ENCODING FIREWALL RULES IN PROLOG

Submitted by -
Honnesh Rohmetra
2016B2A70770P
Rachit Agrawal
2016B2A70901P

A firewall rule consists of several statements (or clauses) that define the traffic for which the rule applies.
Firewall rules are encoded in Prolog as facts and rules: each rule may start with accept (allow the incoming packet), reject (send reject information to sender), or drop (silently) followed by a clause. The following are the predicates defined for each firewall clause-:

Our firewall follows a **strict pattern**(i.e. if in the input packet any one of the clauses or conditions is rejected as specified by the rule base, the packet is rejected). A single rejection parameter is enough for the input to get rejected.

- rejectadapter(X,Y) ~ specifies a range of values of the adapter which need to be rejected(it will reject all adapters lying between X and Y(both included)).
- rejectadapter([A]) ~ specifies a specific value of the adapter which has to be rejected (It has to be a list with single element).
- rejectadapter([‘A’,’C’,’F’,’J’]) ~ specifies a list of values of the adapter which need to be rejected
- rejectadapter(any) ~ it will reject all the adapters

- acceptadapter(X,Y) ~ specifies a range of values of the adapter which need to be accepted(it will accept all adapters lying between X and Y(both included)).
- acceptadapter([A]) ~ specifies a specific value of the adapter which has to be accepted (It has to be a list with single element).
- acceptadapter([‘A’,’C’,’F’,’J’]) ~ specifies a list of values of the adapter which need to be accepted
- acceptadapter(any) ~ it will accept all the adapters

- dropadapter(X,Y) ~ specifies a range of values of the adapter which need to be dropped(it will drop all adapters lying between X and Y(both included)).
- dropadapter([A]) ~ specifies a specific value of the adapter which has to be dropped (It has to be a list with single element).
- dropadapter([‘A’,’C’,’F’,’J’]) ~ specifies a list of values of the adapter which need to be drop
- dropadapter(any) ~ it will drop all the adapters

Similar syntax can be followed like-:

FG where F={accept,reject,drop} and G={adapter,Vid,proto,IPV4src,IPV4dst,TCPsrc,TCPdst,UDPsrc,UDPdst,ICMPcode,ICMPtype}

‘Any’ type is supported in all expressions

‘Any’ type predicate can be used to ACCEPT, REJECT or DROP all the possible inputs of a specific component type.
Eg-

- acceptadapter(any).
- rejectadapter(any).
- dropadapter(any).
- acceptIPV4dst(any).

Range of IP Addresses are supported

Eg -

rejectIPV4dst('127.233.212.222','134.333.321.111').

Blocks *all the IP addresses* having destination address lying between the above 2 values.

Netmasking is also supported

n.n.n.n/<netmask>

- /24 means taking ‘and’ with 255.255.255.0 so only first three parts of the ip address will be considered.
- /16 means taking ‘and’ with 255.255.0.0 so only first two parts of the ip address will be considered.
- /8 means taking ‘and’ with 255.0.0.0 so only first part of the ip address will be considered.

- acceptIPV4srcmasking(’123.XXX.XXX.XXX/8’) ~ will accept all ip addresses having source address starting with 123.
- acceptIPV4srcmasking(’123.132.XXX.XXX/16’) ~ will accept all ip addresses having source address starting with 123 AND 132.

- acceptIPV4srcmasking('123.132.143.XXX/24') ~ will accept all ip addresses having source address starting with 123 , 132 AND 143.

Netmasking is supported for ~ acceptIPV4src, rejectIPV4src, dropIPV4src, acceptIPV4dst, rejectIPV4dst and dropIPV4dst.

- ICMP types vary from 0 to 254 and the ICMP codes also vary from 0 to 254 for the same type(taken arbitrarily).

Input packet contains the following information-: **Adapter ,Ethernet ,IPv4 datagram clause (Containing source and destination address,protocol), TCP/UDP (source/destination address and port numbers), ICMP type and conditions**
(Assumed that all these values are present in the input packet and considered independently).

Input packet is of the form –

input(Adapter, EthernetVid, EthernetProtocol, IPV4 source address, IPV4 destination address, IPV4 protocol Type, X, Y).

Where
- Adapter – Adapter Type, Ranges from 'A' to 'P'
- EthernetVid –

Specifies a virtual LAN (VLAN) identifier to match the 802.1 frame.

- EthernetProtocol –

The ether types(in hexadecimal) are as given on the IANA website.
(https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xml)

Eg - acceptproto(['0x876D']).~Accepts the packet containing this particular ether protocol.

- IPV4 source address – Of the form 'xxx.xxx.xxx.xxx'
- IPV4 destination address – Of the form 'xxx.xxx.xxx.xxx'
- IPV4 protocol type – Can be ***any one*** of the below 3(**only** these 3 are allowed)-
  Protocoltype – **'ICMPV4'**
  X – ICMP type, ranges from 0-254
  Y – ICMP type code, ranges from 0 - 254
  Protocoltype – **'TCP'**
  X – TCP source port, ranges from 0 - 254
  Y – TCP destination port, ranges from
  Protocoltype – **'UDP'**

X – UDP source port, ranges from 0 - 254
Y – UDP destination port, ranges from

Sample input ~

Input to the firewall engine is to be given as-:
input('P',123,'0x876D','123.222.222.145','134.111.234.333','TCP',103,111).

Sample output ~

'Adapter accepted''Vid accepted''Ethernet protocol accepted'''Source IP Accepted''''Destination IP rejected'''TCP source port Behaviour not recognized''TCP destination port Behaviour not recognized'''----FINAL RESULT---''''PACKET REJECTED---''
true

Output format explanation –

Output gives information about each of the components of packet.
- When a particular component is accepted, message displayed is – "X accepted"
- When a particular component is rejected, message displayed is – "X rejected"
- When a particular component is accepted, no message is displayed.

Where X is the component of i/p packet.

Final Output gives information about whether the packet is ACCEPTED, REJECTED or DROPPED.

"----FINAL RESULT---""PACKET REJECTED---" means Packet is rejected
"----FINAL RESULT---""PACKET ACCEPTED---" means Packet is accepted
"----FINAL RESULT---" means Packet is dropped

Policy of FireWall engine –

- Anyone component is rejected, Packet is rejected.
- No component is rejected and anyone component is dropped, Packet is dropped.
- All components are accepted, Packet is accepted.
- Any packet's behaviour not recognized(Not specified in the rule base),then also the packet will be rejected.

<u>Default Configuration file must be present</u> ~

A default configuration file is needed to declare a set of predicates. Further changes can be made into this configuration file as per the user's need to accept, reject or drop some specific components.
The default file is given below -

```
/*adapter sample*/
rejectadapter('','').
rejectadapter(['','']).

acceptadapter('','').
dropadapter('','').
rejectadapter('','').
acceptadapter(['']).
rejectadapter(['']).
dropadapter(['']).

/*ethernet sample Vid*/

acceptVid([0,0]).
rejectVid([0]).
dropVid([0]).
acceptVid(0,0).
rejectVid(0,0).
dropVid(0,0).

/*ethernet sample Protocol*/

acceptproto('','').
dropproto('','').
rejectproto('','').
acceptproto(['0']).
rejectproto(['']).
dropproto(['']).

acceptproto('','').
dropproto('','').
rejectproto('','').
```

```
acceptproto(['0x876D']).
rejectproto(['']).
dropproto(['']).

/*IPV4*/
acceptIPV4srcmasking('').
rejectIPV4srcmasking('').
dropIPV4srcmasking('').
acceptIPV4src('','').
dropIPV4src('','').
rejectIPV4src('','').
acceptIPV4src(['']).
rejectIPV4src(['']).
dropIPV4src(['']).


rejectIPV4dstmasking('').
acceptIPV4dstmasking('').
dropIPV4dstmasking('').
acceptIPV4dst('','').
dropIPV4dst('','').
rejectIPV4dst('','').
acceptIPV4dst(['','','']).
rejectIPV4dst(['']).
dropIPV4dst(['']).

/*ICMP*/
acceptICMPtype(any).
acceptICMPtype(-1,-1).
dropICMPtype(-1,-1).
rejectICMPtype(-1,-1).
acceptICMPtype([0]).
rejectICMPtype([-1]).
dropICMPtype([-1]).

dropICMPcode(any).
acceptICMPcode(-1,-1).
dropICMPcode(-1,-1).
rejectICMPcode(-1,-1).
acceptICMPcode([-1]).
rejectICMPcode([-1]).
dropICMPcode([-1]).
/*UDP*/
```

```
acceptUDPsrc([0,0]).
rejectUDPsrc([0]).
dropUDPsrc([0]).
acceptUDPsrc(0,0).
rejectUDPsrc(0,0).
dropUDPsrc(0,0).

acceptUDPdst([0,0]).
rejectUDPdst([0]).
dropUDPdst([0]).
acceptUDPdst(0,0).
rejectUDPdst(0,0).
dropUDPdst(0,0).

/*TCP*/

acceptTCPsrc([0,0]).
rejectTCPsrc([0]).
dropTCPsrc([0]).
acceptTCPsrc(0,0).
rejectTCPsrc(0,0).
dropTCPsrc(0,0).

acceptTCPdst([0,0]).
rejectTCPdst([0]).
dropTCPdst([0]).
acceptTCPdst(0,0).
rejectTCPdst(0,0).
dropTCPdst(0,0).
```