« Previous 1 2 3

Customizing PortSentry

Customization

I came across a few limitations with the way a third party script would run when triggered, but with a little bit of environment bashing (no pun intended) you can work your way around any issues.

For example, I had a LAN with machines on it that I mostly trusted, but occasionally they would get infected with malware or be compromised and become a risk to my other servers. PortSentry can offer a simple “ignore IP” function where CIDR-formatted IP address blocks, or just single IP addresses, can be added to */etc/portsentry/portsentry.ignore.static* . This method is fine for most scenarios; however, first it was important that I knew about any LAN machines that were port scanning my servers in an attempt to get into them or discover running services for a future compromise. Second, I wanted to disconnect all access to my locally offending servers briefly (some were mail servers that LAN machines used to send mail through via SMTP) so that when PortSentry sent me an email about the port scan, the users on the potentially compromised machines also were informed about the problem but then allowed to resume service again afterward without my manual intervention. With a little bit of help from Cron

and a shell script, this was easily achieved (after dealing with a few fiddly environment issues).

```
KILL_RUN_CMD="(/usr/bin/whois $TARGET$) | /usr/bin/mailx -
s 'Ganymede banned $TARGET$ for scanning port $PORT$'alarm
s@mail.domain.com; /etc/portsentry/temp_ban_length $TARGET
$ $PORT$;"
```

This customized *KILL_RUN_CMD* is split into two main elements. First is a WHOIS lookup on the offender's IP address and then puts that inside an email to the admins with a useful subject line, detailing what's happened. Second, directly from PortSentry, it spawns the first custom script, which checks to see whether the offending IP address is one of the recognized local IPs. If it is, then it just bans it from speaking for 10 minutes and then re-opens access to the services again. Once the temporary ban is removed, *logger* writes to syslog, so it's easy for an admin to see that the IP address ban has been lifted.

Bear in mind that this *KILL_RUN_CMD* and the script in Listing 1 might be improved with things like lock files if a certain number of email messages are generated by PortSentry for one host – or one set of ports, for example), but it met my needs nicely because of the low frequency of alerts being generated.

Listing 1: Custom Script Spawned by PortSentry: *`/etc/portsentry/temp_ban_length`*

```
#!/bin/bash

# Local IPs are only banned briefly
if [ `echo $1 | grep '123.45.67.' ` ] || [ `echo $1 | grep
'89.91.23.' ` ] || [ `echo $1 | grep '45.67.78.91' ` ]
then
/bin/sleep 1200
```

LPIC-1 Systems Administrator certification.

Topics

```
/sbin/iptables -D INPUT -s $1 -j DROP
/usr/bin/logger "portsentry: Removed the ban for IP: $1"
exit 0
fi

# Otherwise ban permanently
exit 0
```

The second script is fired by Cron every 10 minutes (obviously, that's adjustable). It's needed because PortSentry keeps a log of its banned connections in the usefully named */etc/portsentry/portsentry.blocked.atcp* and */etc/portsentry/portsentry.blocked.audp* files. It seems to reference these rather than look up firewall rules (for greater efficiency, I suppose), so it can say that an IP address is "already banned" if a returning IP address comes back at some point in the future for a repeat visit. For the purposes here, that means a LAN IP can attack once, be banned for 10 minutes, then come back and attack with impunity because PortSentry has it listed as already dealt with. The following Cron entry spawns the script to fix that issue:

```
# Flush PortSentry of LAN IPs
*/10 * * * * root /etc/portsentry/perm_ban_length
```

The script in Listing 2 then runs through each of the *blocked* files holding PortSentry's history and deletes any entries with the *sed -i* command. The script is laid out so that it's easy to see the two short sections – one for TCP and one for UDP history. Finally a quick mention is made of what has been removed in syslog so LAN IP addresses are banned again if they attempt to attack in the future.

Listing 2: Custom Script Spawned by Cron: */etc/portsentry/perm_ban_length*

```
#!/bin/bash
# Cron script for flushing recognized LAN IPs from PortSen
```

```
try history
# because they're not permanently banned and PortSentry ig
nores connections from them otherwise

bantcp="/etc/portsentry/portsentry.blocked.atcp";
banudp="/etc/portsentry/portsentry.blocked.audp";

# TCP
tcp=`tail -n1 $bantcp | awk '{ print $6 }' | cut -d / -f 2
`;
if [ `echo $tcp | grep '123.45.67.' ` ] || [ `echo $tcp |
grep '89.91.23.' ` ] || [ `echo $tcp | grep '45.67.78.91'
` ]
then
/bin/sed -i '/'$tcp'/ d' /etc/portsentry/portsentry.blocke
d.atcp
/usr/bin/logger "portsentry: Removed TCP events ($tcp) fro
m the PortSentry history file"
exit 0
fi

# UDP
udp=`tail -n1 $banudp | awk '{ print $6 }' | cut -d / -f 2
`;
if [ `echo $udp | grep '123.45.67.' ` ] || [ `echo $udp |
grep '89.91.23.' ` ] || [ `echo $udp | grep '45.67.78.91'
` ]
then
/bin/sed -i '/'$udp'/ d' /etc/portsentry/portsentry.blocke
d.audp
/usr/bin/logger "portsentry: Removed UDP events ($udp) fro
m the PortSentry history file"
exit 0
```

```
fi

exit 0
```

Conclusion

The use of PortSentry out of the box is sufficient for lots of server scenarios, but hopefully this quick look at how to customize it might give you food for thought to help you solve a problem of your own. It's important that you continue to monitor who is connecting to your servers, and it's equally important that you have the ability to respond in some way – sometimes manually and sometimes automatically. Since implementing the script shared in this article, the LAN machines I temporarily ban rarely cause any problems; whereas, they frequently did in the past. Hopefully you'll reap the same rewards.

« Previous 1 2 3

Related content

Port Knocking

To ensure that the data on your computers remains accessible only by you and those with whom you want to share, we look at the advantages of combining TCP Wrappers and port knocking.

more »

Secure Your Server with TCP Wrappers

TCP Wrappers are versatile, sophisticated, and surprisingly easy to use, and they can secure your servers from attack with run-time ACL reconfiguration.

more »

Pen Testing with netcat

Once you have successfully exploited a target machine, you might be faced with a dilemma common among penetration testers: Do I have shell access or terminal access? These are not the same, and careful knowledge must be used when interacting with a shell compared with a terminal.

more »

Spanning Tree Protocol

Ethernet is so popular because it simply works and is inexpensive. However, the administration side looks a bit more complicated: For the network to run smoothly, the admin might need to make important decisions about the Spanning Tree protocol.

more »

TCP Stealth hides open ports

Port scans for finding vulnerable services are nothing new, and port knocking as a defense has been around for a while, too. TCP Stealth tries to do something similar, but it takes a more sophisticated approach. We take a closer look.

more »