ISSN 1220-2169

BULETINUL I.P.I, Tomul L(LIV), fasc. 1-4, AUTOMATICĂ şi CALCULATOARE, 2004

BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI

Editat de
UNIVERSITATEA TEHNICĂ "GH. ASACHI", IAŞI

Tomul L(LIV)
Fasc. 1 - 4

Secţia AUTOMATICĂ şi CALCULATOARE

2004

BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Tomul L (LIV), Fasc. 1-4, 2004
AUTOMATICĂ şi CALCULATOARE

FIREWALL GENERATOR (II)

BY
EUGEN PETAC and BOGDAN MUŞAT

Abstract

The proposed PHP Firewall Generator is a very useful tool designed for LINUX system administrator. The Web interface designed in PHP and JavaScript allows the easily composure of firewall rules, as well as their easily manipulation. With few mouse clicks any rule from the firewall can be written, edited or deleted. Linking of a computer to the Internet implies, generally, the use of the Linux operating system and of the TCP/IP protocol suite. These components have their own security problems. Access to the Internet also implies the use of a set of dozens of services, programs, with various security problems, either due to software errors or due to mis-incorporation of some appropriate security facilities. Iptables is used to configure, administer and inspect the packet filtering rules tables from the Linux kernel. Each of the defined tables can contain a number of predefined chains but it can also contain user-defined chains.

Keywords: Firewall Generator, LINUX system, Iptables.

1. Introduction

A firewall is a system or a group of systems that imposes an access control policy between two networks. Firewall administrators that protect a large number of hosts have therefore a large responsibility.

Iptables (http://www.iptables.org) is used to configure, administer and inspect the packet filtering rules tables from the Linux kernel. Various tables can be defined. Each table contains a number of predefined chains but it can also contain user-defined chains. Each chain is a list of rules that can match a set of packets. Each rule specifies what to do with a matching packet. This is called "a target", which can be a jump to a user-defined chain in the same table.

The novelty that the PHP *Firewall Generator* application brings is the possibility of dynamic manipulation of firewall rules through a web interface.

2. Description of the PHP Firewall Generator Application – Details

We present, step by step, the chains and the possible user defined chains. In each table the name of chain is written, followed by the number of packets conveyed, through the chain, as well as the traffic, [Mbytes], done in that chain. A blue table follows with three columns and multiples lines, on each line being a rule. In the first

column there is the current number of the rule, in the second column there is the rule itself and in the third column there is a check box and an icon for its editing. At the pressing of a rule's icon, the application will display a page for editing this rule.

Under the blue table there are three commands accessible through buttons: delete all rules from the chain, erase the chain (only if it is a user defined chain and the chain is empty), and rename the chain (only if it is a user defined chain). At the end of the page there is an "Erase" button that deletes all checked rules.

2.1. Adding Rules to the FILTER Table

In this page we can compose our own firewall rules. We have three commands that can be performed by pressing one of the buttons at the end of the page. These commands are: "Send", that sends the rule generated in the "Generated rule" box to the system, "Visualize", that generates a rule starting from all options in the page and "Erase form", which resets the form to the default values. The description of the options is given in what follows.

In the "Rule no." editing box, if no value is specified, the action will be to add the rule at the end of the chain, and if a number is specified (which must be at most equal to the number of rules in chain), the action will be to insert the rule in the *no.* position.

E x a m p l e.
iptables -A INPUT (add to the end of the INPUT chain)
iptables -I INPUT 5 (add the rule in the 5 position in the INPUT chain)

We can choose one of the predefined INPUT, FORWARD or OUTPUT chain or any chain defined by the user from a drop-down box.

The source address of the packets can be specified (*-s*) by filling in the first editing box from the source with an IP address or a network address, case in which we can also specify a network mask in the second editing box. Also, we can choose from few predefined IP addresses or few predefined network masks from the drop-down boxes at the right of the editing boxes.

We can specify the destination address of the packets (*-d*) by filling in the first editing box from destination with an IP or network address, case in which we can also specify a network mask in the second editing box. Also, we can choose from few predefined IP addresses or few predefined network masks from the drop-down boxes at the right of the editing boxes.

E x a m p l e s.
iptables -A INPUT -s 192.168.1.1/255.255.255.255 -j ACCEPT
iptables -A INPUT -d 192.168.1.0/255.255.255.0 -j ACCEPT
iptables -A INPUT -d 192.168.1.0/255.255.255.0 -j ACCEPT

We can choose the interface through which the packets are coming (*-i*) or the interface through which the packets are leaving (*-o*). Many interfaces are available: eth0, eth1, eth+, ppp0, ppp1, ppp+. eth0 is the first Ethernet card, eth1 is the second, and eth+ is any Ethernet card. Ppp0 is the first PPP (Point-to-Point Protocol)

interface, ppp1 is the second interface, ppp+ is any PPP interface. The entering interface is valid only in the INPUT, OUTPUT and PREROUTING chains, and the exit interface is valid only in the FORWARD, OUTPUT and POSTROUTING chains.

O b s e r v a t i o n s.

On the INPUT chain only choosing the entry interface makes sense. If an exit interface is chosen, the rule will not be accepted.

On the OUTPUT chain only choosing the exit interface makes sense. If an entry interface is chosen, the rule will not be accepted.

On the INPUT chain only choosing the entry interface makes sense. If an exit interface is chosen, the rule will not be accepted.

On the FORWARD chain an entry interface as well as an exit interface.

The protocols and ports contain four options: tcp, udp, icmp, or all of which only one can be chosen, through a set of ratio buttons. The all option is default. The *-p tcp* option specifies the use of TCP protocol and has the following options:

a) *-s*port - specifies the source of the packets;

b) *-d*port - specifies the destination port of the packets;

c) *-tcp*-flags - there are seven flags: SYN, ACK, FIN, RST, URG, PSH, ALL, NONE.

The first argument after *tcp*-flags are the flags that need to be examined, separated by commas and the second argument is a list of flags, separated by commas, that need to be set. Thus, the command: iptables -A FORWARD -p tcp -tcp-flags SYN, ACK, FIN, RST, SYN -j ACCEPT will match only to the packets with the SYN flag set and the ACK, FIN, and RST unset.

a *-sys* - if this option is checked, the *-tcp*-flags option is ignored; it matches only the packets with the SYN flag set and the ACK, FIN flags unset. Such packets are used to initiate a connection. Blocking this type of packets will ignore the initiation of TCP connections from outside, but the outgoing TCP connections will be unaffected. It is equivalent to specifying the *-tcp*-flags SYN, ACK, FIN SYN. For example, the rule: iptables -A INPUT -p tcp -dport 25 -syn -j DROP will block all connection on port 25 from outside.

The *-p udrp* option specifies the UDP protocol and has the following options:

a) *-s*port: the source port of the packets; it can specify one port or a series of ports;

b) *-d*port: the destination port of the packets; it can specify one port or a series of ports.

E x a m p l e s.
The iptables -A INPUT -p udp -sport 34 -dport 137:139 -j DROP rule will forbid all packets entering the Linux machine (but not those that convey through, for this it will be used -A FORWARD), using the UDP protocol and having the source port 34 and the destination port 137, 138, or 139.

The *-p icmp* option specifies the *icmp* protocol which is a protocol used for signaling errors. It uses the following messages: *echo-reply, destination-unreachable, network unreachable, host-unreachable, protocol-unreachable, port-unreachable, fragmen-*

tation-needed, source-route-failed, network-unknown, host-unknown, network-prohibited, host-prohibited, TOS-network-unreachable, TOS-host-unreachable, communication-prohibited, host-precedence-violation, precedence-cutoff, source-quench, redirect, network-redirect, host-redirect, TOS-network-redirect, TOS-host-redirect, echo-request, router-advertisement, router-solicitation, time-exceeded, ttl-zero-during-tranzit, ttl-zero-during-reassembly, parameter-problem, ip-header-bad, required-option-missing, timestamp-request, timestamp-reply, address-mask-request, address-mask-reply.

E x a m p l e.

iptables -A INPUT -p icmp -icmp-type icmp-echo-request-j DROP will forbid all ping packets coming towards the Linux machine.

Iptables can use modules to extend the filtering possibilities. These are loaded by default, by using the *-m* or *-meci*, followed by the name of the module. After this, other options become available, depending on the specified module. In a single line many can be specified.

A. **The mac module** (*-m mac*) has the following options: *-mac*-source address.

The *mac* addresss has to be like xx:xx:xx:xx:xx:xx and the rule matches the packets that come from this *mac* address. The rule makes sense only for the packets that enter the PREROUTING, FORWARD or INPUT and that come from one Ethernet device.

B. **The limit module** (*-m* limit)

This module limits the packets by using a token bucket filter. It can be used in combination with the LOG target to limit the logs. One rule using this expression will match until this limit is reached (only if the negation flag is not specified!). It has the following options:

a) -limit-rate: average rate of matching. It is a number with an optional suffix: /second, /minute, /hour, /day. Default is 3/hour.

b) -limit-burst number: the maximum initial packet number that mach. This number is increased by one each time the above-specified limit is not reached, until this number; default is 5.

C. **The multiport module** matches to one set of source or destination ports. Up to 15 ports can be specified. It can be used only together with "*-p tcp*" or "*-p upd*". It has the following options:

a) -source-port port, port; it matches only if the source port is one of the given ports;

b) -destination-port port,port; it matches only if the destination port is one of the given ports;

c) -port port,port; it matches only if both the source and the destination port are equal and are one of the given ports.

D. **The owner module** matches with the packets that were created by a certain creator, only the local generated packets. It is valid only in the OUTPUT chain and even here some packets (like the ping replies) have no owner, therefore don't match.

a) *-uid*-owner userid: it matches only if the packet was created by a process with the given userid;

b) *-gid*-owner userid: it matches only if the packet was created by a process with the given userind.

c) *-pid*-owner processid: it matches only if a process with the given pid created the packet.

d) *-sid*-owner sessionid: it matches only if the packet was created by a process from the given session group.

E. **The state module**, when used together with connection tracking, allows the access to the state of tracking the connection for this packet.

-state state: where state is a list of connection statuses separated by comma that chackes up with the transited packets. Possible states are:

a) INVALID means that the packet is not associated to any connection;

b) ESTABLISHED means the packet is associated to a connection that has transferred packets both ways;

c) NEW means that the packet has begun a new connection or, otherwise, that the packet is associated to a connection that has not transfered packets in both ways;

d) RELATED means that the packet initiates a new connection, but is associated with an existing connection, such as a data transfer through FTP or an error message through ICMP.

A target (*-j*) can be finally chosen from the ACCEPT, DROP, QUEUE, RETURN options or the REJECT, MIRROR, LOG options.

F. **NAT**: there are three options for the NAT table:

a) change the policy of the PREROUTING, POSTROUTING or OUTPUT chain to ACCEPT or DROP;

b) resets the counters of the PREROUTING, POSTROUTING,OUTPUT chain or of the user defined chains;

c) creates a new user defined chain.

The PREROUTING chain is for altering the packets as soon as they enter the Linux machine.

The OUTPUT chain is for altering the local generated packets before routing.

The POSTROUTING chain is for altering the packets before they exit the Linux machine.

2.2. Adding Rules to the NAT Table

A. **MANGLE**: there are three options for the MANGLE table:

a) change the policy of the PREROUTING or OUTPUT chain to ACCEPT or DROP;

b) resets the counters of the PREROUTING, OUTPUT chains or of the user defined chains;

c) creates a new user defined chain.

The PREROUTING chain is for altering the packets that enter before routing.

The OUTPUT chain is for altering packets before routing.

B. **The Mark Module**: –mark value/mask.

Matches to the packets with the given mark value (if a mask has been specified, then logical AND is made with the mask before comparison).

C. **The tos Module** matches to the 8 bits from the "Type of Service" field from the IP header. *–tos tos*: the argument is either a standard name, or a numeric value.

a) Minimize-Delay 16 (0×10).
b) Maximize-Throughput 8 (0×08).
c) Maximize-Reliability 4 (0×04).
d) Minimize-Cost 2 (0×02).
e) Normal-Service 0 (0×00).

2.3. Adding Rules to the MANGLE Table

This page is similar to the "Adding rules to the FILTER table" page, with the following differences: the following MIRROR. TOS, MARK targets may be used.

In the topside of the display we have four more commands available through buttons:

A. **Home** – displays the homepage.

B. **Log Connections** – shows a list of all users of the application that have connected, if there were incorrect logins, the *ip* from which they connected, date and hour.

C. **Log Commands** – displays all commands executed by tcpServer.

D. **Lougout** – leaves the application, erasing the PHP session.

3. Description of Code

The **tcpServer** program written in C listens to requests at address 127.0.0.1, port 1500. If the program is launched with *root* rights it is obvious that he too will be able to launch programs that require *root* rights.

tcpServer is compile with the following command: gcc tcpServer.c -o tcpServer

Such a program can be dangerous from the security point of view, because any command from the system can be executed. That's why at least two security measures are in order:

1. The program will have the *root* owner and the *root* group: chown root.root tcpServer.

2. The access rights will be read, write, execute for the owner and nothing for the rest: chmod 500 tcpServer.

The result of the la –lsa tcpServer command is:

20 -rwxr-xr-x 1 root root 17147 Jun 13 04:02 tcpServer

3.1. Function Send_cmd($comanda)

```
{   $fp = fsockopen ("127.0.0.1", 1500, &$errno, &$errstr,
30);
  if (!$fp)
    {
```



```
    echo "$errstr ($errno)<br> \n"; exit;
    }   else
    {
    fputs ($fp, "parol"); //trimit parola
    fputs ($fp, "$comanda");
    while(!feof($fp))
      fgets($fp,1800);
    fclose ($fp);
    }
}
```

a) The send_cmd($comanda) function allows the dispatch of a command to *tcp*Server by creating a socket at the destination address 127.0.0.1, port 1500. with the help of the fsockopen function.

b) The prototype of the fsockopen function:

int fsockopen (string [udp://]hostname, int port [, int errno [, string errstr [, double timeout]]])

c) The prototype of the fputs function:

int fputs (int fp, string str [, int length])

d) Then in the while cycle is received the data resulted from the execution of the command through the fclose function. The prototype of the fclose function: bool fclose (int fp)

3.2. The Create Preview Function from the Adăugare_filter.php File

```
function createPreview() {
...
if (document.filter_table.newproto[2].checked=="1")
//protocol ICMP
    if (documet.filter_table.icmp_source.value!="  ")
      newproto="  -p icmp "+"--icmp-type "+
            document.filter_table.icmp_source.value;
    else newproto="  -p icmp";
...
document.filter_table.regulafin.value="iptables "+
 lit+indev+outdev+src+dest+newproto+modul_mac+modul_limit+
 modul_multiport+modul_owner+modul_state+action;

}
```

The createPreview function, written in JavaScript, creates a firewall rule beginning from the options filled in the form. In the above example, if the ICMP protocol is selected from the set of radio buttons (called newproto in the form) and if a ICMP message is selected from the *-icmp*-type box (called icmp_source in the form) which contains the type of the ICMP messages, then the option will become, let's say. newproto=" -p icmp -icmp-type echo-request". If the type of ICMP messages is not selected, then the protocol stays "-p icmp":":else newproto=" -p icmp";

3.3. Fragments from TcpServer.c

Example 1

```
...
/* create socket */
```

```
  sd = socket(AF_INET, SOCK_STREAM, 0);
    if (sd<0) {
      perror("cannot open socket ");
      return ERROR;
  }
  /* bind server port */
  servAddr.sin_family = AF_INET;
  servAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
servAddr.sin_port = htons(SERVER_PORT);

  if(bind(sd, (struct sockaddr *) &servAddr,
sizeof(servAddr))<0) {
      perror("cannot bind port ");
      return ERROR;
  }
listen
...
```

In the above example a socket is created by invoking the socket function from the C language. In case of error at the creation of the socket, the program displys a message of error (perror("cannot open socket ");) and terminates its execution by invoking the return function. The socket is endowed with the servAddr structure, which contains the type of the socket (AF_INET), the address (127.0.0.1) and the port (SERVER_PORT=1500). This endowment is done by invoking the bind() function. In case of success, goes into the listening mode by invoking the listen() function.

Example 2

```
while(1) {
  cliLen = sizeof(cliAddr);
  new_socket = accept(sd, (struct sockaddr *) &cliAddr,
&cliLen);
  if(new_socket<0)
    {
    perror("cannot accept connection ");
    return ERROR;
    }
  /* init line */
  for (y=0;y<bufsize+1;y++) buffer[y]=0;
//receives the password in the passwd variable
recv(new_socket, passwd,5,0);

if (strcmp(passwd,"parol")==0)
{
//receives the command in the buffer variable
recv(new_socket, buffer, bufsize,0);

//write the command in the log
write_log(buffer);

//execute the command
teava=popen(buffer,"r");
//sends the result of the command through socket to php
while (fgets(buffer, 180, teava))
send(new_socket, buffer,strlen(buffer),0);
pclose(teava);

close(new_socket);
}
else //password is wrong, write in log "unauthorized access
and
  //we close the connection
{
...
strcpy(buffer,"Acces neautorizat. Ati fost logat!");
send(new_socket, buffer,strlen(buffer),0);
close(new_socket);
}
...
```


When a client connects to **tcpServer**, a new socket is created by invoking the accept() function. Then a five characters string which is verified with the "*parol*" password is received from the socket, through the recv() function. If they match, that is if the correct password has been sent, the received command is executed, otherwise an error message is printed, followed by the closing of the connection.

After the password checking the command is received in the buffer variable and it is written in a log file by invoking the write_log(buffer) function.

Linux allows the redirection of the inputs and outputs with the help of some special operators. One of them is the vertical bar "|" (*pipe*), or the connection operator of two or more commands. Example: command1|command2.

Thus, the output of the first command is directly connected to the input of the second command.

There is pipe also in C: Linux allows this by using the popen() function.

The prototype of the popen function:

FILE *popen(char *command, char *type)

The popen function opens an input/output "duct" (*pipe*) where the command command is the son process at which the parent process will connect, thus creating the "duct".type is the type of the "duct":"r" - for reading or "w" for writing.

The result of the popen command is NULL in case of error or pointer to a stream. A "duct" opened by popen() must always be closed by the pclose(FILE *stream) function. To communicate with the "duct's" stream the functions fprintf() and fscanf() or fgets() and fputs() are used.

In the above example, the line teava=popen(buffer,"r"); creates a "duct" between the *tcp*Server and the process resulting from the execution of the command contained in the buffer variable.

fgsets(buffer,180,teava) and then sent to the new_socket socket like this: send(new_socket,buffer,strlen(buffer),0);.

After the son process has finished sending the output data, the "duct" is closed by invoking the pclose():pclose(teava); function.

Finally, the new_socket: close(new_socket); socket is also closed and the *tcp*Server program continues to listen to port 1500.

4. Conclusions

Because Linux is by definition an operating system for computer networks, its network possibilities are countless: TCP/IP, TCP/IP v6, IPX/SPX, Apple Talk, WAN, X.25, Frame Relay, ISDN, PPP, SLIP, PLIP, Radio Amateur, ATM, file sharing, email, www, FTP, NEWS, DNS, DHCP, NIS, LDAP, Telnet, X, VNC, Router, Multicasting, Bridge, IP Masquerade, IP Accounting, IP Aliasing, Traffic Shaping, Firewall, Port Forwarding, Load Balancing, EQL, Proxy, Diald, Tunneling, IP Mobil, VPN, SNMP, RAID.

Linking of a computer to the Internet implies, generally, the use of the Linux operating system and of the TCP/IP protocol suite. These components have their own security problems. Access to the INTERNET also implies the use of a set of dozens of services, programs, with various security problems, either due to software errors or due to mis-incorporation of some appropriate security facilities. In general, with a view that a user be capable to take the appropriate security measures at network connection, he must understand the way in which the LINUX operating system works with the Internet.

The "PHP Firewall Generator" is a very useful tool designed for LINUX system administrators. The Web interface designed in PHP + JavaScript allows the easily composure of firewall rules, as well as their easily manipulation. With few mouse clicks any rule from the firewall can be written, edited or deleted.

The application has been successfully tested on the RedHat 7.2 and RedHat 9 operating systems, with Apache 1.3.20 and PHP 4.0.6. The minimum hardware requirements to run the application in good conditions are Pentium II, 64 MB RAM.

Received, November 3, 2003

„Ovidius" Univesity, Constantza, Romania

REFERENCES

[1] Buraga S., Ciobanu G., *Computer Networks Programming Workshop* (in Romanian). Ed. Polirom, Bucharest, 2001.

[2] Jurca I., *Programming Computer Networks* (in Romanian). Ed. de Vest, Timişoara, 2000.

[3] Ogletree T.W., *Firewalls Protection of Networks Connected to the Internet* (in Romanian, transl. from English). Ed. Teora, Bucureşti, 2001.

[4] Petac E., Petac D., *Principles and Techniques of Data Protection in Computer Networks* (in Romanian). Ed. MatrixRom, Bucureşti, 1998.

[5] Schneier B., *Applied Cryptography*. J. Wiley a. Sons, New York, 1994.

[6] Tanenbaum A.S., *Computer Networks* (in Romanian, transl. from Russian). Ed. Agora. Tg. Mureş, 1988.

[7] Ziegler R.L., *Firewalls - Protecting Linux Systems* (in Romanian, transl. from English). Ed. Teora. Bucureşti, 2001.

[8] - - http://www.seifried.org/lasg Linux Administrators Security Guide.

[9] - - http://www.redhat.com RedHat Network

[10] - - http://www.iptables.org IPTables Home Page

[11] - - http://netfilter.samba.rog/ NetFilter.

[12] - - http://linuxfocus.org/ About Linux.

[13] - - http://www.linux.org/ About Linux.

[14] - - http://securityfocus.com About computer security.

[15] - - http://www.bugtraq.com/ About exploits.

[16] - - http://www.freshmeat.net/ Software for Linux.

GENERATOR DE FIREWALL (II)

(Rezumat)

Generatorul de firewall propus este destinat administratorilor de sisteme LINUX. Interfaţa Web, realizată în limbajele PHP şi JavaScript, permite compunerea regulilor firewall precum şi manipularea cu uşurinţă a acestora. Printr-un mecanism simplu, orice regulă a firewall-ului se poate scrie, modifica sau şterge. Conectarea unui calculator la Internet presupune, în general, folosirea sistemului de operare LINUX şi a suitei de protocoale TCP/IP. Aceste componente au propriile lor probleme de securitate. Accesul la Internet presupune şi folosirea unui set de câteva zeci de servicii, programe, cu numeroase probleme de securitate, fie datorită unor erori în software, fie datorită neîncorporării unor facilităţi de securitate potrivite. Iptables este folosit pentru a configura, administra şi inspecta tabelele de reguli de filtrare a pachetelor din kernel-ul LINUX-ului. Fiecare din tabelele definite poate conţine un număr de lanţuri predefinite dar poate conţine şi lanţuri definite de utilizator.