Global Information Assurance Certification Paper

A Framework to Collect Security Events for Intrusion Analysis

GIAC Certified Intrusion Analyst

Gold Practical Assignment

Advisor: Jeff Holland

Table of Contents

I. Abstract

It becomes a problem when you have several firewalls, intrusion sensors or servers and to top it off, not all firewalls and intrusion sensors generate logs in a standard format. This means you may need several tools to analyze data – maybe even one tool per each device per vendor. This can be a mess.

This paper assumes you need a way to consolidate event logs from these devices and present them to the people who are chartered to analyze and take action when it becomes necessary. Many organizations have a firewall, at a minimum, while others are fortunate enough to have intrusion sensors. As the number of network devices increases—and perhaps the number of vendors—it becomes increasingly difficult to use the information that these devices provide for analysis in an efficient manner.

This paper describes a framework to help security personnel have a starting point with which to collect and view security events from devices capable of reporting via syslog. Ideally, the reader will be able to follow along and use this paper in a way similar to a how-to reference guide.

Because this paper is based off of free software—assuming the vendor agreements are honored—any organization will be able to set up an infrastructure for monitoring their networks for intrusion. This can be successfully accomplished with the following four resources:

- An IT professional with basic Linux skills
- A knowledgeable intrusion analyst
- A few (or more) spare computers to be used as sensors/correlation servers
- Some time dedicated to setting it all up

II. Document Conventions

When you read this document, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style
URL	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.
»	Continuation of a single line that would not fit the width of this paper.

III. Introduction

This paper is based of the actual installation and configuration of a functioning lab environment. The three primary tools used to accomplish this was Red Hat's Fedora Core 3 Linux operating system with Snort used as an intrusion sensor and Remote Assessment's Aanval product for correlating events.

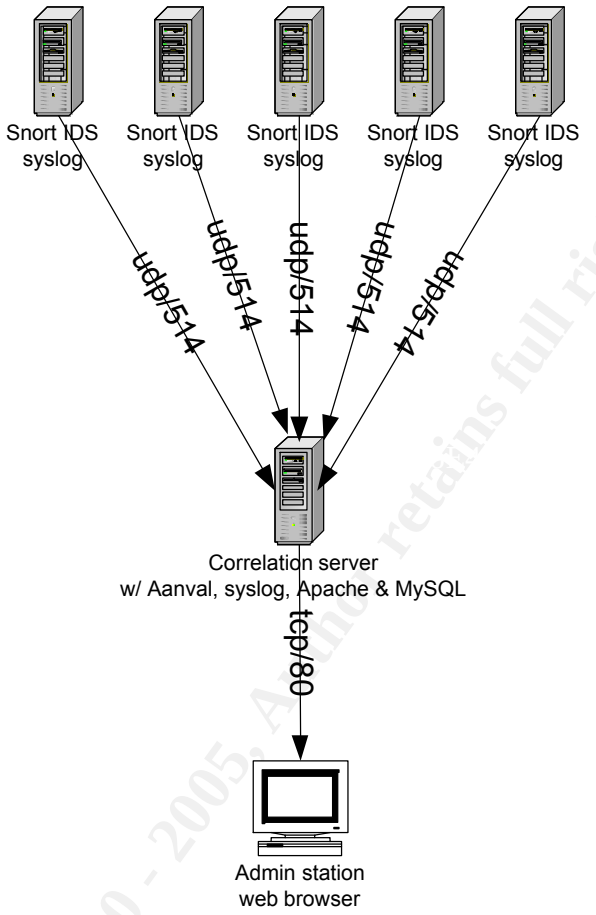
For this paper, I decided to go with Aanval as the collection server to gather and display the results. Aanval has a free version, was built with Snort in mind and includes programs/agents that work well with Snort. By working well I mean they not only collect the bare essentials such as the information found in the alert_fast mode (source/dest IP, source/dest port, alert name and classification and SID), they include detailed information found in the alert_full mode (such as the IP options and payload of the packet that triggered the alert) [6].

Being a paper for the Intrusion Analyst certification for SANS, my main concern is gathering information from intrusion sensors. However, I know much information can be gleaned from drop logs off a firewall such as a worm trying to spread that may or may not have a snort signature. Most security devices (such as firewalls) have the ability to syslog, and Aanval has the ability to receive syslog alerts [7].

Aanval includes a module that understands snort database output. You may decide to use this module for your snort sensors and then syslog for everything else—that's up to you. But knowing that Snort and most firewalls can syslog, I decided to cover as many devices possible and will write about Snort sysloging to Aanval. This will also allow us to cover other intrusion sensors that use syslog for reporting—not just snort.

There are commercial products available, namely Security Information Management (SIM) products. These all allow you to gather and correlate security events for viewing, remediation and archiving. I included a brief section titled "Commercial Security Information Mantement (SIM)" for users that have a need that extends beyond Aanval's capabilities.

However, the goal of this paper is to allow any organization to set up a simple yet functional SIM-type solution to monitor their network for security events for free. The system I propose is illustrated by the following diagram:



This solution allows for the correlation of events on the “correlation server” by way of a SQL database. Having all security events in a central location such as this provides a way for them to be viewed, backed up or mined for information if the user is proficient with SQL and scripting (shell, perl, python, etc.). Furthermore, this can add additional security if the correlation server is hardened and listening to very few ports.

A correlation server with GUI interface is much more convenient than logging on to each sensor to view events or even using a monitoring solution for different device types such as one for ISS intrusion sensors, Cisco IDSes or Snort machines. I, for one, would prefer to interact with a GUI that has information correlated from all those devices, and at the same time has detailed information as if I were to log on to each device.

Additionally, backing up the data is quick and easy because it is all on one database on one server. You can even add redundancy by backing up or replicating the database to another location.

IV. Software

Some things to consider first are software dependencies, security and reliability. Everything I write about was set up and functioning by me – all for free. I am basing everything on Linux and free software. With that in mind, the three main pieces of software needed are Linux, Snort and Aanval. Each of these has their own dependencies which are shown below:

Linux

- Fedora Core 4 - http://fedora.redhat.com/download/
- Debian - http://www.debian.org/
- MandrakeLinux - http://www.mandrakesoft.com/

Snort - (www.snort.org) [10]:

- Libpcap (for Linux) - http://www.tcpdump.org/release/libpcap-0.9.4.tar.gz
- Winpcap (for Windows) - http://www.winpcap.org/
- PCRE (Perl Compatible Regular Expressions) - http://www.pcre.org/

Aanval (www.aanval.com) [7]:

- MySQL - http://www.mysql.com/
- Syslog – Included with your Linux distro.
- PHP - http://www.php.net/
- Perl - http://www.cpan.org/src/stable.tar.gz
- Webserver – http://www.apache.org/

The installation of the above software is beyond the scope of this document. I will say, however, that in my case I had the option of including perl, syslog, MySQL and Apache with my Fedora Core 3 installation [5]. The rest of this document is under the assumption that Linux, Snort, Aanval and their dependencies have been installed.

V. Infrastructure

Below in Figure 1 is an illustration similar to what I had set up. In my situation, I installed VMware so that one of my computers was turned into four virtual snort sensors. This was a 3GHz Pentium 4 with Hyperthreading and 1 Gig of RAM. Each sensor had slightly less than 256Mb RAM, but I still had no problems.

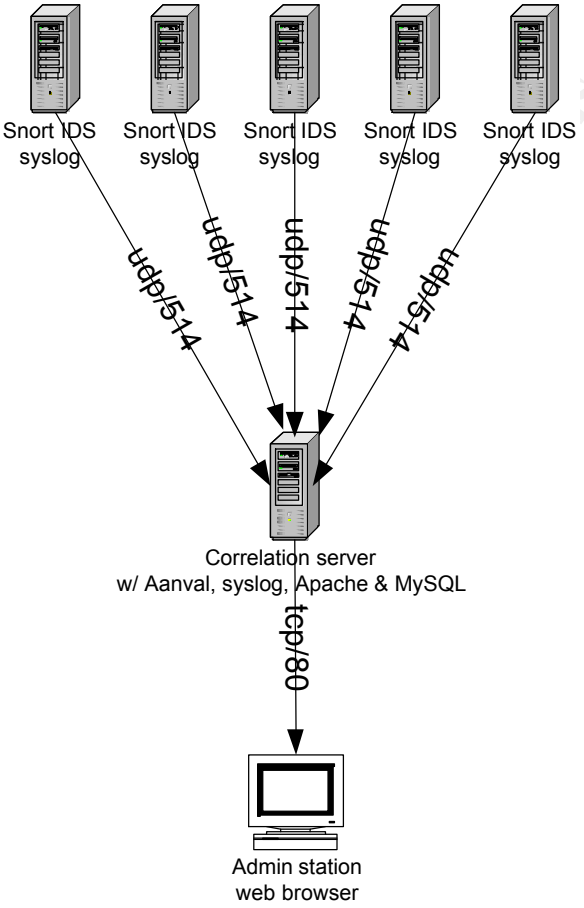


Figure 1

The correlation server was an older 800MHz Athlon with 512Mb RAM and a 40 Gig hard disk. Again, this worked fine in my lab of the 4 Snort sensors. Even while running MySQL, Apache, Aanval (and its syslog and other background scripts) and syslog, I had plenty of computing power for all of this.

My admin station with Web browser was just a laptop with a wireless adapter. I used both MS Internet Explorer and Mozilla Firefox without any problems. I could have even skipped this step and used the correlation server because I had X/KDE and Firefox on there as well.

VI. Configuration

I am going to discuss how everything was configured, so I am assuming the snort boxes, a correlation server and all the software is installed and ready to be configured.

First, on the snort boxes, they should be up and running to a point to where they are seeing traffic and logging locally. I was able to test this by running a false positive generator—specifically `sneeze.pl` [1]. In my `snort.conf` file, I set the alerts to be written to `/var/log/snort`. I did a "`tail -f`" to read the alert files as they were being written and was able to verify that all my snort sensors were indeed listening and logging as they should.

After verifying that snort was generating alerts, my next step was to set up and verify that it would generate syslog alerts. Furthermore, I needed to set up syslog on the sensors to send data to the correlation server and make sure the correlation server would listen and accept these events.

A. Setting up Snort to syslog

In the `snort.conf` file, I needed to make a change in section/step three titled "*Configure output plugins*" where the format is [2]:
`output <name_of_plugin>: <configuration_options>`. The change I needed to make was to add and then configure the "*alert_syslog*" plugin. This plugin has the following format:
`output alert_syslog: FACILITY PRIORITY`

The following facilities and priorities are understood by Snort [2]:

Facilities	Priorities
log_auth	log_emerg
log_authpriv	log_alert
log_daemon	log_crit
log_local0	log_err
log_local1	log_warning
log_local2	log_notice
log_local3	log_info
log_local4	log_debug
log_local5	
log_local6	
log_local7	
log_user	

Figure 2

The specific line I added was:
`output alert_syslog: LOG_local1 LOG_ALERT`. I next configured syslog to log to the file "`/var/log/snort_log`" by adding the following line to my "`/etc/syslog.conf`" file:
`local1; local1.*  /var/log/snort_log`
Local1 was a facility on the snort machine that was available for testing. I then restarted syslog and verified the configuration file was correct because it created the `snort_log` file like I expected. Next I started up snort and used Sneeze to generate some alerts. Upon doing a "`tail`" on the `snort_log` file, I was able to verify that Snort was correctly passing alerts to syslog, and syslog was correctly writing them to the file I had specified.

Now that Snort and syslog were working correctly, I now needed to get the sensor to send these events to my correlation server. To do this, I needed to configure syslog on the Snort sensor to send events to my correlation server whose IP address was 192.168.0.101.

Having syslog on the snort sensor send events to the syslog daemon on the correlation server was quite easy. After referring to the syslog man page, I found that I simply needed to change the output file location to the IP address of the server I wanted the data sent to, preceded by an "@" symbol. I changed the line in `syslog.conf` from [4]:

```
local1; local1.*                /var/log/snort_log
```

to

```
local1; local1.*                @192.168.0.101
```

B. Setting up correlation server

Now at this point, I'm still not able to test this out until I set up syslog on the correlation server to receive syslog data from the snort sensors. I needed to add the "`-r`" option to the `syslogd` execution command on the correlation server [4]. The default for syslog is to receive local messages only. The "-r" option tells syslog to open a network socket and listen for messages on the network. Note that vanilla syslog uses UDP which is what I used and configured.

To add the "-r" option, I edited the `/etc/init.d/syslog` file. In this file there is a variable named `SYSLOGD_OPTIONS`. I found the variable already had a "`-m 0`" so I appended a "`-r`" to it. To be clear, the old line looked like:

```
SYSLOGD_OPTIONS="-m 0"
```

and the new line looked like

```
SYSLOGD_OPTIONS="-m 0 -r"
```

In the `/etc/syslog.conf` file, I added the following line:

```
local1; local1.*                    /var/log/snort_log
```

Upon restarting the syslog service, I generated alerts on the snort sensor by way of `sneezel.pl`, checked the `/var/log/snort_log` file on the correlation server and found that it was logging as expected.

An easier way to ensure that syslog messages are being transmitted by the snort sensor would be to run tcpdump on the correlation server and see if anything is destined for it on UDP port 514. I did this as well, but setting up syslog on the correlation server may grant some readers of this paper more flexibility. For example if there are other products out there to analyze events from syslog log files rather than binding to a port and listening, or if you just want to store logs somewhere and that's it—this is how you would have done it.

To use the syslog module for Aanval, the `idsSyslog.pl` perl script binds to UDP port 514 and will generate errors if syslog is already listening on this port (due to the "`-r`" option). I ran into this and once I removed the "`-r`" from the syslog startup script and the Aanval `idsSyslog.pl` script functioned properly. Note that if a user needs to receive syslog messages with the syslog daemon AND use the Aanval syslog script, you would need to figure out a way to have either syslog or the Aanval script bind to another port. I discuss this more in the section titled "Next Steps."

At this point, I had syslog on the snort sensors sending events to my correlation server and verified this using syslog (with the "`-r`" option) on the correlation server and by using `tcpdump`. To prepare for actual use, I had syslog running on the correlation server without the "`-r`" option.

C. Setting up Aanval

I was then ready to start up the two Aanval scripts – `idsSyslog.pl` and `idsBackground.pl` which are found in the "apps" directory within the location where Aanval was installed. (`/var/www/html/aanval/apps` in my case). `idsBackground.pl` will automatically start `processorA.php`, `processorB.php` and `processorC.php` so be sure not to run those even though they appear in the "apps" directory.

Now log in to Aanval and go to System -> Status. You should see status information for Background Processors A through C [7]. Assuming Aanval is reporting no errors with the background processors and the `idsSyslog.pl` script is running, proceed to System -> Modules -> Syslog.

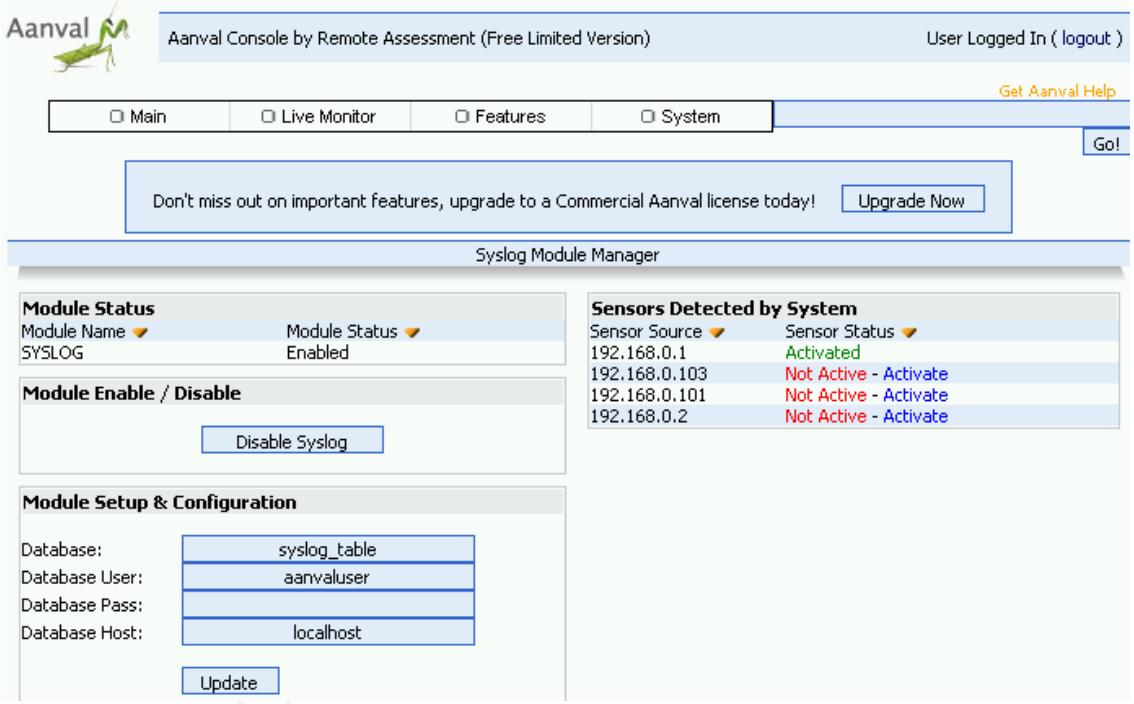


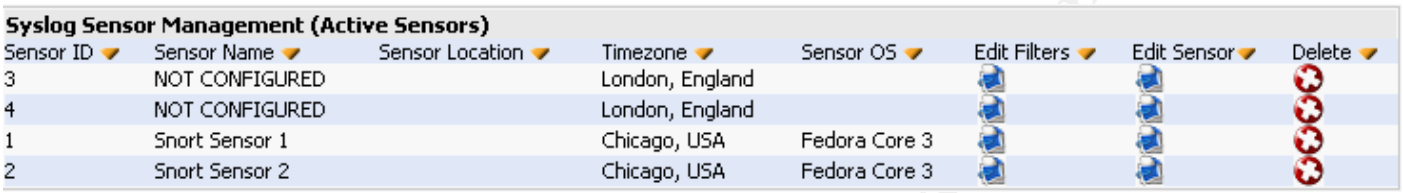
Figure 3

To write this paper, I was using Aanval version 1.55 Maven and the above screenshot (figure 3) is where I configured syslog. You can see in the section "Module Enable / Disable" there is a button. Initially the syslog module is off/disabled. Clicking on the button will enable it and this is reflected by the "Enabled" show in the "Module Status" column in the "Module Status" section.

Once the module is enabled, the "Sensors Detected by System" begins to populate with all the sensors reporting to the correlation server. In this section you will see two columns: "Sensor Source" and "Sensor Status." When Aanval first sees a sensor—or anything sysloging to it—it will display the IP address

and show its status as inactive. To activate, you must click on the blue "Activate" link. Once you click on this and it gets activated, it reads "Activated" in green.

All active sensors are displayed in the bottom of the frame, as shown in figure 4.

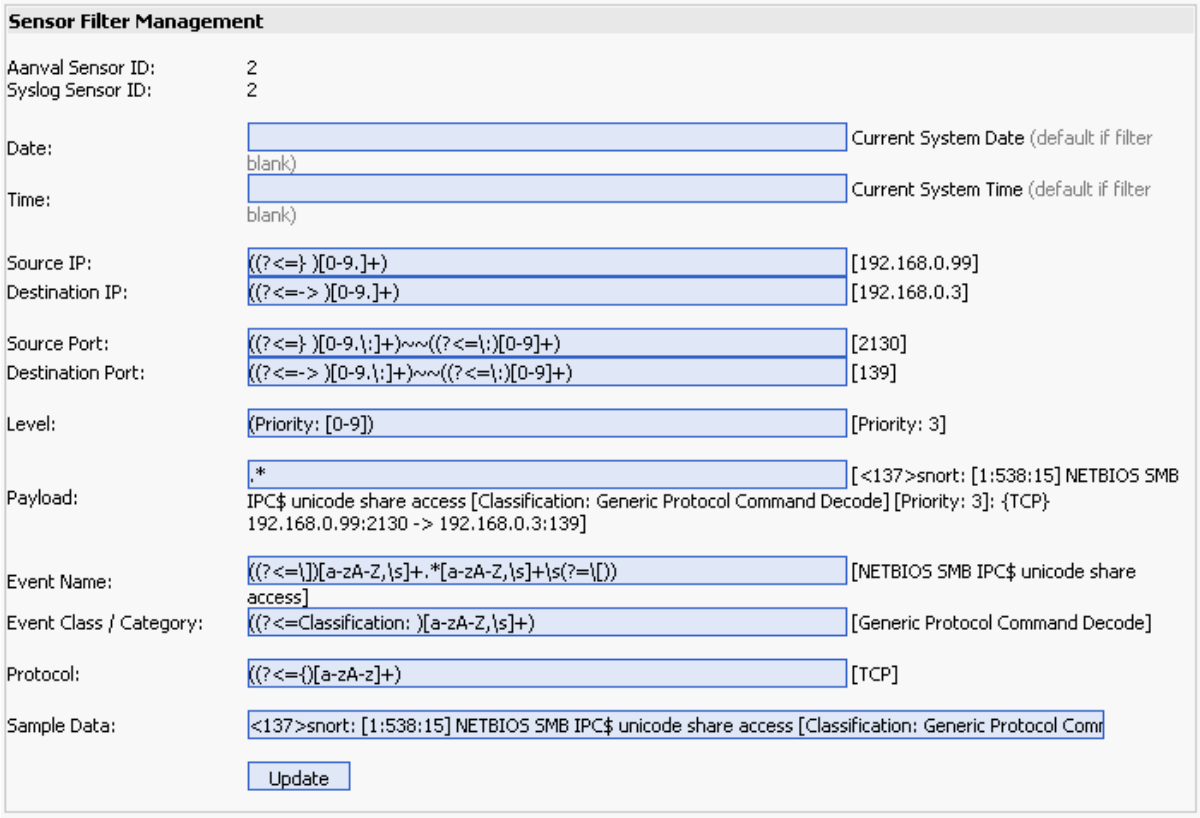


Syslog Sensor Management (Active Sensors)

Sensor ID	Sensor Name	Sensor Location	Timezone	Sensor OS	Edit Filters	Edit Sensor	Delete
3	NOT CONFIGURED		London, England				
4	NOT CONFIGURED		London, England				
1	Snort Sensor 1		Chicago, USA	Fedora Core 3			
2	Snort Sensor 2		Chicago, USA	Fedora Core 3			

Figure 4

You can configure sensor names, specify the timezone and label the sensor's OS by clicking "Edit Sensor." But more importantly, I needed to set up Aanval to interpret the syslog messages correctly. To do this, click "Edit Sensor" and the "Sensor Filter Manager" appears. On this page, I am concerned with the right half of the screen titled Sensor Filter Management" section as shown in figure 5.



Sensor Filter Management

Aanval Sensor ID:	2	
Syslog Sensor ID:	2	
Date:		Current System Date (default if filter blank)
Time:		Current System Time (default if filter blank)
Source IP:	((?<=})[0-9.]+)	[192.168.0.99]
Destination IP:	((?<=->)[0-9.]+)	[192.168.0.3]
Source Port:	((?<=})[0-9.\:]+)~~((?<=\:)[0-9]+)	[2130]
Destination Port:	((?<=->)[0-9.\:]+)~~((?<=\:)[0-9]+)	[139]
Level:	(Priority: [0-9])	[Priority: 3]
Payload:	.*	[<137>snort: [1:538:15] NETBIOS SMB IPC$ unicode share access [Classification: Generic Protocol Command Decode] [Priority: 3]: {TCP} 192.168.0.99:2130 -> 192.168.0.3:139]
Event Name:	((?<=\])[a-zA-Z,\s]+.*[a-zA-Z,\s]+\s(?=\[))	[NETBIOS SMB IPC$ unicode share access]
Event Class / Category:	((?<=Classification:)[a-zA-Z,\s]+)	[Generic Protocol Command Decode]
Protocol:	((?<={)[a-zA-z]+)	[TCP]
Sample Data:	<137>snort: [1:538:15] NETBIOS SMB IPC$ unicode share access [Classification: Generic Protocol Comr	

Update

Figure 5

Aanval uses regular expressions to match and extract information from syslog

messages. It appears most Aanval users are using the snort module for gathering messages from Snort machines, so it was difficult to find the correct regular expressions on the Aanval discussion forum.

However, after experimenting, I was able to come up with a regular expression for each of the fields except time and date. I chose to use the system time for those. The regular expressions and fields are summarized in the table in figure 7 below. The data that is shown in the "Matches" column is based off the Snort event that was sent via syslog shown in figure 6.

Source Syslog Message	[<137>snort: [1:538:15] NETBIOS SMB IPC$ unicode share access [Classification: Generic Protocol Command Decode] [Priority: 3]: {TCP} 192.168.0.99:2130 -> 192.168.0.3:139]

Figure 6

Field	Regular Expression	Matches
Date	*blank*	*System time*
Time	*blank*	*System time*
Source IP	((?<=})[0-9.]+)	192.168.0.99
Destination IP	((?<=->)[0-9.]+)	192.168.0.3
Source Port	((?<=})[0-9.\:]+)~~((?<=\:)[0-9]+)	2130
Destination Port	((?<=->)[0-9.\:]+)~~((?<=\:)[0-9]+)	139
Level	(Priority: [0-9])	3
Payload	.*	[<137>snort: [1:538:15] NETBIOS SMB IPC$ unicode share access [Classification: Generic Protocol Command Decode] [Priority: 3]: {TCP} 192.168.0.99:2130 -> 192.168.0.3:139]
Event Name	((?<=\])[a-zA-Z,\s]+.*[a-zA-Z,\s]+\s(?=\[))	NETBIOS SMB IPC$ unicode share access
Event Class / Category	((?<=Classification:)[a-zA-Z,\s]+)	Generic Protocol Command Decode
Protocol	((?<={)[a-zA-z]+)	TCP

Figure 7

Once all the regular expressions are entered, clicking "update" saves them. Once they are saved, any new syslog messages coming from the IP address of the sensor just configured will be effective. Any past events will not reflect the new settings.

To quickly see if your regular expressions are working, go to the main screen. Before any regular expressions are set up, all events show up as the source IP of 255.255.255.255 and destination IP of 255.255.255.255. Once the regular expressions are set up, normal IP addresses will start to show up because Aanval is now using the regular expressions to extract the fields it needs.

VII. Next Steps

A. Going further with Aanval and Syslog

Note that most of this section is concerned with syslog and how Aanval handles syslog. It may be a good idea to skip ahead and read through Appendix A first.

For the reader following along, there are other things that could be implemented to make this a more robust solution. For instance, the one major downfall I see is that syslog uses UDP datagrams. What if an important event was lost in transit? TCP would add an extra layer of protection.

Since Aanval uses a perl script to receive syslog messages, the number of syslog facilities available would not be an issue. You could have 100 snort sensors and as long as they have a unique IP, they are good to go with Aanval [6]. However, syslog-ng is available and does use TCP for sending syslog messages. This may be one option if you want to reliably send syslog messages to a correlation server.

I hope that all events would be sent across private network links and not the public Internet, but even within a private corporate network, it may also be a good idea to encrypt the traffic. Stunnel (http://www.stunnel.org) is a tool I found that would accomplish this [3].

Stunnel will not work with UDP packets, hence my syslog-ng suggestion [12]. There is plenty of documentation out there on using both syslog-ng and Stunnel. In fact, there is a webpage with information on using stunnel with syslog-ng (see Appendix B) [8]. For my paper, I only researched the two packages but never actually implemented and here's why:

I was able to trick Aanval into thinking the `idsSyslog.pl` script was running because I had both my script and syslog (not syslog-ng) writing files (`indata.<IP ADDRESS>`….) to the correct location (`<AANVAL HOME DIR>/syslog`) and it worked without a problem. The issue became getting syslog events out of the file syslog is writing events to.

Syslog continually writes to a file (e.g. `/var/log/snort_events`). Logrotate will periodically rotate and compress (if your system is configured this way) the files and create a new file. As time goes on, the log file gets larger and larger and the newest events are added to the end. The script I created simply moved that file so that the entire contents of the file were put in place for Aanval to copy the events to the database. I decided to move (not copy!!) the file so that I wouldn't be copying old data and creating duplicates in the Aanval database. Well, this messed up syslog because the next time it received an event, it

expected the logfile (e.g. /var/log/snort_events) to be there when it was time to write the new event. When it discovered the file wasn't there as it should, it did not create a new file nor did it write the new syslog events anywhere else (/var/log/messages for example).

The ideal solution would be a script that copies only the newest events and leaves syslog's file intact. "Newest events" in this case is anything that has arrived since the script last ran and copied events to Aanval's syslog location. If such a solution was found, it could be incorporated into the simple script I wrote and syslog should work fine.

One possibility would be to restart the syslog process every time the script moves files around. Such a script would be similar to logrotate. If logrotate is moving the file from underneath syslog, logrotate has a similar problem, so it must kill off the syslog process and restart it.

Perhaps using a "kill -HUP" would do the trick. Using the ps command and awk should allow you to obtain the PID number of the syslog process to be used with the kill command. However, logrotate runs once in a great while, not every 60 seconds (maybe even sooner) like we would like in order to keep our events coming in to Aanval somewhat fresh. So I am unaware of how this would affect system resources.

And one last possibility is syslog-ng has an option of creating files that don't exist [9]. I did not test it out, but it might replace the /var/log/snort_events file like we need each time we copy events.

So I hope this extra section may have assisted somebody in creating a more robust solution that is reliable, secure and versatile. Reliable in that it uses TCP and does not "lose" events, secure in the fact that an encrypted tunnel is used between the snort sensor and the correlation server and versatile in that the usage of syslog allows the collection of events from many device types.

B. Commercial Security Information Management (SIM)

Some users have a need beyond just seeing security events to act on, such as companies who are audited and/or must comply with government regulations. There are several commercial products similar to Aanval available for use. I have listed a few companies below as a starting point.

- ArcSight - http://www.arcsight.com/
- e-Security - http://www.esecurityinc.com/
- Intellitactics - http://www.intellitactics.com
- netForensics - http://www.netforensics.com/
- TriGeo - http://www.trigeo.com/

Some features of commercial SIM products include, but are not limited to:

- Encryption of logs in transit
- Use of commercial databases like Oracle, MS SQL, DB2
- Ability to alert users in multiple ways based on specific events
- Support of commercial database backup (such as Veritas)
- Database partitioning and replication
- Supports delivery methods above and beyond syslog (ie ODBC/JDBC connections, RDEP, OPSEC LEA, daemons that read log files, etc)

Appendix A - idsSyslog.pl script

The following information is background information on Aanval's `idsSyslog.pl` script that I compiled with my own experimentation. The information below was taken and edited from an email to my SANS advisor:

How Aanval works with syslog:

- `idsSyslog.pl` runs in the background, listening for anything on UDP port 514.
- When `idsSyslog.pl` receives syslog events from the listening port, it puts them in a file named "`indata-<IP_ADDRESS_OF_SOURCE>.syslog`.
- Another process (`idsBackground.pl` or one of the processors) reads that `indata.*.syslog` file, puts the data into the Aanval table in MySQL, then deletes that file.
- When `idsSyslog.pl` gets more data, it creates another file and the whole process repeats.

My testing for a workaround:

- What I did, and found that it works, is set up snort to syslog to a file (`/var/log/snort_syslog` in my test).
- I then did a "`cat /var/log/snort_syslog > /var/www/html/aanval/syslog/indata.192.168.0.101.syslog`" which created a file with the contents of the snort syslog data.
- Aanval read the contents of the file, put it in the database, then deleted the file.
- Upon going into Aanval, I saw my data which told me it worked!
- Next, I changed `/etc/syslog.conf` to direct snort syslog data directly to `/var/www/html/aanval/syslog/indata.192.168.0.101.syslog` and restarted the syslog service.
- I generated a false positive, watched it go into that `indata.*.syslog` file, saw it get deleted and then appear in the Aanval DB).
- Now that the file was deleted, syslog has nowhere to write the data to. I created a new file, but it just gets deleted right away as Aanval reads the data from it and deletes it. Syslog never has a chance to write to it

So it appears that Aanval is reading the file for data, then deleting it. So what I tried to do is this:

- Set up syslog to write the data to a different location, say `/var/log/snort_local1_syslog`
- Research how and create a script (see below) that moves `/var/log/snort_local1_syslog` to `/var/www/html/aanval/syslog/indata.192.168.0.101.syslog` and then creates a new `/var/log/snort_local1_syslog` file so syslog has something to write to.
- Set up a cron job that runs this script every so often - depending on if the user wants real-time or near real-time. I figure once every minute should be okay for my purposes.

Here is the script I created to move around the files:

```
[root@server syslog]# cat script
mv /var/log/snort_local1_syslog »
/var/www/html/aanval/syslog/indata_192.168.0.101.syslog
touch /var/log/snort_local1_syslog
```

Appendix B - Links

Using Stunnel with syslog-ng

Example from stunnel website - http://www.stunnel.org/examples/syslog-ng.html

Quick and dirty syslog-ng with stunnel howto - http://wiki.osuosl.org/pages/viewpage.action?pageId=1369

Software websites

Aanval (www.aanval.com)
Apache web server – http://www.apache.org/
Libpcap (for Linux) - http://www.tcpdump.org/release/libpcap-0.9.4.tar.gz
Linux
- Fedora Core 4 - http://fedora.redhat.com/download/
- Debian - http://www.debian.org/
- MandrakeLinux - http://www.mandrakesoft.com/

MySQL - http://www.mysql.com/
PCRE (Perl Compatible Regular Expressions) - http://www.pcre.org/
Perl - http://www.cpan.org/src/stable.tar.gz
PHP - http://www.php.net/
Snort - (www.snort.org)
Winpcap (for Windows) - http://www.winpcap.org/

References

[1] Bailey, Don & Caswell, Brian, "sneeze.pl - Snort False-Positive Generator", URL: http://www.securiteam.com/tools/5DP0T0AB5G.html (August 22, 2005)

[2] Green, Chris & Roesch, Martin, "Snort Users Manual 2.4.0:, URL: http://www.snort.org/docs/snort_htmanuals/htmanual_2.4/, (August 19, 2005)

[3] Hatch, Brian, "Stunnel.org Frequently Asked Questions", URL: http://www.stunnel.org/faq/ (August 20, 2005)

[4] NetAdminTools.com, "SYSLOG.CONF", URL: http://www.netadmintools.com/html/5syslog.conf.man.html (August 17, 2005

[5] Redhat, Inc., "The Fedora Project", URL: http://www.redhat.com/en_us/USA/fedora/ (August 17, 2005)

[6] Remote Assessment, "Aanval Forum", URL: http://www.aanval.com/forum/phpBB2/ (August 20, 2005)

[7] Remote Assessment, "docuWeb – Aanval Online Documentation", URL: http://www.aanval.com/?op=pub_docuweb, (August 17, 2005)

[8] Seberino, Christian, "Encrypting traffic to a remote syslog-ng server including SSL peer authentication", URL: http://www.stunnel.org/examples/syslog-ng.html (September 1, 2005)

[9] Scheidler, Bal za, "syslog-ng reference manual", URL: http://www.balabit.com/products/syslog_ng/reference-2.0/syslog-ng.html/index.html (October 1, 2005)

[10] Sourcefire, Inc., "Snort Registered User Forums", URL: http://www.snort.org/reg-bin/forums.cgi (August 20, 2005)

[11] Tipping Point, "Tipping Point Technology Partners", URL: http://www.tippingpoint.com/partners_technology.html (December 6, 2005)

[12] Trojnara, Michal, "Stunnel-3.26 Man Page:. URL: http://www.stunnel.org/faq/stunnel-3.xx.html (October 1, 2005)