The Free International Online Linux Monthly ISSN: 1934-371X Main site: http://linuxgazette.net

Rule-based DoS attacks prevention shell script

By Yoshiyasu Takefuji

Introduction

In this article, I describe a simple rule-based DoS attack-prevention shell script. However, the proposed shell script is not a perfect tool for preventing DoS attacks, but a powerful tool for alleviating DoS attacks overheads of the Linux servers significantly.

[Dealing with dynamic threats and automatically taking evasive action is very difficult to do and requires some thought. The article focuses on text patterns and log files. There are other methods that lead to similar results and are mentioned after the author's conclusion. -- René]

In order to understand DoS or DDoS attacks, it is useful to see the log files in /var/log. In this article, an ssh DoS attack-prevention shell script is mainly mentioned. We have been observing the behavior of ssh DoS attacks through three Linux servers in the last six months. We have been manually manipulating iptables commands for disabling the access from specific IPs, after detecting DoS attacks. The proposed shell script is to automate the whole manipulated commands for DoS attacks prevention. If the proposed simple shell script detects DoS attacks that match predefine rules in the shell script, then the DoS attack IPs are added to the detected-IPs file and have their access to the server disabled. Since detecting DoS attacks is rule-based, it is expandable. Kernel-mode implementation of the proposed idea is expected, for immediate DoS attacks prevention, instead of using crontab in this article.

How to detect DoS attacks from /var/log/secure file

In order to see /var/log/secure file, you have to be a root. In this article, "grep", "awk", and "sed" commands are often used for building rules in the proposed shell script. The shell script is composed of a part of DoS attack detection rules, a part of reducing redundant IPs, and a part of disabling detected IPs. The following is an example of the typical ssh attack using the dictionary, where every second user name is changed from root, delta, admin,,,, after the system did not receive identification string from 64.34.200.202.

```
Feb 18 09:14:08 neuro sshd[8978]: Did not receive identification string from 64.34.200.202
Feb 18 09:18:22 neuro sshd[9012]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost
Feb 18 09:18:24 neuro sshd[9012]: Failed password for root from 64.34.200.202 port 43353 ssh2
Feb 18 00:18:24 neuro sshd[9013]: Received disconnect from 64.34.200.202: 11: Bye Bye
Feb 18 09:18:25 neuro sshd[9015]: Invalid user delta from 64.34.200.202
Feb 18 00:18:25 neuro sshd[9016]: input_userauth_request: invalid user delta
Feb 18 09:18:25 neuro sshd[9015]: pam_unix(sshd:auth): check pass; user unknown
Feb 18 09:18:25 neuro sshd[9015]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost
Feb 18 09:18:27 neuro sshd[9015]: Failed password for invalid user delta from 64.34.200.202 port 43875 ssh2
Feb 18 00:18:28 neuro sshd[9016]: Received disconnect from 64.34.200.202: 11: Bye Bye
Feb 18 09:18:29 neuro sshd[9017]: Invalid user admin from 64.34.200.202
Feb 18 00:18:29 neuro sshd[9018]: input_userauth_request: invalid user admin
Feb 18 09:18:29 neuro sshd[9017]: pam_unix(sshd:auth): check pass; user unknown
Feb 18 09:18:29 neuro sshd[9017]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost
Feb 18 09:18:31 neuro sshd[9017]: Failed password for invalid user admin from 64.34.200.202 port 44300 ssh2
```

The following command sends the disabled IPs information from /etc/sysconfig/iptables into tmp file.

```
grep DROP /etc/sysconfig/iptables|awk '{print $5}' >tmp
```

If the system did not receive identification string from the specific IPs, the machine access should be disabled. The following command adds the detected IPs to a temporary file. Detected IPs using rules will be added in that file.

```
grep Did /var/log/secure|awk '{print $12}' >>tmp
```

A new rule can be added to the tmp file using the simple command. The dictionary attacks can be easily detected by "Invalid user" from /var/log/secure. If you misspell in ssh login name, you may not be able to login any more from your machine. In order to re-enable login from the machine again, you must delete all the lines including your machine IP from /var/log/secure file and flush the iptables by /sbin/iptables -F.

```
grep "Invalid user" /var/log/secure|awk '{print $10}' >>tmp
```

Maximum login DoS attacks can be detected by the following command.

```
grep "Maximum login" /var/log/secure|awk '{print $7}'|sed 's/.*\[\(.*\)\])/\1/g' >>tmp
```

How to reduce redundant detected IPs from the temporary file

The following commands reduce redundant detected IPs from the temporary file, and store the unique detected IPs in file ttt. The variable "size" indicates the number of lines in the tmp file.

```
size=`wc tmp|awk '{print $1}'`
i=0
while test $i -lt $size
do
      us=`sed -n 1p tmp`
      sed /$us/d tmp >tmps
      echo $us >>ttt
      cp -f tmps tmp
      size=`wc tmp|awk '{print $1}'`
done
```

How to activate /sbin/iptables

DoS attack IPs are stored in file ttt. The following simple loop activates /sbin/iptables.

```
size=`wc ttt|awk '{print $1}'`
size=`expr $size + 1`
/sbin/iptables -F
i=1
while test $i -lt $size
do
        ip=`sed -n "$i"p ttt`
        i=`expr $i + 1`
/sbin/iptables -A INPUT -s $ip -j DROP
done
```

How to install the proposed shell script

You have to be root. "crontab -e" command sets the crontab, where the proposed shell script test is stored in /var/log in our system. The following setting means, that every five minutes, the shell script test is activated, everyday.

```
0-59/5 * * * * /var/log/test
```

The entire shell script /var/log/test is as follows.

```
#!/bin/bash
rm -f ttt
touch tmp
# disabled IPs can be obtained from /etc/sysconfig/iptables
grep DROP /etc/sysconfig/iptables|awk '{print $5}' >tmp
# ------------------------ DoS attacks rule -------------------------
#identity mismatch in secure
grep Did /var/log/secure|awk '{print $12}' >>tmp
```

```
#Invalid user
grep "Invalid user" /var/log/secure|awk '{print $10}' >>tmp
# Maximum login
grep "Maximum login" /var/log/secure|awk '{print $7}'|sed 's/.*\[\(.*\)\])/\1/g' >>tmp
#
# ------------------ reduce redundant IPs from tmp file -------------
size=`/usr/bin/wc tmp|awk '{print $1}'`
i=0
while test $i -lt $size
do
      us=`sed -n 1p tmp`
      sed /$us/d tmp >tmps
      echo $us >>ttt
      cp -f tmps tmp
      size=`/usr/bin/wc tmp|awk '{print $1}'`
done
rm -f tmp tmps temp0 temp
#
# ------------------ activate detected IPs --------------------------
size=`wc ttt|awk '{print $1}'`
size=`expr $size + 1`
/sbin/iptables -F
i=1
while test $i -lt $size
do
        ip=`sed -n "$i"p ttt`
        i=`expr $i + 1`
/sbin/iptables -A INPUT -s $ip -j DROP
done
# -----------------end of shell script test -------------------------
```

Examine shell script

Before running the cron shell script, you must be root and should examine the shell script by:

```
sh /var/log/test
```

In order to see the current iptables, type the following command.

```
/sbin/iptables -nL
```

Conclusion

Since the proposed shell script is portable, it can be placed in every server or router. In order to share the blacklist of IPs, the shell script must be placed in every router, and router-to-router communication is needed to assemble the blacklist of IPs for the larger framework of network security.

[The system described in this article relies on predefined patterns that have to be extracted from the log files; this, of course, requires that the syslog server on the machine not drop any log messages. Both assumptions can lead to problems when log entries are dropped or the text pattern of the failed login attempts isn't detected properly. The author mentioned a way to do this automatically from the kernel; the Linux Netfilter provides a module than can be used to automatically deny repeated login attempts. There is an article that describes this method. Whatever you decide to do, keep in mind that a resilient security measure should not depend on less than fully-reliable grounds. Parsing log files makes a fine part of your security measures; just don't make it a cornerstone of your security considerations. -- René]

Talkback: Discuss this article with The Answer Gang

Yoshiyasu Takefuji was heavily involved in developing a unix based color workstation in 1983 at University of South Florida. Recently he has been monitoring three Linux servers to see the behavior of DOS attacks. He is a chair of SecurityExpo in Japan since 2004 and also a chair of OECD TrustE security product evaluation committee chair in Japan, and advisor of Japan Network Security Association and CMU in Japan.

<-- prev | next -->