Remote Syslogging - A Primer

By Armando Ortiz
10/28/2002 9:54

The syslog daemon is a very versatile tool that should never be overlooked under any circumstance The facility itself provides a wealth of information regarding the local system that it monitors.

However, what happens when the system it's monitoring gets compromised?

When a system becomes compromised, and the intruder obtains elevated root privileges, he now has the ability, as well as the will, to trash any and all eviden ce leading up to the intrusion, on top of erasing anything else thereafter, including other key system files.

That's where remote system logging comes in, and it's real super-easy to set up.

This example uses Slackware Linux 8.1 as the base to begin, but can be applied easily to other distros usi the syslog daemon that was ported to Linux by Mar tin Schulze (joey@linux.de). As of this writing, I do know if Mr. Schulze's address is still valid since I've not written to anyone that has provided key fea ture to Linux, so if it's not, please contact me so I can revise my writing.

Step One: Pick and Choose Your Parent

Since this is going to require more than one server, it's logical to conclude that you will have to pick and choose a server with which to make the centralize d parental tattle-tale of your network. Pick the server w the least amount of traffic, particularly, any server that doesn't utilize any heavy UDP traffic, si nce this i what syslogd will be using. Database servers should NEVER be used since they rely on space as much as the syslog facilities do, as will become appare nt later on in this document. Double check your services fi in /etc to see if port 514 is listed. If it's not, put the following entry into it:

```
syslog 514/udp
```

Since all the distributions I've played with have it listed, chances are, you won't need to modify it.

Step Two: Let's Listen to the Children!

There is probably already a script somewhere during your startup processing that starts the syslogd. By default, the syslogd no longer listens to remote conne ctions on port 514. In fact, it's been disabled until i explicitely turned on.

Find the script that starts the syslogd and add -r to the end of the line that executes it:

```
/sbin/syslogd -r (plus anything else that comes up with it)
```

The -r switch tells the syslogd to start listening to the world on UDP port 514.

Step Three: ONLY MY KIDS, PLEASE!

The problem with Step Two is that implementing it just like that opens up a wealth of problems, not the least of which is the potential for someone to step up to the plate and DoS the hell out of you on that port with bogus packets.

The quickest way to prevent this is simple packet filtering. Since iptables is now a defacto in almost ever distribution with a 2.4.x kernel, I'll use it as my base, however, ipchains should be just as simple to set u

```
iptables -A INPUT -p udp -s 192.168.2.0/24 -d 0/0 --destination-port 514 -j ACC
iptables -A INPUT -p udp -s 0/0 -d 0/0 --destination-port 514 -j REJECT
```

or

```
ipchains -A INPUT -p UDP -s 192.168.2.0/24 -d 0/0 514 -j ACCEPT
ipchains -A INPUT -p UDP -s 0/0 -d 0/0 514 -j REJECT
```

Summary: let's listen to all the machines on our own network, but reject everything else on port UDP 51 This keeps our head above water in listening to the computers that belong to us. If you put this into a star script, remember that the REJECTion line MUST come LAST in the process or the packets won't get thr gh. Unfortunately, ordering is everything, but then again, order can be a good thing.

Step Four: Show Us Our Parent!

Now that the parent is ready to listen, it's time to prep our children. For every child computer you have t is going to talk to our parent, you may need to tell it who the parent is that it will be talking to. Afterall, v wouldn't want our children talking to strangers, now would we?

For this, we have /etc/hosts. Modify it with the IP of the listening parent server:

```
192.168.2.23        parent.somedomain.com
```

The child can then relate its listening parent to @parent instead of the whole IP address or domain name, which will become apparent in the next step.

Step Five: Talk To Me...Tell Me What's Going On!

The final step in our remote syslogging setup is to actually tell the children what type of information is going to be sent back to our parent. The following is my example of a default /etc/syslog.conf for all of t child systems that will be sending data to our parent logging facility:

```
# /etc/syslog.conf

# For info about the format of this file, see "man syslog.conf"
# and /usr/doc/sysklogd/README.linux.
# Uncomment this to see kernel messages on the console.

# kern.*                                                /dev/console

# We don't want to uncomment it - makes a messy screen.

# Log anything 'info' or higher, but lower than 'warn'.
# Exclude authpriv, cron, mail, and news.  These are logged elsewhere.

*.info;*.!warn;\
 authpriv.none;cron.none;mail.none;news.none           @parent

# Log anything 'warn' or higher.
# Exclude authpriv, cron, mail, and news.  These are logged elsewhere.
*.warn;\
   authpriv.none;cron.none;mail.none;news.none         @parent

# Debugging information is logged here.
 *.=debug                                              @parent

# Private authentication message logging:
 authpriv.*                                            @parent

# Cron related logs:
 cron.*                                                @parent

# Mail related logs:
 mail.*                                                @parent

# Emergency level messages go to all users:
 *.emerg                                               @parent

# This log is for news and uucp errors:
 uucp,news.crit                                        @parent

# You can include any others that you need as long as they follow the
# format stated by the syslog.conf man page.
```

After setting up something like this, you can then restart the children's syslogging facility and they shoul start talking to the parent on UDP port 514.

Check to make sure this is the case. I did find that not specifying who the parent was in step four above w important enough to disallow the child sending a nything out to the world. Could be because my DNS wasn't set up properly to reverse resolve.

Explaining what each line does is beyond the scope of this document. What this document basically is, is primer to get the remote syslog working on your net work. After you have it set up, go read the syslog.co man page for more information on what the parameters are.

The pros of setting up remote logging are quite simple, and make strategically logistical sense to administrators not wanting to waste time connecting to diff erent machines. Let's take a look:

- You can consolidate all logging to one machine, thereby making it simple to access a centralized source of activity on your network that your child ker nels will catch.

- Logging is easily greppable by IP. All log transfer that occurs is flagged by the child's IP address.

- If an intruder breaches the security of a child system, the attempts are remotely logged and the intru will then have to break into the parent to tr ash the log files since none are kept locally. By the time does that, the jig is pretty much up.

- Breached computers can have listening daemons on them that can email back to the intruder the roo password or the passwords to other accounts on the system if you try to log into it the compromised computer to see just what in the hell happened to it, or if you're just visiting. By centralizing each child's system logs to the parent, you can view untampered events provided that you have adequate secured the parent enough that it too hasn't been compromised.

The cons are seriously outweighed by the pros in terms of merit, however, they are worth mentioning:

- By consolidating the logging to a single system, you must ensure that you will have adequate space contain the logs. Logfiles can get rather large, especially from mail servers where activity is in a constantly increasing state. On a casually busy network, I would recommend a harddrive of not less than 10GB in size, only a 10th of which is taken up by the OS and installed applications, for up to 1 screaming children. This number may vary. This should also give you adequate rotation space for a logging in the network.

- Large networks may want to break up the children into groups of 10 or so and have a single listenir parent for each group. This will help curtail the large logfiles, but will also mean setting up more th one listening parent.

- Centralizing the logging to a single parent doesn't mean you can get lazy on security for any of the other machines in your network. You should always make sure each one is secured enough, but especially ensure that the listening parent is secured from outside intruders that can break into it and trash the plac e. Once the parent is trashed, the credibility of the parent is lost.

- Using a server that's hardly doing anything can be a waste of resources for some networks, howeve consider the risks of not having such a facility in the interim.

- You increase UDP traffic for each child you bring up, which isn't much, but can be noticeable on larger networks. You may want to consider installing s eperate ethernet cards on a privatized netwo block and send all syslog traffic out through that instead of letting it become part of the world.

- Rotating out the logs means that you only have a limited time to view events of the past month for a the systems, however, you can configure logrotat e to either increase or decrease the rotation time, depending on your preferences, and you only have to do this for the parent. Read the man pages on logrotate f or more information.

- Distributed denial of service attacks can easily take out the parent if there isn't enough space or adequate bandwidth to handle all requests. Make sur e the parent is protected and that each child is protected to prevent against ddos attacks as much as possible. In other words, limit private types of

data to th e network, such as SSH or NFS.