v 1.0.0

Building a Check Point Firewall Log Analysis Server using Debian Linux, fw1-loggrabber, and MySQL

Author: Mark Stingley, GCIH GCIA GPEN

ggiac _AT_ altsec.info

Outline

1.Introduction

It is no wonder that firewall log analysis is becoming a lost art. In popularity, it runs a dismal fourth place behind Google searches of Sex, Drugs, and Rock and Roll[1]. Hopefully, these pages will inspire you to shake the bonds of a summarized management console and get your hands dirty with the raw firewall log data. Though Checkpoint's OPSEC is the featured data transport technology, the principles of analysis and MySQL queries presented are not only portable, they are essential to network security.

This paper goes beyond the time honored, yet dated, attacker-centric focus on dropped inbound traffic and proposes greater attention to host profiling, automated snapshot analyses, and network traffic audits.

2.Justification

Properly implemented, firewall log analysis is just one piece of a balanced defense-in-depth[2], nothing more, but nothing less. It is no substitute for Intrusion Protection System (IPS) or Intrusion Detection System (IDS) alerts, security event management, or network traffic capture analysis. But, it is an indispensable companion.

Firewall logs can provide connection detail that is critical to network forensics, and the open-source implementation described herein is capable of providing exactly the kind of investigations needed simply and quickly. Further, the flexibility of the system lends itself easily to automation of alerting and reporting and integration with other systems.

Granted, there are impressive (and expensive) commercial systems that collect and analyze logs, learn traffic patterns and alert on variances, and correlate information with other sensors. Some are literally amazing, and we all should be able to afford one. But, even the fanciest network security gadget demands audit. That means that firewall log hits on a workstation's web traffic should match those recorded by a browser security appliance. An IPS might show a variety of attacks on public servers, but the firewall logs will reveal connection detail on additional traffic to or from the attackers. The context of 'audit' is 'cross check'. This inexpensive and simple tool makes that possible.

So, the question now reverses. Instead of being asked to justify why fw1-loggrabber and OPSEC LEA should be implemented, you have to answer the question, "Why not?"

3.Caveats

There are two golden rules for keeping firewall connection data in perspective.

A. Never forget that the data can be misleading if not carefully cross-checked and correlated in context.

B. Always look at the data from every possible perspective.

Some questions simply cannot be answered reliably by looking at the firewall log data alone. Legitimate peer-to-peer and malicious botnet traffic can often express identical connection patterns, yet flow and content based tools can help differential between the two.

Data from firewall interfaces are susceptible to spoofed traffic and should never be the basis for sending an abuse complaint without confirming a three-way handshake from another sensor.

4.Methods of Log Access

Offline Export

The process for obtaining Checkpoint firewall logs most familiar to many is likely to be the export of text logs from the management server with the command ‘fw log’. Most CPFW admin’s are familiar with use of this utility from the firewall management server to export the logs to a text file that is then copied elsewhere for analysis, or even pipe the output to syslog.

The firewall logs can also be exported to a text file via the Smartview Tracker GUI, but this process is even slower than using ‘fw log’from the management server console login.

The major downsides to this method are time and resources demands. FW log exports are relatively slow, and it could take hours for each log. Then, considerable programming time could be required for analysis if a packaged solution is not used. And, though there are a number of fine systems for correlation or analysis of textual data, they are always going to be slower than searching through a structured database.

There is also the question of data security if clear text logs are transported across the network.

One important thing to remember even if you intend only to conduct offline log analysis is that fw1-loggrabber is going to be faster than the native Checkpoint export methods. My personal experience has been several hours for log export with ‘fw log’ or ‘Smartview Tracker export’compared to an hour or less using fw1-loggrabber, whether writing to a text log file or MySQL database.

Online, Real-Time

Online, real-time solutions must speak Checkpoint’s OPSEC to securely transport connection log data from the Checkpoint firewall management server to a client console. Checkpoint Smartview Tracker is a fine example, and it is suitable for very simple queries.

There are various commercial SIM/SEM/SEIM (security information/ event management) products on the market that utilize OPSEC. Most of them are very expensive. Some are relatively inexpensive, such as Splunk.

There are a few open-source systems available, such as OSSIM (which has an fw1-loggrabber plugin).

Then, there is the fw1-loggrabber setup described here.

5.<u>Implementing fw1-loggrabber</u>

The extraordinarily talented and ambitious Linux admin might take the approach of downloading the bare fw1-loggrabber source code and the stock Checkpoint OPSEC SDK then go through all the requisite tuning of the environment, configuration files, and scripts to achieve a working system. From experience, I can tell you that approach can take a fair amount of time and skill.

As a working Linux system administrator for many years, I prefer the simple and easy approach of the Splunk prepackaged fw1-loggrabber. The fine folks at Splunk have combined almost all of the necessary parts in a relatively easy to make bundle. Splunk's original intent was to make it easier for their customers to implement an OPSEC pathway into their affordable event management solution. Even though their licensing puts no obligation on use of their efforts with fw1-loggrabber, many organizations could benefit from their commercial product and it would only be fair to have a look.

That being said, it's time to get fw1-loggrabber built and working.

There are two decisions to make before starting:

32-bit or 64-bit.

The 32-bit or 64-bit decision is one of preference to be decided by your architecture. The predominant criteria are RAM and speed, and RAM will affect speed. Most 32-bit software and hardware combinations are going to limit each application to 4GB or less of RAM. This can be a serious problem for query caching, temporary data, and other database analysis activities. A complex query, or even concurrent simple queries can throw MySQL into disk paging, which can backlog the real-time log writes to the database. You do not want this happening if you rely on real-time data for alerting.

Since 64-bit applications can improve performance and the architecture increases the amount of available RAM, I personally recommend that solution for the database server.

One system or two.

You can run fw1-loggrabber on one system and write the firewall data to a separate database server, or run fw1-loggrabber on the database server itself. Two factors affect this decision.

At present, fw1-loggrabber and Checkpoint's OPSEC SDK are compatible with 32-bit libraries only. Running fw1-loggrabber standalone from a 64-bit database server removes the complexity of implementing the supporting 32-bit libraries for fw1-loggrabber. But, identifying those libraries and getting fw1-loggrabber working on the 64-bit database server isn't all that hard.

In addition, having the database server separate from the loggrabber client introduces network latency. Your network topology will decide if that could be a problem.

Building the client host

fw1-loggrabber on Linux is no harder to compile and configure than any other relatively simple source code package. Using the Splunk adaptation helps bypass the tuning of many environment variables and configuration file settings.

For simplicity, the software should be compiled on a 32-bit Linux installation, which will also serve to provide the few necessary libraries when running it on a 64-bit database server.

These instructions are for the Debian 5 Lenny i386 distribution with the install options "Desktop Environment" and "Standard System" selected. Adjust accordingly for other distributions.

Untar the package fw1-loggrabber-splunk.tar.gz to '/usr/local/src'.

The resulting directory structure for building the software will look like this:

```
fw1-loggrabber-splunk
drwxr-x--- 2 507 507   4096 2007-09-11 12:33 lea-bundle
drwxr-xr-x 2 507 507   4096 2007-07-18 15:01 doc
drwxr-xr-x 2 507 507   4096 2007-07-18 18:00 config
drwxr-xr-x 4 507 507   4096 2007-07-18 18:45 bin
drwxr-xr-x 5 507 507   4096 2007-09-11 00:18 opsec-tools
drwxr-xr-x 6 507 507   4096 2006-01-08 04:24 pkg_rel_linux
drwxr-xr-x 6 507 507   4096 2006-01-17 12:19 pkg_rel_solaris_gcc
-rw-r--r-- 1 507 507  15938 2005-02-21 13:41 fw1-loggrabber.h
-rw-r--r-- 1 507 507  18349 2005-02-21 13:41 LICENSE
-rw-r--r-- 1 507 507   2625 2007-07-18 18:34 Makefile.solaris
-rw-r--r-- 1 507 507   2731 2007-07-18 18:25 Makefile.linux
-rw-r--r-- 1 507 507   3837 2007-09-11 14:26 README.splunk
-rwxr--r-- 1 507 507 193915 2007-07-18 15:16 fw1-loggrabber.c
```

Debian 32-bit Build Host Setup

Assuming a fresh install of Debian 5.0 'Lenny', standard Workstation selection default with the following additional packages:

openssh-server, gcc-4.3, make, libpam-dev, libelf-dev, libstdc++6-4.3-dev, and unixodbc-dev

(if you wish, this can all be done with the single command: aptitude install openssh-server gcc-4.3 make libpam-dev libelf-dev libstdc++6-4.3-dev unixodbc-dev)

Before compiling fw1-loggrabber, there are some important considerations as to the source code. I recommend reviewing the appendix Hacking fw1-loggrabber before continuing.

From the fw1-loggrabber-splunk directory

Edit Makefile.linux for:

```
CC_CMD = gcc-4.3

LD_CMD = gcc-4.3
```

Uncomment the DYNAMIC ODBC lines and correct them to read:

```
ODBC_CFLAGS = -DDYNAMIC_UNIXODBC -DODBCVER=0x0351
-DUSE_ODBC -I/usr/include

ODBC_LIBS   = /usr/lib/libodbc.so /usr/lib/libodbcinst.so
```

Change the line:

```
LIBS = -lpthread -lresolv -ldl -lpam -lnsl -lelf -lstdc++
```

to read:

```
LIBS = -lpthread -lresolv -ldl -lpam -lnsl -lelf -lstdc++ $
(ODBC_LIBS)
```

At this point, the command ‘make -f Makefile.linux’should complete with no errors and you will have the following executables:

```
/usr/local/src/fw1-loggrabber-splunk/bin/linux/fw1-loggrabber

/usr/local/src/fw1-loggrabber-splunk/opsec-tools/linux22/opsec_pull_cert
```

If this host will be the working log grabber system, fw1-loggrabber and opsec_pull_cert should be copied to a suitable directory for the application, such as ‘/opt/loggrabber’ or

'/usr/local/loggrabber', or whatever you prefer.

If you are going to use fw1-loggrabber on this or another 32-bit system, skip the following section that relate to adding the needed 32-bit libraries to a 64-bit server. Otherwise, copy the executables fw1-loggrabber, opsec_pull_cert and the libarary files /lib/libelf.so.1, libmysqlclient_r.so.15, and libodbc.so to a USB drive and proceed to the 64-bit database server setup. Alternatively, you can use 'scp'from the 64-bit server if the 32-bit build workstation will be available online.

Setting up fw1-loggrabber on a 64-bit server

NOTE: don't forget about NTP.

Assume the following data throughout the configuration process, substituting your actual data where necessary.

```
192.168.1.30   loggrabber.mydomain.net  #the mysql database
               server

192.168.1.75   fw1mgmt.mydomain.net     #the Checkpoint
Management Server
```

One critical item to check on the mysql server is the /etc/hosts file. It should contain a valid network host entry for the server, not a default 127.0.1.1 entry. Ensure that the ip address and hostname match reality, then reboot. If the hostname and ip address don't match the actual network setup, the certificate you retrieve later from the Checkpoint Management Server will NOT work.

These instructions are for the Debian 5 Lenny AMD-64 (64-bit) distribution with the install options "Desktop Environment"de-selected and "Standard System" selected. Adjust accordingly for other distributions.

The following packages added after the post-install reboot:

openssh-server, mysql-server, unixodbc, lib32stdc++6, ia32-libs

(if you wish, this can all be done with the single command:

```
aptitude install openssh-server, mysql-server, unixodbc,
lib32stdc++6, ia32-libs)
```

Continue the server setup by making the working directory /opt/loggrabber and copying the fw1loggrabber and opsec_pull_cert executables to it from the 32-bit build server.

You will then need to copy the following 32-bit libraries from the build system /usr/lib/ to the /lib32 directory of the 64-bit mysql server:

```
libelf.so.1
libmysqlclient_r.so.15
libodbc.so
libreadline.so.5
```

And, you will need to copy the following executable to the /opt/loggrabber directory:

```
/usr/bin/isql
```

I recommend that your rename it to isql32 to prevent any confusion with the native 64-bit isql executable.

In the working directory, use the Linux utility ‘ldd’to verify that the dependent libraries for both fw1-loggrabber and opsec_pull_cert are present. The output should resemble this:

```
/opt/loggrabber# ldd fw1-loggrabber
        linux-gate.so.1 =>  (0xf7f3b000)
        libpthread.so.0 => /lib32/libpthread.so.0 (0xf7f1a000)
        libresolv.so.2 => /lib32/libresolv.so.2 (0xf7f07000)
```

```
            libdl.so.2 => /lib32/libdl.so.2 (0xf7f03000)
            libpam.so.0 => /lib32/libpam.so.0 (0xf7ef8000)
            libnsl.so.1 => /lib32/libnsl.so.1 (0xf7edf000)
            libelf.so.1 => /emul/ia32-linux/lib/libelf.so.1
(0xf7ecb000)
            libstdc++.so.6 => /usr/lib32/libstdc++.so.6 (0xf7ddd000)
            libodbc.so.1 => /usr/lib32/libodbc.so.1 (0xf7d7d000)
            libgcc_s.so.1 => /usr/lib32/libgcc_s.so.1 (0xf7d70000)
            libc.so.6 => /lib32/libc.so.6 (0xf7c1e000)
            /lib/ld-linux.so.2 (0xf7f3c000)
            libm.so.6 => /lib32/libm.so.6 (0xf7bf9000)
            libltdl.so.3 => /usr/lib32/libltdl.so.3 (0xf7bf2000)
```

```
/opt/loggrabber# ldd opsec_pull_cert
        linux-gate.so.1 =>  (0xf7fbb000)
        libpthread.so.0 => /lib32/libpthread.so.0 (0xf7f9a000)
        libresolv.so.2 => /lib32/libresolv.so.2 (0xf7f87000)
        libdl.so.2 => /lib32/libdl.so.2 (0xf7f83000)
        libpam.so.0 => /lib32/libpam.so.0 (0xf7f78000)
        libnsl.so.1 => /lib32/libnsl.so.1 (0xf7f5f000)
        libc.so.6 => /lib32/libc.so.6 (0xf7e0d000)
        /lib/ld-linux.so.2 (0xf7fbc000)
```

If there are no‘Not found’entries, you should be able to enter the command ‘./fw1-loggrabber –h’and see the fw1-loggrabber help menu and proceed to the configuration steps.

Creating the mysql database:

Run mysql as root, then enter the following command:

```
CREATE DATABASE fw1logs;

GRANT ALL PRIVILEGES ON fw1logs.* TO 'fwlogger'@'localhost'
IDENTIFIED BY 'Sn00Py';

flush privileges;
```

Configuring ODBC

The files you will be working with here are /etc/odbc.ini and

/etc/odbcinst.ini.

First make odbc.ini look like this:

```
[FW1-Logs]
Description     = FW1NG Log Connection
Driver          = MySQL
Trace           = Yes
TraceFile       = /tmp/odbc.log
Database        = fw1logs
Server          = localhost
Port            = 3306
User            = fwlogger
Password        = Sn00Py
```

Then, edit odbcinst.ini as follows:

```
[MySQL]
Description     = MySQL driver
Driver          = /lib32/libmyodbc.so
Setup           = /lib32/libodbcmyS.so
UsageCount      = 2
```

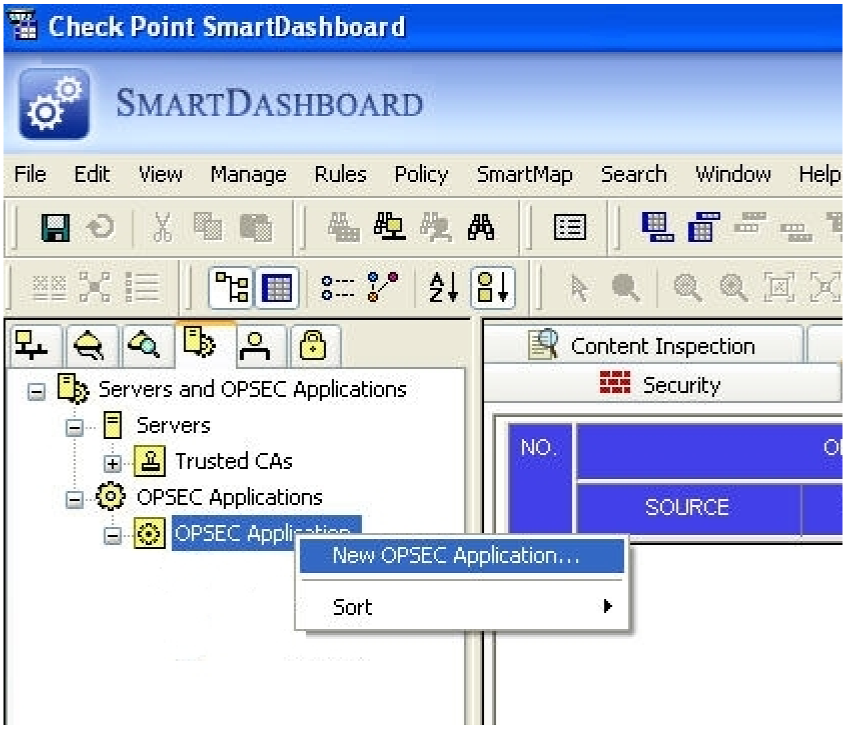
Configuring the Checkpoint Firewall

I recommend doing this phase of configuration from a Windows workstation, so you can run the Checkpint SmartDashboard and ssh to the fw1-loggrabber/MySQL server on the same screen.

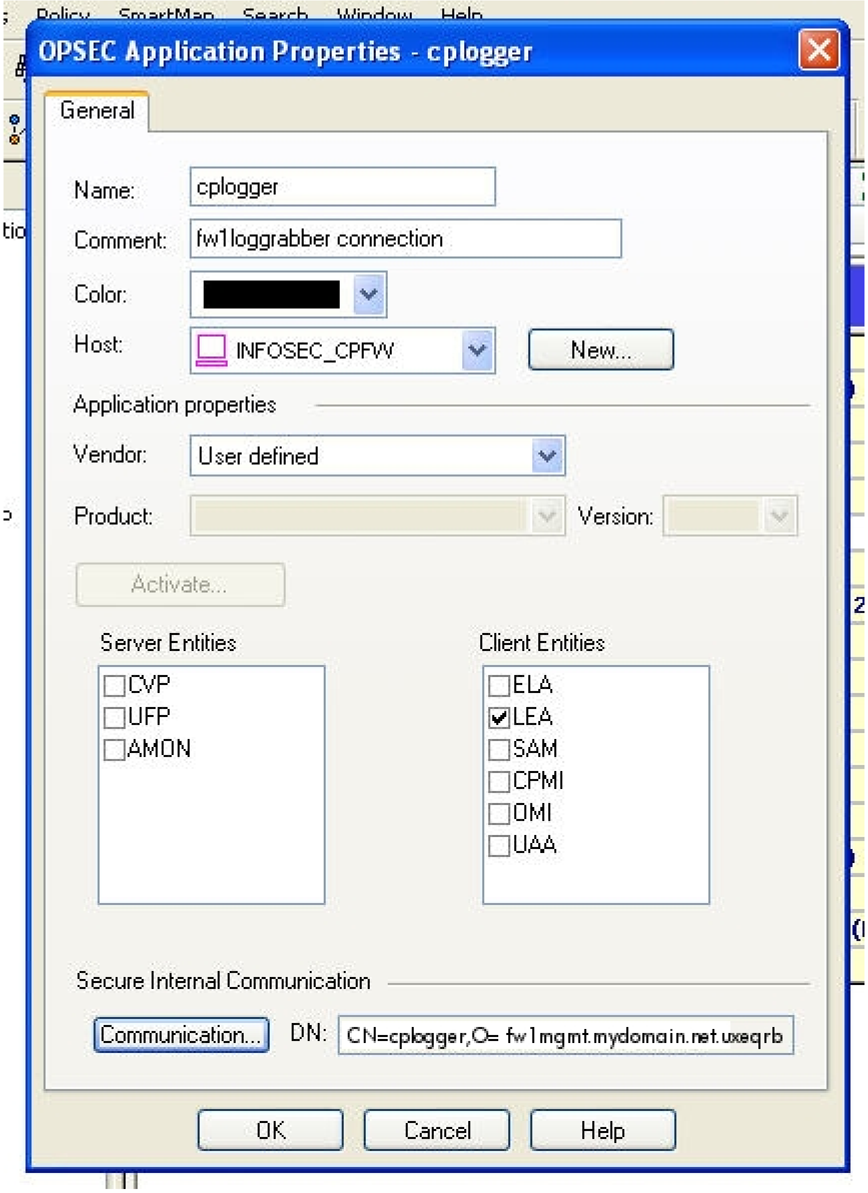
Before the fw1loggrabber database server can be configured to pull firewall logs, an Opsec Application Object must be created in the Checkpoint SmartDashboard.

First, select the ‘Servers and OPSEC Applications’ tab.

Right-clicking on the ‘OPSEC Applications’pull-down will present the option ‘New OPSEC Application’ which should be selected.



After the ‘New OPSEC Application’option is selected, the following configuration window will pop up:



OPSEC Application Properties - cplogger

Name: cplogger

Comment: fw1loggrabber connection

Color: ■

Host: INFOSEC_CPFW

Application properties

Vendor: User defined

Product:

Version:

Server Entities

□CVP
□UFP
□AMON

Client Entities

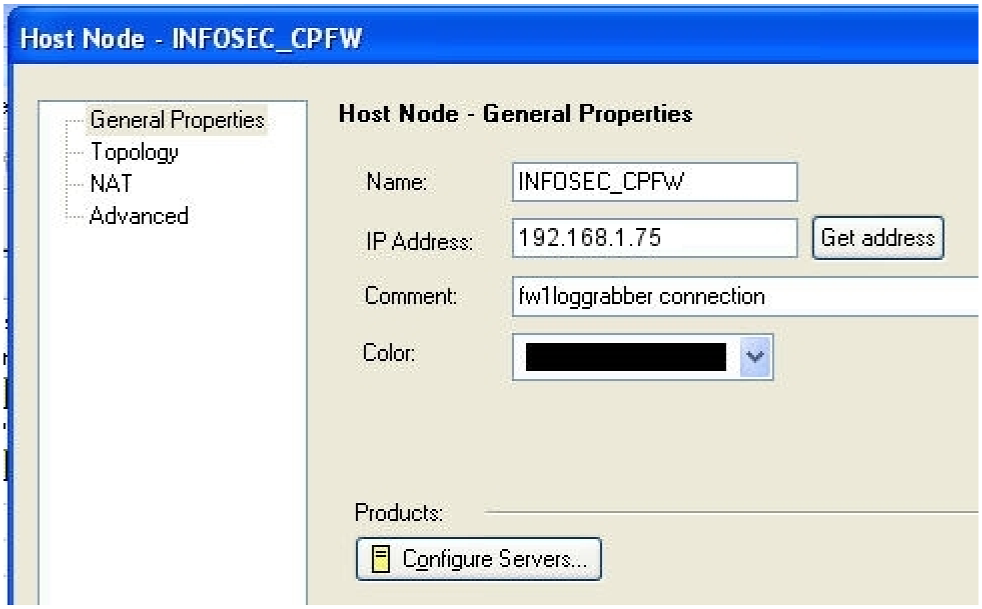
□ELA
☑LEA
□SAM
□CPMI
□OMI
□UAA

Secure Internal Communication

DN: CN=cplogger,O= fw1mgmt.mydomain.net.uxeqrb

The fields you will complete will be Name, Comment, a New Host, Client Entries 'LEA', then the Security Internal Communication section.

Click on the New Host button and enter a suitable object name for the host, the ip address of the fw1-loggrabber server, and a suitable comment to describe the host such as shown:



Host Node - General Properties

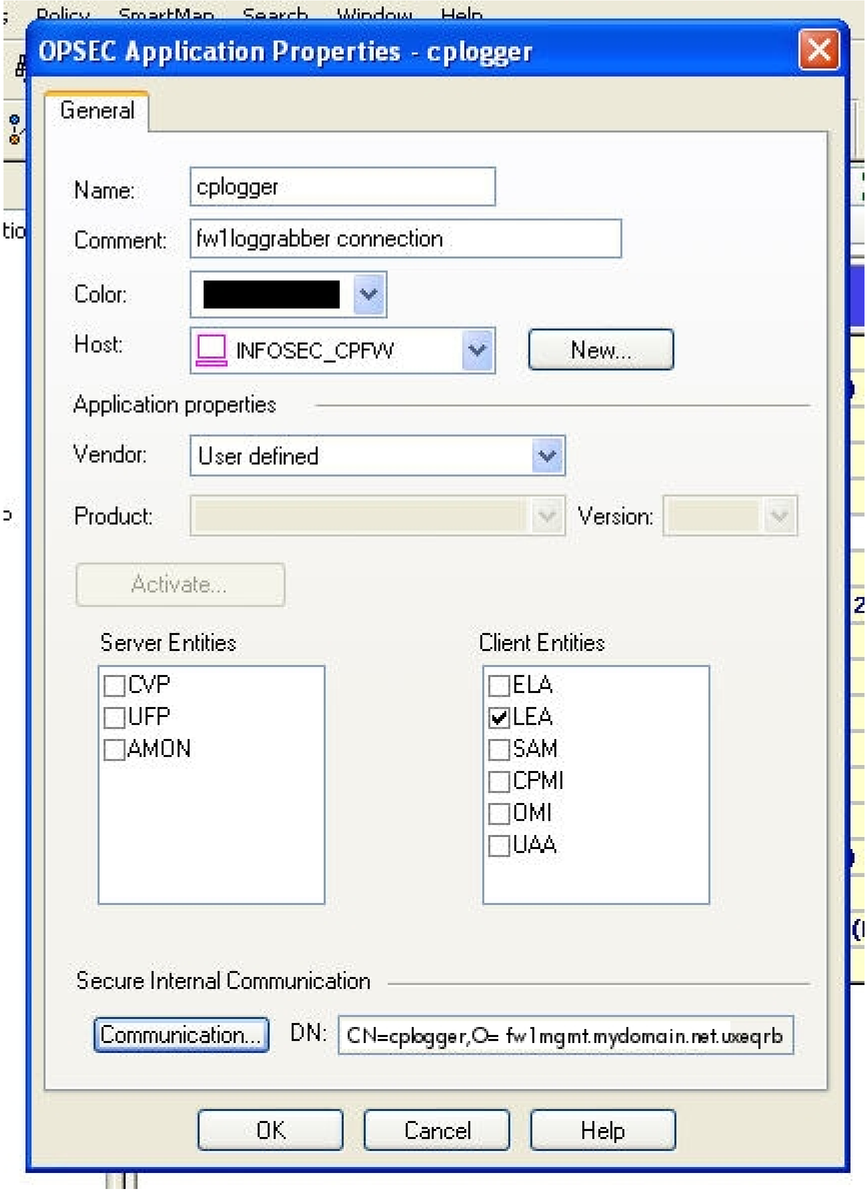
Name: INFOSEC_CPFW

IP Address: 192.168.1.75

Comment: fw1loggrabber connection

Color: ■

Click on the [OK] button to return to the OPSEC Application dialogue.



OPSEC Application Properties - cplogger

Name: cplogger

Comment: fw1loggrabber connection

Color: ■

Host: INFOSEC_CPFW

Application properties

Vendor: User defined

Product:

Version:

Server Entities

□ CVP
□ UFP
□ AMON

Client Entities

□ ELA
☑ LEA
□ SAM
□ CPMI
□ OMI
□ UAA

Secure Internal Communication

DN: CN=cplogger,O= fw1mgmt.mydomain.net.uxeqrb

The last step of the configuration is to coordinate setup of the Secure Internal Communication section with setup on the MySQL server LEA client side. At this point, the lea.conf and fw1-loggrabber.conf config files should be completed. Be sure to copy or write down the above information in the DN: field before proceeding.

Configuring fw1-loggrabber

In /opt/loggrabber, edit lea.conf as follows:

```
lea_server auth_type sslca
lea_server ip 192.168.1.75    fw1mgmt
lea_server auth_port 18184
lea_server port 18184
opsec_sic_name "CN=cplogger,O= fw1mgmt.mydomain.net.uxeqrb"
opsec_sslca_file /opt/loggrabber/opsec.p12
lea_server opsec_entity_sic_name "cn=cp_mgmt,o=
fw1mgmt.mydomain.net.uxeqrb"
```

The lea_server ip and host name are of the Checkpoint firewall management server. The opsec_sic_name entry comes from the ‘DN:’field in the OPEC Application dialogue. The lea_server opsec_entity_sic_name comes from the common name ‘CN’of the Checkpoint firewall management server and the organization ‘O’ entry.

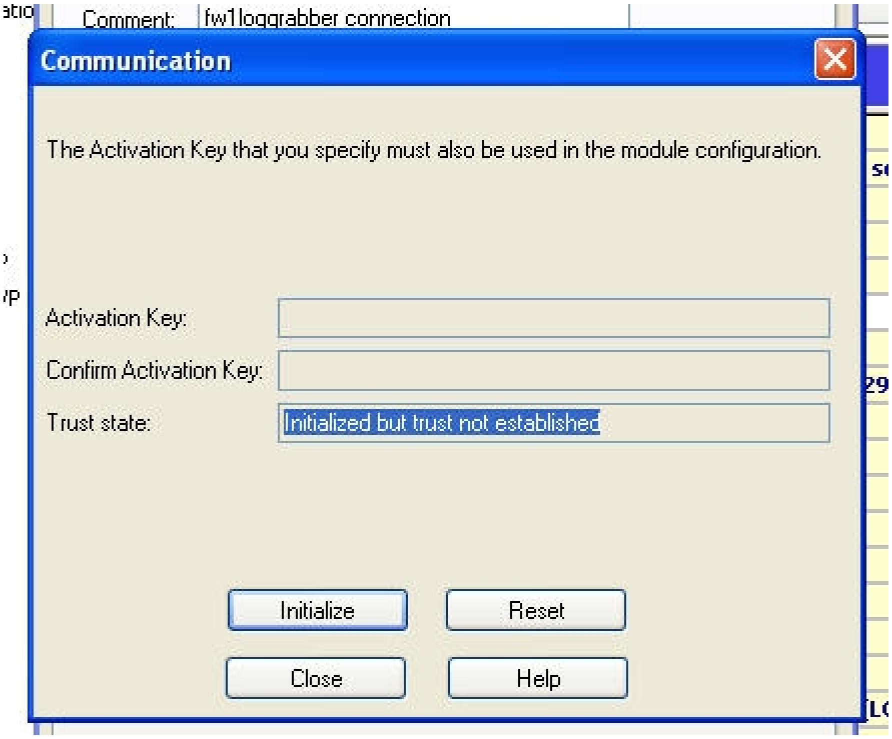
In /opt/loggrabber, edit fw1-loggrabber.conf as follows:

```
DEBUG_LEVEL="0"

#
# FW1 configuration settings
#
FW1_LOGFILE="fw.log"
FW1_OUTPUT="logs"
FW1_TYPE="ng"
FW1_MODE="normal"
ONLINE_MODE="yes"
RESOLVE_MODE="no"
SHOW_FIELDNAMES="no"
RECORD_SEPARATOR="|"
DATEFORMAT="std"
LOGGING_CONFIGURATION=odbc
OUTPUT_FILE_PREFIX="fw"
OUTPUT_FILE_ROTATESIZE=10000482
SYSLOG_FACILITY="LOCAL1"
ODBC_DSN=FW1-Logs
```

Once the above configuration files are edited on the MySQL/fw1-loggrabber server, the next steps will alternate between that server and the Checkpoint firewall management server OPSEC Application dialogue.

Click on the [Communication] button, and an 'Activation Key'dialogue will pop up. Here, you will define a one-time password that you will use on the MySQL/fw1-loggrabber server. Create a password, write it down, then enter it into both the 'Activation Key:' and 'Confirm Activation Key:' fields of the 'Communication'dialogue, then click on the [Intialize] button. After seeing that the 'Trust State:'has changed, click on the [Close] button. It's time now to opsec_pull_cert the certificate you just created.



Comment: fw1loggrabber connection

The Activation Key that you specify must also be used in the module configuration.

Activation Key:

Confirm Activation Key:

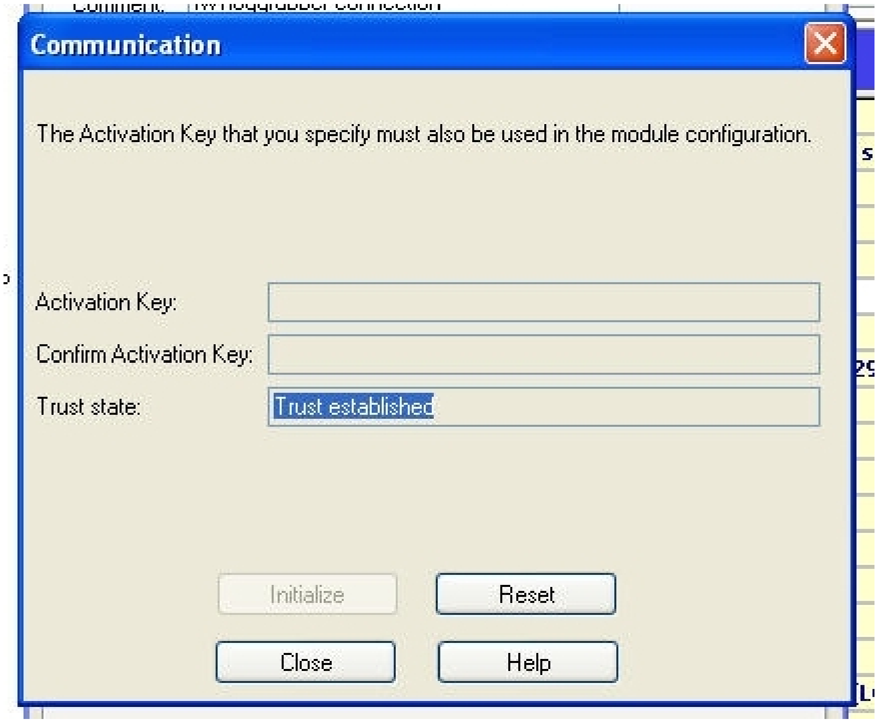
Trust state: Initialized but trust not established

From the MySQL/fw1-loggrabber server, change to the /opt/loggrabber directory and run opsec_pull_cert as follows:

./opsec_pull_cert -h 192.168.1.75 -n cplogger -p <one-time password here>

If there are no errors, opsec_pull_cert will use the one-time password to grab the certificate from the Checkpoint firewall management server and store it in /opt/loggrabber as opsec.p12. That certificate is used by OPSEC to encrypt communication between the Checkpoint firewall management server and the fw1-loggrabber client running on the MySQL server.

Back on the Checkpoint firewall management server OPSEC Application dialogue, it's time to click on [Communication] again. This time, you should see 'Trust Established'as below:



The Activation Key that you specify must also be used in the module configuration.

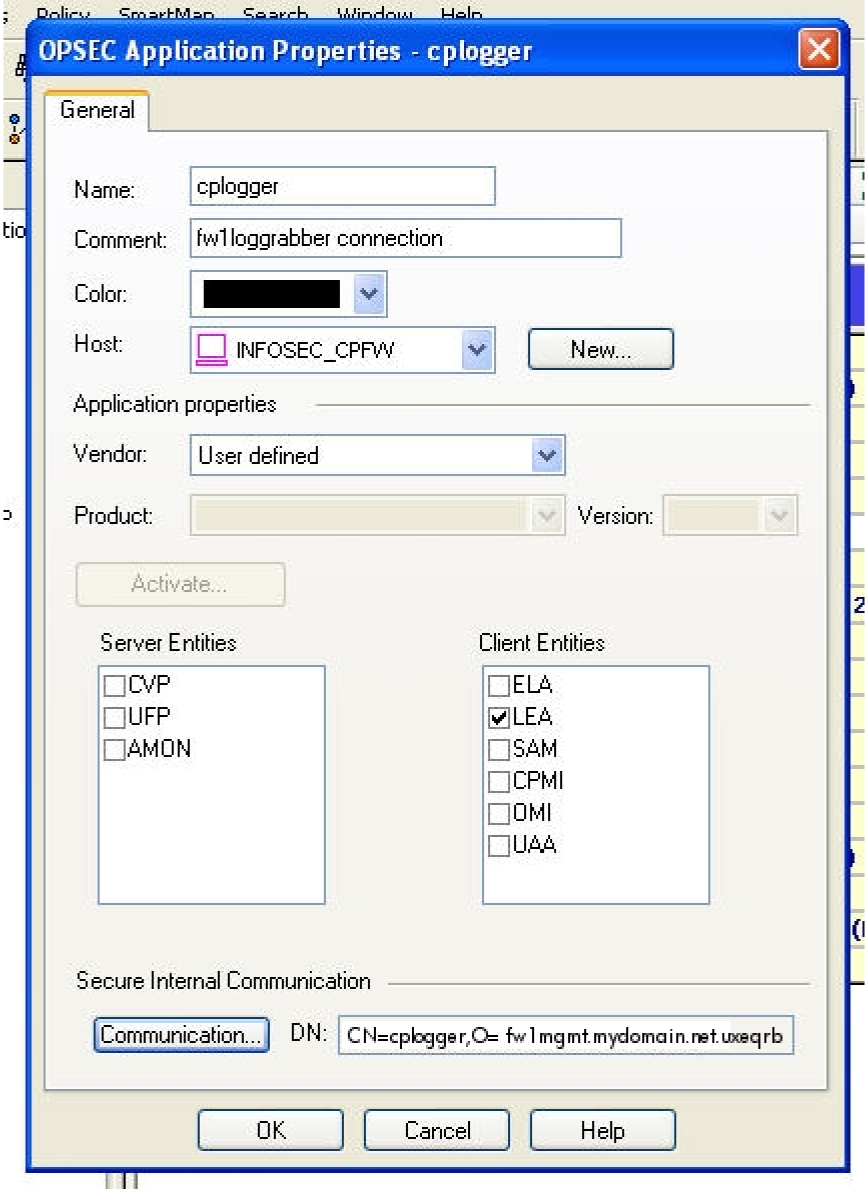
Activation Key:

Confirm Activation Key:

Trust state: Trust established

You can now click on [Close] to quit the Communication dialogue.

And, now that the new OPSEC Application is fully defined, you can click on [OK] to close this dialogue.



OPSEC Application Properties - cplogger

Name: cplogger

Comment: fw1loggrabber connection

Color: ■

Host: INFOSEC_CPFW

Application properties

Vendor: User defined

Product:

Version:

Server Entities

□CVP
□UFP
□AMON

Client Entities

□ELA
☑LEA
□SAM
□CPMI
□OMI
□UAA

Secure Internal Communication

DN: CN=cplogger,O= fw1mgmt.mydomain.net.uxeqrb

IMPORTANT: Now that you've done all this work, be certain to save your configuration changes when you quite the Checkpoint firewall management server's SmartDashboard.

6.Hacking the fw1-loggrabber code

I had two problems with the original fw1-loggrabber source code in the ODBC ONLINE_MODE.

The first problem was with the selection and number of fields retrieved. In ONLINE_MODE, or real-time log retrieval, all of the defined fields are grabbed. Some of those fields are firewall specific, while others pertain to VPN, Smart Defense, and other capabilities. Since I only wanted the firewall connection specific fields, this presented a problem. In addition, I particularly wanted the object named ‘Rule_UID’, which is not configured in the stock fw1-loggrabber source code.

The second problem I had was the indexing. To speed up searches in the MySQL database, all critical fields should be indexed. In the original source code, this was not the case.

The solution was to hack the source code, which is a wonderful aspect of open source software. It also helps to have quite a bit of C programming experience, which I have.

To demonstrate the field problem, observe the sample output below from a stock fw1-loggrabber executable:

```
      | fw1number | fw1time              | fw1action | fw1orig        | fw1alert |
fw1if_dir | fw1if_name | fw1product         | fw1src          | fw1s_port | fw1dst
| fw1service | fw1tcpflags | fw1proto | fw1rule | fw1xlatesrc | fw1xlatedst |
fw1xlatesport | fw1xlatedport | fw1nat_rulenum | fw1resource | fw1elapsed |
fw1packets | fw1bytes | fw1reason | fw1service_name | fw1agent | fw1from | fw1to |
fw1sys_msgs | fw1fw_message | fw1internal_ca | fw1serial_num | fw1dn | fw1icmp |
fw1icmp_type | fw1icmp_type2 | fw1icmp_code | fw1icmp_code2 | fw1msgid |
fw1message_info | fw1log_sys_message | fw1session_id | fw1dns_query | fw1dns_type
| fw1scheme | fw1srckeyid | fw1dstkeyid | fw1methods | fw1peer_gateway | fw1ike |
fw1ike_ids | fw1encryption_failure | fw1encryption_fail_r | fw1cookiei |
fw1cookier | fw1start_time | fw1segment_time | fw1client_in_packets |
fw1client_out_packets | fw1client_in_bytes | fw1client_out_bytes | fw1client_in_if
| fw1client_out_if | fw1server_in_packets | fw1server_out_packets |
fw1server_in_bytes | fw1server_out_bytes | fw1server_in_if | fw1server_out_if |
fw1message | fw1nat_addrulenum | fw1user | fw1srcname | fw1vpn_user | fw1om |
fw1om_method | fw1assigned_ip | fw1mac | fw1attack | fw1attack_info |
fw1cluster_info | fw1during_sec | fw1fragments_dropped | fw1ip_id | fw1ip_len |
fw1ip_offset | fw1tcp_flags2 | fw1sync_info | fw1log | fw1cpmad | fw1auth_method |
fw1tcp_packet_oos | fw1rpc_prog | fw1th_flags | fw1cp_message | fw1reject_cat |
```

Now, look at the headers of my hacked version:

```
      | fw1loc | fw1time              | fw1action | fw1orig        | fw1if_dir |
fw1if_name | fw1src         | fw1s_port | fw1dst       | fw1service | fw1tcpflags |
fw1proto | fw1rule_uid                                  |
```

Quite a difference. Now, let's compare the differences in indexing.

In the stock source code, only the fw1number field is indexed. In my hacked code, fw1time, fw1src, fw1dst, and fw1service are indexed.

If you would prefer my hacked version of the code, I am providing a package of the hacking notes, a patch file that includes my hacks, and a copy of the working fw1-loggrabber executable, just in case you don't want to do any hacking or compiling at all.

If there is enough interest in my implementation of fw1-loggrabber, I could be motivated to build a complete Linux installation disc.

Email me at ‘fwmark _AT_ altsec.info’to get a download link to my hack package and a status report on availability of any improvements, or if you would like to help out with the project.

7.Scripts for fw1-loggrabber

fw1-rotate.sh

This script is run from a cron job at 23:59 every night. It not only rotates the current log to an archive table, it compresses the table and restarts fw1loggrabber.

Crontab for root (crontab –e):

```
# m h  dom mon dow   command
59 23 * * * /usr/local/loggrabber/fw1-rotate.sh
```

Script:

```
#!/bin/sh
#fw1-rotate.sh will archive the current ‘fw1logs’ table to a
date named archive table, such as ‘fw1logs.20090529’
#it stops fw1-loggrabber, renames the current table to archive,
recreates the current table and restarts fw1-loggrabber, then
compresses the archived table
TNOW=`date "+%Y%m%d"`
TFILE=`tempfile`

cd /usr/local/loggrabber

echo "rename table fw1logs.fw1logs to fw1logs.$TNOW" > $TFILE
echo "drop table fw1logs.auditlogs" >> $TFILE
echo "drop table fw1logs.loggrabber" >> $TFILE
echo "" >> $TFILE

FPID=`/bin/pidof fw1-loggrabber`

/bin/kill -KILL $FPID

/usr/bin/isql32 FW1-Logs32 fwlogger Pz8_RR-uwB#d < $TFILE
```

```
     /usr/local/loggrabber/fw1-loggrabber --create-tables

     /usr/local/loggrabber/fw1-loggrabber
1>/usr/local/loggrabber/grabber.log
2>/usr/local/loggrabber/grabber.err &

     rm -f $TFILE

     echo "Compressing $TNOW"

     cd /var/lib/mysql/fw1logs

     myisampack -v $TNOW
     myisamchk -rq $TNOW
```

8.References

Firewall Log Analysis Primer.
http://www.secureworks.com/research/articles/firewall-primer/

OPSEC LEA Integration – Splunk. The link to Splunk's packaging of fw1-loggrabber with Checkpoint's OPSEC SDK.
http://www.splunk.com/view/SP-CAAABJV

Splunk fw1-loggrabber package.
http://download.splunk.com/support/OPSEC/fw1-loggrabber-splunk.tar.gz

Checkpoint OPSEC SDK downloads.
http://www.opsec.com/cp_products/90.htm

MySQL Help. http://forums.mysql.com/
http://dev.mysql.com/doc/index.html

Using opsec_pull_cert.
https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk11520

OSSIM (Open Source Security Event Management).
https://www.ossim.net/

Log Analysis. http://www.loganalysis.org/

SEC (Simple Event Correlator). http://www.estpak.ee/~risto/sec/

[1] Google search hits. Sex 783,000,000 - Drugs 187,000,000 - Rock and Roll 41,300,000 – Firewall Log Analysis 380,000. Retrieved February 21, 2009 from www.google.com

[2] Analyze This, Haral Tsitsivas. Retrieved March 17, 2009 from http://www.unisol.com/papers/UNISOL_ED_IWA.pdf