30 July 2010

TIPS AND NOTES FOR SETUP AND USE OF EQUILIBRIUM NETWORKS' NETWORK MONITORING SOFTWARE

Equilibrium's software was developed and internally tested using CentOS, though other flavors of Linux®—most particularly Scientific Linux and Red Hat® Enterprise Linux®—should provide similar environments from the point of view of system installation and initialization. Linux is recommended for all system components, though some of the system components, and particularly the AI, have run with at least partial functionality under other *nix and/or Windows® operating systems.

For the sake of simplicity and specificity these tips and notes assume starting with a clean default installation of 32-bit CentOS 5.4 and running as root, though this document should be of some help for setup and use on any OS.

With this in mind, **our current reference distribution is designed for easy setup of the Snort®-based sensor on the CentOS desktop.** It includes recent (but not necessarily the most recent) versions of components required for the sensor: e.g., libpcap v.1.0.0, PCRE v.8.01, Barnyard v.0.2.0, and Snort v.2.8.5.3 (note that the jNetPcap version provided for the AI is also not the most recent; this is due to a change in the jNetPcap API). The libnet component required for snort is most conveniently installed via yum, and this step is included in the installation scripts, which you may wish to modify.

Before installation

Read this document! Also, it will be necessary to unpack the Snort tarball provided in the SensorFiles directory, to move it, and to place additional files in the resulting directory. The details are covered in the README.txt file in the SensorFiles directory.[1]

EQ's primary software is provided in a NetBeans-friendly directory structure. The remainder of this document focuses in turn on overall system considerations, the AI, the EII, and the sensor.

Overall system

- The scripts install32.sh and install64.sh have been tested at certain points on multiple computers and architectures and should work without problems for the root user, provided that the installation system can access the internet (you will probably see compilation warnings in a terminal running this script, including for subsystem components not produced or modified by EQ). **For reasons outlined in the AI section, the use of a 32-bit operating system and the corresponding install/run scripts are highly recommended, as complete functionality will not**

[1] One aspect that is not covered there are the changes that have been made to Makefile.am and Makefile.in. The changes to Makefile.am basically amount to appending the relevant .c and .h filenames to the end of "libspp_a_SOURCES =" in both files and to append *filename*.$(OBJEXT) statements to the end of "libspp_a_OBJECTS =" in Makefile.in.

otherwise be available without substantial end user effort.[2] The script db_init.sh completes the system initialization by initializing the database and run32.sh or run64.sh starts the system. For full functionality, an appropriate version of jNetPcap (included) and Wireshark® should be installed (see the AI section); the former is installed by default. **To complete system startup, select "Restart Live Data" in the analytic interface.**

- **The fuser is still under development and is not presently included.** The fuser and MySQL® Proxy are intended to enable the consolidation of multiple input packet streams into a single output. As an example, DAG® hardware can multiplex a high-bandwidth link into multiple packet streams for analysis: the fuser is designed with situations like this in mind. Alternatively, multiple low-bandwidth links may be aggregated.

- **The default installation of the system places all of its subsystems** (Snort-based sensor, sensor-database bridge, database, analyzer, analytic and machine interfaces) **on a single machine and subsystem communications are set up to use the loopback interface.** These settings can be changed in order to distribute subsystems among multiple machines and are hoped to be self-explanatory.

- Because all of the subsystems other than the sensor are written in Java, they are expected to show a high degree of portability across platforms. (In particular, functionally complete versions of the analytic interface have successfully run under Windows and other physical and virtual machines, although in some development circumstances improved performance has resulted from disabling native look and feel.)

- **The subsystems have a significant number of options or adjustable parameters** which are passed as arguments on the command line (or the equivalent location in a run script) or through configuration files.

    - For the sensor, a sample snort.conf file might have the following arguments:

```
tree_file test.tree \
n_leaves 18 \ number of leaves should equal number in the tree file[3]
analyzer_port 998 \ communication with the analyzer
sdb_port 999 \ communication with the database
ip1 [10.10.32.0/8] \
ip2 [64.6.0.0/16,74.125.0.0/16] \
ip3 [209.85.0.0/16] \
…
```

[2] Note that 32-bit Linux can work with up to 64 GB of memory, although the memory allocated to a specific process is still limited to 4 GB (and usually 3 GB by default). However, 4 GB is more than adequate for the analytic interface, so this should not present any real difficulty. The other system components should also work properly with 32-bit memory limits. You may want to use a PAE kernel in this context.

[3] You might get away with `n_leaves` exceeding the number of leaves in the tree file, but this is not advised.

```
ip8 [] \
port1 [1024,1431] \ TCP and/or UDP ports
port2 [80,8008,8080] \
port3 [442:443] \
…
port8 [] \
src_ip_occasional_threshold 5 \ an IP threshold
src_ip_frequent_threshold 50 \
dst_ip_occasional_threshold 5 \
dst_ip_frequent_threshold 50 \
sp_occasional_threshold 2 \ a source port threshold
sp_frequent_threshold 20 \
dp_occasional_threshold 2 \ a destination port threshold
dp_frequent_threshold 20 \
ip_history_length 200 \ buffer length for IP thresholds
port_history_length 50 \ for port thresholds
pair_history_length 20 \ for socket pair thresholds
sample_history_length 100 \ for ID'ing "new" leaf pairs
skip 100 number of packets skipped[4] between database writes
```

- The sensor-database bridge[5] is started by a command like

  ```
  java –jar SDB.jar 127.0.0.1 999 1000 127.0.0.1/packets root ""
  ```

  contains the arguments for an IP, a port (see the snort.conf file), a batch number, database information (an IP and the name of the database), and username/password information, respectively (in this case no password is used). The batch number (here, 1000) determines the number of packets to buffer before writing to the database. Some experimentation to determine the optimal tradeoff between minimizing the number of writes and latency may result in improved performance.

  - For the analyzer, the parameters are passed in as command line arguments: these are, in order, the cycle time lower bound (i.e., the least amount of time between successive traffic pattern averaging samples), the IP and port for its communications with the sensor, and a port feed map governing which port a given feed is output on. The run scripts illustrate the syntax.
  - The parameters for the AI are located in the XML file ai-settings.cfg located in the appropriate dist folder. Portions of it can be changed from within the AI itself, for instance through the settings menu or through simply closing the AI (the open display configurations will be saved).

- Undersampling traffic may lead to null results for manual queries of ongoing source/destination pairs (i.e., "cube tubes") with low packet rates. One trivial but

---

[4] Excepting those with new leaf node pairs.

[5] A single initial message "An I/O error occurred when initializing the input stream!" can safely be ignored; the SDB constantly checks for an active connection, which will generally not be established immediately at startup.

potentially undesirable way to avoid this problem is to increase the sampling rate. Another is to account for frequencies in the tree structure.

- For sustained operation, it may be helpful to query every few hours to keep the database connection open. Also, you should keep track of the database size and consider aging older entries with a script.

AI

- In general, the system will be distributed without plugins. As of this writing the plugins include a CubeEngine (3D) display, a MatrixAlertPanel display (for generating ADMs automatically based on intermittent source/destination pair occurrences) and a RangeSlider for the time series displays. **Notes on plugin installation are provided in the Installation and Quickstart guide.**

- Although the system will operate on both 32- and 64-bit operating systems, **the ability to view hex packet headers, save and/or export packet capture files, or produce automatic ADMs using the matrix alert selector requires version 1.2 rc4 of the jNetPcap API, which is only supported (by a third party; see http://jnetpcap.com) for 32-bit operating systems.**[6] Nevertheless, both 32- and 64-bit installation and run scripts are included for CentOS.

- jNetPcap v.1.2 rc4 is included in the MiscPackages directory and is available from SourceForge as an RPM, which should auto-install on CentOS. Because of an API change, later release candidates or versions of jNetPcap are presently unsupported: hence **using yum to install jNetPcap for use with EQ software will not work** (unless you edit and recompile the EQ source to reflect the API changes, for which see footnote 6).

- If both Wireshark and jNetPcap are installed, packet tables can be exported to Wireshark from the AI. To install Wireshark on CentOS, execute yum install wireshark followed by yum install wireshark-gnome.

- **The AI, and especially the CubeEngine display, will perform sluggishly if the graphics card is not properly configured.** It is generally necessary to install drivers for a suitable graphics card.[7] The README document for NVIDIA® Linux drivers outlines driver installation and its style is accessible to new Linux users, although the brief instructions below may suffice for this particular class of graphics cards. (NB. When the driver is successfully installed and configured an NVIDIA splash screen should appear prior to the login screen.)

---

[6] A change in the jNetPcap API between 1.2 rc4 and 1.2 rc5 affects the AI; the relevant source files are PcapFileGenerator.java and SelectionListener.java. Running the AI in a 64-bit OS without modification will result in exceptions being thrown whenever (for example) a packet table is produced.

[7] Although high-end NVIDIA Quadro FX 3700 and 2700M cards were primarily used during development, a much more inexpensive Quadro FX 570 card performed at the same level. Based on experimental installations on a Macbook Pro and a consumer-grade Vista laptop, we expect that any reasonably capable graphics card will provide adequate performance, provided that it is properly configured.

The following steps have been tested as root on a clean default CentOS 5.3 install:

- The NVIDIA driver was saved to the desktop
- `init 3` was executed in a terminal to exit graphical mode
- `yum install gcc-c++` was executed
- `yum install kernel-PAE-devel.i686` (or another appropriate kernel package) was executed.
- the kernel source path `$KERNEL` was found and noted (this can be easily done by searching for kernel.h)
- the NVIDIA installer script was run similarly to instructions with the option `--kernel-source-path='$KERNEL'` added
- `init 5` was executed to return to graphical mode

- The displays are designed to be able to show a considerable time range; however, **for sufficiently large viewing ranges the displays may become unresponsive** and error messages may appear in the parent terminal of the AI. In practice the specific details of the displays suggest that this is only likely to be an issue for the time line displays, though an inadvertently large selection with a range slider might also lead to this in other displays. In such an event you should close the particular display and reopen it, taking care to use a smaller viewing window. (Also, please note that this is one reason why the lack of a global range slider is not a bad thing.) **The file viewer application allows you to view, save, and export replay files and can be used to enable visualization of very large time series via other programs.**

- **Over time the system displays may lag slightly or get out of sync with each other.** (The natures of the continuous [versus discrete] time representation and replay mechanisms make this a necessary tradeoff.) If you want to see current traffic it is always a good idea to click on the global "go to end" button first. Also, the time line displays may display artifacts over time, but the "go to end" button will refresh them.

- **Replay files are set up to cover four hours before a new replay file is created.** This means that to go back more than four hours in time (and usually less, depending on the last time a replay file was completed) requires you to open a replay file. However, this can be done simply via the file menu (look for the replay directory). For determining which file to open, the creation dates of the files will probably be more helpful in than their names. The file extensions .f#s# indicate feed type (at present only one feed is provided) and set number (corresponding to particular time intervals). As other feed types become available it may be more useful to deal with the .rpl files instead.

EII

- At startup the EII console will show the message "Error connecting to TA: connect timed out." This will not affect the system operation: it is retained for anticipated functionality in which the EII can apply simple thresholds on more sophisticated analyzer feeds in order to generate ADMs.

Sensor

- The single most important thing to keep in mind about installing and configuring the sensor is that it is an extension of Snort. Among other things, this means that there is a great deal of latent functionality and configurability in the sensor beyond that offered by the EQ preprocessor. Some familiarity with Snort and its use of preprocessors can be of substantial benefit to users seeking to fully exploit the software.

- **The sensor can very probably be accelerated significantly** through the use of specialized tools such as the PF_RING kernel patch (as of this writing available at http://www.ntop.org/PF_RING.html) or Phil Wood's version of libpcap (as of this writing available at http://public.lanl.gov/cpw/), or by employing hardware such as that offered by Endace. These methods have not been tested and are not supported, but are likely to meet with success under appropriate circumstances. If you do use any of these techniques to successful advantage with our system, please feel free to contact us with details.

  We have taken some pains to optimize the sensor performance and have processed well over 60K 1500-byte packets per second (corresponding to 720+ Mbps) in sustained operation with Snort 2.8.4.1, and burst rates of over 75K 1500-byte packets per second on a workstation with two quad-core 2.83 GHz Xeon processors (though only a single core can be used for a sensor instance) running in graphical mode (this single machine ran the entire framework).[8] Because our tests used an earlier version of Snort and tcpreplay with the `rdtsc` option from another workstation rather than (e.g.) a DAG, it is possible and even likely that (this was the rate limiting factor and) you will be able to process more packets per second on an active link. **Your mileage may also vary according to the sensor parameters, especially the lengths of histories and the sampling parameters for the database.**

  Reports of sensor saturation and (even rough) benchmarks are most welcome. In the event that your instance of the sensor is able to keep up with a gigabit of traffic at link saturation and you would like to try it on a faster link, please give us brief details along with your request for a modified license to permit such use.

---

[8] Using minimum-length packets instead, we obtained much noisier results as the number of packets per second increased. This was likely due (more) to shortcomings from transmitting test data with tcpreplay rather than with the sensor. Still, we achieved burst rates of over 175K packets per second sustained over intervals of 20+ seconds, and moving averages in the same neighborhood over intervals of several minutes (we did not run prolonged tests of this sort due to the noisy results).

- **Default trees are included for TCP/UDP and ICMP traffic.** They are designed more to illustrate the tree format and syntax than to provide an "out-of-the-box" configuration. They correspond to the following figures.

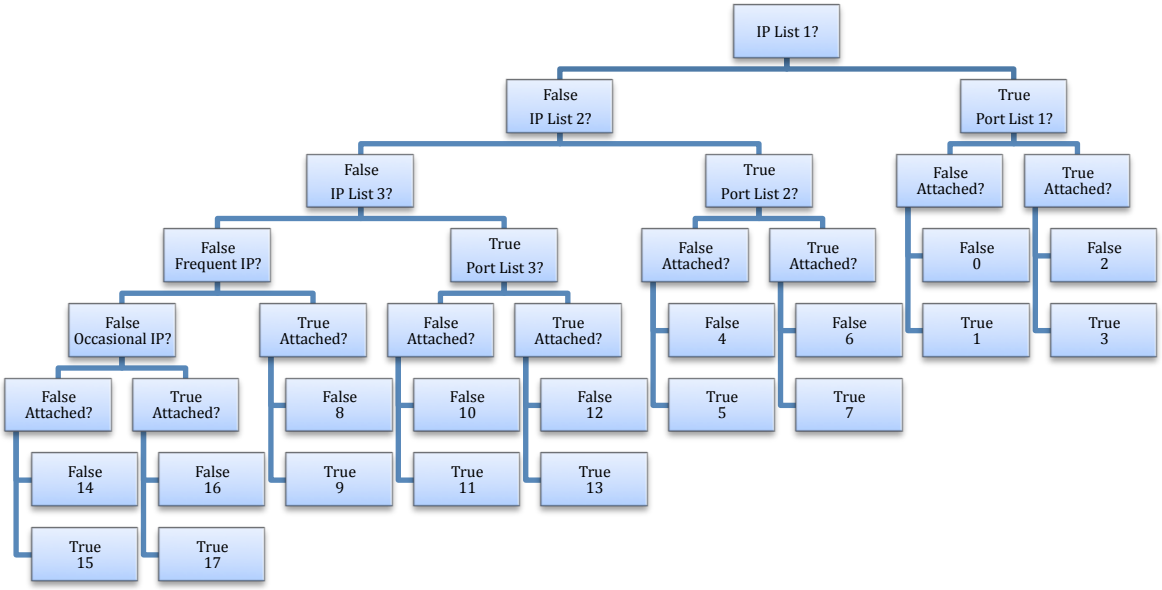

Figure 1. Diagram for `test.tree`.

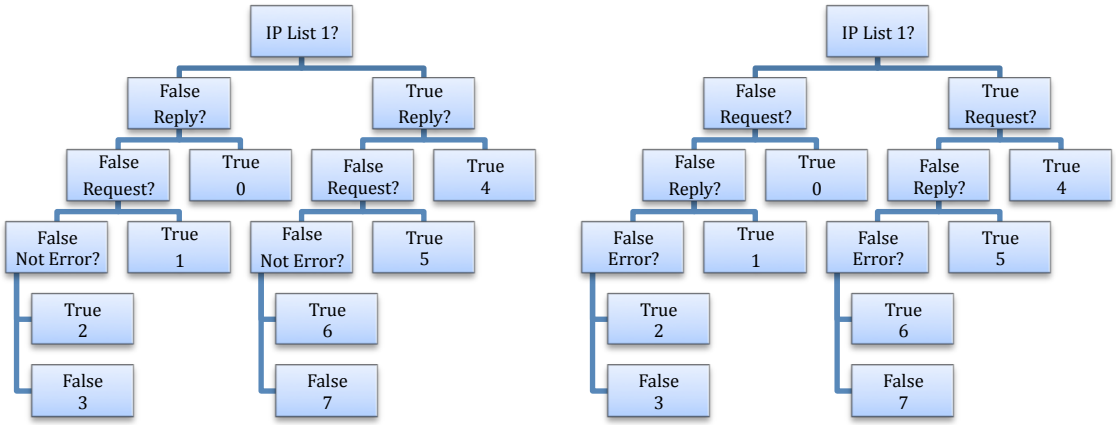

Figure 2. Diagrams for `icmp.tree`. Left: source tree; right: destination tree. ICMP types 8, 10, 13, 15, and 17 correspond to requests; 0, 9, 14, 16, and 18 correspond to replies, and types 3, 4, 5, 11, and 12 correspond to errors.

Additional tree node questions that are supported are isTCP, isUDP, isICMP, isFrequentPort and isOccasionalPort. Even taking the first three of these into account, you should try to use BPF filters whenever appropriate with a given tree.

For more examples of tree files, see our poster presented at VizSec '09, available from our downloads page. For more background on trees, see our whitepaper "Scalable visual traffic analysis" on the same page.

- Known or potential issues include
  - **Tree filenames.** In the unlikely event that you experience problems using long filenames for trees, a filename such as "a.tree" should work.
  - **Sensor connections are limited and cannot be reestablished. For example, the sensor cannot simultaneously log packets and connect to the database.**
  - **Snort does not provide statistics upon exit.**
  - **IPv6 is not handled, and the request/reply/error/other types for ICMPv4 only take types 0-18 into account (though this should be relatively straightforward to change if you so desire). Also, two "new" bit flags for TCP are not accounted for.**

Visit our website at http://www.eqnets.com/ for more information.