

Using Cisco IOS Logging - Part 2

Michael J. Martin

This article is the second in a two part series on implementing Cisco IOS system message log reporting. The IOS supports four types of system message reporting, console, monitor, buffer, and trap. The first three provide for viewing and storing log messages locally. The fourth method, trap, provides remote system message reporting via Unix syslog service. IOS system message log reporting configuration and an overview of Unix syslog was covered in part one. In part two, we will examine a strategy for managing IOS message logging in a large-scale router environment and explore some additional IOS configuration tasks that support system message logging. We'll configure a syslog server to accept IOS generated logging messages using Red Hat Linux 6.2 and then close with a practical example of using IOS message logging to report on potential vulnerability scans against your network.

A Remote Point Of View Of Logging

There is no "best" way to managing system message logging, only opinions. So this approach is put forth as an opinion, with the hope this solution may appeal to you as such or provide a basis for you to develop your own. This solution is designed to structure log reporting in a manner that is functional and scalable (two ideas that are often in conflict when it comes to networking). It provides a centralized means for log collection, monitoring and access in a networking environment where multiple engineers are responsible for supporting and maintaining the infrastructure in parallel with automated monitoring and reporting tools.

Large-scale network environments generate a tremendous amount of logging data. The majority of this provides status and debugging information (info and debug severity) that is generally only looked at when troubleshooting or monitoring a problem. However, the small minority of system messages generated at the notice severity and above provides key information on the operational status of the router. This key data is often missed, leaving potential problems uncorrected because the logging data is generated and displayed locally on the router.

This approach moves all system message reporting to a central logging server. All logging messages are archived to files that can be parsed using standard Unix text tools (more, less, aux, grep, and tail). In addition to centralizing log reporting for easy access, log files may also be monitored by Unix scripts to look for and report on specific severity events. This shifts some of the monitoring work to the computer, which does not need to sleep or finish 1000 other things before it has time to dig through log files. But not all events can be left to the computer, so to ensure that key messages reach the eyes they need to, messages with debug severity, notice severity and higher are directed to shared user accounts that network administrators can log in and view as logging events in real-time. All that's needed is to open a vty session to the server and check the terminal from time to time, what could be easier? So let's get started with the IOS configuration.

IOS Message Reporting Configuration, Redux

In part one, we reviewed the core configuration requirements for system message reporting. Here we will look at some of the supplemental options that enhance the value of message reporting, helping you to get the most out of the log data you are collecting. The configuration examples are provided as part of a solution. The configuration steps are in order along with comments.

Step 1: Configure Loopback Interface

The Loopback interface (in this case) serves as the single network identity for the router. With dynamic routing enabled, its IP address is available through any active router interface (that is also

Using Cisco IOS Logging - Part 2

Michael J. Martin

part of the dynamic routing policy). This availability makes it perfect to serve as the routers vty access interface and provide the source address for logging and SAA (secure system auditing and accounting) reporting and authentication requests. When using Loopback addresses on public networks, the Loopback prefixes should not be reachable from hosts outside of the networks administered routing domain to reduce security exposure. The use of RFC1918 addresses is perfect for this function because they are "un-routable" on the Internet and (in most cases) filtered by AS transit peers. In this configuration example, the Loopback interface is configured using a subnet from the RFC1918 prefix 192.168.0.0/16. To ensure reachability, the interface is added to the routers OSPF policy and will be announced as a host route.

```
Godzilla-ABR(config)# interface loopback 0
Godzilla-ABR(config-if)# ip address 192.168.30.1 255.255.255.255
Godzilla-ABR(config-if)# exit
Godzilla-ABR(config)# router      ospf 3
RT01-outland-n(config-router)# network 192.168.30.1 1 0.0.0.0 area 0.0.0.0
RT01-outland-n(config-router)# exit
```

Step 2: Configure TIME Settings

Time, specifically timestamp, is an valuable piece of information when determining when a problem arose. The idea being that many network problems can often be correlated to system configuration changes, modifications to the networks topology (both intentional and unintentional), etc. While the primary goal of problem resolution is to fix the problem, it is also quite helpful to know the "who, what, and where" of the problem's origin. So if the problem is a result of a management or design error, it can be corrected and avoided in the future. For timestamps to be of use, it is a good idea for all the routers and switches in the network to derive time from a common network time source using the Network Time Protocol (NTP).

Configuring time services on routers is one of those unique tasks that requires both exec and configuration commands. To configure the router's time zone proprieties, the configuration commands <clock timezone {TZ} {gmt offset}> and <clock summer-time {DST TZ} recurring> are used. The configuration commands <ntp server {ip addr}> and <ntp source {interface}> define the NTP server(s) the router should sync with and the source IP address of the NTP requests (important if your NTP server requires that peers be defined). The router's internal clock is set using the exec command <clock set {hh:mm:ss} {day month year}>. The clock must be set when the router is first configured and must be set reasonably accurately to the NTP server in order for time synchronization to function.

To view NTP peer status information, use the exec commands <show ntp associations> and <show ntp status>.

Here is an example of the NTP configuration:

```
Godzilla-ABR(config)# clock timezone EST -5
Godzilla-ABR(config)# clock summer-time EDT recurring
Godzilla-ABR(config)# ntp server 172.30.71.103
Godzilla-ABR(config)# ntp source loopback 0
```

Here we set the system clock:

Using Cisco IOS Logging - Part 2

Michael J. Martin

```
Godzilla-ABR#clock set 21:51:20 7 April 2002
```

Here we configure the timestamp options for system logging and debug messages:

```
Godzilla-ABR(config)# service timestamps log datetime localtime
Godzilla-ABR(config)# service timestamps debug datetime localtime
```

Here is some output from the <show ntp associations > commands. NTP associations list the servers available to the router for synchronization, their availability and desirability status, and the current server being used for time sync.

```
Godzilla-ABR#sh ntp associations
```

```
address ref clock st when poll reach delay offset disp
*~17.254.0.31 17.254.0.49 2 54 64 377 85.2 -4.08 0.8
master (synced), # master (unsynced), + selected, - candidate, ~ configured
```

The IOS requires that the NTP server be a STRATUM3 time source or better. A list of publicly available STRATUM1 servers is available at <http://www.eecis.udel.edu/~mills/ntp/clock1.htm>. A list of publicly available STRATUM2 servers is available at <http://www.eecis.udel.edu/~mills/ntp/clock1.htm>. While there are a number of publicly available timeservers on the Internet, these servers are intended to provide synchronization services for privately maintained NTP servers. It is common best practice for a network to deploy its own NTP server. In the best possible case, the logging server should also function as the NTP server. This can be accomplished quite easily by either installing an RPM or compiling and installing the code for ntpd, which is available via ftp or http at <http://www.eecis.udel.edu/~ntp/>.

Step 3: Configure Local (Router) System Message Reporting

First, disable console and buffered logging. What, you say? Well yes, since you really don't need it. For starters, to view messages sent to the console, you need to be directly connected to the console port. In most large-scale environments, the console port is often connected to a remote access terminal server or a modem for out-of-band access. Leaving console logging enabled sends messages to the console port, and, during an event that generates a large amount of logging messages (which is when you typically want out-of-band access), clutters the command line, making command entry difficult. Buffer logging, which stores the messages in a buffer carved out of the router's DRAM, is a great option if trap logging is not a feasible option. In our example, however, that option would be redundant and consume router resources that are quite valuable on low-end routers. A more practical method for viewing logging messages locally is to enable monitor logging using the configuration command <logging monitor {syslog facility}>, then use the exec command <terminal monitor> to view messages when you need to. To console and buffered logging (or any form of logging, prepend "no" before the configuration command:

```
Godzilla-ABR(config)# no logging console
Godzilla-ABR(config)# no logging buffered
```

To enable monitor logging: `Godzilla-ABR(config)# logging monitor debugging`

Using Cisco IOS Logging - Part 2

Michael J. Martin

Step 4: Configure Remote (Server) System Message Reporting (From Part One)

Configuring trap logging is a four step process:

1. Define a syslog host using the configuration command <logging {hostname or IP address}>.
2. Define the logging severity of the messages to be sent using the configuration command <logging trap {severity}>.
3. Define the IP address that will be associated as the origin address of the logging messages. This is set using the configuration command <logging source-interface {interface type m/s/p}>.
4. The final step defines the syslog "facility" that the messages are sent to on the remote syslog server. Use the configuration command <logging facility {syslog facility}>.

In this configuration example, logging messages are sent to two syslog servers. The system messages are sent to syslog facility local, and the logging messages use the IP address of the

Loopback interface:

```
Godzilla-ABR(config)# logging facility local
Godzilla-ABR(config)# logging source-interface Loopback 0
Godzilla-ABR(config)# logging trap debug
Godzilla-ABR(config)# logging 172.30.71.103
Godzilla-ABR(config)# logging 172.30.71.200
```

Constructing a syslog server

On the hardware side, not much is required. A Pentium 166 MHz PCI-based system with a Fast Ethernet (Intel 100pro if possible) and a minimum of 64M bytes of DRAM will do just fine. As a logging server, a 4G byte or larger hard drive is recommended. For software, Red Hat Linux 6.2 is provides almost everything in terms of the software we need, except for FreeSSH and FreeSSL, which need to be downloaded, compiled and installed to provide secure virtual terminal access to the server. If compiling code is new to you, instructions on compiling SSH and its dependent packages is contained in this Search Networking tip.

When installing RH6.2, select the "Workstation Install" and follow the options for custom file system configuration. Here is an example layout (based on a 4G byte hard disk):

```
/          1000Mb (contains /etc, /usr,/dev, etc)
/boot     128Mb (kernel files)
/home     512Mb (User directories)
Swap     128Mb (assuming a server with 64Mb or DRAM)
/log     2000Mb (routing logging files)
/var     256Mb (system logging files)
```

The install will prompt you to configure X windows and if you want to boot in to X windows on startup. Complete the X windows configuration (if you have problems skip it), but say no to X on start-up. Once the install is complete, the server will boot using the OEM kernel with ipchains support. Internet

Using Cisco IOS Logging - Part 2

Michael J. Martin

access is required to complete the server configuration. But before connecting the server to the network, complete the first configuration section, security modifications.

Step 1: System Security Modifications

There are a number of existing publications on securing Linux installations. One of the best, by one of the HoneyNet Project members, Lance Spitzner, is entitled *Armoring Linux*. It provides some great advice, and is well worth the look. The reason for using the Red Hat 6.2 workstation as a base is because the install provides no access network services by default. So in order to access the server, a virtual terminal service such as telnetd or an equivalent service must be installed. While telnet is the de facto for virtual terminal access, all communication is open and unencrypted. This vulnerability makes SSH the secure and sensible choice for virtual terminal access. It provides not only encrypted session service, but support for secure authentication services such as S-Key, RSA public key and encrypted clear text passwords.

Step 2: Secure Syslog Configuration

Syslog is a standard part of any Unix implementation. On Linux systems, syslogd and klogd (kernel message logger) services are enabled as the system boots. This default configuration only accepts locally generated system messages. This is done largely for security reasons, since syslog provides no means for authentication or access control. So in order for our server to accept externally generated messages, some configuration is required on our part.

Disable Default Logging Configuration

Unix uses a series of scripts to start and stop core operational functions at system boot and shutdown. Under Linux, these scripts are managed by a central process daemon called `init`, which is launched as the last step of the kernel load sequence, giving it the unique distinction of always having a process ID of 1. The `init` service learns what processes to do from its configuration file `/etc/inittab`. The `inittab` file uses a hierarchy known as `runlevels` (this functionality is borrowed from ATT System V Unix; Berkeley Unix uses a slightly different approach with a series of `rc` -- run control -- scripts) to determine what processes should exist. There are 12 runlevels: 0 through 9, S and s, 0 through 6 are used to start up and shut down most System V derived Unixes. Levels 7 through 9 exist but are deprecated and not used. Each runlevel has a collection of execution scripts that load or unload the services associated with the runlevel. These are located in `/etc/rc.d` directory. Specific runlevel scripts are contained in subdirectories named with the corresponding runlevel number, i.e., `rc4.d`. To modify the behavior of the default services first, determine what runlevel the system is operating in (runlevel 3 is the default for multi-user access with network services) using the Unix command `<runlevel>`.

```
[root@parsafal /root]#runlevel
N 3
[root@parsafal /root]#
```

Alternatively, you can `grep` the `/etc/inittab` file, searching for the `initdefault` statement:

```
[root@parsafal /etc]# grep "initdefault" /etc/inittab
id:3:initdefault:
[root@parsafal /etc]#
```

Using Cisco IOS Logging - Part 2

Michael J. Martin

Once the system's runlevel has been determined, we can move to the specific runlevel directory located in /etc/rc.d, review the services associated with the runlevel and disable the services we do not want to execute at boot.

```
[root@parsafal root]# cd /etc/rc.d/rc3.d/
[root@parsafal rc3.d]# ls
K20nfs          S10network     S25netfs      S40crond      S75keytable   S99local
K20rwhod       S11portmap     S30syslog     S45pcmcia     S80sendmail
K45named       S14nfslock     S35identd     S47ipsec      S85gpm
K92ipchains    S16apmd        S35ipsec      S50inet       S90xfs
S05kudzu       S20random      S40atd        S60lpd        S99linuxconf
[root@parsafal rc3.d]#
```

Two types of scripts exist in the runlevel directory "K" or kill scripts (run at shutdown) and "S" or start scripts. To start or stop a service, execute the script by prepending "./" to the script, followed by "start" or "stop" to achieve the desired state.

Syslogd needs to be reconfigured to accept logging messages from remote hosts. To start the current syslogd and klogd processes need to be disabled.

```
[root@parsafal rc3.d]# ./S30syslog stop
Shutting down kernel logger: [ OK ]
Shutting down system logger: [ OK ]
[root@parsafal rc3.d]#
```

Once the services are stopped, we need to disable the service script. The simple way is to just delete the script. The more elegant method is to change the capital "S" to a lowercase "s." Only scripts with a capital "S" or "K" are executed.

```
[root@parsafal rc3.d]# mv S30syslog s30syslog
```

Configuring Syslog Service

When looking at the contents of the runlevel directory, you will notice that along with the service name there is a number. The number determines the order (lowest to highest) that the scripts are executed in. In certain cases, services are dependent on one another to function correctly. If a dependent service is not active when a script executes, the service will often fail. The runlevel script S99local (the last script executed) provides a means for administrators to load custom services. When running non-standard services, it is a good idea to launch them using S99local. The configuration file for S99local is /etc/rc.d/rc.local. The runlevel script S30syslog loads klogd and syslogd, so the S99local script must also load both services. Adding a service is easy -- go to the end of the file and add the full command path and flags:

```
[root@parsafal rc3.d]# cd /etc/rc.d
[root@parsafal rc.d]# vi rc.local
```

```
#Start klogd and syslogd
echo Starting klogd
/sbin/klogd
Revised May 8, 2002
```

Using Cisco IOS Logging - Part 2

Michael J. Martin

```
/sbin/syslogd -hr
:wq
[root@parsafal rc.d]#
```

Klogd requires no special flags. Syslogd requires two flags to support network logging services. The -h flag permits the daemon to forward logging messages to other syslog hosts if the syslogd.conf file is configured to do so. This is handy if your environment is quite large and you want to deploy multiple syslog servers to collect data locally and relay key events back to a central server. The -r flag enables the daemon to accept messages from other devices over the network on UDP port 514.

Once again, there is a more elegant method than using a single command line statement -- a shell "if then" statement. Here is an example (that can be run from rc.local or as a standalone runlevel script) that checks to see if the syslog.conf file exists. If it does, it starts syslogd and klogd; if not, it reports an error.

```
#
VAR=`ls /etc/syslog.conf`
#
kill -9 `cat /var/run/syslogd.pid`
kill -9 `cat /var/run/klogd.pid`
if
[ "$VAR" = "/etc/syslog.conf" ]; then echo "starting syslog" &&
/sbin/syslogd -hr && echo "starting klogd" &&
/sbin/klogd
elif [ "$VAR" = "" ]; then
echo "Starting syslogd failed"
fi
```

Securing Syslog Service

Syslogd uses UDP datagrams for unencrypted transport message and provides no facilities for access filtering or authentication. So while it's great for logging, it falls short on security. Syslogd is vulnerable to denial service attacks and IP spoofing attacks, and there are known buffer overflow vulnerabilities with certain implementations. With the proper scanning techniques employed, syslog will show up if the logging host is scanned. So while there is no perfect defense, something is better than nothing. To defend against potential attacks we can apply inbound access filtering using ipchains, a kernel level packet-filtering tool developed for building Linux firewalls. Here is an example script that applies an inbound access filter that restricts access to the syslog service. Operation is simple -- hosts that are permitted to send syslog messages are added to the "permit statement" section and the script is rerun. The ACL also permits basic ICMP (echo, echo-reply, and host-unreachable) and SSH. All other services ports below 1023 (UDP/TCP) are filtered closed, and above 1023 are open to allow for outbound network connections to function properly.

```
<start copy> #!/bin/sh
# This is a simple filter to restrict access to the syslog service port
# using ipchains. This script should be run at boot time from rc.local.
#
#
# Script Variables
```

Using Cisco IOS Logging - Part 2

Michael J. Martin

```
# Host IP Address
HOST=172.30.71.103
#
# ipchains binary location
#
IPF=/sbin/ipchains
#
# Flushing existing input chain
$IPF -F input
#
# Permit Statement
# Each host that is permitted to send syslog messages should have an entry.
$IPF -A input -p udp -s 192.168.2.1-d $HOST 514 -j ACCEPT
$IPF -A input -p udp -s 192.168.2.3-d $HOST 514 -j ACCEPT

#Default Permit Statements
$IPF -A input -p icmp -s any/0 -d $HOST 0 -j ACCEPT
$IPF -A input -p icmp -s any/0 -d $HOST 8 -j ACCEPT
$IPF -A input -p icmp -s any/0 -d $HOST 3 -j ACCEPT
$IPF -A input -p tcp -s any/0 -d $HOST 22 -j ACCEPT
$IPF -A input -p tcp -s any/0 -d $HOST 1023:65535 -j ACCEPT
$IPF -A input -p udp -s any/0 -d $HOST 1023:65535 -j ACCEPT
#
# Deny Statements For IPCHAINS
$IPF -A input -s any/0 -d any/0 -j REJECT
<end copy>
```

The script was written to run on RH Linux (6.2 and higher). To use the script, just copy the text between the marks, paste into a text file and save as firewall.sh. Then, as root, run the following commands:

```
[trinity:~] root# chown root firewall.sh
[trinity:~] root# chmod o+x firewall.sh
```

Configuring Log Reporting Using Syslog.Conf

The last step of the server configuration is the creation of the syslog.conf file. The default syslog.conf is configured with the message log files stored the directory /var/log. Here is an excerpt from the default syslog.conf file that ships with RH 6.2:

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none /var/log/messages
# The authpriv file has restricted access.
authpriv.* /var/log/secure
# Log all the mail messages in one place.
mail.* /var/log/maillog
```

These log files are managed by /usr/sbin/logrotate, which is run as a daily cron job. The default configuration stores log events for a 7-day period in a text file (with no size restriction). Four weeks of log data is stored on a rotating basis, and log files older than 4 weeks are deleted. The configuration

Using Cisco IOS Logging - Part 2

Michael J. Martin

for logrotate is logrotate.conf. In addition to the default, behavior provisions for custom logfile management such as file size growth restriction, frequency of rotation and mail reporting are also supported.

Saving files to /var is the default behavior for most applications that generate log and other data sets (i.e., PID files) on operational status. Under normal operation, this does not present a problem. However, in addition to log files, the /var partition contains file that are critical to system operation. The corruption of data or the overwriting of files due to a renegade process or denial of service attack that has generated a log file that has filled the file system can create a major problem. You will recall that we created a separate partition for /var (256M Bytes). This was done as a defensive measure to protect the "/" (root) file system from being overwritten and crashing the system. With this problem in mind, the logging files for IOS devices are stored in a separate file system apart from both "/" and /var. An attack or major error event could result in a log size that could exceed the storage of the file system. Granted, these are extreme cases, but sadly I have had customers implement logging and have these very things happen to them. The moral here is to manage your log files either with logrotate or with custom scripts to ensure the proper operation of the logging server. Decide how much data you need to keep on hand and archive the rest to tape, CD or the bit bucket.

Now that we have covered log file safety, let's look a the configuration additions we need to make to the syslog.conf file to make our logging server work:

```
# Logging for NE America Routers LOCAL_1
local1.notice      /log/cisco-core-msg-ne.log
local1.notice      monitor
local1.=debug      /log/cisco-debug-ne.log
local1.=debug      debug
local1.=info       /log/cisco-info-msg-ne.log
```

To start, all messages with the severity of notice and higher are sent to a text file and to the terminal of the user account "monitor." The user account "monitor" is a common account shared by the network administrators. They SSH to the logging server and log in in as "monitor." When a system message with a severity of notice or higher comes in, it is printed out on the terminal display. System messages with the severity of debug only are directed to a text file and the user account "debug." This, like the "monitor" account, is a common account used by all of the network administrators. And, finally, system messages with the severity of info only are directed to a text file.

In terms of reporting structure, all of the routers can report to the same syslog facility, or you can organize the routers in some kind of hierarchy and have them report to different syslog facilities. If you have a small environment, you may want each router to report to its own syslog facility (local0 thru local7 are available). Another option is to have all of the devices in a physical location or geographical area report to a specific syslog facility. In the end, it largely depends on how much data you are collecting and how much, if any, post processing you expect to do. The rule with reporting is the more specific the data, the easier it is to report on.

Using Cisco IOS Logging - Part 2

Michael J. Martin

Three Last Tips

To view the logging messages, the best bet is the Unix command tail with the -f flag ("f" for follow), which opens the file and prints the last 10 lines of the file and any other messages that come in after. This is a great way to look at log data when troubleshooting.

Whenever you make a change to the syslog.conf file, you need to restart the syslog service. To restart the service use the following command: kill -HUP `cat /var/run/syslogd.pid`.

Once the syslog.conf file has been updated and the service has been restarted, you will want to test it. This can be easily done using the logger command, which was discussed in part one. The basic command format is <logger -p {facility.severity} "message text">. Here is the terminal output from a test sent to the monitor user:

```
[root@parsafal /monitor]# logger -p local1.notice "this is a test"
[root@parsafal /monitor]# Apr 16 22:22:18 parsafal monitor: this is a test

[root@parsafal / monitor]#
```

Practical Use For Logging

Well, that's it for the IOS and server configuration. At this point, the server should be up and running and the routers pumping out logging messages. So, in place of a typical conclusion, we'll show you a simple network vulnerability scan reporting script. It uses IOS ACL logging to collect data on connection external (non-trusted) attempts to connect to services that have known network security vulnerabilities: SNMP, NetBIOS, PORTMAPPER, ECHO, distributed denial of service attacks, and syslog. The script works by reading the log reports from an inbound ACL applied to the inbound interface of a sites Internet router. The ACL can either permit or deny the interesting traffic. In the example ACL, the connection attempts are dropped. Here is the ACL:

```
ip access-list extended internet-inbound
remark RFC1918 and Bogus Address Prefixes
permit ip any host 157.191.2.254 log
deny ip 127.0.0.0 0.255.255.255 any log
deny ip 10.0.0.0 0.255.255.255 any log
deny ip 192.168.0.0 0.0.255.255 any log
deny ip 172.16.0.0 0.15.255.255 any log
deny ip 0.0.0.0 0.255.255.255 255.0.0.0 0.255.255.255
deny ip 169.254.0.0 0.0.255.255 255.255.0.0 0.0.255.255
deny ip 224.0.0.0 31.255.255.255 224.0.0.0 31.255.255.255
remark Anti-Spoofing Entry's add your network prefixes here
deny ip 147,225.0.0 0.0.255.255 any log
deny ip 63.181.64.0 0.0.0.128 any log
remark Filter Elements
deny tcp any any eq 27665
deny tcp any any eq 65000
deny tcp any any eq sunrpc
deny udp any any eq netbios-ns log
deny udp any any eq netbios-dgm log
deny udp any any eq netbios-ss log
Revised May 8, 2002
```

Using Cisco IOS Logging - Part 2

Michael J. Martin

```
deny udp any any eq echo log
deny tcp any any eq echo log
deny udp any any eq snmp log
deny udp any any eq snmptrap log
deny udp any any eq syslog log
deny udp any any eq 666 log
deny tcp any any eq 666 log
permit ip any any
```

The ACL filters all RFC1918 addresses, reserved multicast addresses and traffic with a source address from the network prefixes within the routing domain (anti-spoofing). The entries in the "Filter Elements" section filter out the service ports for network vulnerabilities. Connection attempts to these service ports are dropped and a log message reporting the violation and its destination and origin address are sent to the syslog server.

To use the script, copy the text between the <start copy> and <end copy> tags, paste it into a text file, save the file as SNMP-SCAN.sh and, as root, run the following commands:

```
[trinity:~] root# chown root SNMP-SCAN.sh
[trinity:~] root# chmod o+x SNMP-SCAN.sh
```

Then run the script (assuming that syslog is configured correctly) using:

```
[trinity:~] root# ./SNMP-SCAN.sh
```

If the script runs successfully, you will get the "Report Complete" output and you should receive an e-mail with the report output. Be sure to read the script and follow the instructions before running. Good Luck.

<start copy>

```
#
# This script generates access violation reports from syslog
# events generated by Cisco IOS. Violations are recorded as logged
# events against an inbound or outbound Cisco interface access control list
# It is run as a daily cron job
# To run the script TRAP logging needs to be configured on the router and a #
# filter ACL # that looks for target traffic must be installed on the
# Internet access router interface. The script was written to run on RH Linux (6.2
# and higher).
#
# Define script variables before running
#
#
# Who to send reports to add e-mail addresses here:
USERS=user@host.com, user2@host.com
# Uncomment what protocol port number you want to look for
# NetBIOS
#PROT="(13[7.8.9])"
#PROTN="NetBIOS Name, Datagram, Session Services"
# SNMP
```

Revised May 8, 2002

Page 11 of 12

Using Cisco IOS Logging - Part 2

Michael J. Martin

```
PROT="(161)"
PROTN="SNMP"
#
#PROT="(27665)"
#PROTN="trinoo DDOS"
#
#PROT="(65000)"
#PROTN="stacheldraht DDOS"
#
#PROT="(79)"
#PROTN="FINGER"
#
#PROT="(111)"
#PROTN="Portmapper"
#
#PROT="(7)"
#PROTN="echo"
#
#PROT="(666)"
#PROTN="Doom Server"
#
# Set Routers Hostname
HOSTN="MCK-BWG-AS6395"
# Set Routers syslog output file
LOGFILE=/var/log/gw2.log
# Working File
REPFIL=/var/tmp/$PROTN-$HOSTN
#
# File Management
kill -9 `cat /var/run/syslogd.pid`
#
cp -rf $LOGFILE $REPFIL
#
# Delete Logfile (uncomment the "rm" and "touch" commands)
# or Leave Logfile (leave file if you are
# running multiple reports)
#rm -rf $LOGFILE
#touch $LOGFILE
#
#
/sbin/syslogd -r
# Temp Files
TEMP=/var/tmp/access-vio.tmp
#
# Run Report
more $REPFIL | grep $PROT | awk 'BEGIN {print "Prot SrcHost-Port Dst
Host-Port PktSent "}
{print $14,"t"$15,$16,$17,$18,$19} END {print "
nend-of-report"}' > $TEMP
#
# Send Report
mail -s "`more $REPFIL | grep -c $PROT` $PROTN Violations Reported By $HOSTN"
$
```

Using Cisco IOS Logging - Part 2

Michael J. Martin

```
USERS < $TEMP
#
# Clean Up
rm -rf $TEMP
#
# Report Management
#
# You need to choose how to manage the report files "Save" or "Delete"
# Uncomment one of the options. If you are running multiple report instances
# you will need to delete the report file and have the last instance perform
# the archive.
#
# Delete Report File
rm -rf $REPPFILE
#
# Or Archive the Logfiles
#mkdir /var/log/ciscologs > /dev/null 2>&1
#ARCHDIR=/var/log/ciscologs
#mv $REPPFILE $ARCHDIR
#
echo "Report Complete"
#
<end copy>
```