Snorby, OpenFPC, and Pulled Pork bring Snort back to the table

Fresh Pork

Snort is the de facto standard for open source network intrusion detection. The developer community has kept a fairly low profile for a couple of years, but extensions like Snorby, OpenFPC, and Pulled Pork have given the old hog a new lease on life. *By Ralf Spenneberg*

Snort is old – on an IT timescale, even ancient. Marty Roesch started developing the network sniffer [1] back in 1998. His original plan was "just" to program a network sniffer that would run on a variety of operating systems. The initial version, released back in 1998, comprised just 1,200 lines of code, but one of the most powerful network IDS engines of all time arose from these humble beginnings. In 2001, Roesch founded Sourcefire [2], a company that is today synonymous with successful network intrusion prevention appliances based on Snort. Sourcefire continues to develop Snort as a way of giving back to the open source community.

Snort itself is just an engine that analyzes and standardizes network traffic and then refers to signatures to identify suspicious activity. Snort doesn't provide tools for signature management, storing and analyzing messages in a database, or forensic logging. In the past, administrators used tools like Oinkmaster [3] or BASE [4] for processing Snort information; however, these projects are orphaned or dormant. Fortunately, some new projects have stepped into the gap: Snorby [5], OpenFPC [6], and Pulled Pork [7]. In this article, I take you on a tour of the latest generation of Snort support projects.

Sniffer

Version 2.9 saw the Snort developers introduce the new Data Acquisition Library (DAQ), which replaces the classical libpcap. As of version 2.9, Snort depends on the DAQ library, and you can't install the sniffer without it (see the "DAQ" box).

The current version of Snort is 2.9.0.3. In contrast to the previous version, the developers simply fixed a couple of bugs. Before you build Snort, make sure you have the DAQ, Dnet, and PCRE libraries installed on your machine. Table 1 discusses possible modifications to the `./configure` script for Snort. Even though various database configurations are possible, you will probably want to avoid them; I'll show you how to use Barnyard [12] as the resident database.

The Snort source code archive provides a start script for Red Hat- and Fedora-based distributions. You might need to modify the script to suit your own installation – or

THE AUTHOR

Ralf Spenneberg is a freelance Unix/Linux trainer, consultant, and author and the CEO of Ralf Spenneberg Open Source Training. Ralf has published several books on the subjects of intrusion detection, SELinux, firewalling, and virtual private networks. The second edition of his latest book *VPN on Linux* was published a few months ago.

even write your own start script. The important thing to remember is that you need to launch Snort via `/usr/local/bin/snort -c /etc/snort/snort.conf`. Before starting Snort for the first time, copy the contents of the `etc` directory in the source code to `/etc/snort/` and create a `/var/log/snort` directory. Information in the "`snort.conf`" box will help you modify the `snort.conf` configuration file.

Signatures

Of course, Snort now needs signatures that will provide a roadmap for intrusion testing. There are basically three signature sources:

- The VRT Ruleset from the Sourcefire Vulnerability Research Team
- The Emerging Threats Rules
- The GPL Ruleset

You need to keep all these rules up to date on an ongoing basis; I will be using Pulled Pork for updates later in this article. If you are aiming for production deployment of Snort, you will probably want to use either the VRT or Emerging Threats ruleset. Both are available in commercial and free versions. The free VRT ruleset simply becomes available 30 days later than its commercial counterpart, but it gives you access to the whole ruleset. The commercial edition of the Emerging Threats ruleset includes additional rules that are not provided with the free version.

The following example uses the VRT rules, which you can either download as part of a commercial subscription or obtain for free with a delay of 30 days (after registering). After you subscribe or register, Sourcefire sends you an "Oink Code," which you need to pass in with the download request:

```
wget http://www.snort.org/reg-rules/snort ⮐
  rules-snapshot-2903.tar.gz/⮐
  <oinkcode here> ⮐
  -O snortrules-snapshot-2903.tar.gz
```

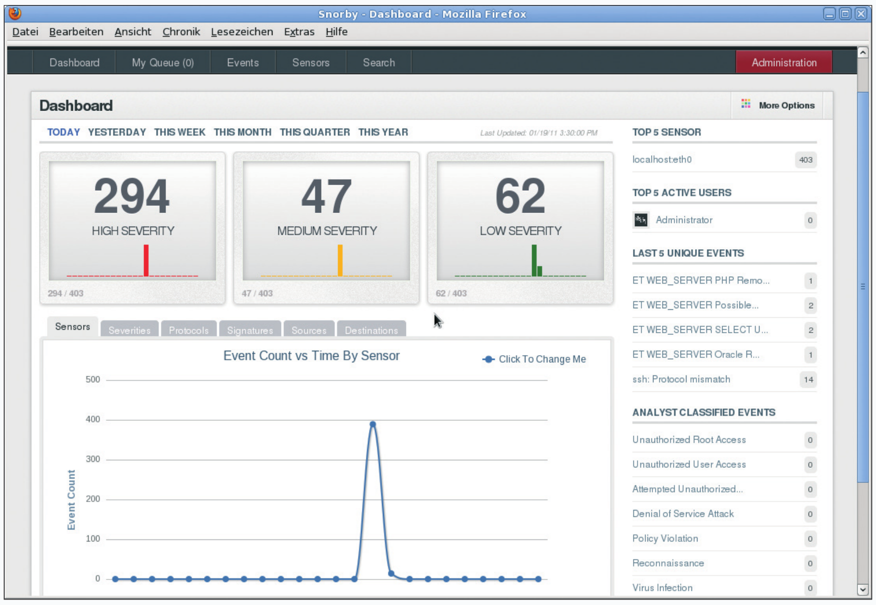
The VRT ruleset contains four subdirectories: `/etc` contains the configuration files that Snort and Barnyard require for this ruleset. You need to copy them to the `/etc/snort` directory. `rules`, which contains the text-based rules for Snort, needs to be stored in `/etc/snort/rules`; `preproc_rules` gives you the preprocessor rules – just copy the directory to `/etc/snort/preproc_rules`.

Finally, the binary rules for Snort are available in `so_rules`. You need to copy the subdirectory for your distribution (e.g., `so_rules/precompiled/Ubuntu-10-4/x86-64/2.9.0.3/*`) to `/usr/local/lib/snort_dynamicrules` and the `*.rules` files to the `/etc/snort/so_rules/` directory.

The Sourcefire Vulnerability Research Team offers some rules in binary format only. There are two reasons for the binary format: for one thing, the Snort rule language is a very powerful language but not capable of identifying suspicious packages in some attacks. The binary libraries considerably improve the detection rate. The second reason is that a third party might have disclosed a vulnerability to Sourcefire subject to a non-disclosure agreement (NDA). If the VRT team were to distribute the signature in the clear, they would disclose details without prior authorization.

Updates with Pulled Pork

Most users rely on Oinkmaster to update their rulesets. Because development work on this tool is dormant, Pulled Pork now handles the task in many environments. Pulled Pork also processes binary rules and organizes things in the right directories. To install, you just need to download the package from Google Code. The current version of Pulled Pork (0.5.0) requires Perl and the Perl Archive::Tar, Crypt::SSLeay, and LWP::Simple modules, which you can install on most distributions via the package manager. To install Pulled Pork, copy the `/etc` directory to `/etc/pulledpork` and the Perl `pulledpork.pl` script to `/usr/local/bin`. The `pulledpork.conf` configuration file is also self-explanatory. The only important tasks are to modify the rule URL with the Oink code, the paths for the rules to be installed, and the shared object rules, depending on your distribution. Pulled Pork can modify rules during the install. To do so, it references four additional configuration files, which you can enable:



Dashboard

Last Updated: 01/19/11 3:30:00 PM

294 HIGH SEVERITY 294 / 403

47 MEDIUM SEVERITY 47 / 403

62 LOW SEVERITY 62 / 403

TOP 5 SENSOR	
localhost:eth0	403

TOP 5 ACTIVE USERS	
Administrator	0

LAST 5 UNIQUE EVENTS	
ET WEB_SERVER PHP Remo...	1
ET WEB_SERVER Possible...	2
ET WEB_SERVER SELECT U...	2
ET WEB_SERVER Oracle R...	1
ssh: Protocol mismatch	14

ANALYST CLASSIFIED EVENTS	
Unauthorized Root Access	0
Unauthorized User Access	0
Attempted Unauthorized...	0
Denial of Service Attack	0
Policy Violation	0
Reconnaissance	0
Virus Infection	0

Figure 1: **The Snorby dashboard sorts messages by severity.**

- `dropsid.conf`: Pulled Pork changes all the rules in this file to tell Snort to drop the packets.
- `enablesid.conf` and `disablesid.conf`: Pulled Pork enables and disables the rules in these files.
- `modifysid.conf`: Pulled Pork modifies the rules in the specified files.

LISTING 1: A Host Attribute Table

```
<SNORT_ATTRIBUTES>
<ATTRIBUTE_TABLE>
  <HOST>
      <IP>10.1.2.3</IP>
      <OPERATING_SYSTEM>
          <NAME>
              <ATTRIBUTE_VALUE>Windows</ATTRIBUTE_VALUE>
          </NAME>
          <FRAG_POLICY>Windows</FRAG_POLICY>
          <STREAM_POLICY>Win XP</STREAM_POLICY>
      </OPERATING_SYSTEM>
      <SERVICES>
          <SERVICE>
              <PORT>
                  <ATTRIBUTE_VALUE>80</ATTRIBUTE_VALUE>
              </PORT>
              <IPPROTO>
                  <ATTRIBUTE_VALUE>tcp</ATTRIBUTE_VALUE>
              </IPPROTO>
              <PROTOCOL>
                  <ATTRIBUTE_VALUE>http</ATTRIBUTE_VALUE>
              </PROTOCOL>
          </SERVICE>
      </SERVICES>
  </HOST>
</ATTRIBUTE_TABLE>
```

The program is launched by typing:

```
pulledpork.pl ⮧
  -c /etc/pulledpork/pulledpork.conf
```

Before downloading the rules, Pulled Pork uses checksums to discover whether the rules on the server are newer. A daily cron job does the rest.

Barnyard

Once you have installed Snort in line with the specs, Snort will log messages in a binary file in Unified2 format. You can't read this file directly and thus need another tool to store the data in a database. Barnyard2 is the tool of choice. The tool assumes the time-consuming process of logging in the database, thus reducing the load on the server. The Barnyard source code archive is available from the SecurixLive website [12]; installing from the source code is not typically a problem.

Barnyard needs to write files in `/var/log/snort` and write the file content to the database. Snort stores files in this directory with a pattern of `merged.log.`*`XXXXXXXX`*. *`XXXXXXX`* is a timestamp Barnyard references to help it ignore old files. If the timestamp is missing, you should remove the `nostamp` option in the configuration file for the Unified output plugin. Because the Unified logfile only contains binary information, Barnyard needs access to other Snort configuration files to translate the file, so enter the paths to these files in the `/usr/local/etc/barnyard2.conf` configuration file and modify the remaining settings:

```
config utc
config daemon
config logdir: /tmp
config waldo_file: /var/log/snort/waldo
#output alert_fast: stdout
output database: log, mysql, user=snort ⮧
                           password=snortpw ⮧
                           dbname=snortdb ⮧
                           host=localhost
```

INSTA-SNORBY

Insta-Snorby [8] is based on the TurnKey Linux Virtual Appliance Library [9], which provides virtual machines for a variety of purposes. Based on Ubuntu Linux, it lets administrators test various applications simply, without having to install.

Insta-Snorby is an ISO image; version 0.6.0 supports uncomplicated installation on an appliance. It contains:

- Snort 2.9.0.3
- Barnyard 2.19
- Snorby 2.2.1
- OpenFPC
- Pulled Pork

To install as a virtual appliance, you will need to give Insta-Snorby at least 512MB of RAM to avoid swapping.

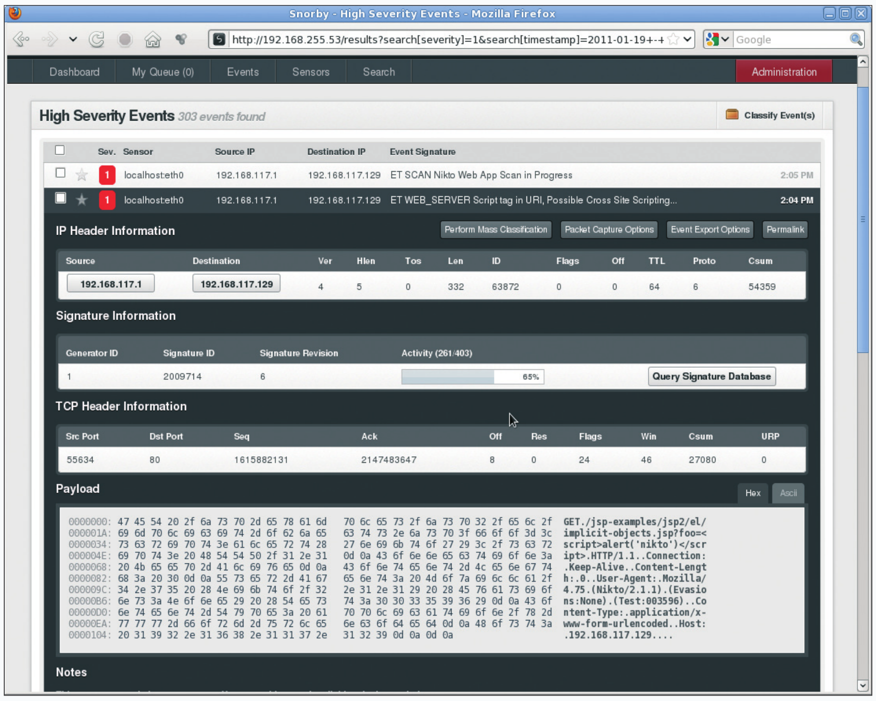
Before you launch Barnyard2, you need to create the database:

```
# cat << EOF | mysql -u root -p
create database snortdb;
grant all on snortdb.* to snort@localhost
   identified by "snortpw";
EOF
# mysql -u snort ⮧
        -password=snortpw snortdb < ⮧
        schemas/create_mysql
```

The following line in the start script automates the start process:

```
barnyard2 -d /var/log/snort/ ⮧
   -f merged.log ⮧
   -c /usr/local/etc/barnyard2.conf ⮧
   -n
```

The -n tells Barnyard2 to process only new messages appended to the Snort logfile. To ignore old messages when restarted, it stores a bookmark in a Waldo file [13].



High Severity Events 303 events found

Sev.	Sensor	Source IP	Destination IP	Event Signature	
1	localhost:eth0	192.168.117.1	192.168.117.129	ET SCAN Nikto Web App Scan in Progress	2:05 PM
1	localhost:eth0	192.168.117.1	192.168.117.129	ET WEB_SERVER Script tag in URI, Possible Cross Site Scripting...	2:04 PM

IP Header Information

Source	Destination	Ver	Hlen	Tos	Len	ID	Flags	Off	TTL	Proto	Csum
192.168.117.1	192.168.117.129	4	5	0	332	63872	0	0	64	6	54359

Signature Information

Generator ID	Signature ID	Signature Revision	Activity (261/403)
1	2009714	6	65%

TCP Header Information

Src Port	Dst Port	Seq	Ack	Off	Res	Flags	Win	Csum	URP
55634	80	1615882131	2147483647	8	0	24	46	27080	0

Payload

```
0000000: 47 45 54 20 2f 6a 73 70 2d 65 78 61 6d  70 6c 65 73 2f 6a 73 70 32 2f 65 6c 2f  GET./jsp-examples/jsp2/el/
000001A: 69 6d 70 6c 69 63 69 74 2d 6f 62 6a 65  63 74 73 2e 6a 73 70 3f 66 6f 6f 3d 3c  implicit-objects.jsp?foo=<
0000034: 73 63 72 69 70 74 3e 61 6c 65 72 74 28  27 6e 69 6b 74 6f 27 29 3c 2f 73 63 72  script>alert('nikto')</scr
000004E: 69 70 74 3e 20 48 54 54 50 2f 31 2e 31  0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a  ipt>.HTTP/1.1..Connection:
0000068: 20 4b 65 65 70 2d 41 6c 69 76 65 0d 0a  43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74  .Keep-Alive..Content-Lengt
0000082: 68 3a 20 30 0d 0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f  h:.0..User-Agent:.Mozilla/
000009C: 34 2e 37 35 20 28 4e 69 6b 74 6f 2f 32  2e 31 2e 31 29 20 28 45 76 61 73 69 6f  4.75.(Nikto/2.1.1).(Evasio
00000B6: 6e 73 3a 4e 6f 6e 65 29 20 28 54 65 73  74 3a 30 30 33 35 39 36 29 0d 0a 43 6f  ns:None).(Test:003596)..Co
00000D0: 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61  70 70 6c 69 63 61 74 69 6f 6e 2f 78 2d  ntent-Type:.application/x-
00000EA: 77 77 77 2d 66 6f 72 6d 2d 75 72 6c 65  6e 63 6f 64 65 64 0d 0a 48 6f 73 74 3a  www-form-urlencoded..Host:
0000104: 20 31 39 32 2e 31 36 38 2e 31 31 37 2e  31 32 39 0d 0a 0d 0a                    .192.168.117.129....
```

Notes

Figure 2: **In Snorby's web interface, administrators can classify results, add comments, and manage sensors and users.**

Snorby

Snorby, which serves as a Snort front end, is only officially available via its own Git repository [14], but you will occasionally stumble across various tarballs. Before you can install Snorby, you need to fulfill a number of dependencies: Ruby (>=1.8.7), RubyGems, and the following gems: `tzinfo`, `builder`, `memcache-client`, `rack`, `rack-test`, `erubis`, `mail`, `text-format`, `bundler`, `thor`, `i18n`, `sqlite3-ruby`, and `rack-mount rails=3.0.0`. The Ruby requirements in particular mean either you need a recent distribution, or you are prepared to install these packages from the source code. Ubuntu as of Karmic and Fedora 14 include Ruby version 1.8.7; neither of these distributions includes Rails version 3.

After unpacking, change to the Snorby directory and type `bundle install`; this step installs a couple of additional packages. `rake snorby:setup` then finally installs Snorby. There are a couple of mainly self-explanatory configuration files – `snorby/config/snorby_config.yml` and `snorby/config/initializers/mail_config.rb` – that you need to modify, but then you can launch Snorby by typing `rails -c` to pop up the login window.

After logging in, you will see the Snorby Dashboard, a state-of-the-art GUI for incoming messages (Figure 1) that lets you view each event in detail (Figure 2).

Forensic Analysis

Snort only detects potential attacks on the network and logs any suspicious packets that trigger an alert. This is often too late because Snort can't turn back the clock and log the packets transmitted previously. You can only modify the signatures to log the subsequent packets from the same connection. This complication makes it nearly impossible to analyze a successful attack at the network level.

TABLE 1: Configuration Options

Option	Function
`--enable-ipv6`	Enables IPv6 support. Even if you don't actually want to use IPv6, you should enable this function. This is the only way to get Snort to support the `ipvar` parameter used by the version 2.9 configuration files.
`--enable-static-daq`	Integrates a static DAQ library, which makes it easier to distribute the binary.
`--enable-zlib`	Puts Snort in a position to analyze the web pages that many servers compress when they serve them up. However, it does mean adding the Zlib library to your system.
`--enable-targetbased`	Enables a host-specific preprocessor configuration, as described in this article.
`--enable-decoder-preprocessor-rules`	Makes it possible to enable and disable individual preprocessor tests individually.
`--enable-reload`	Supports a simple reload in case of Snort configuration changes. Restarting will always entail some packet loss.
`--enable-reload-error-restart`	Restarts if the reload fails.
`--enable-normalizer`	Enables the normalizer code in inline mode. A Snort binary can work in inline mode (IPS) or passive mode (IDS) if you use the DAQ library.
`--enable-inline-init-failopen`	Allows Snort to let packets pass without checking during inline mode initialization.

The recent Open Full Packet Capture project (OpenFPC) aims to close this gap. OpenFPC is the brainchild of Sourcefire staff, just like Pulled Pork; the version number is still fairly low (0.4), and you do need to install several dependencies first. You will probably find the installation easiest on Debian-based distributions like Ubuntu, as in the following:

```
aptitude install apache2 daemonlogger tcpdump tshark ⮒
   libarchive-zip-perl libfilesys-df-perl libapache2-mod-php5 ⮒
   mysql-server php5-mysql libdatetime-perl libdbi-perl ⮒
   libdate-simple-perl php5-mysql libterm-readkey-perl ⮒
   libdate-simple-perl
```

After unpacking the OpenFPC source code, run the `openfpc-install.sh install` installation script and respond to the prompts (e.g., for the username and password). The `/etc/openfpc/openfpc-default.conf` file is for any other settings. You can then start the server and check its availability from the client:

```
openfpc --action start
openfpc-client --action status
```

To start the program at boot time, you will find start scripts in `/etc/init.d`. To integrate with Snorby, you just need to make a minor change via the web GUI. In the administrative interface, enable the OpenFPC plugin and enter the path to OpenFPC: *https://Open-FPC-Server_name/openfpc/cgi-bin/extract.cgi*. You can then download a PCAP file of the whole connection directly from the alert in Snorby (Figure 3) and then send it to Wireshark for analysis.

Security Takes Priority

For reasons of security, you should never run Snort with root privileges; it makes much more sense to create a dedicated account for Snort and add the account name and group to the Snort configuration file. In addition to this, you can lock Snort away in a chroot jail; in the `snort.conf` file, this looks like the following:

```
config set_uid: snort
config set_gid: snort
config chroot: /chroot/snort
```

Of course, you need to modify some permissions and directory structures to avoid Snort tripping over its own feet.

Snort also offers some advanced methods, such as target-based traffic reconstruction. Researchers Vern Paxson and Umesh Shankar discovered in 2003 that different operating systems handle IP defragmentation and TCP reassembly in different ways. This allows an attacker to disguise an attack so that it successfully compromises the target but is not detected by the IDS.

To allow this to happen, the attacker would fragment the exploit. The attacker creates the fragment with the attack payload twice. One fragment contains the payload and another one contains harmless data. If both are sent in succession, the success or failure of the attack, as well as the IDS's ability to detect it, depends on how the target systems respond. Do they prefer the first or second fragment for defragmentation? By carefully setting the parameters, the attacker can run the exploit without the IDS detecting it.

There are similar differences with TCP segments and overlapping fragments and segments. To make sure that Snort always detects these attacks correctly, it needs to defragment and reassemble the packets in the same way the target system would.

Snort has a couple of tricks of its own in response to these advanced attack methods: It supports `first`, `last`, `bsd`, `bsd-right`, `linux`, `windows`, and `solaris` defragmentation modes,

SNORT.CONF

Most defaults in the `snort.conf` file can stay as they are, but you will need to make a few changes. The DAQ configuration uses the AF packet interface and provides a 256MB buffer for caching any packets it has not processed. The `HOME_NET` variable defines the network to protect. The `RULE_PATH`, `SO_RULE_PATH`, and `PREPROC_RULE_PATH` variables point to the right directories. The `unified2` output plugin creates the logfiles:

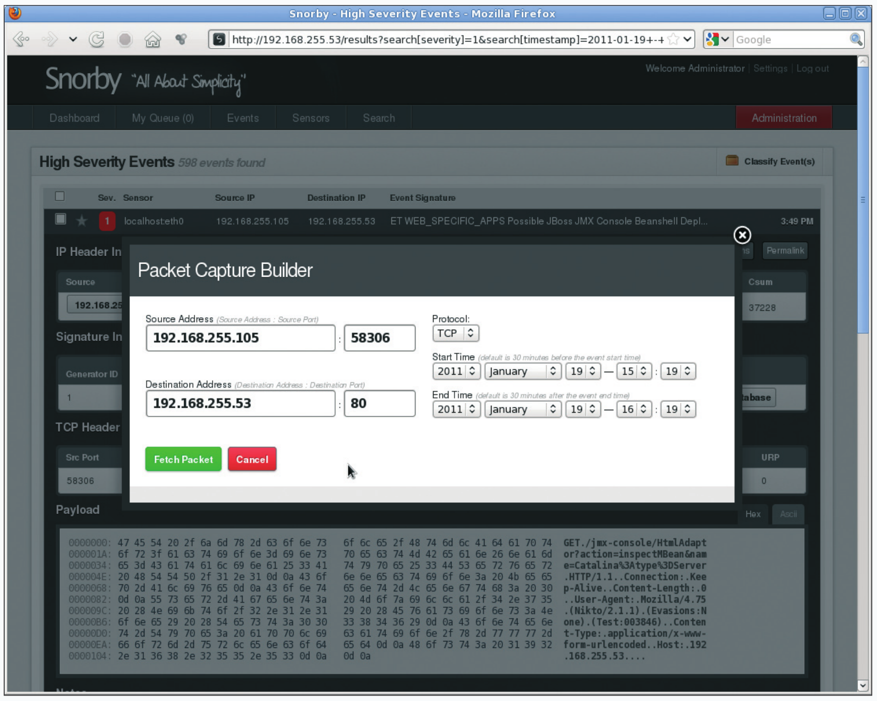
```
config daemon
config interface: eth0
config daq: afpacket
config daq_mode: passive
config daq_var: buffer_size_mb=256
ipvar HOME_NET 192.168.0.0/24
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
output unified2: filename merged.log, limit 128,
                 vlan_event_types
```

To use preprocessor rules and shared object rules, you just need to remove the pound signs in front of the include directives in the corresponding sections.

DAQ

One of the major issues with running Snort on a fast network is Snort's performance. If Snort takes too long to analyze the packets and fails to pick up packets quickly enough from the libpcap library [10], the library drops the packets. Thus far, administrators have used two different approaches to handling this problem: Phil Wood added a memory-mapped ring buffer to Libpcap; the buffer size was specified when loading the library. As of libpcap version 1.0, the ring buffer is built into the library. Another approach is the PF_ring library [11], which replaces the libpcap library.

Snort version 2.9 introduces the Data Acquisition Library (DAQ) as a new and much simpler alternative, thus accessing the network interface card directly without requiring libpcap. DAQ is easy to install; however, prebuilt packages with the right settings are not available for many distributions. In fact, the Snort homepage only offers RPM packages. After downloading the latest 0.5 version of the source code, just follow the standard procedure: `./configure`; `make`; `sudo make install`. There aren't many dependencies to resolve. To make sure the build works, you will probably want to install a C compiler, Flex, Bison, and libpcap >= v1.0. If you are not using libpcap but accessing the NICs directly via the DAQ library, you can disable libpcap with `--disable-pcap-module`.


Snorby - High Severity Events - Mozilla Firefox

http://192.168.255.53/results?search[severity]=1&search[timestamp]=2011-01-19+-+

Snorby "All About Simplicity"

Welcome Administrator | Settings | Log out

High Severity Events *598 events found*

Sev.	Sensor	Source IP	Destination IP	Event Signature	
1	localhost:eth0	192.168.255.105	192.168.255.53	ET WEB_SPECIFIC_APPS Possible JBoss JMX Console Beanshell Depl...	3:49 PM

Packet Capture Builder

Source Address *(Source Address : Source Port)*: 192.168.255.105 : 58306

Destination Address *(Destination Address : Destination Port)*: 192.168.255.53 : 80

Protocol: TCP

Start Time *(default is 30 minutes before the event start time)*: 2011 January 19 — 15 : 19

End Time *(default is 30 minutes after the event end time)*: 2011 January 19 — 16 : 19

```
0000000: 47 45 54 20 2f 6a 6d 78 2d 63 6f 6e 73  6f 6c 65 2f 48 74 6d 6c 41 64 61 70 74  GET./jmx-console/HtmlAdapt
000001A: 6f 72 3f 61 63 74 69 6f 6e 3d 69 6e 73  70 65 63 74 4d 42 65 61 6e 26 6e 61 6d  or?action=inspectMBean&nam
0000034: 65 3d 43 61 74 61 6c 69 6e 61 25 33 41  74 79 70 65 25 33 44 53 65 72 76 65 72  e=Catalina%3Atype%3DServer
000004E: 20 48 54 54 50 2f 31 2e 31 0d 0a 43 6f  6e 6e 65 63 74 69 6f 6e 3a 20 4b 65 65  .HTTP/1.1..Connection:.Kee
0000068: 70 2d 41 6c 69 76 65 0d 0a 43 6f 6e 74  65 6e 74 2d 4c 65 6e 67 74 68 3a 20 30  p-Alive..Content-Length:.0
0000082: 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a  20 4d 6f 7a 69 6c 6c 61 2f 34 2e 37 35  ..User-Agent:.Mozilla/4.75
000009C: 20 28 4e 69 6b 74 6f 2f 32 2e 31 2e 31  29 20 28 45 76 61 73 69 6f 6e 73 3a 4e  .(Nikto/2.1.1).(Evasions:N
00000B6: 6f 6e 65 29 20 28 54 65 73 74 3a 30 30  33 38 34 36 29 0d 0a 43 6f 6e 74 65 6e  one).(Test:003846)..Conten
00000D0: 74 2d 54 79 70 65 3a 20 61 70 70 6c 69  63 61 74 69 6f 6e 2f 78 2d 77 77 77 2d  t-Type:.application/x-www-
00000EA: 66 6f 72 6d 2d 75 72 6c 65 6e 63 6f 64  65 64 0d 0a 48 6f 73 74 3a 20 31 39 32  form-urlencoded..Host:.192
0000104: 2e 31 36 38 2e 32 35 35 2e 35 33 0d 0a  0d 0a                                    .168.255.53....
```

Figure 3: If you require a PCAP file, you can precisely define which packets you want OpenFPC to extract.

with `bsd` as the default. The more complex reassembly methods are shown in Table 2.

If you only use one or two operating systems, the configuration is fairly simple: You just edit the preprocessor section of `snort.conf`:

```
preprocessor frag3_engine: policy linux bind_to ↩
   [10.1.1.12/32,10.1.1.13/32] detect_anomalies
preprocessor frag3_engine: policy windows bind_to ↩
   10.2.1.0/24 detect_anomalies
```

This configuration is fine for a Windows network with two Linux computers. Similarly, you would need to modify the Stream5 preprocessor. For more details, read the `README.stream5` in the Snort source code documentation directory.

If you have many different systems on your LAN, this kind of management is too complex. In that case, you need a host attribute table to define the information in an XML file (Listing 1). The table is defined in `snort.conf` as `attribute_table` *`filename /path/to/file`*. The host attribute table contains information on every single computer. You can use Hogger [15] to create the XML file directly from an Nmap scan.

Run, Piggy, Run!

With an intelligent combination of the right components, you can create a very powerful network intrusion detection system with Snort. Of course, you need to use good hardware on a fast network. Appliances such as the Sourcefire device can monitor networks at speeds of up to 10Gbps with Snort 2.9 running on highly-specialized hardware.

Compared with this, normal hardware will typically reach its limits at 1Gbps. With premium network interface cards, as manufactured by Napatech [16], for example, a generous helping of RAM and a fast CPU are very much recommended. Additionally, administrators who are looking for performance will need to modify the preprocessor parameters to reflect the amount of RAM available.

Getting Snort to handle a 10Gbps network is a challenge, but it is not impossible, given enough experience, the right choice of hardware, and optimum tuning. The answer that Steven Sturges from Sourcefire gives when asked about the biggest challenge is terse and to the point: "Speed." He goes on to explain, it's not difficult analyzing 500Mbps in real time; the challenge is trying to do it at 10Gbps! ■■■

TABLE 2: Target-Based Reassembly

Method	Effect
`first`	Prefer the first overlapping segment
`last`	Prefer the last overlapping segment
`bsd`	Free BSD 4.x, Net BSD 2.x, Open BSD 3.x, AIX
`linux`	Linux 2.4 and 2.6
`old-linux`	Linux 2.2 and older
`windows`	Windows 98, NT, 2000, XP
`win2003`	Windows 2003 Server
`vista`	Windows Vista
`solaris`	Solaris 9.x
`hpux10`	HPUX 10
`hpux`	HPUX 11
`irix`	IRIX 6
`macos`	Mac OS >= 10.3

INFO

[1] Snort: *http://www.snort.org*

[2] Sourcefire: *http://www.sourcefire.com*

[3] Oinkmaster: *http://oinkmaster.sourceforge.net*

[4] BASE: *http://base.secureideas.net*

[5] Snorby: *http://www.snorby.org*

[6] OpenFPC: *http://www.openfpc.org/*

[7] Pulled Pork: *http://code.google.com/p/pulledpork/*

[8] Insta-Snorby from 19.01.2011 *http://www.snorby.org/Insta-Snorby-0.6.0.iso*

[9] TurnKey Linux Virtual Appliance Library: *http://www.turnkeylinux.org*

[10] `libpcap` library: *http://www.tcpdump.org*

[11] `PF_ring` library: *http://www.ntop.org/PF_RING.html*

[12] Barnyard2: *http://www.securixlive.com/barnyard2/index.php*

[13] Waldo file: *http://en.wikipedia.org/wiki/Waldo_file*

[14] Snorby Git repository: *https://github.com/Snorby/snorby/tarball/master*

[15] Hogger: *http://code.google.com/p/hogger/*

[16] Napatech: *http://www.napatech.com/applications/network_security/intrusion_detection.html*