GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at http://www.giac.org/registration/gcia

GIAC (GCIA) Gold Certification

Visualizing Firewall Log Data to Detect Security Incidents

Author: Trenton Bond, bondta@ldschurch.org

Advisor: Joel Esler

© SANS Institute 2009, Author retains full rights.

Abstract

Many organizations are required by corporate policy or regulation to collect and review firewall log data for anomalous traffic, suspicious behaviors, and intrusions. Depending on the size of the infrastructure and organization, the amount of log data to review may be hundreds of thousands, if not millions, of records. How can security analysts properly and effectively review enormous amounts of firewall data quickly and still identify that one event that may be a potential security incident? One approach to this problem is to visually map out the firewall log data. With some knowledge about what log data may be interesting, one can easily create a map of source IP addresses, destination IP addresses, communication ports, state of the connections, denied connections, etc. By clustering common connection data and using predetermined coloring schemes in the visualization, one can clearly see patterns that may be indicative of intrusions such as: the use of certain restricted communication protocols or port scan activity common in worm propagation. For the research in this paper, the Cisco firewall family (FWSM, ASA, PIX) was used for log data samples, and open source utilities like AfterGlow and Graphviz were utilized to facilitate several quick and effective examples of visualizing firewall log data, highlighting areas of potential intrusion. This research should be helpful and relevant to anyone who is looking for ways to successfully and rapidly review enormous amounts of firewall log data for security incidents.

© SANS Institute 2009, Author retains full rights.

1. Introduction

Many security professionals are familiar with open source and commercial technologies that are developed and offered as intrusion detection systems, appliances, or software. With nearly 38,000 documented vulnerabilities in the Common Vulnerabilities and Exposures Dictionary (2009) and countless potential spyware, adware, or malware infections; no doubt, many organizations recognize the essential place in Information Security for intrusion detection systems (IDS) and rely on them heavily. While never a substitute for a well-maintained and implemented IDS, there may be times when it is necessary for incident handlers or security analysts to use alternate methods to identify possible security incidents. For instance, some organizations may interpret the following Payment Card Industry (PCI) Data Security Standard requirements (2008) to mean that they must review system event log data daily to detect possible security incidents even though it also requires the use of intrusion detection systems:

> PCI -10 Track and monitor all access to network resources and cardholder data. Logging mechanisms and the ability to track user activities are critical in preventing, detecting, or minimizing the impact of a data compromise. The presence of logs in all environments allows thorough tracking, alerting, and analysis when something does go wrong. Determining the cause of a compromise is very difficult without system activity logs.
>
> PCI -10.6 Review logs for all system components at least daily. Log reviews must include those servers that perform security functions like intrusion-detection system (IDS) and authentication, authorization, and accounting protocol (AAA) servers (for example, RADIUS).
> *Note: Log harvesting, parsing, and alerting tools may be used to meet compliance with Requirement 10.6.*
>
> PCI - 11.4 Use intrusion-detection systems, and/or intrusion-prevention systems to monitor all traffic in the cardholder data environment and alert personnel to suspected compromises. Keep all intrusion-detection and prevention engines up-to-date.

© SANS Institute 2009, Author retains full rights.

Besides corporate and regulatory compliance, there may be situations when an intrusion detection analyst is asked to investigate an environment where IDS has not been implemented or where it has not been updated regularly and/or tuned well. In such cases alternate methods must be applied to detect and identify potential incidents.

One possible alternate method to detect intrusions or security incidents is to review firewall log events. Generally speaking, firewalls can produce detailed events that can be quite effective at highlighting security incidents, device misconfiguraitons, or malfunctions. However, this method comes with a price. Depending on the size of the environment and/or implementation, the amount of data an incident handler or auditor has to sift through to find the one event that may be of concern can seem daunting, particularly, when a heavily-utilized firewall can generate thousands, if not millions, of log events each day. One possible approach that makes this detection method feasible is visualizing the firewall log data.

When visualizing log data, thousands of lines of firewall events can be viewed in a single graphical image, and relationships between source IP addresses, destination IP addresses, and the ports or protocols they are using can be plainly seen. Graphical representation also helps to quickly highlight events that may be indicative of security incidents such as devices that should not be communicating with each other, protocols that should not be used, firewall rule deficiencies, attempts to access restricted ports, or worms trying to propagate. Most importantly, visualized firewall log data encourages the "right" questions be asked; questions that would likely never be asked otherwise and that often lead to the detection of intrusions.

To demonstrate how firewall log data can be visualized to detect security incidents, a basic four-step process will be presented: Event Collection, Event Preparation, Event Graphing, and Graph Analysis. The setup and details of each of these process steps will be outlined using log events from the Cisco firewall family and the utilities, tools, and scripts found on the live slackware CD distribution "DAVIX" developed and maintained by security data visualization expert Raffael Marty. Event selection will also be discussed and details will be provided about how to properly prepare firewall log events for graphing. The graph analysis step will provide instructions on how to graph the firewall log data and will include examples. Questions raised by these examples will be explored as they relate to possible security incidents or intrusions.

© SANS Institute 2009, Author retains full rights.

Finally, potential future efforts and work related to visualizing event data for intrusions will be described.

2. Event Collection

The first step in visualizing firewall log data is to collect it, and the tool typically used to achieve this is syslog. The "…syslog protocol is a standard to send event notification messages across IP networks to event message collectors - also known as 'syslogd', 'syslog daemon' or 'syslog servers'. The term 'syslog' is often used for both the actual syslog protocol, as well as the application or library sending syslog messages" (Network Protocols Handbook, 2007). In other words, as a firewall or other syslog capable device generates messages, it uses a local syslog daemon to forward those events to a remote syslog daemon, also known as a central or remote syslog collector.

2.1 Central Syslog Collector

To setup a central syslog server to collect Cisco firewall syslog messages, almost any linux or Unix type distribution can be chosen because the syslog daemon is already built-in, and in most instances running, to handle the distribution of system events to local log files. If all the files in the /var/log directory are listed on any linux distribution, one will notice several of these log files. Each event handled by the syslog daemon is logged to one of these files. For the examples in this paper, the live slackware CD "DAVIX" was utilized to create a central syslog server where firewall event log data can be remotely collected.

2.1.1 syslog.conf

To begin configuring a central syslog server on the "DAVIX" distribution, the syslog.conf file needs to be modified so that it collects every message the firewall will be sending. One may also want to specify a new log file for testing purposes. Using a text editor like vim, modify the syslog.conf so there is only one entry in the file that looks like this:

```
*.*          /var/log/syslog_messages.log
```


© SANS Institute 2009, Author retains full rights.

Essentially, this line is telling the local syslog daemon on the central syslog server to log all messages it receives to the file "syslog_messages.log".

2.1.2 syslogd

Now that the syslog.conf file has been setup, the local syslog daemon needs to be restarted to recognize the new configuration and to tell it to listen for remote syslog messages. Start by finding the process ID associated with the syslog daemon (syslogd):

```
root@slax:~/# ps aux |grep syslog
root    11649  0.0  0.0   1848   664 ?        Ss   14:29   0:00 /usr/sbin/syslogd
root    22939  4.0  0.0   1804   560 pts/1    S+   15:28   0:00 grep syslog
```

Kill the process:

```
root@slax:~/Desktop# kill -9 11649
```

Restart the syslog daemon process with the –r option to make it listen for remote incoming syslog messages (i.e. firewall syslog messages):

```
root@slax:~/# /usr/sbin/syslogd –r
```

As defined by RFC 3164, UDP port 514 has been assigned to handle the IP transmission of syslog messages (Lonvick, 2001). Verify that the syslog server is now listening on the expected port using netstat:

```
root@slax:~/# netstat -ano |grep 514
udp        0      0 0.0.0.0:514             0.0.0.0:*                       off (0.00/0/0)
```

If necessary, don't forget to modify the firewall ACLs to allow inbound UDP syslog packets. The iptables can be modified appropriately on the "DAVIX" linux distribution with the following:

© SANS Institute 2009, Author retains full rights.

```
root@slax:~/# iptables -A INPUT -p udp -i eth0 --dport 514 -j ACCEPT
```

It is important to note that because the syslog protocol uses the unreliable transport UDP, it is a "best effort" and there is no acknowledgement of receipt for the messages sent.

2.1.3 Syslog Collector Testing

To test the central syslog server setup from a remote location, a simple client syslog generator like "Kiwi Syslog Message Generator" (http://www.kiwisyslog.com/index.php?option=com_content&task=view&id=55&Itemid=198) can be used. First, specify the "Target IP Address" as the IP address of the central syslog server. Then, type in a sample message that can be easily recognized in the "Message text to send" field. Finally, to send the test syslog message, just hit the "Send" button (see Figure 1).

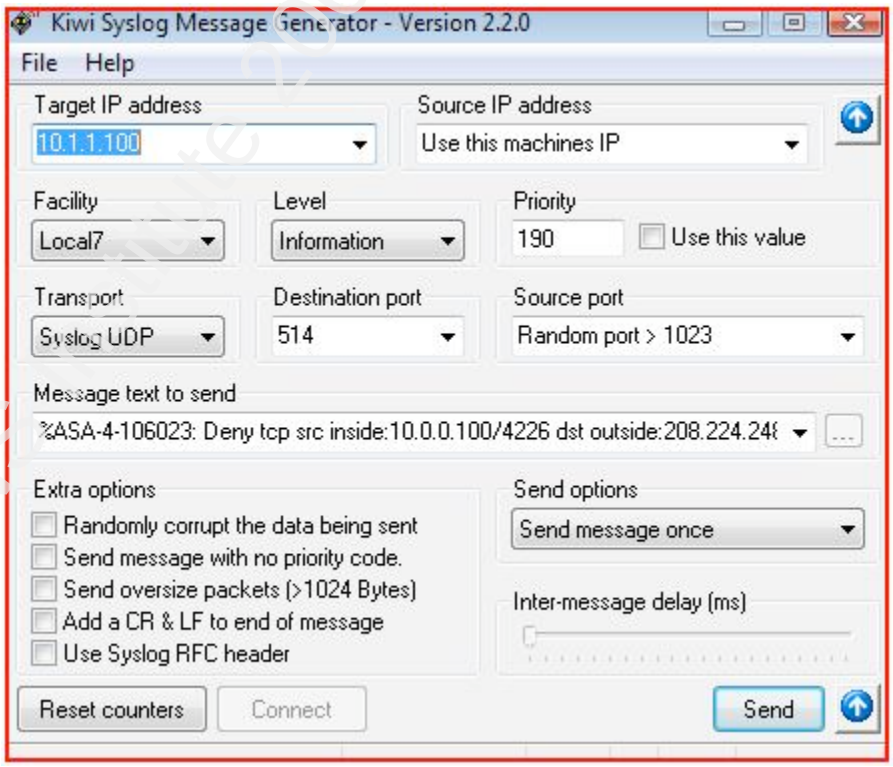


Figure 1

© SANS Institute 2009, Author retains full rights.

Now, on the central syslog server check the /var/log/syslog_messages.log file for the syslog message just sent. There should be an entry like the following:

Aug 4 14:02:58 10.1.1.101 %ASA-4-106023: Deny tcp src inside:10.0.0.100/4226 dst outside: 208.224.248.160/5012 by access-group "insidei-in" [0x0, 0x0]

.2 Log Events of Interest

Before a Cisco firewall is configured to begin logging messages to a remote collector, a decision needs be made concerning what types of events are required to be collected and analyzed. For example, events related to denied or dropped connections on the firewall are of particular interest and will be analyzed for possible intrusions in this paper. There are hundreds of possible events that a Cisco firewall can generate (see http://www.cisco.com/en/US/docs/security/asa/asa70/system/message/logsev.html). After filtering many of these messages, below are a few of the messages specifically related to denied/dropped connections produced by the Cisco family of firewalls:

Logging Level 1 (alert)

%PIX|ASA-1-106021: Deny protocol reverse path check from source_address to dest_address on interface interface_name

%PIX|ASA-1-106022: Deny protocol connection spoof from source_address to dest_address on interface interface_name

%PIX|ASA-1-106101 The number of ACL log deny-flows has reached limit (number).

Logging Level 2 (critical)

%PIX|ASA-2-106016: Deny IP spoof from (IP_address) to IP_address on interface interface_name

%PIX|ASA-2-106017: Deny IP due to Land Attack from IP_address to IP_address

%PIX|ASA-2-106018: ICMP packet type ICMP_type denied by outbound list acl_ID src inside_address dest outside_address

%PIX|ASA-2-106020: Deny IP teardrop fragment (size = number, offset = number) from IP_address to IP_address

Logging Level 3 (error)

© SANS Institute 2009, Author retains full rights.

%PIX|ASA-3-316001: Denied new tunnel to IP_address. VPN peer limit (platform_vpn_peer_limit) exceeded
%PIX|ASA-3-201002: Too many TCP connections on {static|xlate} global_address! econns nconns
%PIX|ASA-3-201004: Too many UDP connections on {static|xlate} global_address! udp connections limit
%PIX|ASA-3-201008: The PIX is disallowing new connections.
%PIX|ASA-3-201009: TCP connection limit of number for host IP_address on interface_name exceeded
%PIX|ASA-3-302302: ACL = deny; no sa created
%PIX|ASA-3-322001: Deny MAC address MAC_address, possible spoof attempt on interface interface
%PIX|ASA-3-322002: ARP inspection check failed for arp {request|response} received from host MAC_address on interface interface. This host is advertising MAC Address MAC_address_1 for IP Address IP_address, which is {statically|dynamically} bound to MAC Address MAC_address_2
%PIX|ASA-3-710003: {TCP|UDP} access denied by ACL from source_address/source_port to interface_name:dest_address/service

Logging Level 4 (warning)
%PIX|ASA-4-106023: Deny protocol src [interface_name:source_address/source_port] dst interface_name:dest_address/dest_port [type {string}, code {code}] by access_group acl_ID
%PIX|ASA-4-407001: Deny traffic for local-host interface_name:inside_address, license limit of number exceeded

Logging Level 6 (informational)
%PIX|ASA-6-106012: Deny IP from IP_address to IP_address, IP options hex
%PIX|ASA-6-106015: Deny TCP (no connection) from IP_address/port to IP_address/port flags tcp_flags on interface interface_name

All of the events above could be of significance and should be analyzed; however, to effectively demonstrate the visualization process, the examples in this paper will focus on two of the most common and prevalent denied/dropped message ids: %PIX|ASA-4-106023 and %PIX|ASA-3-710003.

2.2.1 Logging Severity Level

Once the event messages of importance are identified, the logging severity level needs to be set to ensure those messages are logged. For Cisco firewalls, "…messages are generated according to a severity level, specified by a number (0 through 7) or a keyword, as shown ..." (Hucaby, 2007, p. 587). (See Table 1) Generally speaking, when decreasing in severity, all events with a lower severity are also logged. If the firewall logging severity is set to

© SANS Institute 2009, Author retains full rights.

7 or “debugging”, syslog messages at debug level as well all other severity levels will be generated. On the other hand, if the severity is set to 0 or “emergencies”, only system unusable messages created at severity level 0 will be generated.

Severity Level	Description
0: emergencies	The system is unusable
1: alerts	Immediate action is required
2: critical	A critical condition exists
3: errors	Error message
4: warnings	Warning message
5: notifications	A normal but significant condition
6: informational	Information message
7: debugging	Debug output and very detailed logs

Table 1

Notice the logging severity level of the two messages that will be analyzed in this paper: %PIX|ASA-4-106023 and %PIX|ASA-3-710003. They are set to severity level 4 or “warnings” and severity level 3 or “errors” respectively. By setting the firewall logging severity level to “warnings”, all messages of types “error”, “critical”, “alert”, and “emergency” will also be logged; thereby capturing both messages.

2.3 Firewall Logging Configuration

There are several configuration options for Cisco firewall logging that allow for easy collection. For example, “…logging messages can be sent to any of the following destinations:

- The firewall console
- Telnet or SSH sessions to the firewall
- A temporary buffer internal to the firewall
- SNMP management stations
- Syslog servers

© SANS Institute 2009, Author retains full rights.

- Firewall management applications such as Cisco Adaptive Security Device Manager (ASDM) and Cisco PIX Device Manager (PDM)
- E-mail addresses (ASA 7.0 and above, FWSM 3.1[1] and above)
- An FTP server (ASA 7.0 and above, FWSM 3.1[1] and above)
- Firewall flash (ASA 7.0 and above, FWSM 3.1[1] and above)" (Hucaby, 2007, p. 587).

Of these choices, one of the most useful approaches is to log all of the events to a remote syslog server, similar to the one setup above. However, this may not always be possible. For example, if there is no immediate access to a syslog server, firewall log events can be sent to the local log buffer, and then retrieved using a data transfer protocol like TFTP. Both of these Cisco firewall log collection options are outlined below.

2.3.1 Remote Logging Collection

Configuring remote logging on a Cisco firewall is quite simple. First, ensure that logging is enabled and that the log severity level is set to "warnings", as determined above, to ensure that the two events of focus are captured. Finally, specify a central syslog server as the destination.

To enable remote logging, log into the firewall and enter the following commands:

```
firewall> enable
[enter enable password]
firewall# conf t
firewall(config)# logging on
firewall(config)# logging trap warnings
firewall(config)# logging host <inside|outside> <ip of your syslog server>
firewall(config)# write mem
```

At this point, the Cisco firewall should be sending all "warnings" level and lower messages over port UDP 514 to the IP address of the central syslog server specified. If the syslog.conf configuration example above was utilized, the remote collection of firewall log data can be validated by checking the /var/log/syslog_messages.log file.

© SANS Institute 2009, Author retains full rights.

2.3.2 Local Log Buffer Collection

If a remote syslog server is not available for collecting firewall messages, a local log buffer on the firewall may be useful. If the firewall is configured to use the internal firewall buffer as the output destination, the local log buffer acts as a temporary storage location for system events until they can be transferred using something like TFTP. The number of events the local log buffer can hold is directly related to the size of each message, but the size of the buffer itself can be modified. Below are a few simple steps for setting up a Cisco firewall local log buffer for event collection and transferring the data to a TFTP server.

To enable the log buffer as a log output destination, log into the firewall and enter the following commands:

```
firewall> enable
[enter enable password]
firewall# conf t
firewall(config)# logging on
firewall (config)# logging buffered warnings
```

When the local log buffer is used for event collection, one needs to ensure there is a timestamp added to the events. This is not a issue for the remote collection method because the syslog daemon adds a timestamp by default to each event it receives.

```
firewall(config)# logging timestamp
```

To set the memory size of the logging buffer to 24K:

```
firewall (config)# logging buffer-size 24
firewall(config)# write mem
```

To validate the local logging buffer is collecting events:

```
firewall (config)# show logging
```


© SANS Institute 2009, Author retains full rights.

When events have been collected, they can be copied from the local buffer to another location. To save the current contents of the local log buffer to flash memory, enter the following command:

```
firewall(config)# logging savelog syslog_messages.log
```

It is important to note that when saving the buffer content to another location, the firewall creates log files with names that use the following format: LOG-YYYY-MM-DD-HHMMSS.TXT. To list log files:

```
firewall(enable)# dir
```

To save the log file to a TFTP server of your choice:

```
firewall(enable)# copy flash:syslog_messages.log tftp
IP address of remote host [255.255.255.255]? <ip address of your TFTP server>
filename to write on tftp host? syslog_messages.log
```

At this point, the Cisco firewall should have sent all of the collected "warning" level messages and lower found in the local log buffer to the TFTP server specified in a file called syslog_messages.log.

3. Event Preparation

The next step in the firewall log data visualization process is to prepare the syslog data from a collection of firewall events. In order to visualize the data and use tools like AfterGlow and Graphviz, one must provide the data in a format the utility or tool is expecting. It is also quite useful if the event messages can be parsed with a reusable script to provide any of the fields or elements of interest upon request.

3.1 Message Separation

To start preparing the events for visualization, the two messages of focus (%PIX|ASA-4-106023 and %PIX|ASA-3-710003) will need to be separated from the other syslog

© SANS Institute 2009, Author retains full rights.

messages so they can be parsed properly. Additionally, for clarity and better understanding, the events will also need to be separated again based on whether the denied connections are originating from a host “inside” the network or from some device “outside” the network. This is done based on the access-group name and how it has been applied to each interface. For example, in this paper the access group "inside-in" has been logically applied to connection attempts originating from "inside" the firewall to some "outside" or “dmz” location and the access-group “outside-in” has been logically applied to connection attempts originating “outside” the firewall to some “inside” or “dmz “address (see Figure 2). It should also be noted that while the examples, representations, and graphs in this paper are generated from actual events, all IP addresses have been anonymized and do not reflect ownership by any particular organization or individual.

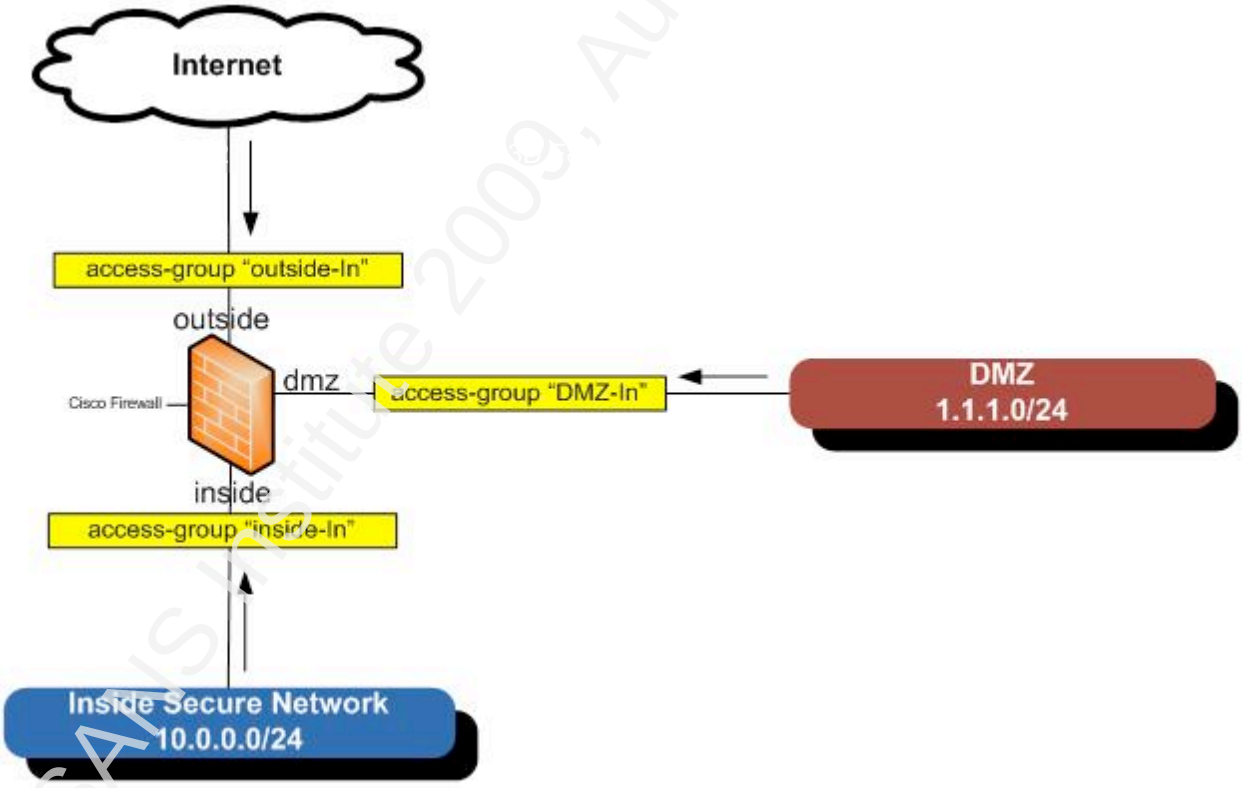


Figure 2

Assuming previously collected Cisco firewall log data is in a file named syslog_messages.log, all “inbound” denied connections originating from a source “outside” of

© SANS Institute 2009, Author retains full rights.

the firewall and of message type %PIX|ASA-4-106023 or %PIX|ASA-3-710003 can be separated from the other data using simple grep commands:

```
root@slax:~/# cat syslog_messages.log |egrep '(106023)|(710003)' |grep "\"outside-in\"" >
inbound_denied.log
```

Assuming previously collected Cisco firewall log data is in a file named syslog_messages.log, all "outbound" denied connections originating from a source "inside" of the firewall and of message type %PIX|ASA-4-106023 or %PIX|ASA-3-710003 can be separated from the other using the below grep commands:

```
root@slax:~/# cat syslog_messages.log |egrep '(106023)|(710003)' |grep "\"inside-in\"" >
outbound_denied.log
```

3.2 Message Parser – ciscofw2csv

The AfterGlow script that will be used to make visualization possible accepts only a two or three column comma delimited file. Now that the relevant events have been separated and the data sorted by direction, event fields that will be visualized need to be parsed into a comma delimited file. While Raffael Marty provided a number of event parsers on the "DAVIX" live CD, there did not appear to be one available to handle Cisco firewall syslog messages. Using Raffael's OpenBSD pf firewall parser pf2csv.pl as a template, a new parser (ciscofw2csv.pl – see Appendix A) was written to handle the two Cisco firewall messages of interest. The following fields can be requested as output when the script is executed:

- Event timestamp (timestamp)
- Source IP address (sip)
- Destination IP address (dip)
- Source port (sport)
- Destination port (dport)
- Firewall type - ASA, PIX, FWSM (type)
- Message severity level (level)

© SANS Institute 2009, Author retains full rights.

- Source interface (src_int)
- Destination interface (dst_int)
- Access group (access_group)
- Remaining message (rest)
- Protocol – UDP, TCP (proto)

Any combination of fields can be chosen, but for the purposes of demonstration in this paper, the following fields have been selected: the source IP (sip), the destination IP (dip) and the destination port that was denied (dport). It should also be noted that while the "Protocol (proto)" field can be requested separately as one of the two or three columns, it has been added by default to the "Destination port (dport)" field. It can be extremely helpful while analyzing a graph for intrusions to understand what protocol is associated with each port and to do so without having to use one of the two or three columns.

Using the previously created inbound_denied.log and outbound_denied.log files, the following command can be used to parse out the selected fields into a comma delimited file using the ciscofw2csv.pl script:

```
root@slax:~/# cat outbound_denied.log |./ciscofw2csv.pl "sip dip dport" > outbound_denied.csv

root@slax:~/# cat inbound_denied.log |./ciscofw2csv.pl "sip dip dport" > inbound_denied.csv
```

Or simply separate the events as necessary and parse out the required elements in one step with the following:

```
root@slax:~/# cat syslog_messages.log |egrep '(106023)|(710003)' |grep "\"inside-in\"" |./ciscofw2csv.pl
"sip dip dport" > outbound_denied.csv

root@slax:~/# cat syslog_messages.log |egrep '(106023)|(710003)' |grep "\"outside-in\"" |./ciscofw2csv.pl
"sip dip dport" > inbound_denied.csv
```

3. Event Graphing

Trenton Bond, bondta@ldschurch.org

© SANS Institute 2009, Author retains full rights.

With the data now properly prepared, a graph of "denied connection" firewall events can now be generated. While there are a number of open source graphing tools and utilities available in the community, two fantastic tools, AfterGlow and Graphviz, found on the "DAVIX live CD will be used to demonstrate how the data can be graphed.

3.1 AfterGlow

The AfterGlow script will be used to format the comma delimited data so it can finally be drawn. AfterGlow developer Raffael Marty described the purpose of the AfterGlow script as follows, "One type of graph which is very powerful to visualize relationships among entities, are so-called linked graphs or network graphs. Another type of visualization can be achieved with Treemaps. Different open source libraries can draw these types of graphs, but all of them require input in a very specific format - generally a graph description language. AfterGlow facilitates the generation of these inputs for graphing libraries" (2007).

The AfterGlow help option (see Appendix B) provides the appropriate syntax for a number of features as well as the appropriate format the input data. Some of these features can be used to get a better view of the data or to remove "noise" that may be distracting while trying to detect potential intrusions or security incidents. However, one of the options that can be helpful before a graph is even generated is the "color config file". In order to differentiate between the source IP addresses and the destination IP addresses, it is possible to color them differently. Below is a sample color config file, named "afterglow.colors", which was used to generate the graphs in this paper.

```
# AfterGlow Color Property File
#
# @fields is the array containing the parsed values
# color.source is the color for source nodes
# color.event is the color for event nodes
# color.target is the color for target nodes
#
# The first match wins
#
#color.source="green" if ($fields[0]=~/^10\..*/);
```


© SANS Institute 2009, Author retains full rights.

```
#color.source="green" if ($fields[0]=~/^172\.16\..*/);
color.source="dodgerblue3"
#color.event="yellow" if ($fields[1]=~/^192\.168\..*/)
#color.event="greenyellow" if ($fields[1]=~/^10\..*/)
color.event="firebrick"
#color.target="blue" if ($fields[2]<1024)
color.target="grey51"
```

Ultimately, the AfterGlow script produces a "graph description language" formatted DOT file that can then be used by a "graphing library" like Graphviz.

3.2 Graphviz

Once the data has been formatted by AfterGlow, a decision about how to draw the data must be made. As Raffael mentioned above there are basically two types of visualization: "link graphs" and "treemaps". The firewall log data which was collected and prepared is likely best understood by seeing lines drawn between the source IP address, destination IP address and destination port in a particular event or, in other words, a "link graph". One tool that is capable of drawing "link graphs" is Graphviz (Gansner, Koren, & North, 2009).

There are a number of different graph layout options available in the Graphviz suite and all of them can be quite valuable depending on the data being viewed and the effect sought by the drawing. A complete list of the algorithms and details about their layouts can be found at http://www.graphviz.org/, however, the "neato" algorithm which is a "spring-model" layout is a good choice to highlight the relationships found in the Cisco firewall events selected earlier. Ultimately, the "neato" utility takes a DOT file like the one created using AfterGlow and produces a number of image file formats such as jpg, png, gif, etc. The following options were used to create the graphs in this paper:

-Tpng (to set the output format as png)
-o (to specific the output file name)

3.3 Outbound Denied Connections Graph

© SANS Institute 2009, Author retains full rights.

Having some understanding of the AfterGlow script and the Graphviz drawing tools, a graph can now be drawn. To create a link graph called "outbound_denied.png" from the outbound denied connection data that was collected and prepared as outlined above, execute the following commands from the "DAVIX" live CD.

To create the DOT file:

```
root@slax:~/# cat outbound_denied.csv |afterglow.pl -c afterglow.colors > outbound_denied.dot
```

To create the graphical PNG image:

```
root@slax:~/# neato -v -Tpng -o outbound_denied.png outbound_denied.dot
```

Or it can all be done in one step with the following:

```
root@slax:~/# cat outbound_denied.csv |afterglow.pl -c afterglow.colors |neato -v -Tpng -o
outbound_denied.png
```

Before actually viewing the graph, it may also be interesting to get a feel for how many lines of log data are being visualized in the drawing. For the example, the generated graph (outbound_denied.png) represents more than 24,000 outbound denied connections.

```
root@slax:~/# cat outbound_denied.csv |wc –l
24738
```

To view the newly created image from the "DAVIX" live CD, simply use an image viewer called "GQview" as follows:

```
root@slax:~/# gqview
```

The generated graph, at a very high level, should look something like Figure 3 below.

© SANS Institute 2009, Author retains full rights.

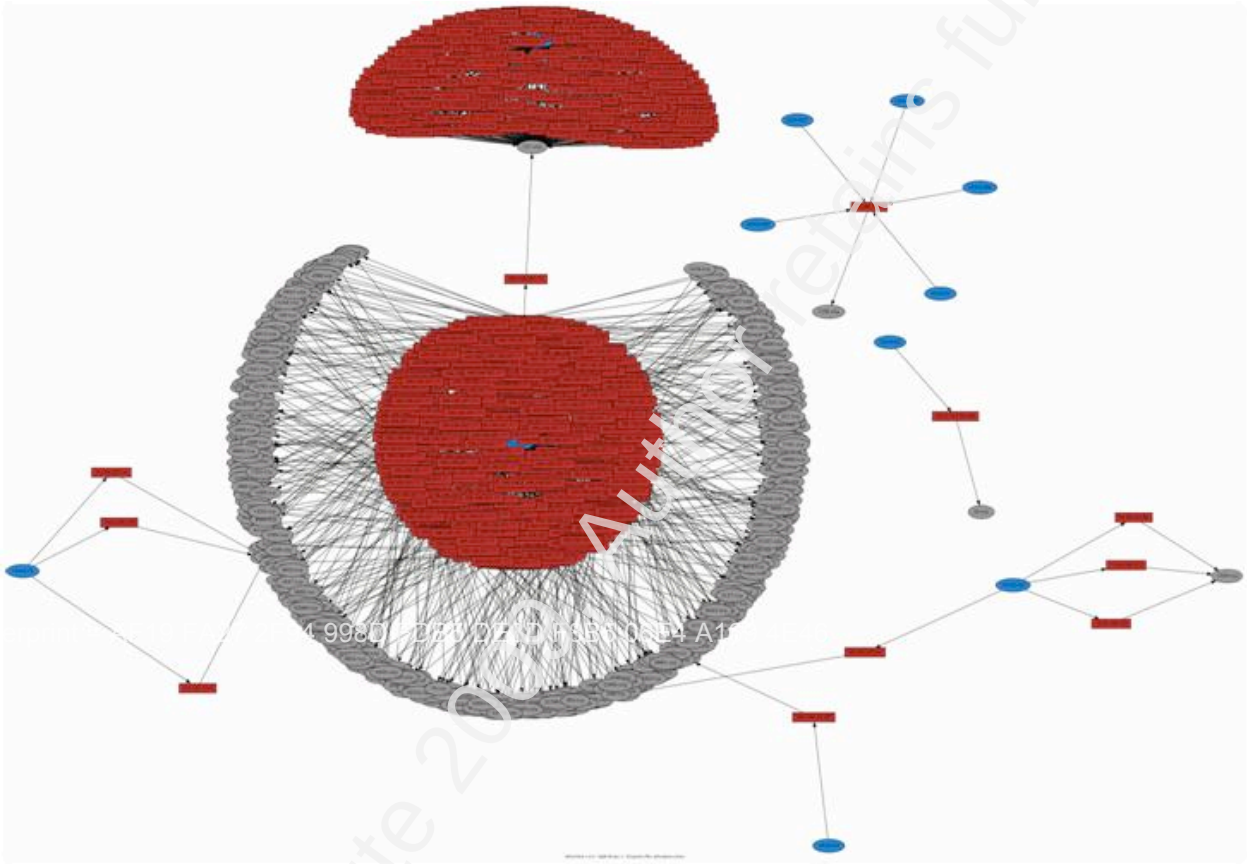


Figure 3

A closer look at the image will be taken below when the actual graph analysis is performed, however, based on the afterglow.color config file above, one can see that the blue-colored objects are the source IP addresses "inside" the firewall. The red-colored objects are the destination IP addresses "outside" the firewall and the gray-colored objects are the destination ports along with the associated IP protocol.

3.3 Inbound Denied Connections Graph

Trenton Bond, bondta@ldschurch.org

© SANS Institute 2009, Author retains full rights.

To create a similar graph for the inbound denied connection data that was collected and prepared, execute the following commands from the "DAVIX" live CD.

To create the DOT file:

```
root@slax:~/# cat inbound_denied.csv |afterglow.pl -c afterglow.colors > inbound_denied.dot
```

To create the graphical PNG image:

```
root@slax:~/# neato -v -Tpng -o inbound_denied.png inbound_denied.dot
```

Or it can all be done in one step with the following:

```
root@slax:~/# cat inbound_denied.csv |afterglow.pl -c afterglow.colors |neato -v -Tpng -o
inbound_denied.png
```

Once again, it may be helpful to know how many lines of log data are visualized in the graph. For the example here, the number of denied connection events represented in the generated graph (inbound_denied.png) was over 25,000.

```
root@slax:~/# cat inbound_denied.csv |wc –l
25422
```

To view the newly created image from the "DAVIX" live CD, use the image viewer "GQview" as follows:

```
root@slax:~/# gqview
```

The graph, at a very high level, should look something like Figure 4 below.

© SANS Institute 2009, Author retains full rights.

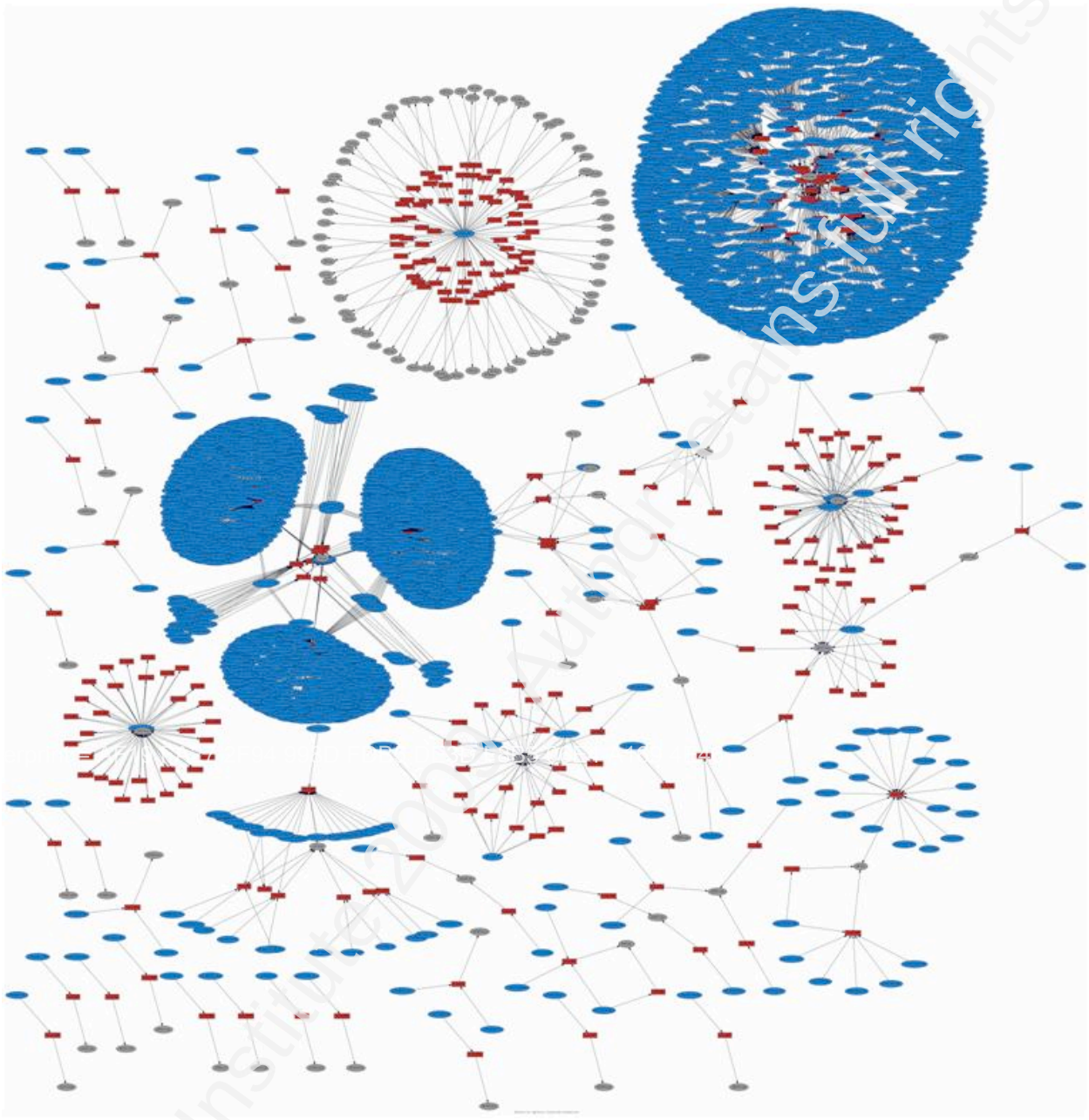


Figure 4

4. Graph Analysis

The whole idea of visualizing firewall log data is to make it easier to identify areas of consequence and raise appropriate questions without having to review every single log record. Specifically for this paper, the focus was on outbound denied connection attempts through a

© SANS Institute 2009, Author retains full rights.

firewall to help detect potential intrusions or security incidents. To do that, the link graph was generated from two of the most common Cisco firewall dropped/denied event ID's %PIX|ASA-4-106023 or %PIX|ASA-3-710003. To provide even more clarity, the data was also separated into an inbound perspective and an outbound perspective. Now, looking closer at the example graphs, there are some very interesting questions that might otherwise have gone unnoticed or unasked.

4.1 Outbound Denied Connections Analysis

The "outbound_denied.png" graph by itself is fascinating, even without analyzing it. The fact that there is data to create this graph could be a concern. Keep in mind that this is outbound traffic from "inside" the network to some destination "outside" the network. Theoretically, all of the devices on the "inside" network should be configured properly, patched, scanned, and managed by the organization with the appropriate firewall rules in place to allow required communication. In other words, this view of the firewall log data should, for the most part, contain little or no data. So, the fact that there are denied connections originating from "inside" the firewall is concerning, and every event represented in this graph is probably worth investigation. In some instances, it may turn out to be a simple device or application misconfiguration, but it could also be subtle residue from an actual intrusion or compromise.

With a quick glance of the "outbound_deniedn.png" graph, it is easy to see several areas to investigate for possible security incidents or intrusions:

1. The first couple of patterns that are probably most noticeable in the graph are the two areas with a large number of red objects clustered together. Taking a closer look at the cluster at the top of the graph (see Figure 5) one can see that there is a large number of IP destination addresses (the red objects) that all appear to be of the same class C network (31.126.62.x) and a destination of UDP port of 137.

© SANS Institute 2009, Author retains full rights.

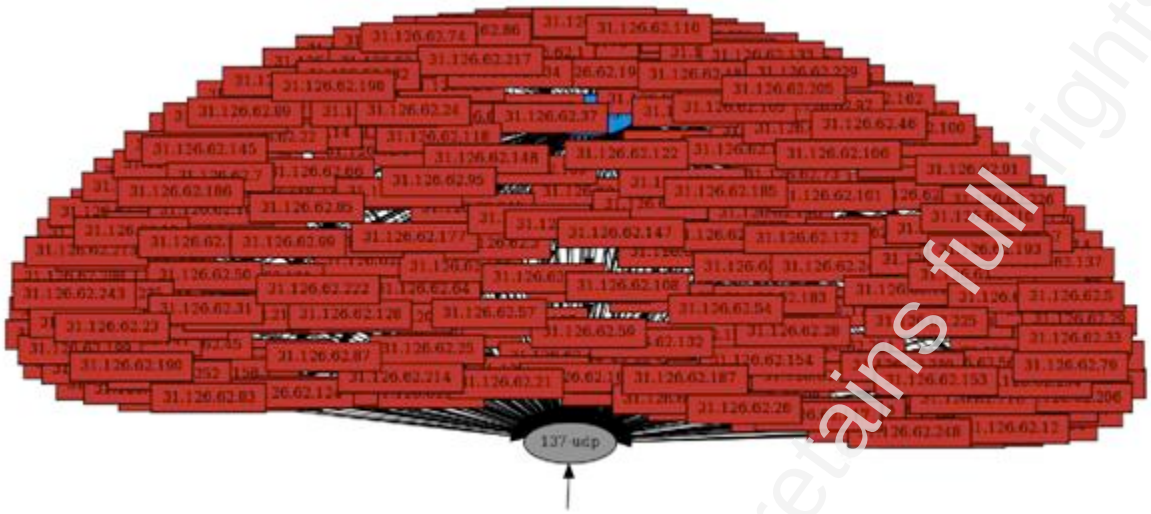


Figure 5

So where is the source? Unfortunately, in some instances there is so much data that objects can be drawn tightly clustered making it is impossible to clearly identify either a source or destination IP address. Based on the other visible characteristics of the pattern in question, and if necessary, one can quickly find the "hidden" information or object from the original log data using grep.

```
root@slax:~/# cat syslog_messages.log |egrep '(106023)|(71003)' |grep "\"inside-in\"" |grep "\/137
"
```

As snippet of the original log data is below:

```
….
Aug 04 02:29:44 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.52/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:46 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.53/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:46 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.47/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:46 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.46/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:46 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.44/137 by access-group "inside-in" [0x0, 0x0
```


© SANS Institute 2009, Author retains full rights.

```
Aug 04 02:29:48 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.45/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:48 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.43/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:48 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.42/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:48 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.40/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:49 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.34/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:49 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.32/137 by access-group "inside-in" [0x0, 0x0
Aug 04 02:29:49 10.0.0.5 %ASA-4-106023: Deny udp src inside:10.0.0.45/137 dst outside:
31.126.62.33/137 by access-group "inside-in" [0x0, 0x0
….
```

It is pretty easy see from the original log data that the source IP address is 10.0.0.45 and that it is the host that appears to be scanning all of those destinations on UDP port 137.

Although it appears that the source is trying unsuccessfully in this case to connect to "outside" hosts, it appears to be very systematic and to have happened in a short period of time. There are also numerous known worms that actively try to propagate by exploiting vulnerabilities that exist in Windows services that use this port. Is the device infected? It seems likely. With what malware exactly is unknown, but now that the source IP address and the behavior is known, it can be forensically investigated for evidence of compromise. Additionally, a network trace could also be collected to provide even more details from a network span port or tap on the "inside" network using tcpdump if desired.

```
root@slax:~/# tcpdump –i <interface> –s 0 –nX src host 10.0.0.45 and udp dst port 137
```

2. The other very obvious pattern in the graph is the cluster in the middle (see Figure 6). With a quick look at the destination IPs, it is obvious that they are all the same (200.190.207.77), but the destination port varies.

© SANS Institute 2009, Author retains full rights.

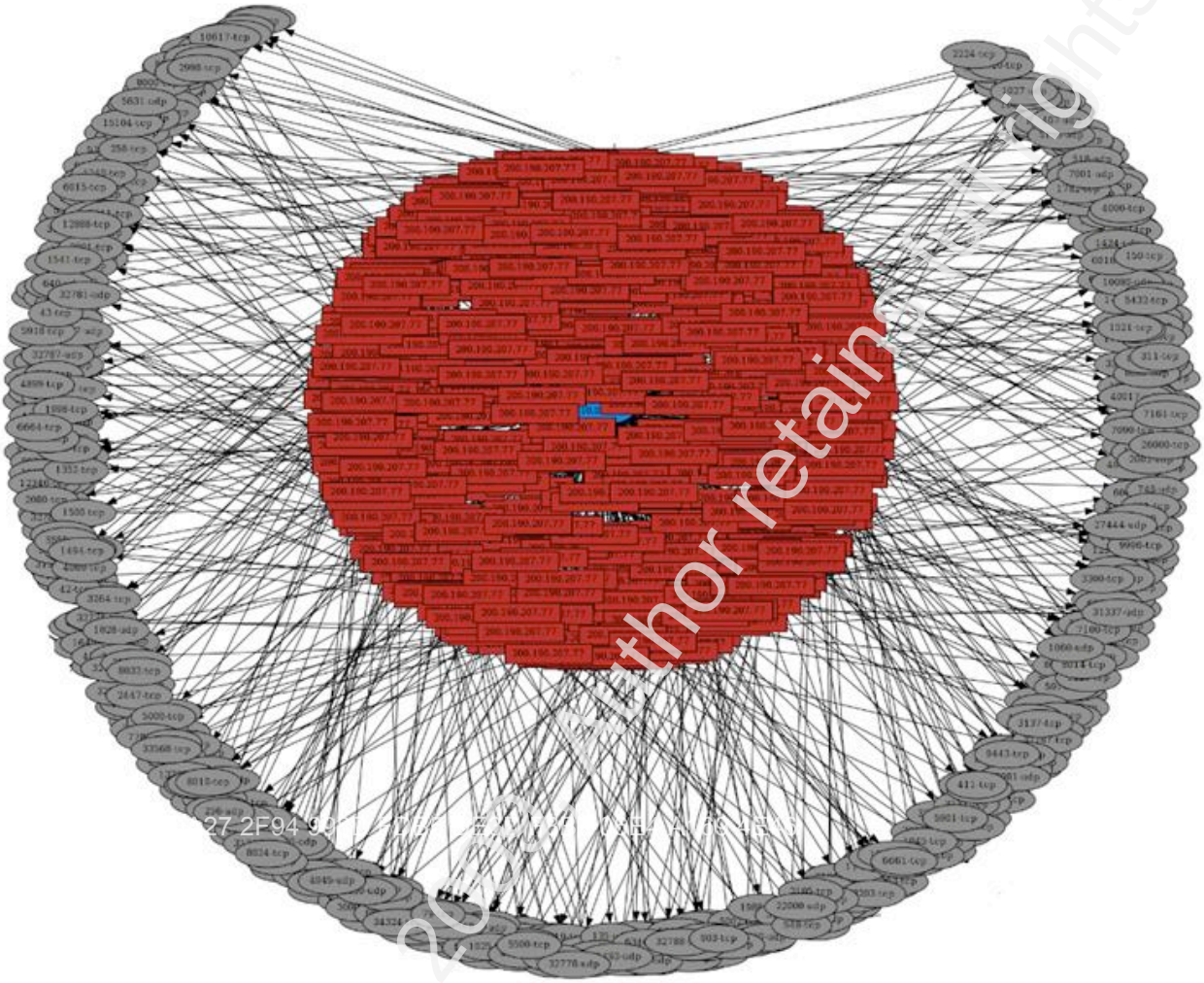


Figure 6

Once again, there is so much data that it is hard to see the source, but that is no problem. There does not appear to be any other denied connection attempts to the destination "200.190.207.77", so with a quick search through the comma delimited data created by the ciscofw2csv.pl script, the source should be found (this can also be found easily in the original syslog data).

```
root@slax:~/# cat outbound_denied.csv |grep "200.190.207.77"
```

A snippet of the outbound_denied.csv data is below:

© SANS Institute 2009, Author retains full rights.

….
10.0.0.13,200.190.207.77,389-tcp
10.0.0.13,200.190.207.77,106-tcp
10.0.0.13,200.190.207.77,1025-tcp
10.0.0.13,200.190.207.77,2003-tcp
10.0.0.13,200.190.207.77,1755-tcp
10.0.0.13,200.190.207.77,49-tcp
10.0.0.13,200.190.207.77,113-tcp
10.0.0.13,200.190.207.77,8594-tcp
10.0.0.13,200.190.207.77,1030-tcp
10.0.0.13,200.190.207.77,6664-tcp
10.0.0.13,200.190.207.77,32780-tcp
10.0.0.13,200.190.207.77,9999-tcp
….

The source appears to be 10.0.0.13 and the ports are seemingly random or perhaps "well known" ports. Regardless, the source is defiantly scanning the destination 200.190.207.77 for open ports. The question is, was it intentional or a sign that the device has been compromised? Correlating this information with a network trace, intrusion detection events, if available, or actually performing a forensic investigation of the source device in question should yield a great deal of additional data. Most importantly, one knows where to focus his or her attention.

3. Another relevant find may be the five devices attempting to access a single device outside the firewall on UDP port 1230 (see Figure 7). Why are only five devices from "inside" the network trying to access a device "outside" the network on UDP port 1230?

© SANS Institute 2009, Author retains full rights.

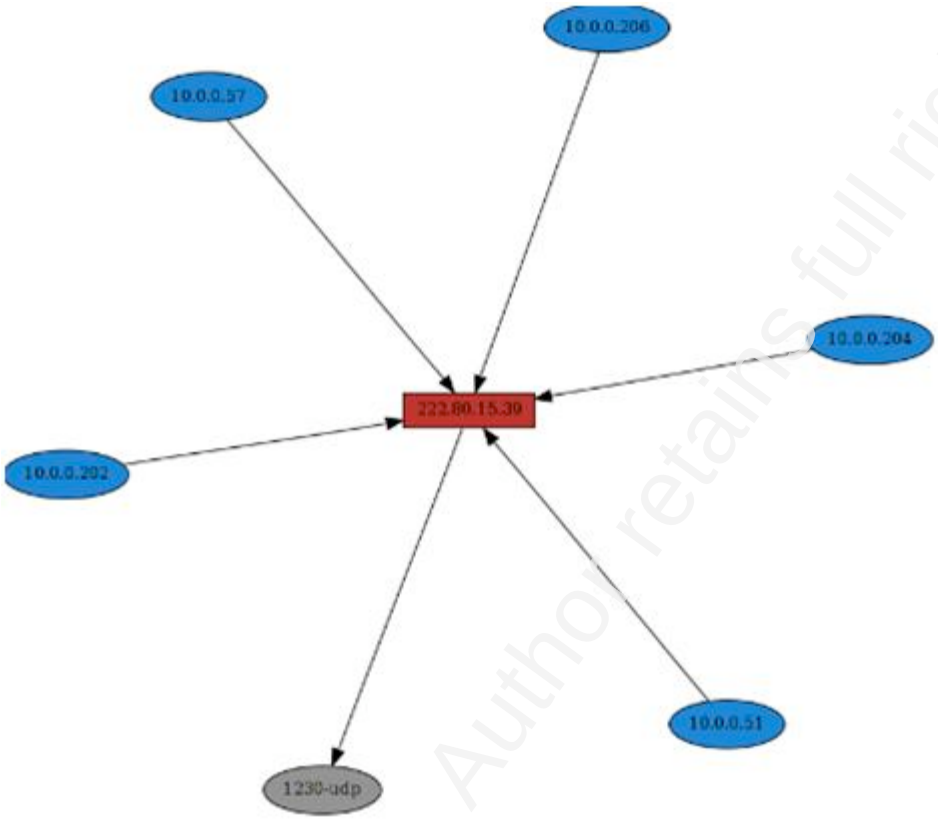


Figure 7

The IANA list of registered ports says that UDP port 1230 is Periscope, but this seems unlikely. Who is 222.80.15.39?

4. There are also several smaller relationships that are represented in the outbound denied connections graph that could be indicative of a possible security incident. For example, source IP 10.0.0.63 is trying to connect to an SMTP (TCP port 25) server outside of the local network at IP 223.173.176.184 (See Figure 8).

© SANS Institute 2009, Author retains full rights.



Figure 8

Is this evidence of a simple misconfiguration or is this a malicious attempt to mail out collected sensitive data or keylogged data. What is the approximate physical location of the destination IP? Who owns this address space? Even though this representation is not as eye-catching as other features in the graph, it is easy to see that it is still important to follow up on all displayed events.

The items above are not an exhaustive list of areas or relationships to focus on in this graph. As stated earlier, everything in this graph is of interest and should be investigated as security incidents. The objective of graph analysis is that with one quick look at the visualized

© SANS Institute 2009, Author retains full rights.

firewall data, it is obvious that there are issues, and that there is enough information to know where to focus potential investigation efforts.

4.2 Inbound Denied Connections Analysis

Typically, the reason firewalls are used is to segment a secure and controlled network from a hostile or less secure environment. Denied connections "inbound" into a network from "hostile" networks on the "outside" can be expected; however, visualizing this inbound denied data can still highlight some very appealing questions that probably deserve deeper analysis and investigation.

Looking at the inbound_denied.png graph, it is easy to see several areas of possible concern:

1. One of the first patterns that comes to attention is the large blue cluster in the upper right corner of the graph (see Figure 9). Zooming in to the cluster, one can see what appears to be hundreds of random internet addresses trying to access TCP port 445 on several destination IP addresses residing in the "dmz" (see Figure 2). Port 445 is commonly SMB over TCP and, unfortunately in the case of Windows, has historically had a number of exploitable vulnerabilities associated with it. The first question that comes to mind here is, why aren't attempts seen to every IP address in the "dmz" subnet? Typically, if a device(s) is infected with malware that has worm-like propagation features, it tries to hit every IP address regardless of its state. Was there a previous ping sweep that indicated these hosts were "live"? Or was there a previous fingerprint scan that identified these devices at most likely to be a running a Windows operating system and thus most likely to be exploitable? Another possibility is that these destination addresses have already been compromised and "phoned home", and this is simply residue of that communication.

   Regardless, an investigation of the "dmz" devices is likely warranted here. Are the operating systems and applications found in the "dmz" fully patched and configured properly to help avoid an intrusion? If there is an intrusion detection system, are there any alerts for the destinations or related to the destination port TCP 445?

© SANS Institute 2009, Author retains full rights.

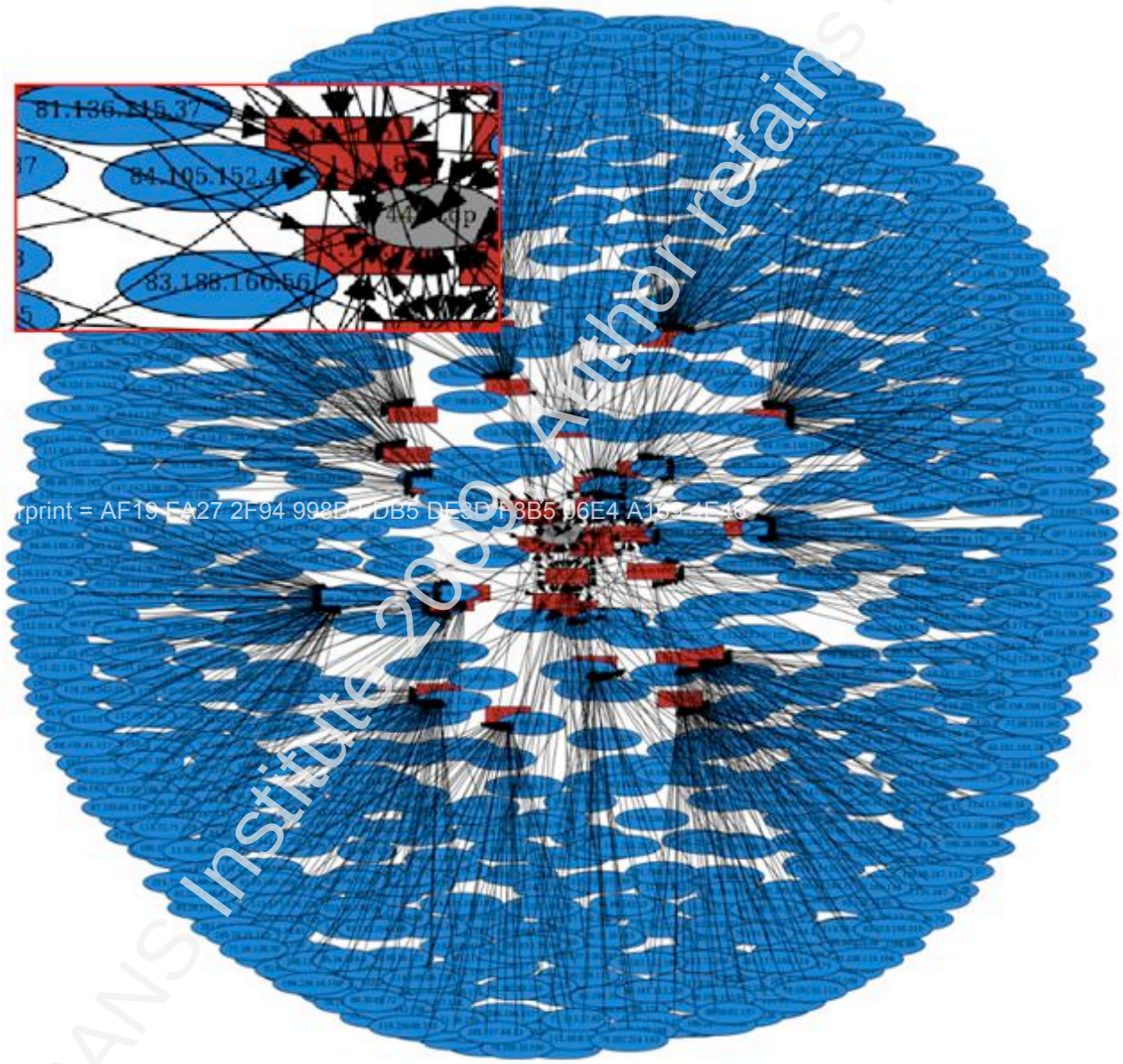
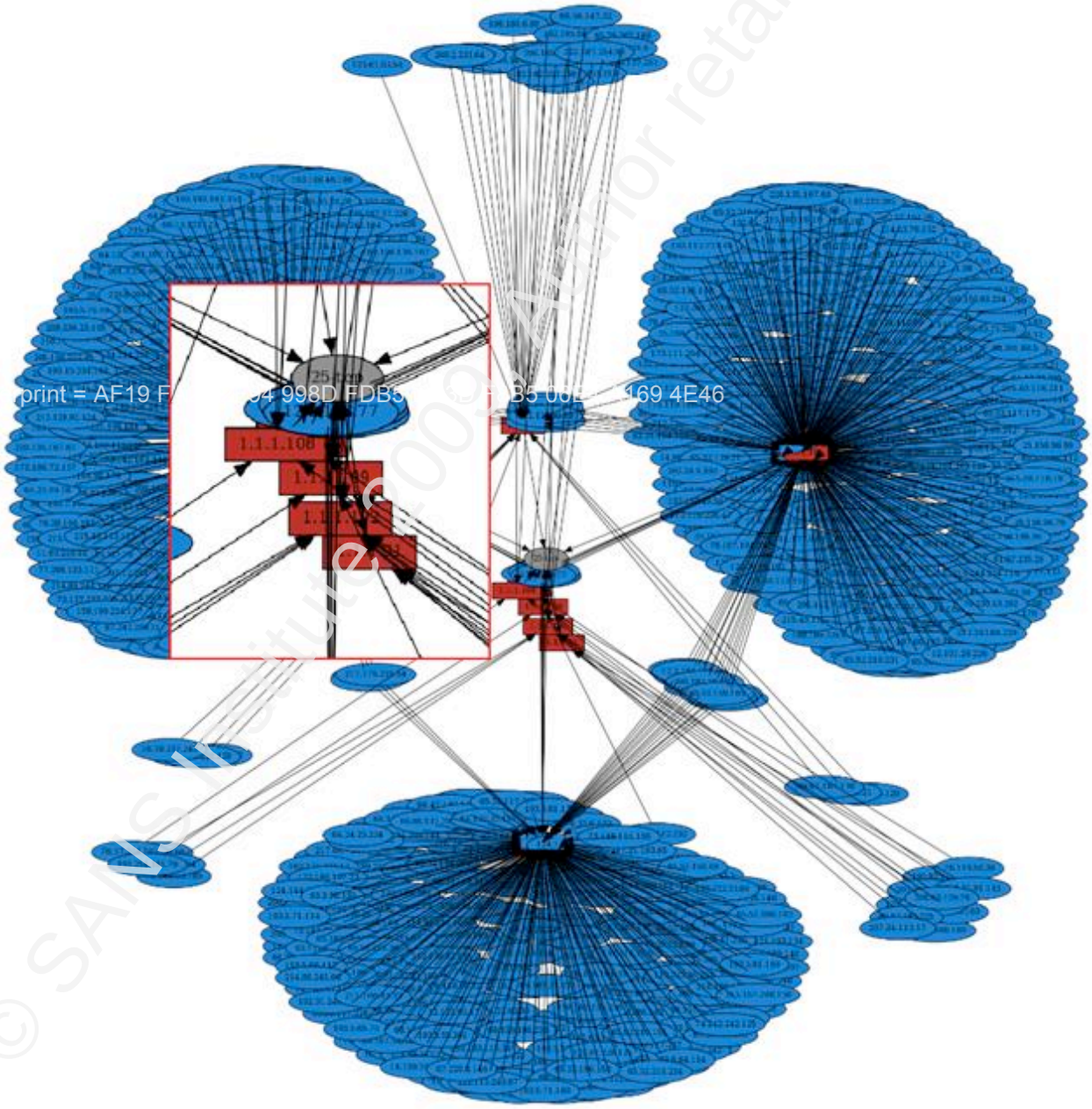


Figure 9

2. It is also easy to see a three node cluster near the middle of the graph and centered around connection attempts to access a couple of "dmz" addresses on TCP port 25 (typically

© SANS Institute 2009, Author retains full rights.

SMTP) (see Figure 10). Why are so many outside sources trying to access an SMTP server in the "dmz" network? Was it previously allowed? Are DNS mail records misconfigured? Are these devices compromised and being advertised as SMTP relays even though TCP port 25 is denied inbound? There are countless possibilities here, but a brief investigation of the destination devices may provide additional clues.



Trenton Bond, bondta@ldschurch.org

© SANS Institute 2009, Author retains full rights.

Figure 10

3. The "wheel spoke" pattern in the graph is also easy to identify in the inbound_denied.png graph (see Figure 11). It looks like one outside source IP (1.246.1.101) is scanning a single "dmz" host (1.1.1.76) on many different ports. Unfortunately, "neato" does not draw the target objects (in this case the destination ports) sequentially around the source IP address, so it can be hard to tell if the scan was sequential. A quick search through the source data for source IP address 1.246.1.101 using grep shows that it is a very sequential TCP port scan from 1-72.

© SANS Institute 2009, Author retains full rights.

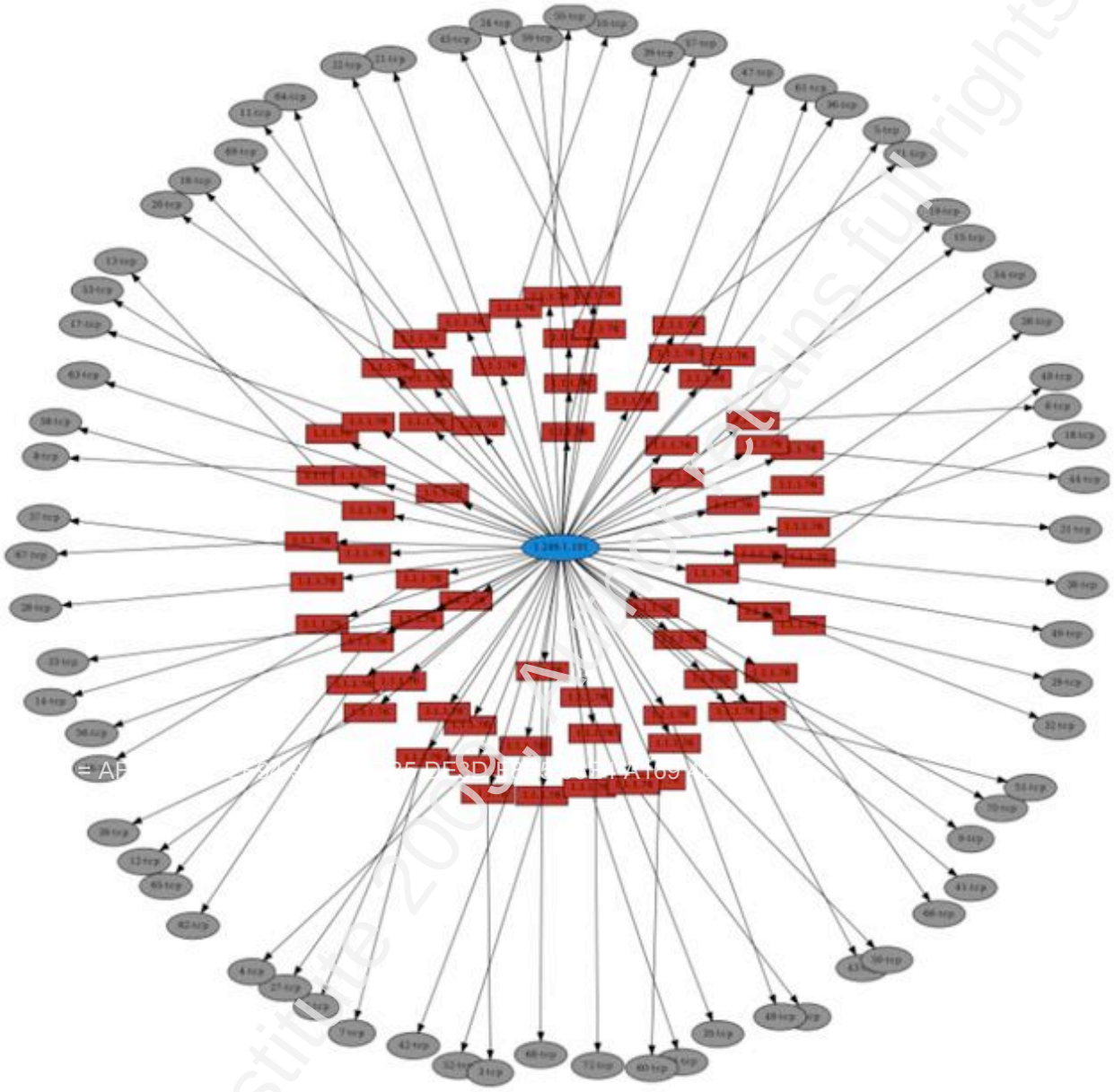


Figure 11

The curious thing here is, a denied inbound connection for every port between one and 72 was graphed except for TCP port 21 (usually FTP) and TCP port 23 (usually telnet). Is it because these ports are allowed through the firewall? Are telnet and FTP firewall ports open for the "dmz" destination 1.1.1.76? Is this intentional or has this contributed to a host compromise?

Trenton Bond, bondta@ldschurch.org

© SANS Institute 2009, Author retains full rights.

There are many more connection attempts that ought to be investigated further from the inbound_denied.png graph. However, from the examples above it is apparent what the most obvious issues are, which devices on the "inside" network need to be validated, and which access group configurations need to be reviewed.

5. Future Efforts

There are a number of areas surrounding the idea of visualizing security event log data that can still be explored and expanded upon. Below are a few ideas for future visualization efforts:

1. While the research performed for this paper focused on the Cisco firewall family (ASA, FWSM, and PIX), the process of event collection, event preparation, event graphing, and finally analysis can be applied to nearly any security device or application event logs. Two of the most intriguing might be the Windows operating system events and Oracle database event logs. Research would need to be done on the possible events, but failed login attempts may provide some significant information that, if visualized, could also help detect possible security incidents.
2. There are several additional denied/dropped type messages that can be generated by the Cisco family of firewalls (See section 2.2 above). Only two of the most common denied/dropped type events were used in this paper (%PIX|ASA-4-106023 and %PIX|ASA-3-710003). Many more events could be added to the ciscofw2csv.pl parser.
3. Besides AfterGlow and Graphviz, there are other graphing tools that can be used to visualize security events. Some of these additional tools are found on the "DAVIX" live CD. There are also others like Flare (http://flare.prefuse.org/), an actionscript library that produces beautiful graphs and other visualizations which can be used in Adobe Flash Player.
4. This paper focused on denied/dropped type connection attempts. Possibly even more interesting would the connection "teardown" type events. First, it can always be inferred that if there is a connection "teardown", there must have been a "build up". Are these "build ups" allowed? Who has the most "build ups"? Secondly, there may be particular notice in the reason why the connection was torn down. For example, several connections

© SANS Institute 2009, Author retains full rights.

torn down as a result of a SYN timeout can be an indicator that a device may be infected with malware, performing a SYN scan, or attempting to use a service not currently available.

6. Conclusion

"The goal of data visualization is to provide an at-a-glance understanding of accurate and relevant statistics so that viewers can draw conclusions and appropriately act on them" (Trost, 2009). As has been demonstrated in this paper, visualizing firewall log data to detect potential security incidents is a fairly simple process of collecting event data, preparing the event data, graphing it, and finally analyzing it. Thousands of security events can be reviewed simply and quickly by making a graphical image of the data using tools such as AfterGlow and Graphviz to map relationships like the source IP addresses, destination IP addresses, and destination ports that may be of concern. While graphs of firewall log data may lack many of the details required to fully understand a potential security incident, they do provide enough data to be a catalyst for asking the right questions. These questions will inevitably lead to other event correlating tools that can provide more of the necessary information to determine if the event is truly a security incident. Raffael Marty stated best what log data visualization is all about in his book, "Applied Security Visualization", when he cleverly modified the old Chinese proverb "A picture is worth a thousand words" to "A picture is worth a thousand log records" (2008).

6. References

Common Vulnerabilities and Exposures. (2009). Retrieved August 10, 2009, from The Standard for Information Security Vulnerability Names Web site: http://cve.mitre.org/

Gansner, Emden, Koren, Yeheda, & North, Stephen. (2009). Graphvis – Graph Visualization Software (neato). Retrieved August 1, 2009, from Graphviz Web site: http://graphviz.org/

Hucaby, David. (2007). Cisco ASA, PIX, and FWSM Firewall Handbook, Second Edition. Cisco Press.

Javvin Technologies, Inc.(2007). Network Protocols Handbook. Javvin Press.

Trenton Bond, bondta@ldschurch.org

© SANS Institute 2009, Author retains full rights.

Lonvick ,C. (2001). The BSD syslog Protocol – RFC 3164. Retrieved August 11, 2009, from Internet Engineering Task Force (IETF) Web Site: http://www.ietf.org/rfc/rfc3164.txt

Marty, Raffael. (2007). AfterGlow. Retrieved August 1, 2009, from Sourceforge Web Site: http://afterglow.sourceforge.net/about.html

Marty, Raffael. (2008). The DAVIX Live CD. Retrieved June 4, 2009, from SecViz – Security Visualization Web Site: http://secviz.org/content/the-davix-live-cd

Marty, Raffael. (2008) Applied Security Visualization. Addison-Wesley Professional.

Payment Card Industry (PCI) - Data Security Standard, Requirements and Security Assessment Procedures . (Version 1.2 October 2008). Retrieved August 10, 2009, from PCI Security Standards Council Web site: https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html

Trost, Ryan. (2009). Intrusion Analysis: Prevention and Detection for the Twenty-First Century. Addison-Wesley Professional.

Trenton Bond, bondta@ldschurch.org

© SANS Institute 2009, Author retains full rights.

Appendix A – ciscofw2csv.pl

```
#!/usr/bin/perl
##############################################################################
# Title:   CISCOFW2CSV
# File:    ciscofw2csv.pl
# Version:          1.0
#
# Description:     Takes specific cisco firewall (asa, fwsm, pix) messages (see below) and
# parses the events into comma separated values. Specific fields wanted in the output are
# provided at execution.
#
# %PIX|ASA-3-710003: {TCP|UDP} access denied by ACL from source_address/source_port to
interface_name:dest_address/service
# %PIX|ASA-4-106023: Deny protocol src [interface_name:source_address/source_port] dst
interface_name:dest_address/dest_port [type
#{string}, code {code}] by access_group #acl_ID
#
# Usage: cat /var/log/enterprise.log | ./ciscofw2csv.pl ["field list"]
#
# Possible fields:
#                   timestamp  sip  dip  sport  dport  type  level src_int dst_int  access_group
#                   rest  proto report_ip
##############################################################################
#
#
use strict vars;
my $output=$ARGV[0];
our  ($timestamp,$report_ip,$level,$type,$src_int,$sip,$sport,$dst_int,$dip,$dport,$access_group,$rest,$proto);

while (<STDIN>) {
        chomp;
        my $input = $_;
        if ($input =~ /106023/){
        ($timestamp, $report_ip, $type, $level, $proto, $src_int, $sip, $sport, $dst_int, $dip, $dport, $access_group,
$rest) =
                $input =~ /^(.* \d{2}:\d{2}:\d{2}).*? (\d+\.\d+\.\d+\.\d+) \%(ASA|FWSM|PIX)\-(4)\-106023\:
Deny (tcp|udp) src (\D+)\:(\d+\.\d+\.\d+\.\d+)\/(\d+) dst (\D+)\:(\d+\.\d+\.\d+\.\d+)\/(\d+) by access-group \"(\D+)
\" (.*)/;
        }
        if ($input =~ /710003/){
        ($timestamp, $report_ip, $type, $level, $proto, $sip, $sport, $dst_int, $dip, $dport) =
                $input =~ /^(.* \d{2}:\d{2}:\d{2}).*? (\d+\.\d+\.\d+\.\d+) \%(ASA|FWSM|PIX)\-(3)\-710003\:
(TCP|UDP) access denied by ACL from (\d+\.\d+\.\d+\.\d+)\/(\d+) to (\D+)\:(\d+\.\d+\.\d+\.\d+)\/(\d+)/;
        }
        $dport = $dport."-".$proto;
        my @tokens = split / /,$output;
        print ${shift(@tokens)};
        for my $token (@tokens) {
                print ','.$$token;
        }
        print "\n";
}
```

Trenton Bond, bondta@ldschurch.org

© SANS Institute 2009, Author retains full rights.

Appendix B – afterglow.pl -h

```
Afterglow $version ------------------------------------------------------------

A program to visualize network activitiy data using graphs.
Uses the dot graph layout program fromt the Graphviz suite.
Input data is expected to be in this simple CSV-style format:

   [subject],  [predicate], [object]
   10.10.10.10, ACCEPT, 216.239.37.99

Usage:   afterglow.pl [-adhnstv] [-b lines] [-c conffile] [-e length] [-f threshold ] [-g threshold] [-l lines] [-o
threshold] [-p mode] [-x color] [-m maxsize]

-a              : turn off labelelling of the output graph with the configuration used
-b lines        : number of lines to skip (e.g., 1 for header line)
-c conffile     : color config file
-d              : print node count
-e length       : edge length
-f threshold    : source fan out threshold
-g threshold    : event fan out threshold (only in three node mode)
-h              : this (help) message
-l lines        : the maximum number of lines to read
-m              : the maximum size for a node
-n              : don't print node labels
-o threshold     : omit threshold (minimum count for nodes to be displayed)
                        Non-connected nodes will be filtered too.
-p mode         : split mode for predicate nodes where mode is
                        0 = only one unique predicate node (default)
                        1 = one predicate node per unique subject node.
                        2 = one predicate node per unique target node.
                        3 = one predicate node per unique source/target node.
-s              : split subject and object nodes
-t              : two node mode (skip over objects)
-v              : verbose output
-x              : text label color

Example: cat somedata.csv | afterglow.pl -v | dot -Tgif -o somedata.gif

The dot executable from the Graphviz suite can be obtained
from the AT&T research website: http://www.graphviz.org
```

Trenton Bond, bondta@ldschurch.org

© SANS Institute 2009, Author retains full rights.