Global Information Assurance Certification Paper

Case study of analysis methods for firewall retrofit and operation at ABC University

GCFW Gold Certification

Author: Kim Cary, cary00@gmail.com

Adviser: Dominicus Adriyanto

Accepted: August 30th 2006

Abstract

Universities have traditionally provided open environments in support of intellectual growth. University computer installations have followed this open tradition. Adding a security layer, such as a firewall, to this culture of 'unlocked doors' has unique challenges, especially for young and entrepreneurial Universities, like ABC. This paper shows how to meet the challenge of low fiscal impact by using open source tools and re-purposing equipment in-hand. The challenge of low service impact was met through three layers of analysis before cutover. The methods used to develop this analysis are structured for re-use in other firewall projects and presented for use by others with similar challenges. The cultural challenge was the hardest – how to make an open community feel good about closing access? This paper shows how to leverage pre-install analysis data collection systems for post-install response via a self-service security information application. This application was useful in securing and retaining the open community's good will for future security projects (without the motivation of a incident). This paper shows how the same self-service application is also useful for ongoing security operations.

Figures

Tables

1. Statement of the Problem

Product Installer: "I will need a port opened on your firewall."
System Administrator: "We don't have a firewall."

Those words could have reflected a number of conversations vendor systems engineers doing installations over the years at ABC University. I'm sure that in the not too distant past this could have been many an institution of higher learning. Slowly, the Universities have come to the realization that good security is as vital to the mission of education as the free flow of information.

When the University was audited and charged with putting up an Internet firewall, the Information Security (InfoSec) staff proposed starting with a firewall at the datacenter, both as of greater priority and as practice for the more politically sensitive perimeter firewall. With the support of University administration, firewall hardware was repurposed and the data center firewall project was launched.

Downtime was not to be a major resource for the project. How do you put a running datacenter behind an effective firewall? Three layers of analysis of the server ports in legitimate use, allowed InfoSec to construct ACLs to allow the good traffic, without blocking legitimate traffic.

2. Project Resources and Approach

There could be a number of ways to go from a 'hardened-host, deny known-bad network' datacenter subnet, to a 'default-deny network'. One of the better ways I'm acquainted with is exemplified by the work reported by Robert Winding at Notre Dame (Winding, R., 2005). At Notre Dame they erected an application-aware high-availability firewall (Sidewinder G2) around an empty datacenter. Migration to the protected network consisted of backing up a system, wiping it out, bringing it inside the firewall, rebuilding the system & applications, fingerprinting the file system with Tripwire, restoring the data and then

opening ports to allow any required traffic. Like the fictional Star Trek Transporter, this method not only rebuilds the target in a different place but gives you a record of the exact original configuration of the target, and provides a chance to filter out any 'microbes' that don't belong in the system.

This method was not available to our team. Our server engineering group was in the middle of several large implementations that could not be stopped to allow such a thorough process. Instead we opted for what might be called, the Iron Curtain approach – drop the firewall around the datacenter subnet. However, this approach required some careful analysis to make sure that required traffic could still get through. Extended downtime was not a resource provided for this project – neither for all systems as a whole nor any of the 178 individual servers.

Perhaps this approach and order of events seems a bit sloppy, but remember where we started, our goal and constraints.

Constraints:

- Low budget, repurposing hardware in hand.
- Low impact to server engineering projects.
- Low impact to service to the University.

Goal:

- Create a default-deny network around the datacenter.
- Develop methods to be reused in the Internet firewall.

Given our start as a patched-host, deny-known-bad network, the benefits of moving 178 hosts immediately to a protected network were so large that an approach where we were unable to verify/audit each host before protecting its network was a valid compromise.

3. Pre-Install: Analysis Methods

Multiple sources of data were used to construct and refine the firewall ACLs. The objective was to avoid service outage once we went to default-deny. The System Administrators (Sysadmins) provided the starting list of ports in use and metadata about

which should be reachable across the firewall from the intranet or Internet. This list was refined using Netflow data from the core router, which added unlisted ports. The Netflow data also showed that some ports listed were receiving no legitimate traffic and did not need to be exposed beyond the datacenter. Finally, after systems were moved over to the data center router/firewall in 'permit and log' mode, the firewall logs were analyzed to refine the ACLs to their final state before turning on default-deny.

3.1. Sysadmins Port/IP Table

The Manager of Server Engineering provided us with a table of ports in use by each server IP address and whether those ports should not be reachable at all or be reachable from the intranet or Internet. This very helpful list was necessarily incomplete, but listed the majority of application-related ports and a baseline understanding of what traffic was to be permitted to zones beyond the firewall. The list of ports came from the Sysadmins understanding of the primary applications on the machine and, where time permitted, a survey of the active ports using the netstat command on the Windows and Unix hosts. The table received from Server Engineering showed the system IP, port, protocol and whether the port required access from Intranet, Internet or was to be closed. Data for two sample servers from this list is found in Table 1, below.

Table 1. System Administrator initial port table

Legend	
underline	intranet
bold	Internet
plain	Data Center Only

Fileserver2 (7.7.88.90):	
Port	Service
111/tcp	sunrpc
135/tcp	loc-srv
139/tcp	netbios-ssn
867/tcp	unknown

13722/tcp	VeritasNetbackup
13782/tcp	VeritasNetbackup
13783/tcp	VeritasNetbackup

Mailfilter3	**(7.7.88.241):**
Port	Service
22/tcp	**ssh**
25/tcp	**smtp**
80/tcp	**http**
111/tcp	sunrpc
587/tcp	submission
898/tcp	unknown
4045/tcp	lockd
5432/tcp	postgres
13722/tcp	VeritasNetbackup
13782/tcp	VeritasNetbackup
13783/tcp	VeritasNetbackup
32771/tcp	sometimes-rpc5
32787/tcp	sometimes-rpc27

The Sysadmin's port list was converted to a MySQL database table with fields for system name, IP address, IP protocol, port, and access filter (0=none, 1=intranet, 2=Internet). This table was updated as additional information about required service ports was discovered. Table 2 shows the conversion of sample data from the Sysadmin's list to a MySQL table.

Table 2. MySQL version of port table

hostname	ip	proto	port	filter
fileserver2	7.7.88.90	tcp	111	0
fileserver2	7.7.88.90	tcp	135	0
fileserver2	7.7.88.90	tcp	139	1
fileserver2	7.7.88.90	tcp	867	0
fileserver2	7.7.88.90	tcp	1034	0
fileserver2	7.7.88.90	tcp	2707	0
fileserver2	7.7.88.90	tcp	5168	0
fileserver2	7.7.88.90	tcp	5169	0
fileserver2	7.7.88.90	tcp	13722	0
fileserver2	7.7.88.90	tcp	13724	0
fileserver2	7.7.88.90	tcp	13782	0
fileserver2	7.7.88.90	tcp	13783	0
mailfilter3	7.7.88.241	tcp	22	0
mailfilter3	7.7.88.241	tcp	25	0
mailfilter3	7.7.88.241	tcp	80	2
mailfilter3	7.7.88.241	tcp	111	0
mailfilter3	7.7.88.241	tcp	113	2

mailfilter3	7.7.88.241	tcp	587	0
mailfilter3	7.7.88.241	tcp	898	0
mailfilter3	7.7.88.241	tcp	4045	0
mailfilter3	7.7.88.241	tcp	5432	0
mailfilter3	7.7.88.241	tcp	5987	0
mailfilter3	7.7.88.241	tcp	5988	0
mailfilter3	7.7.88.241	tcp	13722	0
mailfilter3	7.7.88.241	tcp	13724	0
mailfilter3	7.7.88.241	tcp	13782	0
mailfilter3	7.7.88.241	tcp	13783	0
mailfilter3	7.7.88.241	tcp	17777	0
mailfilter3	7.7.88.241	tcp	18080	2
mailfilter3	7.7.88.241	tcp	28080	2
mailfilter3	7.7.88.241	tcp	32771	0
mailfilter3	7.7.88.241	tcp	32787	0
mailfilter3	7.7.88.241	tcp	36064	0

3.2. Netflow

Netflow data was used to get an actual measure of traffic to ports. Netflow is a Cisco system for reporting network traffic, based on flows. A network flow is defined as a chronologically related group of network packets, which have 5 criteria in common: source and destination IP addresses, source and destination ports and IP protocol. Flow data is maintained in a table on the network router. Network packets passing through the router that match existing flow criteria, increment counters for that flow within the table. New packets to the device that don't match an existing flow cause a new flow entry to be created in the table (Netflow, 2006).

Netflow sends the record of a flow via UDP to a Netflow collector. The flow record is output once the router determines a flow is complete.

A complete flow is determined by flow aging. When a new packet arrives for an existing flow, the router resets the flow-aging timer to its maximum value. When no new packets are received for the flow, the timer counts down. When the aging timer has run down, a flow record is output to the collector in the designated Netflow format. Optionally, a second timer can be configured to generate intermediate records for a long-running flow (Cisco, 2006).

Since the main router for the campus was directly connected to the data center subnet, Netflow could provide a good picture of the existing traffic to the servers to be protected. This router was running in hybrid CATOS/IOS mode, so there were two parts to the Netflow data export configuration. Commands for configuring Netflow export on the CATOS router and the IOS switch are shown in Table 3.

Table 3. Netflow export configuration

CATOS Router

```
ip flow-cache timeout active 5
```

```
ip flow-export version 5
ip flow-export destination 7.7.88.236 2055
```

IOS Switch

```
set mls flow full
set mls nde version 5
set mls nde 7.7.88.236 2055
set mls nde enable
```

For the CATOS router subsystem, the first command sets the intermediate timer to report on active (long-running) flows, such as a long file transfer, every 5 minutes. The second command specifies Netflow V5 format exports; the relevant data in this format are the flow criteria, plus packet and byte counts for the flow. The third command specifies the Netflow collector and udp port.

On the IOS switch subsystem, the Cisco Multiprotocol Layer Switching (mls) system exports flow data in Netflow formats. The first command sets flow accounting to the maximum information, the second specifies Netflow V5 format exports, the third command specifies the Netflow collector and udp port and the fourth command activates the flow data collection.

The flow-split tool from the ubiquitous open source flow-tools (Fullmer, 2005) was used to duplicate the flow data and send them to different collectors for summarization in different database tables according to source IP. The following command was set up in a startup script. It instructs flow-fanout to listen on port 2055 on all interfaces and send Netflow records received from the router IP address at 7.7.41.2 to collectors listening on localhost ports 2054 and 2056:

```
flow-fanout -p fanout.pid 0/7.7.41.2/2055 0/0/2054 0/0/2056
```

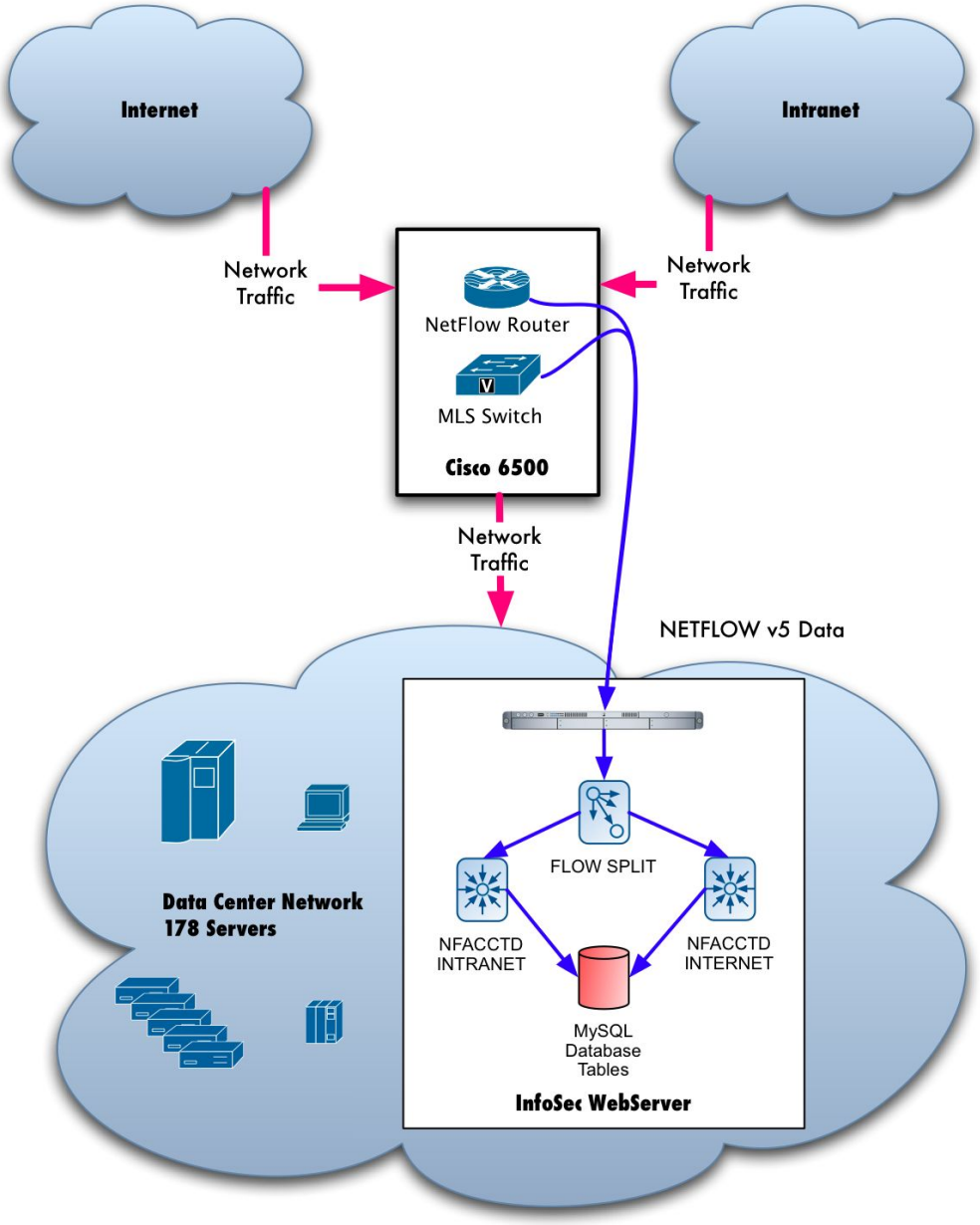
The collectors listening on those ports were nfacctd instances from the excellent open source flow accounting software, PMACCT (Lucente, P., 2006). There are many Netflow collectors in the open source space. The key feature for this project of the nfacctd collector in the multi-faceted PMACCT suite was its accounting capability. It had the ability to

additively summarize the packet counts and byte totals for all flows. However, the configuration of this software is flexible enough ignore (or combine) some flow sources. The configuration used in this project grouped the source IP addresses by zone and ignored the source ports, while calculating totals for all traffic sharing the same destination port and IP address. High totals quickly identify which ports are in active use versus traffic from scans and misconfigured devices.

The traffic totals are created using a SQL update call against a matching database record. If a matching record exists for the flow, the byte & packet counts from the current flow are added to that record. If a matching flow record does not already exist in the database, a new record is created.

Two nfacctd configuration files were used to divide our source (client) IPs into Internet and Intranet zones. These configurations are listed in Appendix 3. The nfacctd collectors were then able to summarize traffic from each source category by destination (server) port and IP. MySQL database tables on the Information Security web server were set up using the script included with the PMACCT suite. A port was opened in this host's firewall to receive the Netflow from the router interface. Flow tools' flow-fanout utility was used to clone the Netflow stream and direct a copy to the two nfacctd processes. A diagram of the flow export and collection is shown in Figure 1.

Figure 1. Network traffic and Netflow collection



The following commands were added to the startup script to make sure the nfacctd processes were started after the flow-fanout process:

```
sleep 15
nfacctd -f nfacctdIN.conf &
sleep 15
nfacctd -f nfacctdEX.conf &
```

The data collected in the MySQL database was then queried to filter out probes and scans and show which ports were actually being used for production services. A sample SQL query created by my partner, John Vannoy, for searching the database is shown in Table 4. This query combines the data on packet and byte counts with the database of the ports and security zones. The results for our sample servers are shown in Table 5 and 6.

Table 4. SQL query for traffic analysis

```
SELECT s.hostname, s.ip, s.proto, s.port, s.filter, SUM( intin.packets ) AS
int_pkts_in, SUM( intin.bytes ) AS int_bytes_in, SUM( extin.packets ) AS
ext_pkts_in, SUM( extin.bytes ) AS ext_bytes_in
FROM srvportfilt AS s
LEFT OUTER JOIN intracct060501 AS intin ON s.ip = intin.ip_dst
AND s.proto = intin.ip_proto
AND s.port = intin.dst_port
AND intin.mac_src = "0:0:0:0:0:0"
AND intin.mac_dst = "0:0:0:0:0:0"
AND intin.ip_src = "0.0.0.0"
AND intin.src_port =0
LEFT OUTER JOIN extracct060501 AS extin ON s.ip = extin.ip_dst
AND s.proto = extin.ip_proto
AND s.port = extin.dst_port
AND extin.mac_src = "0:0:0:0:0:0"
AND extin.mac_dst = "0:0:0:0:0:0"
AND extin.ip_src = "0.0.0.0"
AND extin.src_port =0
GROUP BY s.ip, s.proto, s.port
      ORDER BY INET_ATON( s.ip ) , s.proto, s.port
```

Table 5. Analysis for Fileserver

hostname	ip	proto	port	int_pkts_in	int_bytes_in	ext_pkts_in	ext_bytes_in
fileserver2	7.7.88.90	tcp	111	5	300	NULL	NULL
fileserver2	7.7.88.90	tcp	135	5	236	NULL	NULL
fileserver2	7.7.88.90	tcp	139	1428090	323782731	NULL	NULL
fileserver2	7.7.88.90	tcp	867	5	300	2	80
fileserver2	7.7.88.90	tcp	1034	NULL	NULL	4	160
fileserver2	7.7.88.90	tcp	2707	NULL	NULL	NULL	NULL
fileserver2	7.7.88.90	tcp	5168	NULL	NULL	NULL	NULL
fileserver2	7.7.88.90	tcp	5169	NULL	NULL	NULL	NULL
fileserver2	7.7.88.90	tcp	13722	12	720	3	136
fileserver2	7.7.88.90	tcp	13724	NULL	NULL	NULL	NULL
fileserver2	7.7.88.90	tcp	13782	5	300	1	40
fileserver2	7.7.88.90	tcp	13783	6	360	2	80

Table 6. Analysis for Mailfilter

hostname	ip	proto	port	int_pkts_in	int_bytes_in	ext_pkts_in	ext_bytes_in
mailfilter3	7.7.88.241	tcp	22	866304	1165197864	770784	67308615
mailfilter3	7.7.88.241	tcp	25	32	1920	945	37800
mailfilter3	7.7.88.241	tcp	80	191445	20537670	428574	164934372
mailfilter3	7.7.88.241	tcp	111	6	360	NULL	NULL
mailfilter3	7.7.88.241	tcp	113	2	120	78	3180
mailfilter3	7.7.88.241	tcp	587	6	360	3	120
mailfilter3	7.7.88.241	tcp	898	7	420	2	80
mailfilter3	7.7.88.241	tcp	4045	5	300	3	120
mailfilter3	7.7.88.241	tcp	5432	6	360	4	160
mailfilter3	7.7.88.241	tcp	5987	NULL	NULL	NULL	NULL
mailfilter3	7.7.88.241	tcp	5988	NULL	NULL	NULL	NULL
mailfilter3	7.7.88.241	tcp	13722	7	420	2	80
mailfilter3	7.7.88.241	tcp	13724	NULL	NULL	NULL	NULL
mailfilter3	7.7.88.241	tcp	13782	7	420	2	80
mailfilter3	7.7.88.241	tcp	13783	6	360	2	80
mailfilter3	7.7.88.241	tcp	17777	NULL	NULL	NULL	NULL
mailfilter3	7.7.88.241	tcp	18080	12355	1503459	NULL	NULL
mailfilter3	7.7.88.241	tcp	28080	369707	60248019	189696	29976732
mailfilter3	7.7.88.241	tcp	32771	6	360	66	2640
mailfilter3	7.7.88.241	tcp	32787	12	720	5	200
mailfilter3	7.7.88.241	tcp	36064	NULL	NULL	NULL	NULL

The Netflow data above allows us to see that what traffic is actually passing during the survey period on each system. Let's take a look at the fileserver example. The Sysadmins' port list recommended port 135 TCP be open to intranet. The data in Table 6 shows that port 135 is not getting any appreciable traffic. Port 135 may be closed to this system, saving the exposure of a historically vulnerable port.

In the mailfilter3 example, the data in Table 7 confirms the traffic expected on the low ports. However, it reveals that the filter service management port 18080 and end user application port 28080 were overlooked in the original list. Further, it shows that the management port need only be open to the intranet.

3.3. Firewall Logs

Our second key source for network traffic data was the firewall logs. Towards the end of our analysis phase, the datacenter subnet was transferred to a different Cisco 6509 IOS (not CATOS) router with a pair of Firewall Service Module (FWSM)

blades. This new 'firewall' router had all the servers and switches from the datacenter connected to it and it was connected to the old core router with a large Etherchannel (multiplexed Gigabit) link.

The FWSM firewalls were initially configured with a permit all and log at notification level' rule at the end of the ACL list. Firewall rules, developed from the Sysadmin's port/IP table and the Netflow data were added to 'permit specific traffic and log at debug level'. Our syslog server kept distinct files for the debug and notification level logs from the FWSM.

With this syslog setup, any traffic that was not permitted by an ACL would trigger the 'permit all' rule and show up in the notification logs. The entries in the notification logs were helpful with two kinds of traffic. The first type was that traffic that was below our thresholds for detection with Netflow. Small amounts of TCP traffic, which actually pass data, are easily distinguishable from scans using this log. The second type of traffic that is easier to distinguish in these logs is traffic to ports higher than 1024. Cisco ACLs, in contrast to Netflow, are able to indicate which end of the network began the conversation. With Netflow analysis recognizing the server end of the conversation was done by trying to spot the port associated with a known application or well known service port. High port traffic, such as that to port 18080 on mailfilter3 could easily be FTP traffic, for example, if you don't know which end started the conversation.

At this stage in project installing new firewall rules became an iterative process: Install rules to allow newly discovered traffic, search the notification logs for traffic that doesn't look like a scan and install new rules to allow it.

3.4. Going Live

As the go-live date approached for our default-deny datacenter firewall, the data in the notification logs seemed to be all scans and traffic we did not intend to permit. Moving to default deny was as simple as changing the 'permit all and log at

notification' rule at the bottom of the ACL list on the FWSM to a 'deny all and log at notification' rule.

How well did we do at finding the traffic to 178 servers and some 3180 network ports? Very well! We only had two items that were not discovered; neither system received data during our discovery period. One was a financial reporting system that was accessed quarterly. The other was an abandoned syslog server that had only one remote system sending authentication logs; no one logged in on the remote system during our measurement period.

The InfoSec team was prepared for worse. We had created a self-service security-information web application that would allow department technical leads to check the firewall to see if it was blocking a client IP address. This application would provide enough detail that InfoSec could quickly open a port that we had overlooked. Would this system get used much for fixing ports we missed? Would it have a role in operational changes to the firewall once it was up and running?

4. Post-Install: Operational Methods

Customers we spoke with during the planning and analysis phases of the datacenter firewall project were very supportive of having the firewall put in place. However, we thought that once a firewall was in place, it would generate a lot of questions along the line of "isn't the firewall doing something to stop my computer from getting to its data?" Chasing the 'wild hares' among those questions could absorb a lot of the security staff resources, plus hurt customer service and security. For this reason, we wanted to construct a self-service application that would allow desktop technical support staff to answer this question directly. The system would reliably report whether any client IP address had traffic blocked recently, and if so, what ports and IP addresses were blocked. If there was a block, leads could request consulting from the security team. If there was no block, they could go straight to troubleshooting the client computer.

How did InfoSec collect and serve this information? As seen above, all the denied traffic on the firewall was hitting a rule that logged the denies at the notification level. A script read these logs and input them to a database that could be searched with a pre-constructed query by someone who didn't know regular expression syntax or SQL queries. Technical leads were trained on the use of the web pages and given authenticated access to them over an SSL connection.

To get the denied logs into the database, they were first selected by source and sent to a named pipe using syslog-ng. These output from this pipe was parsed and uploaded to a MySQL database with the fisq.pl script by Activeworx (2004). The MySQL database was queried by a PHP script constructed using open source ADODB technology (Lim, J., 2004) on the back end via the freeware PHAKT plug-in (Interakt, 2005) for the web page editor. The web page containing the PHP script was authenticated and served over SSL via Apache.

The relevant syslog-ng configuration for logging different levels and logging to the named pipes is shown in table 7.

Table 7. Configuration for syslog-ng files and pipes

```
# Selected syslog-ng log entries related to logging from the datacenter
firewall

# label the source for log entries received on IP 7.7.88.5 as s_udp_dc
source s_udp_dc {
                        udp(ip(7.7.88.5));
                };

# label the log entries received at notice an above as f_notice
filter f_notice  { level (notice .. emerg);  };

# label the log entries received from the firewall as f_dc_pix
# Range includes failover partner on adjacent IP address
filter f_dcPIX  { netmask (7.7.88.16/255.255.255.252); };


# files to receive firewall logs
destination r_dcPIX
        { file ("/var/log/fwlog/dcPIX.$YEAR$MONTH$DAY" owner("root")
group("log") perm(0660)); };
destination r_dcPIXacl
        { file ("/var/log/fwlog/dcPIXacl.$YEAR$MONTH$DAY" owner("root")
group("log") perm(0660)); };

# pipe to receive firewall logs
```

```
destination r_dcPIX_deny2db
        { pipe("/var/run/dcpix.pipe" owner("root") group("log") perm(0660));
};

#
# log ALL DcPix messages to local file
log {
        source (s_udp_dc);
        filter (f_dcPIX);
        destination (r_dcPIX);
        };

# separate out ALL notice level DcPix traffic to local file
log {
        source (s_udp_dc);
        filter (f_dcPIX); filter (f_notice);
        destination (r_dcPIXacl);
        };

# log dcPIX notice traffic to pipe for fisq
log { source (s_udp_dc);
        filter (f_dcPIX); filter (f_notice);
        destination (r_dcPIX_deny2db);
        };
```

A brief HOWTO for the setup of the fisq.pl script is covered in a GIAC paper (Plytas, G., 2004, p.23). However, with our strategy of varying the log levels between explicit permit rules (debug) and final 'deny and log' at notification level meant that the fisq.pl code needed some adjustment to match our approach. A classic PIX firewall uses log messages of the format PIX-n-NNNNNN where n is a numeric syslog level and NNNNNN is the message ID (Cisco, 2005. p. 1-15). The fisq script uses a fixed digit (the Cisco default log level for that message) for the numeric level. Therefore, to log our non-standard level messages to the database we need to make a modification to the script to accept any digit for the level. In addition, the FWSM does not use the same log message labels as the PIX, so fisq.pl had to be modified for use on this project. The modified program listing is in Appendix A. The following diff excerpt from the critical 106100 'permit or deny by acl' message shows the required change:

```
233,234c238,239
<   elsif (/^(\w+)\s+(\d+)\s+([\d:]+)\s(\S+)\s\%PIX-4-106100:.+
(permitted|denied) (tcp|udp)/){
<     /^(\w+)\s+(\d+)\s+([\d:]+)\s(\S+)\s\%PIX-4-106100:.+
(permitted|denied) (tcp|udp)
(\w+)\/([\d\.]+)\((\d+)\).+\/([\d\.]+)\((\d+)\)/i
;
```

```
      ---
      >   elsif (/^(\w+)\s+(\d+)\s+([\d:]+)\s(\S+)\s\%FWSM-\d-106100:.+
(permitted|denied) (tcp|udp)/){
      >     /^(\w+)\s+(\d+)\s+([\d:]+)\s(\S+)\s\%FWSM-\d-106100:.+
(permitted|denied) (tcp|udp) (\w+)\/([\d\.]+)\((\d+)\).+\/([\d\.]+)\((\d+)\)/
```

Leveraging the fisq database notification log of blocked connections has been powerful in allowing authorized field support to answer users questions about whether the firewall was blocking their access. This has meant less stress for customers, field support, Sysadmins and more time to focus on security analysis for InfoSec. In addition, the same system is probably most heavily used by Sysadmins and developers in their process of building of new systems. Often, vendor documentation is vague about inbound/outbound permits required, is generally inscrutable or assumes that all traffic does not pass through a firewall. Once a system goes online and the initial port-open requests have been fulfilled, the Sysadmins are able to use the web page to empirically test what ports are still being blocked by the firewall. By utilizing the firewall check system described above, the Sysadmins and developers can then make very concrete requests for additional open ports, regardless of the shortcomings of the product documentation. Figure 2 shows the information request screen, Figure 3 shows a negative result (no blocks) and figure 4 shows a positive result (created by telnet to port 667 on mailfilter3 from outside the firewall).

Figure 2. Technicians firewall query screen

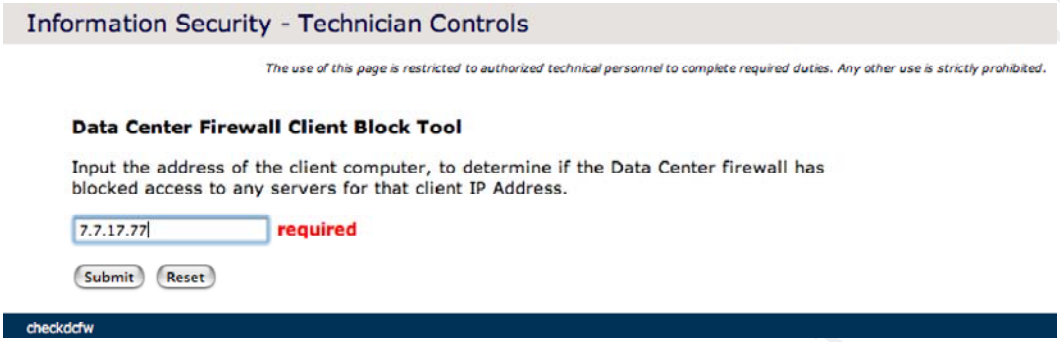


Figure 3. Clear negative result => troubleshoot client system.

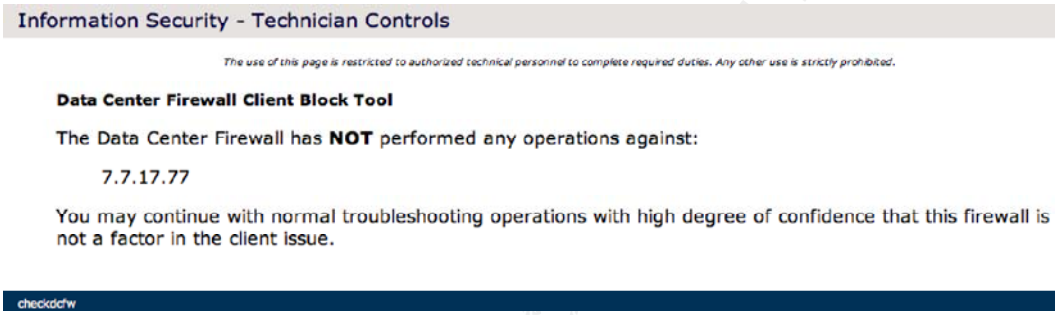
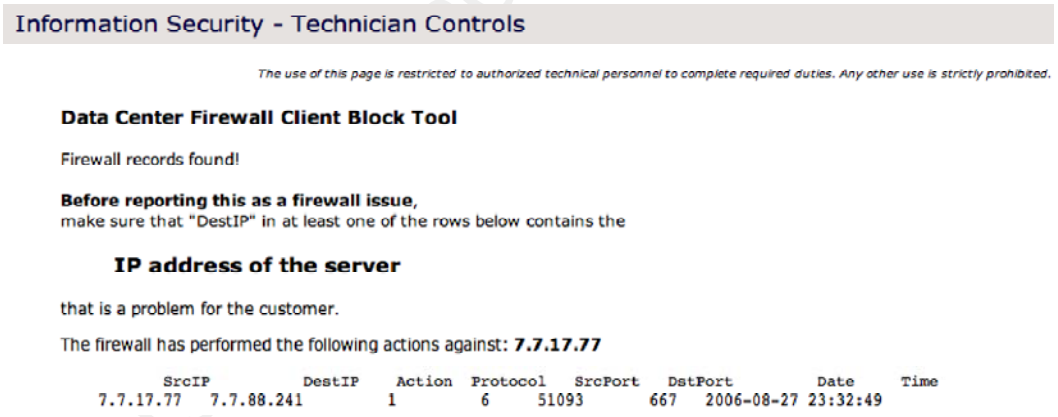


Figure 4. Positive result detail => consult security team



We are now on our third firewall installation and have created these 'firewall check' pages for each firewall, since they have been very helpful to the support staff. The query used by the web page against the standard fisq table structure is shown in the text of the web page in Appendix B.

5. Lessons Learned

5.1. General Analysis Method

I don't recommend our "Iron Curtain" method of dropping a firewall around not quite trusted hosts, if you can use the "Star Trek Transporter" backup, wipe and rebuild method. You'll have a lot more confidence in your systems with the latter. However, for those with similar constraints, a summary description of our now standardized approach follows:

1. Divide the network into zones, e.g. Internet, DMZ, User LAN, Wireless, Residences, etc.
2. Those who know the systems to be protected provide a table of which ports should be made available to which zones. This works best if the data for each 'port' is provided in easily machine convertible format, like CSV text, as: IP address, protocol, port, zone.
3. Audit the accuracy of that table with a traffic accounting system installed near the servers to be protected (we continue to use nfacctd). It's likely there will legitimate traffic discovered that the most knowledgeable people left off the list.
4. Construct preliminary ACLs to permit traffic based on the above analysis. Place a temporary 'permit everything' ACL just in front of your final rule (which should be 'deny all'). Have the firewall log traffic permitted by the preliminary ACLs at debug and traffic hitting the temporary 'permit everything' ACL at a higher log level, like notification. Direct syslog to divert the higher level log entries to a different file.
5. Audit your preliminary ACLs by summarizing the log that shows the traffic allowed by your 'permit everything' ACL. Anything in this log is either unwanted traffic, for example scans and probes, or it is production traffic that was missed earlier.

6. Add/change ACLs according to your findings in step 5 and repeat step 4.
7. When results from step 6 are only unwanted traffic, make sure you are ready to remove the 'permit everything' ACL: customers have been notified of the cutover date, technical staff are trained in how to follow up firewall 'complaints' and differentiate between client computer problems and 'unintended' firewall blocks.
8. Delete or disable the permit-all rule.

5.2. Problems and Benefits

The method used to measure traffic and move the datacenter behind a firewall has some issues that need to be considered. Foremost is that idea that the hosts have been exposed for years are not entirely trusted. It would have been better to backup, wipe, rebuild, fingerprint and restore data for each host as it entered the protected DMZ zone. Second, there was not time to do the planning required to move hosts that were not customer-facing into a separate network apart from the datacenter DMZ. Mixing these hosts goes against best-practice. Separating them is a project that remains on hold in the face of other priorities.

The traffic analysis methods used share one main weakness: they ignore high-granularity processes. This issue could be addressed by either lengthening the time of the traffic measurement period, or by doing a more exhaustive search of running processes on the servers. However, despite our constraints on both methods of finding running network applications, I feel that our results were strong enough to recommend these methods. The fact that we had a major production application and an abandoned syslog server that went unaccounted is targeted to be addressed in a server inventory project at a future date.

The major benefit of these projects has been that the community very pleased with low impact of the first firewall

project. This positive result created support for the more perimeter firewall project, which would have a greater impact on customer computers. The methods learned in analysis and operation have been useful in both the perimeter and our secondary datacenter firewalls projects.

Routine use of firewall log web interface to expedite the proper port-open requests during server-build process has benefited the System Administrators. They have evangelized the web interface to user technical leads and to our administrative software developers. Overall, the web interface has helped end-user service by either 1) pointing immediately to the firewall as an impediment or 2) immediately exonerating the firewall allowing technicians to focus on client-system issues. We highly recommend appropriately controlled security self-service applications as a way to provide better service while allowing the security team to focus on analysis and implementation.

There is room to further leverage the infrastructure created for the firewall traffic analysis. For example, on our back burner is the thought that we'll use the Netflow data we continue to collect to supplement our IDS as a security analysis tool. This will require creating SQL queries tuned to our finer-grained security zones (there are now 11 zones, not just intranet and Internet). We would also need to change the nfacctd configuration to account down to the pure flow level to show data for individual clients. A high packet count, but low byte count to a service port could show authentication attacks, for example.

6. Appendices

6.1. Appendix A. fisq.pl modifications

The modified fisq.pl script, featuring two main changes: 1) using a digit regexp instead of the fixed numeric value in the original script and 2) substituting FWSM for PIX in the script.

```
#!/usr/local/bin/perl
```

```
# By Activeworx, Inc.
# fisq.pl
# Copyright (c) 2003-2004 Activeworx <fisq@activeworx.org>
# Firewall-Sql Import Script
# Version 0.9.1 Beta
# web: www.activeworx.org
# email: fisq@activeworx.org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

## Changed all occurrences of %FWSM to %FWSM globally. 050615 kcary
## Changed 106100 messages to use FWSM-6 instead of FWSM-4. 050615 kcary
## Changed 106100 messages to use FWSM-\d instead of FWSM-6. 050617 kcary
## Removed code for non-PIX and irrelevant selects 060513 kcary

use Getopt::Std;
use FileHandle;
use DBD::mysql;
use DBI;
use strict;

use vars qw( %opts $dst_ip $dst_port $src_ip $src_port $type $status
             $year $mon $day $verbose $infile $outfile
             $dbh $db_server $db_username $db_password $db_database
             $timestamp $icmptype $icmpcode $hid $daemon
             $noutfile $proto $action $tmpval $ofh $fisq_pid
             $sql $result $line $tail $ifh $eventmsg $static_hid $logmsg
             $Daemon $pidfile $sent_data $rsvd_data $log_count
            $pass_count $read_count
);
my $prog_ver = "0.9.1 Beta";

my %month_array = (
        'Jan' => 1,
        'Feb' => 2,
        'Mar' => 3,
        'Apr' => 4,
        'May' => 5,
        'Jun' => 6,
        'Jul' => 7,
        'Aug' => 8,
        'Sep' => 9,
        'Oct' => 10,
        'Nov' => 11,
        'Dec' => 12);

$db_server   = "137.159.8.236";       #IP Address of Database Server
$db_username = "logdata";             #Username for Database
$db_password = "flogbata";            #Password for Database
$db_database = "dcpix";                       #Database name
$eventmsg    = "true";                #Log Event/Admin Messages
$logmsg      = "true";                #Traffic Log Messages
$Daemon      = "0";                   #Fork Process
$infile      = "/var/run/dcpix.pipe";
$outfile     = "/data/fwlog/fisq-dc-output.txt";
$noutfile    = "/data/fwlog/fisq-dc-parse.txt";
$verbose     = "1";                   #1-verbose : 0-non-verbose
$tail        = "false";               #true-tail file : false-Cat file
$pidfile     = "/var/run/fisqdc.pid";
$static_hid  = "";                    #staticly set the host id

####################################
# Don't change anything below here #
####################################

getopts('?Deltvi:h:u:p:', \%opts) || &prog_usage;
```

```
if ($opts{'?'}){
  &prog_usage;
  exit;
}
if ($opts{'e'}){
  $eventmsg = "true";
}
if ($opts{'D'}){
  $Daemon = "true";
}
if ($opts{'l'}){
  $logmsg = "true";
}

if ($opts{'h'}){
  $static_hid = $opts{'h'};
}

if ($opts{'i'}){
  $infile = $opts{'i'};
}

if ($opts{'p'}){
  $db_password = $opts{'p'};
}

if ($opts{'t'}){
  $tail = "true";
}
if ($opts{'u'}){
  $db_username = $opts{'u'};
}
if ($opts{'v'}){
  $verbose = "0";
}

$read_count = $log_count = $pass_count = 0;
# ---------- Prepare inputs and outputs -------
print "* Process Event Messages       : $eventmsg\n" unless $verbose;
print "* Process Traffic Messages     : $logmsg\n" unless $verbose;
print "* Opening No-Parse output file : $noutfile\n" unless $verbose;
my $nofh = new FileHandle $noutfile, "w" || die "Unable to open file : $noutfile $!";

#print "* Opening Database Connection  : $db_database\n" unless $verbose;
#$dbh = DBI->connect("DBI:mysql:database=$db_database;host=$db_server",$db_username,$db_password)
|| die "Error Connecting to Database!\n";

# ------- Fork Process -------

if ($Daemon){

  print "Forking process please wait.\n" unless $verbose;
        $verbose = 1;

  FORK: {
    if ($fisq_pid = fork) {
      # Parent here
      # child process pid is available in $smd_pid
      my $ofh = new FileHandle $pidfile, "w" || die "Can't open $pidfile $!";
      print $ofh $fisq_pid;
      close $ofh;
      exit;
    }
    elsif (defined $fisq_pid) { # $smd_pid is zero here if defined
      # child here
      # parent process pid is available with getppid
    }
    elsif ($! =~ /No more process/) {
      # EAGAIN, supposedly recoverable fork error
      sleep 5;
      print "redo\n";
      redo FORK;
    }
    else {
      # weird fork error
      die "Can't fork: $!\n";
    }
```

```
  }
  $0 = "fisq($infile)";
}

# --------- Read files -------

if ($tail eq "true") {
  print "* Opening tail input file      : $infile\n" unless $verbose;
  open(STDIN,"tail -f $infile|") || die "Unable to tail : $infile - $!";
  while(1) {
    print "* Opening Database Connection  : $db_database\n" unless $verbose;
    until ($dbh = DBI-
>connect("DBI:mysql:database=$db_database;host=$db_server",$db_username,$db_password)) {
      print "Can't connect: $DBI::errstr. \nPausing before retrying.\n" unless $verbose;
      sleep (5);
    }
    eval {
      print "* Connected to Database        : $db_database\n" unless $verbose;
      while (<STDIN>) {
        &parse_logs;
      };
      exit;
    }
  }
}
else {
  print "* Opening input file           : $infile\n" unless $verbose;
  $ifh = new FileHandle $infile,  "r" || die "Unable to open file : $infile - $!";
  while(1) {
    print "* Opening Database Connection  : $db_database\n" unless $verbose;
    until ($dbh = DBI-
>connect("DBI:mysql:database=$db_database;host=$db_server",$db_username,$db_password)) {
      print "Can't connect: $DBI::errstr. \nPausing before retrying.\n" unless $verbose;
      sleep (5);
    }
    eval {
     print "* Connected to Database        : $db_database\n" unless $verbose;
     while (<$ifh>) {
        &parse_logs;
     };
     &cleanup;
     exit;
    }
  }
}

################
sub cleanup {
  print "\n* Read lines                   : $read_count\n" unless $verbose;
  print "* Logged lines                 : $log_count\n" unless $verbose;
  print "* Passed lines                 : $pass_count\n\n" unless $verbose;
  print "* Closing input file           : $infile\n" unless $verbose;
  undef $ifh;
  print "* Closing No-Parse output file : $noutfile\n" unless $verbose;
  undef $nofh;
}

################
sub parse_logs {
  &reset_vars;
  #print "$_\n";
  $read_count++;
  #PIX 6.3 ICMP Message
  #Feb 10 00:00:41 126.44.64.1 :May 09 22:30:50 UTC: %FWSM-4-106100: access-list inside_access_in
permitted icmp inside/126.44.33.29(0) -> outside/126.144.11.131(8) hit-cnt 10 (300-second
interval)
  if (/^(\w+)\s+(\d+)\s+([\d:]+)\s(\S+)\s\%FWSM-\d-106100:.+ (permitted|denied) icmp/){
    /^(\w+)\s+(\d+)\s([\d:]+)\s(\S+)\s\%FWSM-\d-106100:.+ (permitted|denied) icmp
(\w+)\/([\d\.]+).+\/([\d\.]+)/i;

      $mon = $month_array{$1};
      $day = $2;
      $hid = $4;
      $timestamp = "$year\/$mon\/$day $3";
      $status = set_action($5);
      $type = set_direction($6);
      $proto = "1";
      $src_ip = $7;
```

```
      $src_port = "0";
      $dst_ip = $8;
      $dst_port = "0";
      &db_log;
  }

  #PIX 6.3 UDP/TCP Message
  # Feb 10 00:00:45 pix :May 09 22:30:54 UTC: %FWSM-4-106100: access-list outside_access_in
permitted udp outside/10.10.1.2(123) -> inside/10.40.3.30(123) hit-cnt 2 (300-second interval)
  elsif (/^(\w+)\s+(\d+)\s+([\d:]+)\s(\S+)\s\%FWSM-\d-106100:.+ (permitted|denied) (tcp|udp)/){
    /^(\w+)\s+(\d+)\s+([\d:]+)\s(\S+)\s\%FWSM-\d-106100:.+ (permitted|denied) (tcp|udp)
(\w+)\/([\d\.]+)\((\d+)\).+\/([\d\.]+)\((\d+)\)/i;

      $mon = $month_array{$1};
      $day = $2;
      $hid = $4;
      $timestamp = "$year\/$mon\/$day $3";
      $status = set_action($5);
      $proto = set_proto($6);
      $type = set_direction($7);
      $src_ip = $8;
      $src_port = $9;
      $dst_ip = $10;
      $dst_port = $11;
      &db_log;
  }

  #PIX TCP/UDP Denied
  #Apr  1 16:08:50 pix Apr 01 2003 14:40:31: %FWSM-2-106007: Deny inbound UDP from 10.1.1.1/32774
to 10.1.2.1/53 due to DNS Query
  elsif (/^(\w+)\s+(\d+) ([\d:]+) (\S+).+ \%FWSM-\d-10600(6|7): (\w+) (inbound|outbound)
(TCP|UDP)/) {
    /^(\w+)\s+(\d+) ([\d:]+) (\S+).+ \%FWSM-\d-10600(6|7): (\w+) (inbound|outbound) (TCP|UDP)
from ([\d\.]+)\/(\d+) to ([\d\.]+)\/(\d+)/i;
      $mon = $month_array{$1};
      $day = $2;
      $hid = $4;
      $timestamp = "$year\/$mon\/$day $3";
      $status = set_action($6);
      $type = set_direction($7);
      $proto = set_proto($8);
      $src_ip = $9;
      $src_port = $10;
      $dst_ip = $11;
      $dst_port = $12;
      &db_log;
  }


  #Pix tcp/udp
  elsif (/^(\w+)\s+(\d+) ([\d:]+) (\S+).+ \%FWSM.* Deny (inbound|outbound) (tcp|udp)/) {
    /^(\w+)\s+(\d+) ([\d:]+) (\S+).+ \%FWSM.* Deny (inbound|outbound) (tcp|udp) src
outside:([\d\.]+)\/(\d+) dst inside:([\d\.]+)\/(\d+)/i;
      $mon = $month_array{$1};
      $day = $2;
      $timestamp = "$year\/$mon\/$day $3";
      $hid = $4;
      $type = set_direction($5);
      $proto = set_proto($6);
      $src_ip = $7;
      $src_port = $8;
      $dst_ip = $9;
      $dst_port = $10;
      $status = "1";
      &db_log;
  }


  #PIX
  #Apr  1 16:09:48 pix %FWSM-4-106023: Deny udp src inside:149.122.196.106/1030 dst
outside:172.16.8.27/427 by access-group "100"
  elsif (/^(\w+)\s+(\d+) ([\d:]+) (\S+) \%FWSM-\d-106023: (\w+) (tcp|udp)/) {
    /^(\w+)\s+(\d+) ([\d:]+) (\S+) \%FWSM-\d-106023: (\w+) (tcp|udp) src (\w+):([\d\.]+)\/(\d+)
dst \w+:([\d\.]+)\/(\d+)/i;
      $mon = $month_array{$1};
      $day = $2;
      $timestamp = "$year\/$mon\/$day $3";
      $hid = $4;
```

```
      $status = set_action($5);
      $proto = set_proto($6);
      $type = set_direction($7);
      $src_ip = $8;
      $src_port = $9;
      $dst_ip = $10;
      $dst_port = $11;
      &db_log;

  }

  #PIX TCP/UDP
  #Apr  1 15:32:38 pix Apr 01 2003 14:04:19: %FWSM-2-106001: Inbound TCP connection denied from
10.1.1.1/58740 to 10.1.2.10/113 flags SYN  on interface core
   elsif (/^(\w+)\s+(\d+) ([\d:]+) (\S+).+ \%FWSM-\d-106001: (\w+) (TCP|UDP)/) {
    /^(\w+)\s+(\d+) ([\d:]+) (\S+).+ \%FWSM-\d-106001: (\w+) (TCP|UDP) connection denied from
([\d\.]+)\/(\d+) to ([\d\.]+)\/(\d+)/i;
      $mon = $month_array{$1};
      $day = $2;
      $timestamp = "$year\/$mon\/$day $3";
      $hid = $4;
      $type = set_direction($5);
      $proto = set_proto($6);
      $src_ip = $7;
      $src_port = $8;
      $dst_ip = $9;
      $dst_port = $10;
      $status = "1";
      &db_log;

  }

# Not interesting for checkfw purposes 060213, log, but change status to 4
# PIX TCP/UDP
# May 12 11:33:18 10.241.15.20 %FWSM-6-106015: Deny TCP (no connection) from 126.145.65.133/1371
to 126.144.66.38/1177 flags PSH ACK  on interface outside
   elsif (/^(\w+)\s+(\d+) ([\d:]+) (\S+) \%FWSM-\d-106015: (\w+) (TCP|UDP)/) {
    /^(\w+)\s+(\d+) ([\d:]+) (\S+) \%FWSM-\d-106015: (\w+) (TCP|UDP) \(no connection\) from
([\d\.]+)\/(\d+) to ([\d\.]+)\/(\d+).+interface (\w+)/i;
      $mon = $month_array{$1};
      $day = $2;
      $timestamp = "$year\/$mon\/$day $3";
      $hid = $4;
#     $status = set_action($5);
      $status = 4;
      $proto = set_proto($6);
      $src_ip = $7;
      $src_port = $8;
      $dst_ip = $9;
      $dst_port = $10;
      $type = set_direction($11);
      &db_log;

  }

  #PIX ICMP
  #Apr  1 15:32:38 pix %FWSM-4-106023: Deny icmp src outside:10.1.1.1 dst inside:10.1.2.10 (type
0, code 0) by access-group "101"
  elsif (/^(\w+)\s+(\d+) ([\d:]+) (\S+) \%FWSM-\d-106023: (\w+) icmp/) {
    /^(\w+)\s+(\d+) ([\d:]+) (\S+) \%FWSM-\d-106023: (\w+) icmp src (\w+):([\d\.]+) dst
\w+:([\d\.]+)/i;# \(type (\d+), code (\d+)/i;
      $mon = $month_array{$1};
      $day = $2;
      $timestamp = "$year\/$mon\/$day $3";
      $hid = $4;
      $status = set_action($5);
      $type = set_direction($6);
      $src_ip = $7;
      $dst_ip = $8;
      #$icmptype = $9;
      #$icmpcode = $10;
      $proto = "1";
      &db_log;
  }


    #PIX Admin Messages
    # elsif (/^(\w+)\s+(\d+) ([\d:]+) (\S+) *.*PIX-5-111(\d+):/) {
```

```
    if (/^(\w+)\s+(\d+) ([\d:]+) (\S+) *.*PIX-5-111(\d+):/) {
      /^(\w+)\s+(\d+) ([\d:]+) (\S+) *.*PIX-5-111(\d+): (.*)/i;
      $mon = $month_array{$1};
      $day = $2;
      $timestamp = "$year\/$mon\/$day $3";
      $hid = $4;
      $status = set_severity($6);
      $type = "4";

      &db_admin_log;
    }

    else{
    &no_parse;
    }
  }
  else{
    &no_parse;
  }
}

####################
#
sub reset_vars {
  #print "* Resetting variables\n" unless $verbose;
  $dst_ip = $dst_port = $src_ip = $src_port = $type = $status = $sent_data = $rsvd_data =
  $mon = $day = $timestamp = $icmptype = $icmpcode = 0;
  $hid = $static_hid;
  #$year = 2004;
  #$year = 2005;
  $year = 2006;
}

####################
#
sub set_direction{
       my ($sDir) = @_;
       my ($sType);
       if    (lc($sDir) eq "inbound")  { $sType = "0"}
       elsif (lc($sDir) eq "untrust")  { $sType = "0"}
       elsif (lc($sDir) eq "outside")  { $sType = "0"}
       elsif (lc($sDir) eq "trust")    { $sType = "1"}
       elsif (lc($sDir) eq "dmz")      { $sType = "1"}
       elsif (lc($sDir) eq "outbound") { $sType = "1"}
       elsif (lc($sDir) eq "inside")   { $sType = "1"}
       elsif (lc($sDir) eq "outg")     { $sType = "1"}
       elsif (lc($sDir) eq "drop")     { $sType = "2"}
       else                            { $sType = "0"}
       return $sType;
}
####################
#
sub set_proto{
       my ($sProto) = @_;
       my ($sType);
       if    (lc($sProto) eq "icmp") { $sType = "1"}
       elsif (lc($sProto) eq "tcp")  { $sType = "6"}
       elsif (lc($sProto) eq "udp")  { $sType = "17"}
       else                          { $sType = "0"}

       return $sType;
}
####################
#
sub set_action{
       my ($sAction) = @_;
       my ($sType);
       if    (lc($sAction) eq "permit")    { $sType = "0"}
       elsif (lc($sAction) eq "permitted") { $sType = "0"}
       elsif (lc($sAction) eq "deny")      { $sType = "1"}
       elsif (lc($sAction) eq "denied")    { $sType = "1"}
    elsif (lc($sAction) eq "drop")      { $sType = "2"}
    elsif (lc($sAction) eq "vpn")       { $sType = "3"}
       elsif (lc($sAction) eq "tunnel")    { $sType = "3"}
       else                                { $sType = "0"}

       return $sType;
}
```

```
sub set_severity{
       my ($sSev) = @_;
       my ($sType);
       if    (lc($sSev) eq "high")     { $sType = "1"}
       elsif (lc($sSev) eq "notice")   { $sType = "4"}
       else                            { $sType = "4"}

       return $sType;
}
####################
# Line was not parsed, output line to a file
sub no_parse{
    $pass_count++;
    print $nofh $_."\n";
    print "-* No Parse *- $_\n" unless $verbose;
}

####################
# Prep Log event to mysql Database
sub db_log{
  if ($logmsg){
    if ($static_hid){
      $hid = $static_hid
    }
    $sql = "INSERT INTO fwlogs (hid, timestamp, direction, action, proto, srcip, srcport, dstip,
dstport, sentdata, rsvddata) VALUES (\"$hid\",
\"$timestamp\",\"$type\",\"$status\",\"$proto\",inet_aton(\"$src_ip\"),\"$src_port\",inet_aton(\"
$dst_ip\"),\"$dst_port\", \"$sent_data\", \"$rsvd_data\")";
    print "$hid $timestamp d=$type s=$status p=$proto $src_ip $src_port $dst_ip $dst_port\n"
unless $verbose;
    &sql_insert;
  }
}

####################
# Prep Log Admin message to mysql Database
sub db_admin_log{
  if ($static_hid) {
    $hid = $static_hid
  }
  if ($status eq "") {
    &no_parse;
  }
  else {
    print "$eventmsg\n";
    $sql = "INSERT INTO fweventlogs (hid, timestamp, severity, message) VALUES
(\"$hid\",\"$timestamp\",\"$type\",\"$status\")";
    print "$hid $timestamp $type $status\n" unless $verbose;
    &sql_insert;
  }
}

####################
#Log event to Mysql

sub sql_insert{
  $result = $dbh->do($sql) || print $nofh "DB Error $logmsg on line $line: \"$sql\"";
   $log_count++;
}

####################
sub file_log{
#print $ofh "$hid $year\/$mon\/$day $hour:$min:$sec $type Permit $proto $src_ip $src_port $dst_ip
$dst_port\n";
}

####################
sub prog_usage {
  print "fisq v$prog_ver\n";
  print "By Activeworx (fisq\@activeworx.org, www.activeworx.org)\n";
  print "Usage: $0 -[?elvupD] -i <input file> -h <Host ID>\n";
  print "Options:\n";
  print " -?         Help and Exit\n";
  print " -e         Enable Event Message Logging\n";
  print " -l         Enable Traffic Message Logging\n";
  print " -i [file]  Input file name\n";
  print " -v         Verbose output\n";
  print " -h <name>  Host ID\n";
```

```
  print " -u         DB Username\n";
  print " -p         DB Password\n";
  print " -D         Fork process\n";
  exit;
}
```

6.2. Appendix B Firewall Check Web Page

Web Page used for querying the fisq database table for blocks against a specific IP. The page accepts a post argument on SSL for authenticated clients:

```
<?php
//Connection statement
require_once('../Connections/dcpix.php');

//Aditional Functions
require_once('../includes/functions.inc.php');

// begin Recordset
$maxRows_Recordset1 = 50;
$pageNum_Recordset1 = 0;
if (isset($_GET['pageNum_Recordset1'])) {
  $pageNum_Recordset1 = $_GET['pageNum_Recordset1'];
}
$startRow_Recordset1 = $pageNum_Recordset1 * $maxRows_Recordset1;
$colname__Recordset1 = '2308903175';
if (isset($_POST['clientip']))  {
  $colname__Recordset1 = sprintf('%u', ip2long($_POST['clientip']));
}
$query_Recordset1 = sprintf("SELECT * FROM fwlogs WHERE srcip = %s and action = 1 ORDER BY
`timestamp` DESC",  $colname__Recordset1)
;
$Recordset1 = $dcpix->SelectLimit($query_Recordset1, $maxRows_Recordset1, $startRow_Recordset1)
or die($dcpix->ErrorMsg());
if (isset($_GET['totalRows_Recordset1'])) {
  $totalRows_Recordset1 = $_GET['totalRows_Recordset1'];
} else {
  $all_Recordset1 = $dcpix->SelectLimit($query_Recordset1) or die($dcpix->ErrorMsg());
  $totalRows_Recordset1 = $all_Recordset1->RecordCount();
}
$totalPages_Recordset1 = (int)(($totalRows_Recordset1-1)/$maxRows_Recordset1);
// end Recordset

//PHP ADODB document - made with PHAkt 3.6.5
?><!-- InstanceBegin template="/Templates/OmniPopUp.dwt.php" codeOutsideHTMLIsLocked="false" -->
<HEAD>
<!-- InstanceBeginEditable name="doctitle" -->
<TITLE> Data Center Firewall Blocked Client Result</TITLE>
<!-- InstanceEndEditable -->
<META http-equiv="Content-Type" content="text/html; charset=utf-8" />
<META http-equiv="Content-Language" content="en-us" />
<META name="robots" content="all" />
<META http-equiv="imagetoolbar" content="false" />
<META name="MSSmartTagsPreventParsing" content="true" />
<STYLE type="text/css" media="screen, projection, tv"> @import
url("http://www.ABC.edu/css/advanced.css");</STYLE>
<LINK rel="stylesheet" type="text/css" media="print" href="http://www.ABC.edu/css/print.css" />
<!-- InstanceBeginEditable name="head" -->

<STYLE type="text/css">
<!--
.style2 {font-family: "Courier New", Courier, mono}
.style3 {font-size: 12px}
-->
</STYLE>
<!-- InstanceEndEditable -->
<STYLE type="text/css">
<!--
.style1 {font-size: 9px}
-->
</STYLE>
</HEAD>
```

```
<BODY>
<TABLE cellpadding="0" cellspacing="0" width="100%" border="0">
<TR><TD><DIV id="container-str">
<DIV id="header-str"><IMG src="/media/header-left-str.gif" width="322" height="72" alt="ABC
University word mark" /></DIV>
<DIV id="sectiontitletopbar-str"><IMG src="/wwwimages/template/1pxtrans.gif" width="1" height="1"
alt="" /></DIV>
<!-- InstanceBeginEditable name="pagetitle" -->
<DIV id="sectiontitle-str">
  <H1>
    Information Security - Technician Controls
</H1>
</DIV>
<!-- InstanceEndEditable -->
<P align="center"><I><SPAN class="style1">The use of this page is restricted
      to authorized technical personnel to complete required duties. Any other
      use is strictly prohibited. </SPAN></I></P>
<TABLE width="90%" align="center"><TR><TD>
<!-- InstanceBeginEditable name="pagebody" -->
<H3><FONT face="Verdana, Arial, Helvetica, sans-serif">Data Center Firewall Client Block
Tool</FONT></H3>

<?php if ($totalRows_Recordset1 > 0) { // Show if recordset not empty ?>
    <P> <FONT face="Verdana, Arial, Helvetica, sans-serif"><SPAN class="style3">Firewall records
found!</SPAN></FONT></P>
    <P><FONT face="Verdana, Arial, Helvetica, sans-serif"><SPAN class="style3">
            <B>Before reporting
              this as a firewall issue</B>, <BR>
      make sure that &amp;quot;DestIP&amp;quot; in at least one of the rows below contains the
</SPAN></FONT></P>
    <BLOCKQUOTE>
      <P><FONT face="Verdana, Arial, Helvetica, sans-serif"><B>IP address of
            the server</B></FONT></P>
    </BLOCKQUOTE>
    <P class="style3"><FONT face="Verdana, Arial, Helvetica, sans-serif">that is a problem for
        the customer.</FONT></P>
    <P class="style3">The firewall has performed the following actions against: <B><?php echo
long2ip ($Recordset1->Fields('srcip'))
; ?></B></P>
    <PRE><SPAN class="style3"><?php
        $c00 = 'SrcIP';
        $c0 = 'DestIP';
        $c1 = 'Action';
        $c2 = 'Protocol';
        $c3 = 'SrcPort';
        $c4 = 'DstPort';
        $c5 = 'Date';
        $c6 = 'Time';
        $hdrline = sprintf("%16s%16s%10s%10s%10s%10s%13s%9s\n",$c00,$c0,$c1,$c2,$c3,$c4,$c5,$c6);
        echo $hdrline; ?>
<?php
while (!$Recordset1->EOF) {
?>
<?php
        $outputline = sprintf("%16s%16s%10s%10s%10s%10s%22s\n", long2ip($Recordset1-
>Fields('srcip')),long2ip($Recordset1->Fields('d
stip')), $Recordset1->Fields('action'), $Recordset1->Fields('proto'), $Recordset1-
>Fields('srcport'), $Recordset1->Fields('dstport')
, $Recordset1->Fields('timestamp') );
        echo $outputline; ?>
<?php
    $Recordset1->MoveNext();
  }
?>
    </SPAN></PRE>
    <?php } // Show if recordset not empty ?>
<?php if ($totalRows_Recordset1 == 0) { // Show if recordset empty ?>
    <P><FONT face="Verdana, Arial, Helvetica, sans-serif">The Data Center Firewall
        has <B>NOT</B> performed any operations against:</FONT></P>
    <BLOCKQUOTE>
      <P><FONT face="Verdana, Arial, Helvetica, sans-serif"><?php echo
long2ip($colname__Recordset1); ?></FONT></P>
    </BLOCKQUOTE>
    <P><FONT face="Verdana, Arial, Helvetica, sans-serif">You may continue with
        normal troubleshooting operations with high degree of confidence that
        this firewall is not a factor in the client issue.</FONT></P>
    <P> </P>
```

```
    <?php } // Show if recordset empty ?>
<!-- InstanceEndEditable -->
</TD>
</TR></TABLE>
<DIV id="footer-str"><A href="<?php   $urlpath=spliti('/',$_SERVER['PHP_SELF']); echo
"/$urlpath[1]/"; ?>"><?php echo $urlpath[1]; ?
></A> </DIV>
</DIV></TD></TR></TABLE>
<DIV id="copyright">Copyright &copy;</a>ABC University 2006</DIV>

</BODY>
<!-- InstanceEnd --></HTML>
<?php
$Recordset1->Close();
?>
```

6.3. Appendix C PMACCT Netflow Accounting Configuration

The nfacctd configurations below are commented using the ! character as the comment escape. These configurations show how to select traffic to a target subnet from an intranet (include network range) and then the Internet (exclude same range).

```
!
! nfacctd configuration
! intranet traffic to the datacenter
!
daemonize: true
syslog: daemon
pidfile: /tmp/nfacctdIN.pid

nfacctd_port: 2054
sql_db: pmacct
sql_table: intracct
sql_table_version: 1
sql_passwd: <removed>
sql_user: <removed>
sql_refresh_time: 90

! Total the flows for a month
sql_history: 1M
sql_history_roundoff: d

!
! src and dest info for 88 vs. on-campus intranet only
!
aggregate[inbound]: dst_host,dst_port,proto
aggregate[outbound]: src_host,src_port,proto
aggregate_filter[inbound]: dst net 7.7.88.0/23 && src net 7.7.0.0/16
aggregate_filter[outbound]: src net 7.7.88.0/23 && dst net 7.7.0.0/16
plugins: mysql[inbound], mysql[outbound]

!
! nfacctd configuration
! Internet traffic to the datacenter
!
daemonize: true
syslog: daemon
pidfile: /tmp/nfacctdIN.pid

nfacctd_port: 2054
sql_db: pmacct
```

```
sql_table: intracct
sql_table_version: 1
sql_passwd: <removed>
sql_user: <removed>
sql_refresh_time: 90

! Total the flows for a month
sql_history: 1M
sql_history_roundoff: d

!
! src and dest info for 88 vs. Internet only
!
aggregate[inbound]: dst_host,dst_port,proto
aggregate[outbound]: src_host,src_port,proto
aggregate_filter[inbound]: dst net 7.7.88.0/23 && !src net 7.7.0.0/16
aggregate_filter[outbound]: src net 7.7.88.0/23 && !dst net 7.7.0.0/16
plugins: mysql[inbound], mysql[outbound]
```

7. References

Activeworx. (2004). "fisq.pl" Retrieved, August 2006, from http://www.activeworx.org/downloads/index.htm

Cisco. (2005). "Catalyst 6500 Series Switch and Cisco 7600 Series Router Firewall Services Module System Messages Guide, 2.3". Retrieved, August 2006, from http://www.cisco.com/en/US/products/hw/switches/ps708/products_system_message_guide_book09186a00802c3062.html

Cisco. (2006). Cisco IOS Netflow Introduction". Retrieved August 2006, from http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html

Fullmer, M. (2005). "flow-tools". Retrieved, August 2006, from http://www.splintered.net/sw/flow-tools/docs/flow-tools.html

Interakt. (2005). "PHAKT freeware web-database IDE for Macromedia Dreamweaver. Retrieved, August 2006, from http://www.interaktonline.com/Products/Free-Products/PHAkt/Overview/

Lim, J. (2004). "ADOdb Database Abstraction Library for PHP". Retrieved, August 2006, from http://adodb.sourceforge.net/

Lucente, P. (2006). "Promiscuous Mode IP Accounting". Accessed (2006): http://www.pmacct.net/

Netflow. (2006, August 30). In Wikipedia, The Free Encyclopedia. Retrieved 17:03, August 30, 2006, from http://en.wikipedia.org/w/index.php?title=Netflow&oldid=72834238

Plytas, G. (July 5, 2004). "Enhancing ABC Inc's Security Strategy with IDS and Centralized Syslog". Retrieved, August 2006, from http://maverick.giac.org/rr/whitepapers/casestudies/1456.php

Winding, R. (2005). "Firewalls, VPNs, and Intrusion Detection Systems in a University Environment". Retrieved, August 2006, from http://www.educause.edu/LibraryDetailPage/666?ID=CMR0560