Integrating the Network Security Monitoring Model

Richard Bejtlich

Intrusion detection is a controversial topic. Although intrusion detection systems (IDS) were once hailed as the answer to the shortcomings of firewalls, they are now labeled "dead" by some market analysts and are threatened by intrusion prevention systems (IPS) and "deep inspection" firewalls. In this article, I'll look at the detection and validation of intrusions through an operational model called network security monitoring (NSM). I will briefly explain NSM theory and introduce several tools for integrating NSM concepts into existing prevention and detection systems.

NSM is the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions. NSM is an operational model inspired by the United States Air Force's signals intelligence (SIGINT) collection methods and Todd Heberlein's "Network Security Monitor" (**http://www.attackcenter.com**). SIGINT is the collection of information on communications and the transformation of that information into intelligence products. Similarly, NSM collects and analyzes network traffic to identify and validate intrusions. NSM uses full content, statistical, session, and alert data to help analysts make decisions. Whereas intrusion detection cares more about identifying successful attacks, NSM provides evidence to gauge the extent of an intrusion, assess its impact, and guide effective remediation steps.

Although NSM is effective against "unstructured threats" like script kiddies and worms, it is more concerned with the highest-end intruders. These structured threats employ stealth, encryption, zero-day exploits, and advanced back doors for which no IDS signature has been written. To compensate for this imbalance, the NSM philosophy follows three principles:

1. Some intruders are smarter than you.

2. Intruders are unpredictable.

3. Prevention eventually fails.

Belief in these principles means that traditional alert-centric intrusion detection systems must be supplemented with more content-neutral tools.

The following tools can be used to augment existing detection efforts. The reference platform is FreeBSD 4.9 RELEASE (**http://www.freebsd.org**), but all of the tools work on Linux. Installation of all tools except Argus and Sguil can be done using FreeBSD's ports or packages system. For example, to install the version of Tethereal packaged for use with FreeBSD 4.9 RELEASE, along with any dependencies, use the **pkg_add -r** command:

```
#pkg_add -r tethereal
```

The **-r** switch tells **pkg_add** to fetch the file from a remote site. An alternative is FreeBSD's ports system. Change into the ports tree directory of the desired tool and run **make && make install**. The following installs Tcpdstat from ports:

```
#cd /usr/ports/net/tcpdstat
#make && make install
```

To use the new tool immediately, run **rehash** to re-compute the hash table of the user's path.

Full Content Data

Full content data represents traffic on the wire or transmitted via radio frequency. The packet capture library libpcap

(**http://www.tcpdump.org**) is the standard for reading packets. Three tools facilitate saving entire packet contents:

- Tcpdump -- **http://www.tcpdump.org**
- Ethereal/Tethereal -- **http://www.ethereal.com**
- Snort -- **http://www.snort.org**

You can use any one of the following to capture packets on interface fxp2:

```
#tcpdump -i fxp2 -s 1514 -w pcap.lpc

#tethereal -i fxp2 -w pcap.lpc

#snort -i fxp2 -b -l ./
```

Tcpdump has a default snap length of 68 bytes. Specify the **-s** switch to capture 1514 bytes. Tethereal and Snort capture full packet contents by default. While Snort offers a **-L** option to specify a capture file name, it still appends a timestamp to that file name. The default is acceptable, as it will be **snort.log.TIMESTAMP** (e.g., snort.log.1068742800). The TIMESTAMP represents the seconds since the epoch, or 00:00:00 UTC, January 1, 1970; see **time**(3). To determine when the snort.log file was created, used the **date**(1) command:

```
#date -j -r 1068742800
Thu Nov 13 12:00:00 EST 2003
```

Once full content data is available, it can be reviewed natively using each tool. Here, we request the maximum amount of interpretation:

```
#tcpdump -n  -e -vvv -X -r pcap.lpc

#tethereal -n -x -V -r pcap.lpc

#snort -dve -r snort.log.TIMESTAMP
```

Each tool can read the others' output, as all are in pcap format. Unencrypted full content data is the ultimate network-based incident response resource, as it can be analyzed directly or used to generate other forms of NSM data.

Statistical Data

Statistical data represents broad trends in network activity. It's easy to review dozens or hundreds of packets manually, but an overview is often helpful. Tcpdstat, which can be found at:

```
http://staff.washington.edu/dittrich/talks/core02/tools/tools.html
```

summarizes a pcap file by recognizing well-known protocols. To conserve space, only a subset of the output is shown in Listing 1.

Session Data

The next step up from statistical data is session data. Session data represents conversations or flows between parties. Two formats are used: NetFlow (**http://www.cisco.com/go/netflow**) and proprietary versions. Interface FastEthernet 0/0 on a Cisco 2600 series router can be configured to export NetFlow data to a collector listening on port 9995 UDP at IP 172.27.20.3 using the following commands:

```
enable
configure
interface FastEthernet 0/0
ip route-cache flow
exit
ip flow-export destination 172.27.20.3 9995
```

If a Cisco router or other NetFlow collector is not available, you can deploy the fprobe (**http://fprobe.sourceforge.net/**) NetFlow collector:

```
#fprobe -i fxp2 172.27.20.3:9995
```

To collect the NetFlow data from the router or fprobe in real-time, use the flow-tools collection, which is found at:

```
http://www.splintered.net/sw/flow-tools/
```

First, set up flow-capture to receive the NetFlow exports. Second, use flow-cat and flow-print to review the records:

```
#flow-capture -w /nsm/netflow 0/0/9995
#flow-cat -p /nsm/netflow | flow-print

srcIP        dstIP       prot srcPort dstPort octets packets
172.27.20.3  192.168.60.2  1    0       771     56      1
216.182.1.1  172.27.20.3   17   53      3940    123     1
172.27.20.3  216.182.1.1   17   3940    53      71      1
...continues...
```

Argus is an alternative to NetFlow and is available from:

```
http://www.qosient.com/argus/
```

Argus employs the "argus" server to record traffic in a proprietary flow format, and the "ra" client to read the data. Be sure to use version 2.0.6 as it contains numerous enhancements over previous versions. Here Argus is used to process pcap.lpc. Note that producing session data from a capture file is a feature of Argus but not the NetFlow tools. Also, the column headers were added manually:

```
#argus -r pcap.lpc -w - | ra -n

Timestamp          Protocol  Source IP:Port  Dir  Dest IP:Port    Close
13 Nov 03 12:00:02  udp     216.182.1.1.53   ->  10.1.1.241.1123   TIM
13 Nov 03 12:00:04  tcp     10.1.1.177.2254  ?>  216.239.37.99.80  FIN
13 Nov 03 12:00:04  tcp     10.1.1.177.2255  ?>  216.239.37.99.80  FIN
13 Nov 03 12:00:04  tcp     10.1.1.177.2253  ?>  216.239.37.99.80  FIN
...continues...
```

Session data is especially helpful because its ignorance of application data renders it immune to encryption. Session data can be quickly passed through **grep** to locate IPs or ports of interest. Because it tracks "who talked to whom and when," session data is often the key to understanding an intrusion.

Alert Data

The final form of NSM information is alert data. Alert data consists of a judgment made by an analysis tool, like the network-based IDS Snort. Snort generates alerts using signatures, which embody the opinions of the signature developers. While Snort can be used by itself or in conjunction with tools like ACID, rapid event-based investigation, escalation, and validation of alert data require an NSM-oriented tool like Sguil, which can be obtained from:

```
http://sguil.sourceforge.net
```

Sguil is an open source interface to alert data from Snort, session data collected by Snort's stream4 preprocessor, and full content data collected by a second instance of Snort running in packet logging mode. Sguil is written in Tcl/Tk and stores its data in a MySQL database. The server, sensor, database, and client components can be collocated or run independently. Communication is encrypted using OpenSSL. By deploying the appropriate libraries, the client can even run on Windows workstations. Instructions for installing Sguil's server-side components on FreeBSD 4.9 RELEASE, and its analyst console on Windows, are available from the Sguil Web site.

Sguil is designed to display as much of the information needed to make a validation decision as possible. The main screen shows three alert panes, along with the rule that triggered a Snort alert, a decode of the packet that triggered the rule, and information on the source and destination addresses. The lower left-hand pane shows messages from the server or a real-time chat window for conversation with other analysts logged into the same Sguil server. See Figure 1, which highlights an alert indicating a directory listing passed over port 445 TCP.

If an alert cannot be validated as indicating benign or malicious activity using event data alone, analysts can use session data. Sguil offers many default queries for both event and session data, but Figure 2 shows how to build a query for the IP address associated with the Server Message Block (SMB) alert seen earlier.

Session data helps gauge the extent of an intrusion by showing to whom a target spoke, what ports were used, and how many bytes and packets were passed. Figure 3 shows the results of running the query against session data.

Once a session of interest is found, Sguil can rebuild transcripts of that activity if TCP full content data has been stored. The clear-text HTTP protocol is chosen to demonstrate this capability in Figure 4. For a non-clear text protocol like SMB, analysts can launch Ethereal from within Sguil to help decipher the traffic, as shown in Figure 5.

The end result of any analysis requires an analyst to mark the alert using one of seven predefined event categories, to clear it from the display after judging it benign, or to escalate it to a more experienced analyst. Sguil promotes accountability by marking events in its database with the identity of the user who made the validation decision.

Conclusion

In this article, I've barely scratched the surface of performing network security monitoring with open source tools. However, I hope I've shown that combining the four forms of network-based evidence can help analysts overcome the limitations of alert-centric intrusion detection methods.

Richard Bejtlich is the author of The Tao of Network Security Monitoring: Beyond Intrusion Detection*, and co-author of* Real Digital Forensics*, both to be published in mid-2004 by Addison-Wesley. He publishes the TaoSecurity blog at* **http://taosecurity.blogspot.com** *and reviews security books for Amazon.com through his* **http://www.taosecurity.com** *Web site. He began performing network security monitoring in 1998 as a captain in the Air Force Computer Emergency Response Team and continues by writing the documentation for the Sguil project.*