.151   132.257.85.96   >1000 port scan
124.333.95.0     62.214.34.30    >250 port scan
172.278.206.135  multiple       (3128/tcp)

================================================================================
Connection Log Summary
================================================================================
```

annotation: *The connection log summary gives a general idea of what hosts are most active. The analyst may want to become familiar with any new hosts that appear on the next three lists and services that appear or radically change position on the fourth list*

```
Site-wide connection statistics
    Successful:   4498748
    Unsuccessful: 35941140
    Ratio: 1:7.989

Top 20 Sources
                Host                      IP         Bytes   Conn. Count
  -------------------------------  ---------------  ------  -----------
                 ns1.org_name.org  124.333.34.186    3.7 G       683948
                 ns2.org_name.org  132.257.64.2      165 M       231245
             lemonade.org_name.org  124.333.181.191    88 M       217781
                  nsx.org_name.org  132.257.64.3      371 M       200935
               cinnamon.mining.com  207.5.380.138     4.5 M       103011
       node2.lbnl.nodes.planet.org  198.328.56.12     106 M        75725
       node1.lbnl.nodes.planet.org  198.328.56.11      85 M        73719
     microscope.dhcp.org_name.org  132.257.19.79      61 M        54024
                                   169.299.224.1     2.3 M        40348
                uhuru.org_name.org  132.257.10.97     423 M        39847
                                   132.257.77.246     13 M        29496
           googledev.org_name.org  124.333.41.57      13 M        24930
                                   64.46.248.43       60 M        19785
  ...16-141.sfo4.dsl.contactor.net  66.292.16.141     6.2 M        19048
                      rock.es.net  198.128.2.83      2.8 G        18459
             perry.Geo.college.EDU  124.32.349.11     1.7 M        17326
               google.org_name.org  124.333.41.70     8.5 M        15508
             egspd42212.search.com  65.264.38.212     3.1 M        15138
       hmb-330-042.MSE.college.EDU  124.32.349.20     222 M        14865
          1rodan.dhcp.org_name.org  132.257.19.170    7.7 M        11873

Top 20 Destinations
                Host                      IP         Bytes   Conn. Count
  -------------------------------  ---------------  ------  -----------
                  nsx.org_name.org  132.257.64.3       14 G      1571638
                 ns1.org_name.org  124.333.34.186    1.6 G       264976
```

ns2.org_name.org	132.257.64.2	80 M	218740
lemonade.org_name.org	124.333.181.191	2.6 G	176788
CS.university.EDU	128.312.136.10	10 M	81622
g.old-servers.net	192.42.293.30	11 G	71407
engram.CS.university.EDU	128.312.136.12	7.5 M	61309
aulvs.realthing.com	207.288.24.156	792 M	50493
ns1.college.EDU	124.32.349.9	995 M	39977
rohan.superc.gov	128.550.6.34	4.7 G	32883
sportsmed.starship.com	199.281.132.79	17 M	32152
ns2.yoho.com	66.263.169.170	2.1 G	24361
uhuru.org_name.org	132.257.10.97	58 M	19785
g3.NSDDD.COM	192.342.93.32	488 M	19734
w4.org_name.org	124.333.7.51	447 M	19334
E.TOP-SERVERS.NET	192.303.230.10	195 M	19066
mantis.org_name.org	124.333.7.39	395 M	18811
postala.org_name.org	124.333.41.61	8.0 M	17283
vista.org_name.org	132.257.48.146	488 M	15961
calmail.college.EDU	128.32.349.103	73 M	15154

Top 20 Local Email Senders

Hostname	IP	Conn. Count
mta1.org_name.org	124.333.41.24	3869
postala.org_name.org	124.333.41.61	2850
ci.org_name.org	132.257.192.220	868
postal2.org_name.org	132.257.248.26	376
ee.org_name.org	132.257.1.10	173
math.org_name.org	124.333.7.22	131
rod2.org_name.org	132.257.112.183	121
gigo.org_name.org	124.333.2.54	110
mh1.org_name.org	124.333.7.48	82
stm.org_name.org	132.257.16.51	81

Top 20 Services

Service	Conn. Count	% of Total	Bytes In	Bytes Out
dns	3378522	75.10	30 G	11 G
http	902573	20.06	18 G	11 G
other	92913	2.07	14 G	249 G
smtp	35942	0.80	458 M	196 M
https	33848	0.75	2.3 G	179 M
ssh	25515	0.57	977 M	1.0 G
netbios-ssn	11004	0.24	65 M	9.5 M
pop-3	5494	0.12	58 M	3.6 M
ftp-data	4495	0.10	37 G	34 G
ldap	3549	0.08	740 K	2.0 M

```
  ftp                    1061        0.02      1.3 M      873 K
  ident                   970        0.02     29602       9039
  printer                 834        0.02       837       9176
  time                    645        0.01      2416        166
  imap4                   636        0.01      28 M       47 M
  nntp                    308        0.01     355 M      1.5 M
  pm_getport              238        0.01     13328       6664
  telnet                  164        0.00     469 K       7850
  ntp                      26        0.00      1344       1392
  X11                       6        0.00     652 K      64280
================================================================================
Byte Transfer Pairs
================================================================================
```

annotation: *Once again, this summary gives a general idea of what hosts are most active. Radical changes to this list may indicate malicious activity.*

```
Hot Report - Top 20
                                        Local      Remote     Conn.
  Local Host       Remote Host          Bytes      Bytes      Count
---------------  ---------------  ---------  ---------  ---------
124.333.28.60    128.265.128.131      123 G     5327 K  3930
124.333.28.60    128.265.128.132      123 G     5159 K  3927
132.257.64.3     198.328.2.83        2855 M     11.9 G  15097
124.333.34.186   192.342.93.30       2958 M     10.7 G  40033
132.257.64.3     61.283.32.172       7469 M      10393  11
124.333.41.57    128.256.6.34        12.0 M     4490 M  22360
124.333.181.191  81.257.197.163      1350 M     4430 M  3341
132.257.64.3     130.262.101.6        276 M     2200 M  13064
124.333.34.186   66.263.169.170       389 M     2095 M  17919
132.257.195.68   140.267.28.48       91.3 M     2029 M  6275
132.257.212.232  151.293.199.65       39155     1994 M  24
124.333.41.61    206.290.82.18        3401      1853 M  22
132.257.64.3     61.278.72.30        1798 M          7  1
124.333.181.191  61.263.209.246      16.8 M     1676 M  113
132.257.64.3     261.232.163.3       1544 M      24069  9
132.257.64.3     61.273.210.110      1517 M       4140  7
124.333.34.186   128.342.121.70      1351 M      222 M  14861
132.257.64.3     258.14.200.58       1350 M      24075  14
132.257.64.3     222.330.100.28      1219 M       4077  7
132.257.64.3     210.261.41.131      1162 M         25  3
```

6 Analysis of Incidents and Alarms

Rule one: There are no rules . . .

This section describes a specific procedure that can be followed with each "incident" that Bro uncovers, but one must keep in mind that intrusion detection is not a static problem. The perpetrators of intrusions and malicious network activity are constantly changing their techniques with the express purpose of evading detection. Unexpected activities are often found by investigation of seemingly innocuous network oddities or serendipitous inspection of logs. While Bro is an exceptionally useful tool for collecting, sorting, analyzing and flagging suspect network data, it cannot be expected to flag all new, cleverly disguised attacks. Nor can it be expected to differentiate with 100% accuracy between aberrant, but legitimate, user behavior and a malicious attack. Sometimes a strong curiosity is an analyst's best friend and Bro is the vehicle for allowing him or her to follow that curiosity.

6.1 Two Types of Triggers

There are two ways that alarms can be triggered. One is when network traffic matches a *signatures* that has been converted to work with Bro. The other way is by matching Bro *rules* that are embedded in the Bro analyzers.

6.1.1 Converted Signatures

In the Bro report, converted signatures are identified by the alarm type: `SensitiveSignature` and the existence of a `bro` identification number. Each signature is distinct, targeting one specific set of network events for each alarm. Currently the majority of converted *signatures* are developed from Snort© signatures using the '`snort2bro`' utility. In addition, enhancing have been made by utilizing features in the Bro policy language that are absent in Snort©. Most Bro signatures are found in the '`$BROHOME/site/signatures.sig`', however, they can exist in other '`.sig`' files.

6.1.2 Embedded Bro Rule

Bro rules are typically embedded in the Bro *analyzers* or other '`.bro`' policy files. Several trigger conditions are usually lumped into a grouping of Bro rules within a '`.bro`' file, making it difficult to separate the exact condition that triggered the alarm. Hence, alarms triggered by an embedded Bro rule will not have a specific `bro` identification number, nor will the *signature code* block appear in the report.

Possible types of embedded bro rule alarms:

AddressDropped	AddressScan	BackscatterSeen
ClearToEncrypted_SS	CountSignature	DNS::DNS_MappingChanged
DNS::DNS_PTR_Scan	FTP::FTP_BadPort	FTP::FTP_ExcessiveFilename
FTP::FTP_PrivPort	FTP::FTP_Sensitive	FTP::FTP_UnexpectedConn
HTTP::HTTP_SensitiveURI	HotEmailRecipient	ICMP::ICMPAsymPayload
ICMP::ICMPUnpairedEchoReply	ICMP::ICMPConnectionPair	PrintSensitiveID
LoginForbiddenButConfused	LocalWorm	MultipleSigResponders
MultipleSignatures	OutboundTFTP	PasswordGuessing
PortScan	RemoteWorm	ResolverInconsistency
SSH_Overflow	SSL_SessConIncon	SSL_X509Violation

ScanSummary	SensitiveConnection	SensitiveDNS_Lookup
SensitivePortmapperAccess	SensitiveLogin	SensitiveSignature
SensitiveUsernameInPassword	SignatureSummary	SynFloodEnd
SynFloodStart	SynFloodStatus	TRW::TRWAddressScan
TerminatingConnection	W32B_SourceLocal	W32B_SourceRemote
ZoneTransfer		

6.2 General Process Steps

The following steps will both aid the Bro user with uncovering network activity of interest, and also help acquaint the user with the anomalies that Bro detects, together building up an understanding of what constitutes "normal" network traffic for the local site. The analyst might follow each successive step with each incident until a firm determination is made if the incident is malicious or a "false positive".

- Understand What Triggered the Alarm(s)
- Understand the Intent of the Alarm(s)
- Examine the Session(s) from the HTTP, FTP, or SMTP Logs
- Examine the Connection Logs for Breakin Indicators
- Examine for Connections to Other Computers
- Examine Other Bro Logs for Odd Activity
- Examine the Bulk Trace if Available
- Contact and Question Appropriate People

6.3 Understand What Triggered the Alarm(s)

To understand what triggered the alarm, compare the signature or rule code with *payload*. The network traffic that matches the signature, rule, or policy is known as the payload. The payload that triggers the alarm is usually included in the Bro's incident report. Often it is obvious that the payload is not malicious.

> **Example:** The signature may trigger on the word *shadow*, notifying that someone may be attempting to download the shadow password file. However, the payload may reveal that the actual download is something like *theshadow.jpg*, which is obliviously innocuous.

The two kinds of alarms, converted signatures and embedded rules trigger alarms differently, so they must be treated separately. The following sections describe how to investigate the signature or rule code and payload of each.

6.3.1 Converted Snort Signatures

These signatures are recognizable by the inclusion of a `bro` number and the identification `SensitiveSignature`. A *signature code* and *payload* block should be present in the incident report. To understand what triggered the alarm, compare the payload to the signature code and find the defined signature within the payload. Since some payload lines can get extremely long, the payload lines in the report and notice and alarm logs has been truncated to 250 characters. Sometimes the actual trigger payload is beyond the 250 character cut off. In this case, the protocol sessions log file must be examined. See Section 6.5 [Examine HTTP FTP or SMTP Sessions], page 32.

6.3.2 Embedded Bro Rule

For alarms triggered by an embedded Bro rule the *signature code* block will not appear, and in many cases, neither will the payload. There is currently no direct way to find the specific Bro rule that triggered the alarm other than to search the Bro policy files. Following is a process for conducting that search. The example of the `HTTP_SensitiveURL` is used. In actual practice, this rule appears quite often in the reports.

Read about the specific analyzer: In the Bro Technical Reference Manual there are sections for each type of analyzer. In the case of our example the HTTP analyzer is the obvious choice. In the section on the HTTP analyzer, it is noted that the variables *sensitive_URIs* and *sensitive_post_URIs* are responsible for flagging sensitive URIs.

Find the policy file that defines these variables: Using egrep to search for *sensitive_URIs* and/or *sensitive_post_URIs* yields the following:

```
> egrep "sensitive_URIs | sensitive_post_URIs" http*
http-request.bro:   const sensitive_URIs =
http-request.bro:  # URIs that match sensitive_URIs but can be generated by worms
http-request.bro:   const skip_remote_sensitive_URIs = /\/cgi-bin\/(phf|php\.cgi|
http-request.bro:   const sensitive_post_URIs = /wwwroot|WWWROOT/ &redef;
http-request.bro:   if ( (sensitive_URIs in URI && URI != worm_URIs) ||
http-request.bro:   (method == "POST" && sensitive_post_URIs in URI) )
http-request.bro:   skip_remote_sensitive_URIs in URI )
```

Clearly '`http-request.bro`' is the file of interest. If, in the case of other types of analyzers, more than one file appears, look for the place where the `const` statement is used to declare the variable(s).

Look into the policy file: Search in the section of Bro policy code that describes the rule(s) for the specific notification. In the file '`http-request.bro`', is found:

```
export{
   const sensitive_URIs =
      /etc.*\/.*(passwd|shadow|netconfig)/
      | /IFS[ \t]*=/
      | /nph-test-cgi\?/
      | /(%0a|\.\.)\/(bin|etc|usr|tmp)/
      | /\/Admin_files\/order\.log/
      | /\/carbo\.dll/
      | /\/cgi-bin\/(phf|php\.cgi|test-cgi)/
      | /\/cgi-dos\/args\.bat/
      | /\/cgi-win\/uploader\.exe/
      | /\/search97\.vts/
      | /tk\.tgz/
      | /ownz/        # somewhat prone to false positives
     &redef;

     # URIs that match sensitive_URIs but can be generated by worms,
     # and hence should not be flagged (because they're so common).
     const worm_URIs =
```

```
            /.*\/c\+dir/
            | /.*cool.dll.*/
            | /.*Admin.dll.*Admin.dll.*/
        &redef;
}

redef capture_filters +=  {
        ["http-request"] = "tcp dst port 80 or tcp dst port 8080
                            or tcp dst port 8000"
};

# URIs that should not be considered sensitive if accessed remotely,
# i.e. by a local client.
const skip_remote_sensitive_URIs = /\/cgi-bin\/(phf|php\.cgi|test-cgi)/
 &redef;

const sensitive_post_URIs = /wwwroot|WWWROOT/ &redef;
```

Unfortunately, there isn't any way of knowing exactly which one of these rules triggered the `HTTP_SensitiveURL` alarm. As will be seen in the next section, the triggering payload must be compared against this entire section.

6.4 Understand the Intent of the Alarm(s)

While understanding the technical signature or policy "code" that "triggered" the alarm, it is also useful to understand the reason the trigger was built.

- What attack or malicious behavior is the alarm trying to illuminate?
- What is the normal method of attack ... manual? automated? expert? novice?
- How long has the particular attack existed?
- How often is it seen? How often is it actually used by attackers?

All of these things, and any other information that can be gathered, will help in differentiating attacks from legitimate behavior. Although this process may seem tedious and time consuming in the beginning, the Bro analyst will quickly build up a substantial knowledge of known attacks. Even if the incident in question turns out to be benign, the effort to learn about the attack almost always proves useful in future investigations.

6.4.1 Converted Snort© Signatures

Since Snort© signatures are usually fairly well documented, one way to discover the intent of the signature is to search the web for the title of the signature using any of the common search engines (Yahoo, Google, Teoma, AltaVista, or one of the may others). For instance, a search on the *MS SQL xp_cmdshell* vulnerability yields ~7000 hits. One of those hits is:

```
Zone-H.org * Advisories
... Successful exploitation of this vulnerability can enable an attacker to
execute commands in the system (via MS SQL xp_cmdshell function). ...
www.zone-h.org/advisories/read/id=4243 - 17k - Cached - Similar pages
```

This web site give a fairly detailed description of the exploit and verifies that it can be used to root compromise a computer and hence, is a vulnerability of significant interest. Several other sites also give details about the signature, the attack, and other useful information.

6.4.2 Embedded Bro Rule

Unfortunately, most of the embedded Bro rules have not been documented. The analyst must rely on his/her own understand of network attacks to guess what the intent of the rule is. Sometimes useful comments are written into the Bro policy source.

6.5 Examine HTTP FTP or SMTP Sessions

These three files record session activity on ports 80(http), 21(ftp), and 25(smtp) respectively. If the alarm involves any of these ports, these files may reveal the details of the sessions. The general format of all three files is: date/timeSP%sessionnumberSPMessage

where:

date/time: is the time in UNIX epoch time. The `cf` utility can be used to convert this time to *readable* time. Reference Tech Manual

sessionnumber: is the number assigned to session. All subsequent records in the file that are part of the session will retain this same session number. Session numbers are prefixed with the '`%`' sign.

message: is the message that Bro policy has formed to describe the session event. Typically the message will be:

- the start of the session, including the two ip addresses involved
- an anomolous event
- the full protocol command line that was sent
- short statistics concerning the transaction (e.g. bytes sent)

In an alarm where the session number is given (typically in a SensitiveSignature alarm), a search on the session number in the appropriate file(s) will show the full sessons. See Section A.7 [The bro/logs Directory], page 56.

Example:
Consider the following alarm:

```
Alarm: HTTP_SensitiveURI
       11/22_12.52.42                128.333.48.179 -> 80.143.378.186
                                           3091/tcp -> 80/tcp
       session: %73280
       payload: GET\/NR/rdonlyres/eirownz4tqwlseoggqm2ahj5cqsdbedlaxyye
                7kvdz7rnh6u4o2v2gpvmoggqjlekzdtulryyatiinj3xwimmiavgfb/
                smallshoulders.gif\ (200\ "OK"\ [1134])
```

From the payload shown, it is unclear what triggered the alarm. To investigate further, the entire session can be viewed:

Example:

```
> grep %73280 http.hostname.04-11-22_12.52.42 | cf
Nov 22 15:18:30 %73280 start 128.333.48.179 > 80.143.378.186
Nov 22 15:18:30 %73280 GET /fitness/default.htm (200 "OK" [10473])
Nov 22 15:18:30 %73280 GET /javascripts/cms_common.js (304 "Not Modified"[0])
Nov 22 15:19:47 %73280 GET /food_nutrition/default.htm (200 "OK"[13177])
Nov 22 15:19:47 %73280 GET/NR/rdonlyres/eirwwu3xtlr22dkat5cim4ziupouzxb6kz4xb
zbr4zs255ca57cvv5mhcjcrmrfg6kpcrevyndo2za3yoi5esheiolf/News111904Dairy NotFor
Diet.jpg (200 "OK" [6572])
Nov 22 15:19:51 %73280 GET /NR/rdonlyres/0D25692F-D59A-4B90-AB53-8BBC9E75A286.
gif (200 "OK" [189])
Nov 22 15:19:51 %73280 GET /NR/rdonlyres/eqpbdbex34wpqpagp2fcbxh35omcjtq45feyf7
zgtjff6fhrybfbsvtszeu4rc2clayghhslfimaafkoocae6cv6wof/doctor.jpg (200 "OK" [161
5])/NR/rdonlyres/enhskrfoodzuquvmbli2hasjspusrgsvyhbd3nlue5msoli2ueagrwdxw56gqa
aa7sosee3yn2hwywcg6kgv4wcv6jc/bigback.gif (200 "OK" [8192 (interrupted)])/NR/rd
onlyres/ej2cpd275ghrefp23ezou43haqe6fmj3oyeqxkvopf4bv4zhwbqimfrrbndqpotx55pogc7
xiqvdcovaxo66afyqfof/smallleg.jpg (200 "OK" [1010])
Nov 22 15:22:12 %73280 GET /NR/rdonlyres/eirownz4tqwlseoggqm2ahj5cqsdbedlaxyye7
kvdz7rnh6u4o2v2gpvmoggqjlekzdtulryyatiinj3xwimmiavgfb/smallshoulders.gif (200 "
OK" [1134])
Nov 22 15:22:13 %73280 GET /NR/rdonlyres/49D86A33-AF6C-4873-AD11-F26DDBF222B1.g
if (200 "OK" [167])
```

By examining this session it can clearly be seen that the session is simply a web visit to a fitness website. There is no need to investigate further.

6.6 Examine the Connection and Weird Logs

The connection logs are a record of every connection Bro detects. Although they don't contain content, being able to track the network *movement* of an attacking host is often very useful.

6.6.1 Breakin Indicators

If it is still not clear if a suspect host is an attacker, the connection surrounding the suspicious connection can be examined. Here are some questions that might be answered by the '`conn`' logs. itemize item How many more successful connection the attacker make to the target host? item How much data was transfered? A lot of data means something more than an unsuccessful probe. item Did the target host connect back to the attacker? This is a fairly sure sign of a successful attack. The attacker has gained control of the target and is connecting back to his own host. item What was the time duration? If several attacks occur in a very short time and then slow down to *human* speed, it could indicate the attacker used an automated attack to gain control and then switched to a manual mode to "work on" the compromised target host.

6.6.2 Connections to Other Computers

If a host has been successfully identified as an attacker, it is useful to know what and how many other hosts the attacker has touched. This can be found by grepping through the '`conn`' logs for instances of connections by the suspect host.

```
example here
```

If the attack used a specific, little used, port; another investigation would be to search for other similar connection using that port. Often the attacker might change attack hosts, but will continue to use the same attack method.

`example here`

NOTES: *You may want to go back several days, weeks, months, or even years to see if the attacker has visited (and perhaps compromised) you site earlier without being detected.*
However, be forwarned that the '`conn`*' logs tend to get very large and doing extensive searches can take a very long time.*

6.6.3 Odd Activity

Despite attempts to have the network community adhere to network standards, non-compliant traffic occurs all the time. The '`weird`' logs are a record of instances of network traffic that simply should not happen.

While these logs are usually of interest to the most hard-core of network engineers, if a unique attack is detected, it is sometimes valuable to search the weird logs for other unusual activities by the attacking host. Hackers are not bound by standard protocol and sometimes find ways to circumvent security via *weird* methods.

6.7 Examine the Bulk Trace if Available

For information on using the Bulk trace files for analysis, see Chapter 10 [Bulk Traces and Off-line Analysis], page 48.

6.8 Contact and Question Appropriate People

The final and usually the most definitive investigation is to call the owners of the hosts involved. Often a call to the owner of the local host can reveal that the activity was not normal, but appropriate or a mistake.

7 Customizing Bro

Bro is very customizable, and there are several ways to modify Bro to suit your environment. You can write your own policy analyzers using the Bro language. Most sites will likely just want to do minor customizations, such as changing the level of an alert from "notice" to "alarm", or turning on or off particular analyzers. The chapter describes how to do these types of customizations. Information on how to write your own analyzers can be found in the Bro Reference Manual.

The default policy scripts for Bro are all in $BROHOME/policy. These files should **never** be edited, as your edits will be lost when you upgrade Bro. To customize Bro for your site, you should make all your changes in $BROHOME/site. Many simple changes just require you to *redefine* (using the `redef` operator, a Bro constant from a standard policy script with your own custom value. You can also write your own custom script to do whatever you want.

For example, to add "guest" to the list of forbidden_ids (user names that generate a login alarm), you do this:

```
redef forbidden_ids += { "guest", };
```

In this chapter we give an overview of all the standard Bro policy scripts, what notices they generate, and how to customize the most commonly changed items, and how to write new policy modules.

7.1 Builtin Policy Files

Bro *policy* script is the basic analyzer used by Bro to determine what network events are alarm worthy. A policy can also specify what actions to take and how to report activities, as well as determine what activities to scrutinize. Bro uses policies to determine what activities to classify as *hot*, or questionable in intent. These hot network sessions can then be flagged, watched, or responded to via other policies or applications determined to be necessary, such as calling `rst` to reset a connection on the local side, or to add an IP address block to a main router's ACL (Access Control List). The policy files use the Bro scripting language, which is discussed in great detail in The Bro Reference Manual.

Policy files are loaded using an `@load` command. The semantics of `@load` are "load in this script if it hasn't already been loaded", so there is no harm in loading something in multiple policy scripts. The following policy scripts are included with Bro. The first set are all on by default, and the second group can be added by adding them to your '`site/brohost.bro`' policy file.

Bro Analyzers are described in detail in the Reference Manual. These policy files are loaded by default:

`site`	defines local and neighbor networks from static config
`alarm`	open logging file for alarm events
`tcp`	initialize BPF filter for SYN/FIN/RST TCP packets
`login`	rlogin/telnet analyzer (or to ensure they are disabled)
`weird`	initialize generic mechanism for detecting unusual events
`conn`	access and record connection events
`hot`	defines certain forms of sensitive access
`frag`	process TCP fragments

`print-resources`	on exit, print resource usage information, useful for tuning
`signatures`	the signature policy engine
`scan`	generic scan detection mechanism
`trw`	additional, more sensitive scan detection
`http`	general http analyzer, low level of detail
`http-request`	detailed analysis of http requests
`http-reply`	detailed analysis of http replys
`ftp`	FTP analysis
`portmapper`	record and analyze RPC portmapper requests
`smtp`	record and analyze email traffic
`tftp`	identify and log TFTP sessions
`worm`	flag HTTP-based worm sources such as Code Red
`software`	track software versions; required for some signature matching
`blaster`	looks for blaster worm
`synflood`	looks for synflood attacks
`stepping`	used to detect when someone logs into your site from an external net, and then soon logs into another site
`reduce-memory`	sets shorter timeouts for saving state, thus saving memory. If your Bro is using < 50% of you RAM, try not loading this

These are **not** loaded by default:

Policy	**Description**	**Why off by default**
`drop`	Include if site has ability to drop hostile remotes	Turn on if needed
`icmp`	icmp analysis	CPU intensive and low payoff
`dns`	DNS analysis	CPU intensive and low payoff
`ident`	ident program analyzer	historical, no longer interesting
`gnutella`	looks for hosts running Gnutella	Turn this on if you want to know about this
`ssl`	ssl analyzer	still experimental
`ssh-stepping`	Detects stepping stones where both incoming and outgoing connections are ssh	Possibly too CPU intensive (needs more testing)
`analy`	Performs statistical analysis	only used in off-line alalysis
`backdoor`	Looks for backdoors	only effective when also capturing bulk traffic
`passwords`	Looks for clear text passwords	may want to turn on if your site does not allow clear text passwords
`file-flush`	Causes all log files to be flushed every N seconds	may want to turn on if you are doing "real time" analysis

To modify which analyzers are loaded, edit or create a file in '`$BROHOME/site`'. If you write your own new custom analyzer, it goes in this directory too. To disable an analyzer, add `"@unload policy.bro"` to the beginning of the file '`$BROHOME/site/brohost.bro`', before the line `"@load brolite.bro"`. To add additional analyzers, add them `@load` them in '`$BROHOME/site/brohost.bro`'.

7.2 Notices

The primary output facility in Bro is called a *Notice*. The Bro distribution includes a number of standard of Notices, listed below. The table contains the name of the Notice, what Bro policy file generates it, and a short description of what the Notice is about.

Notice	Policy	Description
AckAboveHole	weird	Could mean packet drop; could also be a faulty TCP implementation
AddressDropIgnored	scan	A request to drop connectivity has been ignored ; (scan detected, but one of these flags is true: !can_drop_connectivity, or never_shut_down, or never_drop_nets)
AddressDropped	scan	Connectivity w/ given address has been dropped
AddressScan	scan	The source has scanned a number of addrs
BackscatterSeen	scan	Apparent flooding backscatter seen from source
ClearToEncrypted_SS	stepping	A stepping stone was seen in which the first part of the chain is a clear-text connection but the second part is encrypted. This often means that a password or passphrase has been exposed in the clear, and may also mean that the user has an incomplete notion that their connection is protected from eavesdropping.
ContentGap	weird	Data has sequence hole; perhaps due to filtering
CountSignature	signatures	Signature has triggered multiple times for a destination
DNS::DNS_MappingChanged	DNS	Some sort of change WRT previous Bro lookup
DNS::DNS_PTR_Scan	dns	Summary of a set of PTR lookups (automatically generated once/day when dns policy is loaded)
DroppedPackets	netstats	Number of packets dropped as reported by the packet filter
FTP::FTP_BadPort	ftp	Bad format in PORT/PASV;
FTP::FTP_ExcessiveFilename	ftp	Very long filename seen
FTP::FTP_PrivPort	ftp	Privileged port used in PORT/PASV
FTP::FTP_Sensitive	ftp	Sensitive connection (as defined in *hot*)
FTP::FTP_UnexpectedConn	ftp	FTP data transfer from unexpected src

HTTP::HTTP_SensitiveURI	http	Sensitive URI in GET/POST/HEAD (default sensitive URIs defined http-request.bro; e.g.: /etc.*\/.*(passwd|shadow|netconfig)
HotEmailRecipient	smtp	XXX Need Example, default = NULL
ICMP::ICMPAsymPayload	icmp	Payload in echo req-resp not the same
ICMP::ICMPConnectionPair	icmp	Too many ICMPs between hosts (default = 200)
IdentSensitiveID	ident	Sensitive username in Ident lookup
LocalWorm	worm	Worm seen in local host (searches for code red 1, code red 2, nimda, slammer)
LoginForbiddenButConfused	login	Interactive login seen using forbidden username, but the analyzer was confused in following the login dialog, so may be in error.
MultipleSigResponders	signatures	host has triggered the same signature on multiple responders
MultipleSignatures	signatures	host has triggered many signatures
MultipleSigResponders	signatures	host has triggered the same signature on multiple responders
OutboundTFTP	tftp	outbound TFTP seen
PasswordGuessing	scan	source tried too many user/password combinations (default = 25)
PortScan	scan	the source has scanned a number of ports
RemoteWorm	worm	worm seen in remote host
ResolverInconsistency	dns	the answer returned by a DNS server differs from one previously returned
ResourceSummary	print-resources	prints Bro resource usage
RetransmissionInconsistency	weird	possible evasion; usually just bad TCP implementation
SSL_SessConIncon	ssl	session data not consistent with connection
SSL_X509Violation	ssl	blanket X509 error
ScanSummary	scan	a summary of scanning activity, output once / day
SensitiveConnection	conn	connection marked "hot", See: Reference Manual section on hot ids for more information.
SensitiveDNS_Lookup	dns	DNS lookup of sensitive hostname/addr; default list of sensitive hosts = NULL
SensitiveLogin	login	interactive login using sensitive username (defined in 'hot')
SensitivePortmapperAccess	portmapper	the given combination of the service looked up via the pormapper, the host requesting the lookup, and the host from which it's requiesting it is deemed sensitive

SensitiveSignature	signatures	generic for alarm-worthy
SensitiveUsername InPassword	login	During a login dialog, a sensitive username (e.g., "rewt") was seen in the user's password. This is reported as a notice because it could be that the login analyzer didn't track the authentication dialog correctly, and in fact what it thinks is the user's password is instead the user's username.
SignatureSummary	signatures	summarize number of times a host triggered a signature (default = 1/day)
SynFloodEnd	synflood	end of syn-flood against a certain victim. A syn-flood is defined to be more than SYNFLOOD_THRESHOLD (default = 15000) new connections have been reported within the last SYNFLOOD_INTERVAL (default = 60 seconds) for a certain IP.
SynFloodStart	synflood	start of syn-flood against a certain victim
SynFloodStatus	synflood	report of ongoing syn-flood
TRWAddressScan	trw	source flagged as scanner by TRW algorithm
TRWScanSummary	trw	summary of scanning activities reported by TRW
Terminating Connection	conn	"rst" command sent to connection origin, connection terminated, triggered in the following policies: ftp and login: forbidden user id, hot (connection from host with spoofed IP address)
W32B_SourceLocal	blaster	report a local W32.Blaster-infected host
W32B_SourceRemote	blaster	report a remote W32.Blaster-infected host
WeirdActivity	Weird	generic unusual, alarm-worthy activity

Note that some of the Notice names start with "ModuleName::" (e.g.: FTP::FTP_BadPort) and some do not. This is becuase not all of the Bro Analyzers have been converted to use the Modules facility yet. Eventually all notices will start with "ModuleName::".

To get a list of all Notices that your particular Bro configuration might generate, you can type:

```
sh . $BROHOME/etc/bro.cfg; bro -z notice $BRO_HOSTNAME.bro
```

7.3 Notice Actions

Notices that are deemed particularly important are called *Alarms*. Alarms are sent to the alarm log file, and to optionally to *syslog*.

The standard Bro distribution supports a number of types of *notice actions*, these are:

NOTICE_IGNORE	do nothing
NOTICE_FILE	send to 'notice' file
NOTICE_ALARM_ALWAYS	send to alarm file and *syslog*

NOTICE_ALARM_PER_CONN	send to alarm file once per connection
NOTICE_EMAIL	send to alarm file and send email
NOTICE_PAGE	send to alarm file and send to pager

It is also possible to define your own custom notice actions.

By default, all notices are set to NOTICE_ALARM_ALWAYS except for the following:

```
ContentGap, AckAboveHole, AddressDropIgnored, PacketsDropped,
RetransmissionInconsistency
```

By default all Alarms are also sent to *syslog*. To disable this, add:

```
redef enable_syslog = F;
```

To change the default notice action for a given notice, add something like this to your 'site/brohost.bro' file:

```
redef notice_action_filters += {
     [[WeirdActivity, ContentGap]] = ignore_notice,
};
```

This will cause the Notices WeirdActivity and ContentGap to no longer get logged anywhere. To send these Notices to the Notice log file only, and not to the Alarm log, add this:

```
redef notice_action_filters += {
     [[WeirdActivity, ContentGap]] = file_notice,
};
```

For NOTICE_EMAIL and NOTICE_PAGE, email is sent using the script specified by the mail_script variable (default: "mail_notice.sh"), which must be in $PATH. To activate this, $mail_dest must be set. Email is only sent if Bro is reading live traffic.

For example, to send email on TerminatingConnection and FTP_Sensitive notices, add something like this:

```
redef mail_dest = "youremail@yoursite.edu";

redef notice_action_filters += {
    [[TerminatingConnection, FTP::FTP_Sensitive]] = send_email_notice,
};
```

7.4 Customizing Builtin Policy

The default policy scripts for Bro are all in $BROHOME/policy. Remember that these files should **never** be edited, as your edits will be lost when you upgrade Bro. To customize Bro for your site, you should make all your changes in $BROHOME/site. Many simple changes just require you to *redefine* (using the redef operator, a Bro constant from a standard policy script with your own custom value. You can also write your own custom script to do whatever you want.

Here are some example of the types of things you may want to customize.

To add "guest" to the list of forbidden_ids (user names that generate a login alarm), you do this:

```
redef forbidden_ids += { "guest", };
```

To add a new rootkit string to HTTP sensitive_URIs:

```
redef HTTP::sensitive_URIs += /^.*rootdown.pl.*$/;
```

7.5 Writing New Policy

For example, if your site only allows external http and mail to a small, controlled lists of hosts, you could write a new .bro file containing this:

```
const web_servers = { www.lbl.gov, www.bro-ids.org, };
const mail_servers = { smtp.lbl.gov, smtp2.lbl.gov, };

const allow_my_services: set[addr, port] = {
        [mail_servers, smtp],
        [web_servers, http],
};
```

Bro can then generate an Alarm or even terminate the connection for policy violations. For example:

```
event connection_established(c: connection)
{
   local id = c$id;
   local service = id$resp_p;
   local inbound = is_local_addr(id$resp_h);

   if ( inbound && [id$resp_h, service] !in allow_my_services )
      NOTICE ([$note=SensitiveConnection, $conn=c,
               $msg=fmt("hot: %s", full_id_string(c)) ]);
    if ( inbound && service in terminate_successful_inbound_service )
            terminate_connection(c);
}
```

To test this you might do the following. First, generate some "offline" data to play with:

```
# tcpdump -s 0 -w trace.out port smtp or port http
```

Kill off the tcpdump after capturing traffic for a few minutes (use ctrl-C). Then add the above Bro code to your hostname.bro file, and run Bro against this captured trace file:

```
# setenv BROHOME /usr/local/bro
# setenv BROPATH $BROHOME/site:$BROHOME/policy
# bro -r trace.out hostname.bro
```

7.6 Signatures

NOTE: Bro Signatures mechanism is still under development

Signatures in Bro are quite different than standard packet matching signatures such as those used in Snort. A Bro signature, or *Rule*, is a *contextual signature* that can include connection-level information. Hence Bro signatures generate **far** fewer false positives.

However, Bro's contextual signatures are fairly CPU and memory intensive, and still generate more false positives than we'd like, so for now they are turned off by default. See the next section for information on how to turn them on.

For example, an packet-level signature of a HTTP attack only looks at the attack packet, where the Bro contextual signature also looks for the HTTP reply, and only generates an alarm if the attack was successful.

In this section we explain how to customize signatures for your site, and how to import new signatures from Snort and bro-ids.org. More information on the details of Bro signatures are in the signature section of the reference manual.

The following files are used to control and customize Bro signatures.

- `$BROHOME/site/signatures.sig`: Bro version of snort signatures
- `$BROHOME/policy/sig-addendum.sig`: Bro supplied signatures
- `$BROHOME/policy/sig-action.bro`: policy file to control signature notification type

Files in `$BROHOME/policy` contain the default Bro signatures, and should not be edited. Files in `$BROHOME/site` contain files you will use to customize signatures for your site. New signatures that you write go here too. All files ending in `.sig` in this directory will be loaded into the signature engine. In fact, all .sig files in any directory in `$BROPATH` (set in `$BROHOME/etc/bro.cfg`) will be loaded.

7.6.1 Turning Signatures ON/OFF

Signature matching is off by default. To use a small set of known, high quality signatures, add the following to your site policy file:

```
@load brolite-sigs
```

To use the full set of converted snort signatures, add both of these lines:

```
@load brolite-sigs
redef signature_files += "signatures";
```

If signatures are turned on, then you can control the signature "action" levels through the file `$BROHOME/site/sigaction.bro`. You can set the signature action to the one of the following:

```
SIG_IGNORE          # ignore this sig. completely
SIG_FILE            # write to signatures and notice files
SIG_ALARM           # alarm and write to notice and alarm files
SIG_ALARM_PER_ORIG  # alarm once per originator
SIG_ALARM_ONCE      # alarm once and then never again
```

All signatures default to action = `SIG_ALARM`. To lower the alarm level of the signature, add an entry to the file `$BROHOME/site/sigaction.bro`. The Bro distribution contains a default sigaction.bro file that lowers the level of a number of signatures from ALARM to FILE (notice) .

To permanently remove a signature you can delete it from the `.sig` file.

7.6.2 Add a New Signature

To add a new signature to a running Bro, add the signature to the file `$BROHOME/site/site.sig` (or create a new `.sig` file in `$BROHOME/site`), and then restart Bro using `"$BROHOME/etc/bro.rc checkpoint"`.

A sample signature looks like this:

```
signature formmail-cve-1999-0172 {
       ip-proto == tcp
       dst-ip == 1.2.0.0/16
       dst-port = 80
       http /.*formmail.*\?.*recipient=[^&]*[;|]/
       event "formmail shell command"
       }
```

For more details, see the reference manual.

7.6.3 Editing Existing Signatures

Bro supplied signatures are in $BROHOME/sigs. You should not edit these, as they will get overwritten when you update Bro. Instead, make your modifications in $BROHOME/site. If you use the same signature ID as an existing signature, the site sig will take precedence.

7.6.4 Importing Snort Signatures

New snort signatures come out almost every week. To add these to Bro, do the following:

(XXX section not done!)

Add instructions for incorporating new sigs from Snort.

7.6.5 Checking for new Signatures from bro-ids.org

note: this functionality is currently under development, and does not yet exist

The Bro team will be constantly updating our set of default signatures and posting them on the Bro web site. To download the latest signatures and incorporate them into your Bro setup, run the script:

```
$BROHOME/scripts/update-sigs
```

This script uses the `wget` command to download the latest signatures and puts them into the required Bro files, and then restarts Bro to load the new signatures..

7.7 Tuning Scan Detection

There are a large number of tunable parameters in the scan analyzer, all of which are described in the reference manual. Most of these parameters should be fine for all sites. The only settings that you may want to tune are:

- report_peer_scan: Generate a log message whenever a remote host has attempted to connect to the given number of distinct hosts. Default = `{` 100, 1000, 10000, `}`.
- report_outbound_peer_scan: Generate a log message whenever a local host has attempted to connect to the given number of remost hosts. Default = `{` 100, 1000, `}`.
- skip_services: list of ports to ignore scans on, because they often gets scanned by legitimate (or at least common) services. The default list can be found in the brolite.bro file.

If you want enable ICMP scan detection, set these:

```
redef ICMP::detect_scans = T;
redef ICMP::scan_threshold = 100;
```

7.8 Other Customizations

There are a number of things you may wish to customize.

hot_ids

The policy file 'hot-ids.bro' contains a number of constants that you might want to customize by "redef"ing them in your brohost.bro policy file. These are all used to generate FTP and login alarms (SensitiveConnection Notice) for suspicious users. The user ID's that are in `hot_ids` and not in `always_hot_ids` are only hot upon successful login. For details see the Bro Reference Manual.

constant	Defaults
forbidden_ids	"uucp", "daemon", "rewt", "nuucp", "EZsetup", "OutOfBox", "4Dgifts", "ezsetup", "outofbox", "4dgifts", "sgiweb" "r00t", "ruut", "bomb", "backdoor", "bionic", "warhead", "check_mate", "checkmate", "check_made", "themage", "darkmage", "y0uar3ownd", "netfrack", "netphrack"
always_hot_ids	"lp", "demos", "retro", "milk", "moof", "own", "gdm", "anacnd", + forbidden_ids
hot_ids	"root", "system", "smtp", "sysadm", "diag", "sysdiag", "sundiag", "sync", "tutor", "tour", "operator", "sys", "toor", "issadmin", "msql", "sysop", "sysoper", + always_hot_ids

Input/Output Strings

The policy files login.bro and ftp.bro both contain a list of input and output strings that indicate suspicious activity. In you wish to add anything to this list, you may want to `redef` one of these.

```
login.bro: see input_trouble and output_trouble
ftp.bro: see ftp_hot_files
```

Sensitive URIs

The policy file http-request.bro contain a list of http URI's that indicate suspicious activity. In you wish to add anything to this list, you may want to `redef` one of these.

```
sensitive_URIs
sensitive_post_URIs
```

Log Files

`redef` this to rotate the log files every N seconds

```
log_rotate_interval (default = 0 sec, don't rotate)
```

`redef` this to rotate the log files when they get this big

```
log_max_size (default = 250e6, rotate when any file exceeds 250 MB)
```

8 Intrusion Prevention Using Bro

Bro includes two important *active response* capabilities that allow sites to use Bro for intrusion prevention, and not just intrusion detection. These include the ability to terminate a connection known to be an intrusion, and the ability to update a blocking router's access control list (ACL) to block attacking hosts.

8.1 Terminating a Connection

The Bro distribution includes a program called `rst` that will terminate a active connection by sending a TCP "reset" packet to the sender. The `ftp` and `login` analyzers look for connections that should be terminated. All connections from a `forbidden_id` get flagged for termination, as well as any service defined in `terminate_successful_inbound_service`.

Connection termination is off by default. To enable it, redefine the following flag in your '`site/site.local.bro`' file:

```
redef activate_terminate_connection = T ;
```

Connections are terminated using the `rst` program, which is installed in $BRO-HOME/bin. To use this program change the file permission to be setuid root. Whenever a connection is terminated you will see a `TerminatingConnection` alarm. If Bro detects a connection that Bro thinks is a candidate for termination, but `activate_terminate_connection = F`, then you will see the alarm: `IgnoreTerminatingConnection`.

You may want to add a number of services to the list of forbidden services. For example, to terminate all successful attempts to access the RPC portmapper via TCP from an external network, you would add this:

```
redef terminate_successful_inbound_service += {
    [111/tcp] = "disallow external portmapper"
};
```

This will prevent NFS connections from external hosts. P2P services such as KaZaa can also be terminated in this manner. You can make exceptions to `terminate_successful_inbound_service` by redefing `allow_services_to`. See `hot.bro` for details.

8.2 Updating Router ACL

Bro can be used to send the IPs of scanning or attacking hosts to your router, so that the router can drop these hosts.

Since every router does this differently, you will need to write a script that works for your router.

To active your custom drop script, add this to your hostname.bro file:

```
@load scan
redef can_drop_connectivity  = T;
redef drop_connectivity_script = "my_drop_script";
```

At LBL we use a program called acld to update the ACLs in our boarder routers on the fly. This code is available at: ftp://ftp.ee.lbl.gov/acld.tar.gz

9 Performance Tuning

NOTE: This chapter still a rough draft and incomplete

If the link you are monitoring with Bro has too many connections per second, or if you have too many policy modules loaded, it is possible that Bro will not be able to keep up, and that the Bro host will drop too many packets to be able to perform accurate analysis.

A "rule of thumb" for Bro is that if CPU usage is < 50% and memory use is < 70% of physical memory, than you should not have any worries.

Otherwise you might want to explore the tuning options below.

For sites with an extremely high load you might consider using multiple Bro boxes, each configured to capture and analyze different types of traffic.

Note that the amount of CPU required by Bro is a function of both the number of connections/second and the number of packets/second. So it's possible that a large site (e.g., 2,000 hosts) on a slow link (e.g., 100 Mbps) would still have performance issues because it has a very large number of connections / second.

9.1 Hardware and OS Tuning

If your CPU load > 50% or your memory footprint is > 70% of physical memory, an obvious solution is to buy a faster CPU or more memory.

If this is not possible, here are some other things to try.

FreeBSD

First, check that your BPF buffer size is big enough. The Bro installation script should set this correctly for you, but to test this, do:

```
sysctl debug.bpf_bufsize
sysctl debug.bpf_maxbufsize
```

They should both be at least 4 MB.

Next, if your Bro host is capturing packets on 2 interfaces and you are running FreeBSD, we provide a patched kernel that bonds both interfaces into a single interface at the BPF level. This reduces CPU load considerably. This patched kernel also increases the default per-process memory limits.

This kernel source is available for download at:

http://www.bro-ids.org/download/FreeBSD.4.10.bro.tgz.

To install this kernel and the BPF bonding utilites, type:

```
tar xfz fbsd.4.10.bond.tgz
cd FreeBSD-4-10-RELEASE/sys/i386/conf
/usr/sbin/config BRO
cd ../../compile/BRO
make depend
make
make install

cd FreeBSD-4-10-RELEASE/local/sbin/bpfbond/
make
make install

reboot
```

For more instructions on rebuilding the kernel, see: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/kernelconfig.html.

Linux

Check that the net.core.rmem_max buffer is big enough. The Bro installation script should set this correctly for you, but to test this, do:

```
sysctl net.core.rmem_max
```

It should be at least 4 MB.

For heavy traffic load, the Linux version of libpcap has a hard time keeping up. There are a couple a options available to improve Linux pcap performance. These include:

Phil Wood's libpcap replacement: (see http://public.lanl.gov/cpw/) Luca Deri's patch to fix libpcap issues. (see http://luca.ntop.org/Ring.pdf)

(Note that Phil Wood's version of libpcap seems to be buggy in non-blocking mode. Build Bro using the –disable-selectloop option to disable non-blocking mode if using this version of libpcap.)

9.2 Bro Policy Tuning

If the hardware and OS tuning solutions fail to bring your CPU load or memory consumption under control, next you will have to start turning off analyzers. Signatures are particularly CPU and memory intensive, so try turning it off or greatly reduce the number of signatures it is processing. The HTTP analyzers are also CPU intensive. For example, to turn off the HTTP reply analyzer, add the following lines at the beginning of the file `$BROHOME/site/brohost.bro`, before any `@`load commands.

```
@unload http-reply
```

Another solution is to modify libpcap filter for Bro. This is done by adding `restrict_filters`. For example, to only capture SYN/FIN packets from a large web proxy, you can do this:

```
redef restrict_filters += { ["not proxy outbound Web replies"] =
     "not (host bigproxy.mysite.net and
           src port 80 and (tcp[13] & 7 == 0))" };
```

This filter will allow you to record the number and size of the HTTP replies, but will not do further HTTP analysis.

Another way to reduce the CPU load of Bro analysis is to split the work across two Bro hosts. An easy way to do this is to take the sum of the source and destination IPs, and monitor even combinations on one host and odd combinations on a second host.

For example:

```
redef restrict_filters += { ["capture even src/dest pairs only"] = "(ip[12:4] + ip[16:4]) &
```

10 Bulk Traces and Off-line Analysis

NOTE: This chapter still a very rough draft and incomplete

Bro is most effective when used in conjunction with bulk traces from your site. Capturing bulk traces just involves using `tcpdump` to capture all traffic entering and leaving your site.

Bulk traces can be very valuable for forensic analysis of all traffic in and out of a compromised host. It is also needed to run some particularly CPU intensive policy analyzers that can not be done in real time (as described in the Off-line Analysis section below).

Depending on your traffic load, you might be able to bulk capture on the Bro host directly, but in general we recommend using a separate packet capture host for this. Unless you want to buy a huge amount of disk, you'll probably only be able to save a few days worth of traffic.

10.1 Bulk Traces

The Bro distribution includes a couple scripts to make bulk capture easier. These are:

`spot-trace`: called by `start-capture-all` script

`start-capture-all`: captures all packets. This script looks for an existing instance of the `spot-trace` program, and if it finds one creates a new capture file name with an incremented filename, and continues capturing data. Bulk capture files can get very large, so typically you run this as a cron job every 1-2 hours.

`bro_bulk_compress.sh`: compress and/or delete old bulk trace files. Run as a cron job.

Since the bulk trace files can be huge, you often will want to run tcpdump on the raw trace with a filter to extract the packets of interest. For example:

```
tcpdump -r bulkXXX.trace -w goodstuff.trace 'host w.x.y.z'
```

If you know that that packets you want are bounded by a time interval, say it occurred 1:17PM-1:18PM, then you can speed this up a great deal using tcpslice. For example:

```
tcpslice 13h15m +5m bulkXXX.trace | tcpdump -r - -w goodstuff.trace 'host w.x.y.z'
```

It is recommend to use a somewhat broader time interval for tcpslice (such as in the above example) than when Bro reported the activity occurred, so you can catch additional related packets cheaply.

10.2 Off-line Analysis

There are some policy modules that are meant to be run as off-line analysis on bulk trace files. These include:

`backdoor.bro`: looks for standard services running on non-standard ports. These services include ssh, http, ftp, telnet, and rlogin.

To run Bro on a tcpdump file, do something like this:

```
# set up the Bro environment (sh or bash)
. /usr/local/bro/etc/bro.cfg
/usr/local/bro/bin/bro -r dumpfile backdoor.bro
```

To use Bro to extract the contents of a trace file, do:

```
bro -r tracefile contents
```

which will load policy/contents.bro. It stores the contents of each connection in two files, contents.H1.P1.H2.P2 and contents.H2.P2.H1.P1, where H1/P1 is the host/port of the originator and H2/P2 the same for the responder.

You can extract just the connections of interest using, for example:

```
bro -f "host 1.2.3.4" -r tracefile contents
```

Appendix A Bro Directory and Files

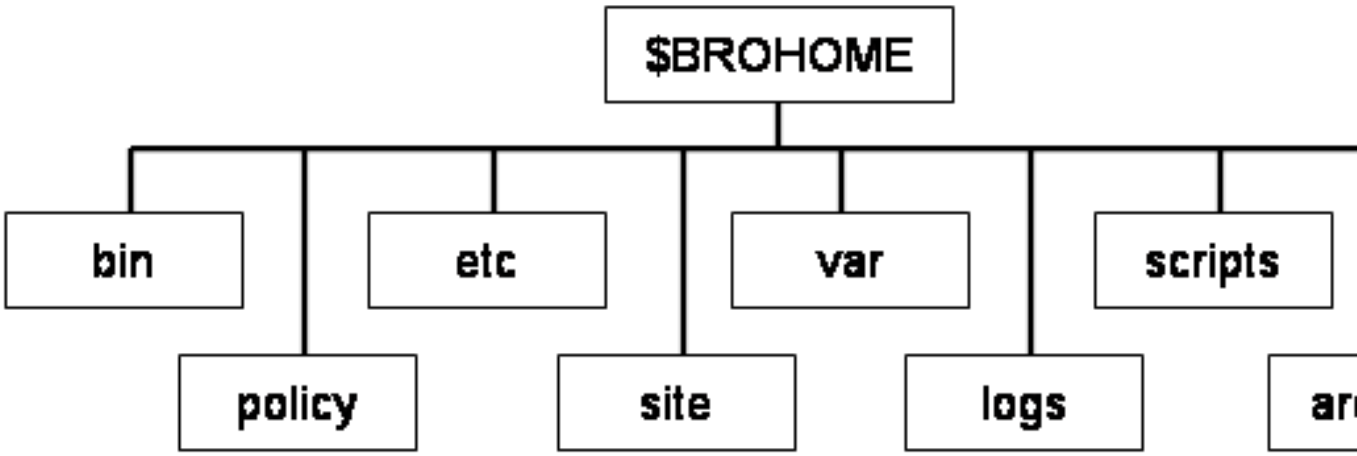

Figure A.1: The Bro Directory Structure

A.1 The bro/bin Directory

The bin directory is the storage area for executable binary files used by Bro.

adtrace

adtrace retrieves MAC and IP address information from tcpdump trace files

usage:

```
adtrace <trace-file>
```

bro

This program is the primary Bro executable. Full use of the bro command is documented in the technical manual.

cf

A program that converts UNIX epoch time into a conventional date. Most of the raw Bro logs record UNIX epoch time as the timestamp for their records. Piping the file through cf will convert the time. Full use of cf is documented in the technical manual.

rst

A program that Bro calls to form and send a reset packet which will tear down a tcp connection. The use of rst is documented in the Technical Manual and in chapter ### of the User Manual.

A.2 The bro/etc Directory

Configuration and other ancillary files are stored in the etc directory. These files are usually changed by supplimentary configuration tools supplied with the Bro distribution. Direct editing of these files is discouraged. If direct edits are made, the changes may be reversed or deleted during subsequent Bro updates.

alert_scores

This file contains ranking numbers for alarms (the use of the term "alert" is vestigial and will be changed in the future). The ranking numbers are used as part of the ranking system for determining the success likelihood of an incident triggering a specific alarm.

bro.cfg

This file contains configuration criteria for operational parameters. Most of the parameters are set during the installation process and can be changed using the bro- config script.

bro.cfg.example

A annotated, generic bro.cfg file. This file is not used by Bro. It is supplied for documentation purposes.

bro.rc

This is the script for controlled starting and stopping of Bro. See section ### for its use.

bro.rc-hooks.sh

This script is called by bro.rc at various points during the starting and stopping of Bro. It is presented as an interface for customizations into the start and stop process.

signature_scores

This file contains ranking numbers for signatures. The ranking numbers are used as part of the ranking system for determining the success likelyhood of an incident triggering a specific signature.

VERSION

A file containing the Bro version number for the installed distribution.

A.3 The bro/var Directory

Temporary information about the current Bro instance is stored in the var directory.

autorestart

Contains the word "ON" if Bro is configured to autorestart.

pid

Contains the process ID number for the current instance of Bro.

start_time

Contains the date and time when the current instance of Bro was started.

A.4 The bro/scripts Directory

This directory contains a number of auxiliary scripts used to suppliment Bro's operation.

bro-config

A utility script for changing the Bro operational parameters in the bro.cfg file.

bro-logchk.pl

Currently, this file does not work
A utility program for searching ftp and http log files for activity by specific ip addresses.

Usage:

```
bro-logchk.pl -[hrDFHds] -f filename -a ipaddr -x ipaddr
    -h          print this usage information
    -F          using ftp log
    -H          using http log
    -r          try to resolve IP addresses to hostnames
    -f file     log file to parse
    -a ipaddr   only output connections from this address
    -s          only want matching source address (used with -a )
    -d          only want matching dest address (used with -a )
    -D          debug option
    -x ipaddr   exclude connections from this address
```

bro_log_compress.sh

A very simple script written to manage log and coredump files. By default it compresses log files older than 30 days and sends them to the archive directory; it deletes log files older than 60 days; and it deletes coredump files older than 4 days.

Restrictions:

- Must be run from a user account that has read/write/execute access to files in the $BROHOME directory.

host-grep

Greps a Bro connection summary log on stdin for two given hostnames.

Usage:

```
host-grep [-a] hostname hostname < connection_log
If -a is specified then we only want lines with *all* of the listed hosts.
```

Restrictions:

- Must have $BROHOME/scripts included in the PATH environment variable.
- Will only work with hostnames. ip addresses are not accepted
- Uses host-to-addrs and ip-grep scripts

host-to-addrs

Finds all ip addresses associated with a given hostname.

Usage:

```
host-to-addrs hostname
```

Restrictions:

- Must have $BROHOME/scripts included in the PATH environment variable.
- Will only work with hostnames. IP addresses are not accepted

ip-grep

Returns an exact grep pattern for matching the IP addresses of the given hosts

Usage:

```
ip-grep hostname hostname ...
```

Restrictions:

- Must have $BROHOME/scripts included in the PATH environment variable.
- Will only work with hostnames. ip addresses are not accepted
- Uses host-to-addrs script

site-report.pl

This script produces the daily consolidated site report. By default, it is run daily via the cron job submitted by the bro user via files in /var/cron/tabs.

The bro/scripts/pm Directory

This directory contains perl modules to support the perl scripts in the scripts directory.

A.5 The bro/policy Directory

This directory contains all standard Bro policy files. For more information about the policy files see section ###, Policy

Signature support files:

sig-addendum.sig

This file contains small support utilities that are used in the implementation of Bro signatures.

sig-functions.bro

To be completed

sig-action.bro

To be completed

Policy files:

- active.bro
- alarm.bro
- analy.bro
- anon.bro

- backdoor.bro
- blaster.bro
- bro.bif.bro
- bro.init
- brolite.bro
- capture-events.bro
- checkpoint.bro
- common-rw.bif.bro
- conn-id.bro
- conn.bro
- const.bif.bro
- contents.bro
- cpu-adapt.bro
- demux.bro
- dns-info.bro
- dns-lookup.bro
- dns.bro
- drop-adapt.bro
- event.bif.bro
- finger-rw.bif.bro
- finger.bro
- flag-irc.bro
- flag-warez.bro
- frag.bro
- ftp-anonymizer.bro
- ftp-cmd-arg.bro
- ftp-reply-pattern.bro
- ftp-rw.bif.bro
- ftp-safe-words.bro
- ftp.bro
- gnutella.bro
- hand-over.bro
- hot-ids.bro
- hot.bro
- http-abstract.bro
- http-body.bro
- http-entity.bro
- http-event.bro
- http-header.bro

- http-reply.bro
- http-request.bro
- http-rewriter.bro
- http-rw.bif.bro
- http.bro
- icmp.bro
- ident-rewriter.bro
- ident-rw.bif.bro
- ident.bro
- inactivity.bro
- interconn.bro
- listen-clear.bro
- listen-ssl.bro
- load-level.bro
- login.bro
- mime.bro
- mt.bro
- netstats.bro
- notice.bro
- notice.bro.old
- ntp.bro
- pcap.bro
- pkt-profile.bro
- port-name.bro
- portmapper.bro
- print-filter.bro
- print-globals.bro
- print-resources.bro
- print-sig-states.bro
- profiling.bro
- reduce-memory.bro
- remote-pcap.bro
- remote-print.bro
- remote.bro
- scan.bro
- secondary-filter.bro
- signatures.bro
- signatures.bro.old
- site.bro

- smtp-relay.bro
- smtp-rewriter.bro
- smtp-rw.bif.bro
- smtp.bro
- snort.bro
- software.bro
- ssh-stepping.bro
- ssh.bro
- ssl-alerts.bro
- ssl-ciphers.bro
- ssl-errors.bro
- ssl-worm.bro
- ssl.bro
- stats.bro
- stepping.bro
- synflood.bro
- tcp.bro
- tftp.bro
- trw.bro
- udp.bro
- vlan.bro
- weird.bro
- worm.bro

A.6 The bro/site Directory

signatures.sig

To be completed

A.7 The bro/logs Directory

All logs take the form

```
type.hostname.start_date/time-end_date/time
```

The date/time stamps for each record in the files are always in UNIX (ticks since epoch) format.

type is one of the following:

alarm

Network occurrences that are determined to be of high importance will be written into the alarm file. The determination is made by the Bro policy scripts. Local site modifications can override default Bro alarms or create new ones that are site specific. Each entry contains the date/time, the alarm type, and a description of the alarm. This file is usually the "starting point" for investigation. Each alarm should be evaluated for further follow-up action.

conn

All network connections detected by Bro are recorded in this file. A connection is defined by an initial packet that attempts to set up a session and all subsequent packets that take part in the session. Initial packets that fail to set up a session are also recorded as connections and are tagged with a failure state that designates the reason for failure. Each entry contains the following data describing the connection: date/time, the duration of the connection, the local and remote ip addresses and ports, bytes transferred in each direction, the transport protocol (udp, tcp), the final state of the connection, and other information describing the connection. This file is often used in forensic analysis to determine network activity by a suspect host beyond the immediate alarm.

ftp

All transactions involving the well known ftp control port (21) are recorded into this file. Each entry is marked by an arbitrary session number, allowing full ftp control sessions to be reconstructed. Each entry contains the date/time, a session number, and ftp connection information or the specific ftp commands transferred. This file is often used to examine details of suspect ftp sessions.

http

All transaction involving the well known http ports (80, 8000, 8080) are recorded into this file. Each entry is marked by an arbitrary session number, allowing the full http session to reconstructed Each entry contains the date/time, a session number, and http connection information or the specific http commands transferred. This file is often used to examine details of suspect web sessions.

info

This file contains information concerning the operation of Bro during the time interval covered by the file. The entries will consist of the Bro version number, startup information, and Bro runtime warnings and errors. This file is helpful in troubleshooting Bro operational difficulties.

notice

Network occurrences that are determined to be of nominal importance will be written into the notice file. The determination is made by the Bro policy scripts. Local site modifications can override default Bro alarms or create new ones that are site specific. The notice files are similar to the alarm files, but of lesser importance. Each entry contains the date/time, a notice type, a notice action, the local and remote ip addresses and ports. Optionally, depending on the type of notice, an entry might contain information about user, filename,

method, URL, and other messages. This file alerts to occurrences that are worth noting, but do not warrant an alarm.

signatures

This file contains information associated with specific signature matches. These matches do not necessarily correspond to all alarms or notices, only to those that are triggered by a signature. Each entry contains the date/time, a description of the signature, the local and remote ip addresses and ports, the signature id number (if available), a description of the signature trigger, a portion of the offending payload data, a count of that particular signature, and a count of the number of involved hosts. This file gives details that are helpful in evaluating if an event triggered by a signature match is a false- positive.

smtp

All transactions involving the well known smtp port (25) are recorded into this file. Each entry is marked by an arbitrary session number, allowing full smtp sessions to be reconstructed. Each entry contains the date/time, a session number, and smtp connection information or the specific smtp commands transferred. This file is often used to examine details of suspect mail sessions.

software

This file is a record of all unique host/software pairs detected by Bro during the time interval covered by the file. Each entry in the file contains the date/time, the ip address of the host, and information about the software detected. This file can be useful for cataloging network software. However, population of this file on a busy network often results in a huge number of entries. Since the relative daily usefulness of the file usually does not warrant the disk space it consumes, the software file is turned off by default. It can be turned on by <<<instructions>>>

weird

Network events that are unusual or exceptional are recorded in this file. A number of these events "shouldn't" or even "can't" happen according to accepted protocol definitions, yet they do. Each entry in the file contains the date/time, the local and remote ip addresses and ports, and a short description of the weird activity. This file is useful for detecting odd behavior that might normally "fly under the radar" and also for getting a general sense of the amount of "garbage" that is on the network.

worm

Bro's worm.bro policy detects patterns generated by specific worms and records the instance in this file. Currently, the worms detected are code red1, code red2, nimda, and slammer. Each entry in the file contains the date/time, the worm detected, and the source ip address of the worm. This file is useful for spotting hosts that have been infected with worms.

Other files in the /logs directory are:

.state

To be completed

active_log

To be completed

A.8 The bro/archive Directory

The archive directory is initially empty. The script bro/script/bro_log_compress.sh populates the archive directory with compressed log files.

A.9 Other Files

Index

P

R

S

T

U

V

W