Portknocking in bash

Written by Michael Shinn
Thursday, 17 April 2008 00:00

So we've been playing around with PortKnocking for some time, trying to ion that didn't cre

Yep, you heard right, BASH scripts. What you got here is a 100% shell based portknocking server and client, with neither directly exposed to the traffic coming into the box its protecting so no need to worry about processing packets. This is a really handy feature, not being a service and not parsing packets directly, because that means we don't have to directly worry about our client and server handling them. Without further delay, here is the server pieces:

```
#!/bin/bash
# Change IP address to your own
# Add this entry into /etc/syslog.conf
# local7.*                              /var/log/boot.log

# touch /var/log/portknocking.log

#create portknock "hook" at top of firewall rules?
# or have user create "hook" in their rulesets?
# or give them a choice?

HOOK=NO
LOG_REJECT=YES

if [ $HOOK = "YES" ]; then
     $IPTABLES -N PORTKNOCK
     #position 1 in the INPUT rules
     $IPTABLES -A INPUT -s 0.0.0.0/0 -d $our_ip -m state --state ESTABLISHED -p tcp --dport 22 -j ACCEPT 1
     #need to put ESABLISHED rule first
     #log the hits?
     if [ $LOG_REJECT = "YES"; then
          $IPTABLES -A INPUT -s 0.0.0.0/0 -p tcp --dport 22 -j LOG --log-level notice --log-prefix "SSH REJECT "
     fi
```

```
      #and drop the baddies
      $IPTABLES -A INPUT -p tcp --dport 22 -j DROP 2
fi



# base port of knocking (range from port0 to port0+4095 must be free)
port0=10000
# password
pass="some_password"
# unique string of knocking logs
id_string="PORT_KNOCKING"
# knocking log file
log_file="/var/log/portknocking.log"
# ip of our interface
our_ip="xxx.xxx.xx.xxx"


# -----------------------------

# allowing only one connect from ip in this time period (seconds)
# and also in other words max period of client and server clock desynchronisation
time_period=100
# max time (in seconds) between two knocks in knocking sequence
delta=2
# time period (in seconds) when door is open
sleep_time=10


# don't touch
time_flag=0
used_time=0
cnt=0

IPTABLES=/usr/sbin/iptables
POLICY=`$IPTABLES -L INPUT | grep policy | awk '{print $4}' | tr -d )`

# iptables initialisation for knocking listening
port1=$(($port0+4095))

$IPTABLES -A INPUT -s 0.0.0.0/0 -d $our_ip -p tcp --syn --dport $port0:$port1 -j LOG
--log-level notice --log-prefix "$id_string "

if [ $POLICY = "ACCEPT" ]; then
      $IPTABLES -A INPUT -p tcp --dport $port0:$port1 -j DROP
fi

# allow only established ssh connection
$IPTABLES -A INPUT -s 0.0.0.0/0 -d $our_ip -m state --state ESTABLISHED -p tcp --dport 22 -j
```

ACCEPT

```bash
tail -n1 --follow=name --retry $log_file |
{
    # read no using line
    read

    # main cycle of reading log lines
    while [ 1 == 1 ]
    do
        read str

        # get time in seconds since `00:00:00 1970-01-01 UTC'
        time=`date +%s`

        # check is it our log line
        ok=`echo $str | grep $id_string`
        if [ -z "$ok" ]; then
            # to next iteration of main cycle
            continue
        fi

        # extract source ip and destination port from log line
        for fld in $str
        do
            case "${fld:0:4}" in
                "SRC=")
                        sip=${fld:4}
                        ;;
                "DPT=")
                        dport=${fld:4}
            esac
        done


        # calculate secure combination of ports and time up to which this combinaton is valid
        if [ $time -ge $used_time ]; then
            time_stamp=$(($time/$time_period))
            sum=`echo $pass$time_stamp | md5sum`
            i=0
            sec_ports=""
            while [ $i -lt 16 ]
            do
                j=${sum:$i*2:2}
                port=$(($port0+0x$j*16+$i))
                sec_ports="$sec_ports $port"
```

```
        i=$((i+1))
    done

    remainder=$(($time%$time_period))
    used_time=$(($time-$remainder+$time_period))
    used_ips=""

fi

# time period from last successsful processing
dtime_flag=$(($time-$time_flag))

if [ $dtime_flag -gt $delta -o $cnt -eq 0 ]; then

    # check if our ip already processed in current time period
    ok=`echo $used_ips | grep $sip`
    if [ "$ok" ]; then
        # to next iteration of main cycle
        continue
    fi

    # begin processing for this ip
    cur_ip=$sip
    ports=""
    cnt=0

else
    # not allowed simultaneously process more then one ip
    if [ $sip != $cur_ip ]; then
        # to next iteration of main cycle
        continue
    fi
fi

# time label of successful processing
time_flag=$time
# list of processing ports
ports="$dport $ports"
# port counter
cnt=$((cnt+1))

# it's time to check port sequence
if [ $cnt -eq 16 ]; then
    cnt=0

    # check if incoming knocking correct
```

```
        for port in $sec_ports
        do
           ok=`echo $ports | grep $port`
           if [ -z "$ok" ]; then
               continue
           fi
        done

        # open our door for some time
        if [ "$ok" ]; then
           used_ips="$used_ips $cur_ip"

           # turn on incoming ssh connects

           if [ $POLICY = "ACCEPT" ]; then
               $IPTABLES -D INPUT -p tcp --dport 22 -j DROP
           fi

           $IPTABLES -A INPUT -s $cur_ip -d $our_ip -p tcp --syn --dport 22 -j ACCEPT

           if [ $POLICY = "ACCEPT" ]; then
               $IPTABLES -A INPUT -p tcp --dport 22 -j DROP
           fi
           sleep $sleep_time
           # turn off incoming ssh connects
           $IPTABLES -D INPUT -s $cur_ip -d $our_ip -p tcp --syn --dport 22 -j ACCEPT
        fi
      fi

   done
}
```

And the client:

```
#!/bin/bash

# program to knock (telnet or netcat)
prog="telnet"
# must be equal to time period on knocking server
time_period=100
# period between knocking sequence and ssh connect
sleep_period=2
# ssh user
username=$1
# destination ip
```

Portknocking in bash

Written by Michael Shinn
Thursday, 17 April 2008 00:00

```
ip=$2

if [ $# -ne 2 ]; then
    echo "usage: ./clientname username ip_address"
    exit
fi

read -p "enter base port of knocking: " -s port0
echo
read -p "enter knocking password: " -s pass
echo

# calculate secure sequence of ports
time=`date +%s`
time_stamp=$(($time/$time_period))
sum=`echo $pass$time_stamp | md5sum`

i=0
ports=""
while [ $i -lt 16 ]
do
    j=${sum:$i*2:2}
    port=$(($port0+0x$j*16+$i))
    ports="$ports $port"
    i=$((i+1))
done

# start knocking
(
for port in $ports
do
    $prog $ip $port &
done
pkill $prog
) >/dev/null 2>&1
echo "knocking done"
sleep $sleep_period

echo "trying to ssh ..."
ssh -l $username $ip
```

As the client is written as a script, we can use it on almost any OS, provided that we have sha1sum on the client and it can parse bash scripts.