Working with Snort

AUSCERT 2004 Conference

Martin Roesch, Sourcefire Inc
Agenda

- Background
- Getting Started with Snort
- Run Modes
- Using Snort as an IDS
- Managing Output
- Working with Snort tools
Background

- **What is Snort?**
  - Open Source packet analysis tool
  - Most notably, a Network Intrusion Detection System (NIDS)

- **Martin Roesch**
  - Author of Snort
  - Founder & CTO, Sourcefire Inc.
Snort History

• Initial implementation in 1998

• First goals of Snort (1998):
  • Home traffic analysis tool
  • Debugger for a honeypot I was developing
  • Learning tool for libpcap
Snort History [cont.]

- Initial open source release in December 1998
- Just a sniffer, no rules language
- Rules implemented in early 1999
- 1.0 release, June 1999
- Basic rules language, stateless
Snort History [cont.]

• Snort 1.5 released December 1999
  • Same basic architecture still in use today
  • Snort implemented as packet analysis pipeline

• Acquisition, decode, preprocess, detect, output stages all defined
Snort History [cont.]

• Snort today

• 2.1.2 stable release (2.1.3 coming soon)

• Highly stateful, 2500 detection rules + anomaly detection

• Full featured detection language

• ~15000 downloads per week
Getting Started
Getting Started

• Website - http://www.snort.org

• Stable release is always available at http://www.snort.org/dl

• Installable packages (RPMs, win32 installer, etc.) and source tarball available

• Also available via CVS
Building Snort

• Get the tarball from snort.org
• Must have libpcap and PCRE installed
• Unpack as normal
  • tar zxvf snort-2.1.2.tar.gz
• Build
  • ./configure && make && make install
Snort CVS

- Staying on the bleeding edge
- Snort CVS available at Sourceforge
- NOTE: Snort CVS is moving to snort.org!

cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/snort login

[<Enter> at password prompt]

cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/snort co snort
Read the Docs!

- Lots of documentation is available for Snort!
- Look in the ‘doc’ directory in the unpacked distro
  - Manual in PDF format
  - README files covering Snort features
  - man page, etc
Snort Run Modes
Understanding Snort Run Modes

- Three basic modes of operation
  - Sniffer
  - Packet Logger
  - NIDS
- Mode is determined at run-time via command line switches
Sniffer Mode

• Simply decodes the packets and prints them to the screen

• Command line switches:
  • -v: verbose mode
  • -d: dump application layer
  • -e: display layer-2 data
Sniffer Mode [cont.]

- **Examples:**
  
  snort -v
  
  snort -vde (snort -v -d -e also works!)

- **Output**

  05/22-11:50:11.320761 127.0.0.1:55786 -> 127.0.0.1:631
  TCP TTL:64 TOS:0x0 ID:16546 IpLen:20 DgmLen:69 DF
  ***AP*** Seq: 0xCE6183EE  Ack: 0x89ECD4F2  Win: 0xFFFF  TcpLen: 32
  TCP Options (3) => NOP NOP TS: 1481027454 1481027454
  50 4F 53 54 20 2F 20 48 54 54 50 2F 31 2E 31 0D  POST / HTTP/1.1.
Packet Logger Mode

• Log packets to disk

• Useful for post-processing captured traffic

• Great for testing Snort

• Command line switches:

  • -l <directory>: log packets in <directory>

  • -b: log packets in pcap format
Packet Logger Mode

- Examples:
  
  ```
  snort -l ~/pktlogs
  snort -l ~/pktlogs -b
  ```

- The `-d` switch is still needed to get packet payloads

- The `-l` switch logs in ASCII output mode to a directory structure based on IP/port combinations, be careful on live networks!
NIDS Mode

- Mode most people think of with Snort
- Command line switches:
  - `-c <conf-file>`: load NIDS config from `<conf-file>`
  - `-A <mode>`: specify alert `<mode>`
  - `-s`: generate syslog alerts (legacy switch)
• Using -c overloads the -l switch; -l now just denotes where output files (alerts & logs) will go

• Binary logging mode (-b) should be used in uncontrolled/live environments to avoid a DoS against the filesystem
NIDS Mode [cont.]

- Examples:
  - `snort -c snort.conf`
  - `snort -c snort.conf -d -l ~/pktlog -s`
  - `snort -c snort.conf -b -A fast`
NIDS Mode [cont.]

- Useful switches
  -D : daemon mode
  -i <intf> : read packets from <intf>
  -r <pcap_file> : read packets from <pcap_file>
  -g <gid> : set group ID of Snort process
  -u <uid> : set UID of Snort process
  -t <dir> : chroot Snort process to <dir>
Using Snort as an IDS
Understanding snort.conf

- The snort.conf file is the primary configuration file for a Snort implementation in NIDS-mode.
- Can be any filename, snort.conf is generic...
- Snort.conf can get complex, but is built on a few simple concepts.
Snort.conf primitives

- Snort configuration consists of a number of directives in a ‘directive keyword: args’ set

- Keywords:
  - config, var, preprocessor, output, include, ruletype, threshold, suppress

- Rules are a special case and we will treat them separately
Config Directive

• Config primitive is used to configure functionality typically defined at the command line

• All config values are specified in the snort_manual.pdf file!

• Example:

  • config daemon - run snort as a daemon
Var Directive

- Define a substitution variable
- Simply defines a string to represent a value
- Reference the variable with by prepending a ‘$’ to the variable name

Example:
```sh
var HOME_NET 10.1.1.0/24
var EXTERNAL_NET $HOME_NET
```
Preprocessor Directive

• Preprocessors perform complex protocol and traffic analysis

• TCP stream reassembly, IP defrag, HTTP protocol parsing, etc

• 13 preprocessors currently available in the Snort distro

• Example:
  proprocessor stream4: disable_evasion_alerts
Output Directive

- Snort has two main types of output, logs and alerts
  - Alerts are for real-time notification
  - Logs are for forensic analysis of attacks
- Multiple output options can be ‘stacked’
- Example:
  
  ```
  output log_unified: file snort.log, limit 128
  ```
Include Directive

• Works much like include/import functionality in most programming languages

• Reads named file during parsing

• Useful for keeping config files tidy

• Example:
  
  include $RULE_PATH/imap.rules
Ruletype Directive

- Creates custom rule types (in addition to alert/log/pass) with individual output settings
- Not used very often...
- Example in snort.conf
Threshold/Suppress

- Powerful capability to do rate-based alerting, limiting or squelching of noisy rules
- Can be applied globally or per-rule
- Applies to all Snort alert generators, not just rules
- Examples (globals only)

  suppress gen_id 1, sig_id 1852, track by_src, ip 10.1.1.54
  threshold gen_id 1, sig_id 1851, type limit, track by_src, count 1, seconds 60
Building snort.conf

- Formatting is typically standard
  Define vars
  Config directives
  Configure preprocessors
  Configure output
  Include classification maps
  Include rules
  Include threshold/suppression maps
Simple Config

var HOME_NET 10.1.1.0/24
var EXTERNAL_NET !$HOME_NET
var RULE_PATH /etc/snort/rules

preprocessor frag2
preprocessor stream4
preprocessor stream4_reassemble

output alert_syslog: LOG_AUTH LOG_ALERT
output log_unified: filename snort.log, limit 128

include $RULE_PATH/netbios.rules
Snort.conf Notes

- Command line switches **override** snort.conf directives!
- If no command line switches are specified, Snort looks for snort.conf in /etc/snort and the local directory
- Default logging directory is /var/log/snort
Snort Rules
Rule Syntax

- Snort’s rule syntax is simple and straightforward
- Unfortunately, it’s also somewhat quirky
- Full docs in snort_manual.pdf
Rule format

Rule Header
- static definition
- has to be in every rule

Rule Options
- variable definition
- not always necessary
- 50 options available
Rule Headers

```
alert tcp $BAD any -> $GOOD any
```

- **Rule action**:Alert
- **Protocol**:TCP
- **Src. CIDR**: $BAD$
- **Src. Port**: any
- **Dest. Port**: $GOOD$
- **Dest. CIDR**: any
- **Direction**: ->
Rule Header Options

- **Actions**
  alert, pass, log

- **Protocols**
  tcp, udp, icmp, ip

- **Directions**
  -> (unidirectional)
  <> (bidirectional)
Rule Options

(flags: SF; msg: “SYN-FIN scan”;

Option start/finish

Option Detail

flags: SF;

Keyword

Separator

Argument

Delimiter
Basic Options

• content - pattern matcher
• msg - alert message to generate on match
• flow - test for TCP connection state, traffic direction
Simple Rule

```
alert tcp !$HOME any -> $HOME any \
(content: "foo"; msg: "detected foo");
```

This rule simply looks for the string “foo” contained in any TCP packet observed on the network.

Upon detection, it generates an alert with the alert message consisting of the string “detected foo”.

NOTE: ‘\’ used to denote rule continuation on next line
Slightly Less Simple

```
alert tcp !$HOME any -> $HOME any \
  (content: "foo"; content: "bar"; msg: "detected foobar");
```

Additional content keyword added
- content keywords (and other options) can be “stacked”
- options are evaluated in order, if any fail the whole rule fails
Rule Option Types

• Informational options (meta-data)
  - msg, sid, rev, classtype, priority, reference

• Payload analysis
  - content, byte_jump, byte_test, distance, within, depth, offset, nocase, rawbytes, uricontent, isdataat, pcre, dsize
Rule Option Types
[cont.]

• Protocol analysis
  - keywords for IP, TCP, ICMP protocols

• Stateful rules
  - flowbits, threshold, flow

• Actions, post-detection
  - tag, resp, etc
Fun with Snort Rules

- Basic detection is fun and easy with Snort rules
- To detect a basic string on the network (network grep..) you just need the content keyword

```snort
alert tcp any any -> any any \
  (content: "foo"; msg: "detected foo";)
```
Getting Focused

• Detecting basic strings is easy, but can result in false positives

• Better method is to try to **focus** the rule as much as possible

• Focus consists of defining the constraints under which the rule may fire
  - TCP state, client/server target, state of application layer protocol are all interesting and necessary to focus!
Getting Stateful with Snort Rules

• Two options available for Snort rules
  - flow: check TCP session state, direction
  - flowbits: set/test/clear application state

• Stream4 preprocessor must be running for the flow keyword to work

• Flow preprocessor must be running to enable flowbits
Stateful Snort Rules

- Flow example:
  ```
  alert tcp any any -> any any \\
  (flow: established, to_server; \\
  content: "foo"; \\
  msg: "detected foo");
  ```

- This rule will only fire for TCP sessions that are in the ESTABLISHED state and for traffic that is headed to the server, all other traffic is ignored!
Stateful Snort Rules

- There is another type of state that can be used, cross-rule state
- Uses the new “flowbits” keyword to set/test/clear bits in Snort rules
- Implications: We can now track application protocol state with Snort!
  We can now perform ad protocol analysis with Snort!
Stateful Snort Rules

• Examples:
  alert tcp any any -> $SMTP 25 \n  (pcre: "/^DATA\n/i"; \n    flowbits: set,smtp.client.mode.data; \n    flowbits: noalert;)
  alert tcp any any -> $SMTP 25 \n  (msg:"SMTP expn decode"; \n    flow: established, to_server; \n    flowbits: isnotset,smtp.client.mode.data; \n    ...


Regular Expressions

- Snort supports PCRE
- Powerful regular expression payload analysis
- Useful for a variety of applications...
- Use with care
  - Maintainability
  - Performance!
alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"VIRUS OUTBOUND bad file attachment"; flow:to_server,established; content:"Content-Disposition|3A|"; nocase; pcre:"/filename\s*==\s*.\*?\.(?=[abcdehijlmnoprsvwxy])(a(d[eip]|s[dfx])|c([ho]m|li|md|pp)|d(iz|ll|ot)|e(m[fl]|xe)|h(lp|sq|ta)|jse?|m(d[abew]|s[ip])|p(p[st]|if|[lm]|ot)|r(eg|tf)|s(cr|[hy]s|wf)|v(b[es]?|cf|xd)|w(m[dfsz]|p[dmsz]|s[cfh])|xl[stw]|bat|ini|lnk|nws|ocx)\[\x27\x22\n\r\s\]/iR"; classtype:suspicious-filename-detect; sid:721; rev:7;)
Byte_test/Byte_jump

- Really useful for analyzing byte streams in Snort
- Really useful for ad hoc protocol analysis, especially RPC
- Really confusing to use due to terse syntax
- We’re working on it!
alert udp $EXTERNAL_NET any -> $HOME_NET 111 \n(msg: "RPC portmap cachefs request UDP"; \ncontent: "|00 01 86 A0|"; depth:4; offset:12; \ncontent: "|00 00 00 03|"; within:4; distance:4; \nbyte_jump:4,4,relative,align; \nbyte_jump:4,4,relative,align; \ncontent: "|00 01 87 8B|"; within:4; \ncontent: "|00 00 00 00|"; depth:4; offset:4; \nreference: bugtraq, 4674; \nreference: cve, CAN-2002-0084; \nclassstype: rpc-portmap-decode; sid: 1746; rev: 9;)
alert udp $EXTERNAL_NET any -> $HOME_NET 111 "
(msg:"RPC portmap cachefs request UDP"; 
content:"|00 01 86 A0|"; depth:4; offset:12; 
content:"|00 00 00 03|"; within:4; distance:4;
byte_jump:4,4,relative,align; 
byte_jump:4,4,relative,align; 
content:"|00 01 87 8B|"; within:4; 
content:"|00 00 00 00 00|"; depth:4; offset:4; 
reference:bugtraq,4674; 
reference:cve,CAN-2002-0084; 
classtype:rpc-portmap-decode; sid:1746; rev:9;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"RPC mountd TCP mount path overflow attempt"; flow:to_server,established; content:"|00 01 86 A5|"; depth:5; offset:16; content:"|00 00 00 01|"; within:4; distance:4; byte_jump:4,4,relative,align; byte_jump:4,4,relative,align; byte_test:4,>,1023,0,relative; content:"|00 00 00 00|"; depth:4; offset:8; reference:bugtraq,8179; reference:cve,CAN-2003-0252; reference:nessus,11800; classtype:misc-attack; sid:2184; rev:6;)
Managing Output
Snort Output

- Two types of output: alerts and logs
  - alerts are for real-time notification
  - logs are for forensics

- Several alert output types available
  - syslog, text (3 types), database, unified

- Several log output types
  - text, pcap, database, unified, CSV, (Null)
Output Notes

• Performance is a big deal in the output mechanisms

• Low performance = dropped packets!

• Snort’s unified format was designed specifically for high-performance output

• Pcap format is for cross platform analysis
Unified Output

- Unified output is setup for high performance and flexibility
- Unified format can be converted to any other output format (DB, pcap, XML...)
- Barnyard is used to process unified files
Barnyard

• Available from snort.org
  - http://www.snort.org/dl/barnyard/barnyard-0.2.0.tar.gz

• CVSROOT at cvs.sourceforge.net:/cvsroot/barnyard

• Used to convert Snort unified files to any output format

• Has input/output plugins
  - alert, log, stream (flow) data
  - DB, CSV, syslog, pcap, SGUIL, text
Thoughts

- Unified/Barnyard are the future of Snort output generation
- Performance is the number one concern in the sensor process
- Unified2 coming soon, Barnyard will change to suit
- Most complex post-processing should move into Barnyard
3rd Party Tools
Tools

• Several classes of tools
  • Log processing/reporting
  • Data analysis
  • Console
  • Policy management
Log Processing

- Swatch/Logsurfer - http://www.cert.dfn.de/eng/logsurf/
- Snortalog - http://jeremy.chartier.free.fr/snortalog/
Data Analysis

• ACID
  http://www.andrew.cmu.edu/user/rdanyliw/snor/snoracidsnort.html

• PHP/MySQL based data analysis GUI for Snort

• Web-based queries and data mining/reporting
### ACID Screenshot

The image displays a query results screen for ACID (Automated Continuity, Integrity, Dependability) with a table showing the following columns:

- **ID**
- **Signature**
- **Time Stamp**
- **Source Address**
- **Destination Address**
- **Layer 4 Proto**

The table entries are for different sessions, each identified by an ID and timestamp, with corresponding IP addresses and transport layer protocol (TCP).

The screenshot also includes a web browser interface with a URL bar showing the query results for a specified date range and criteria.

---

**Example Table Entry:**

<table>
<thead>
<tr>
<th>ID</th>
<th>Signature</th>
<th>Time Stamp</th>
<th>Source Address</th>
<th>Destination Address</th>
<th>Layer 4 Proto</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>TCP</td>
<td>2000-07-31 11:42:49</td>
<td>128.2.237.93</td>
<td>128.2.237.74</td>
<td>TCP</td>
</tr>
<tr>
<td>#2</td>
<td>TCP</td>
<td>2000-07-31 11:42:49</td>
<td>128.2.237.93</td>
<td>128.2.237.74</td>
<td>TCP</td>
</tr>
<tr>
<td>#3</td>
<td>TCP</td>
<td>2000-07-31 11:42:49</td>
<td>128.2.237.93</td>
<td>128.2.237.74</td>
<td>TCP</td>
</tr>
</tbody>
</table>

---

**Query Details:**

- **Queried DB on:** Mon September 11, 2000 23:28:11
- **Meta Criteria:** time = [07 / 31 / 2000 | any time]
- **IP Criteria:** any
- **TCP Criteria:** any
- **Payload Criteria:** any

**Displaying rows 1 - 50 of 2014**
Console

- SGUIL - http://sguil.sourceforge.net/
- TCL/Tk-based analysis and reporting console
- Built for operations/analysts
- Modified DB schema for performance
- Barnyard output plugin available
SGUIL Screenshot
Policy Management

- Oinkmaster - http://oinkmaster.sourceforge.net/
- SnortCenter - http://users.pandora.be/larc/
- Web-based (PHP/MySQL) config/policy management GUI
- Works in concert with ACID
- Provides centralized policy management
Conclusion
Thank You!