ReadMe File (Documentation)

CSF214 Logic in Computer Science – Assignment

Made By:

Aryan Mehra (2017A7PS0077P)

Ayush Jain (2017A7PS0093P)

Topic: Sample Firewall Implementation in Prolog

The following project implements a sample firewall by using the following 8 criteria to accept or reject the given IP packet: **Adapter Type, Ethernet Protocol, Ethernet Virtual LAN ID, IP source address, IP Destination Address, IP type of protocol, TCP UDP port number and ICMP code protocol and message code.**

The code is separated into a firewall engine and knowledge base wherein the knowledge base is well commented explaining every function and predicate used. The explanation is given below as well after the sample inputs. The firewall engine is structured in a way that it is independent of the knowledge base values.

SAMPLE INPUTS AND EXPECTED OUTPUT

INPUT FORMAT:

The input is a String with following prototype. The field are separated by "space" which acts as the delimiter in our code.

"adapter_type ethernet_protocol_number ethernet_VLAN_ID source_IP_address destination_IP_address ip_protocol tcpudp_port_number icmp_code_protocol"

OUTPUT FORMAT: (Combination of conditions for reject & drop is chosen randomly)

PRIORITY: REJECT > DROP > ACCEPT

Accept: This means that all the conditions on all the clauses and information has been checked (multiple conditions taken into account) and the packet is allowed.

Reject: This means that the packet has been rejected if any of these fails: Adapter, Ethernet Protocol Number, Ethernet VLAN ID, IP source address, ICMP Code protocol or IP Protocol. ***The message also tells everything that is wrong with the packet.***

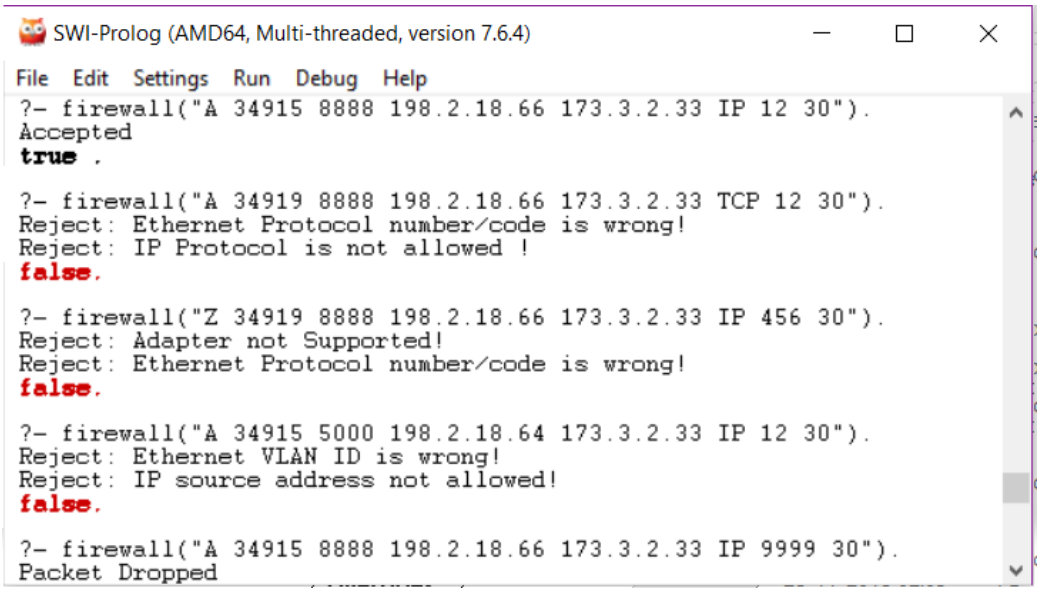
Dropped: This means that there was no rejection but due to error in ***the destination address and/or TCP/UDP Port number***, the packet has been dropped but no error message has been displayed and the reason is left unknown for the user.

[PTO]

10 SAMPLE INPUTS are provided for reference:

SNO	SAMPLE INPUT AND OUTPUT	Remark
1.	?- firewall("A 34915 8888 198.2.18.66 173.3.2.33 IP 12 30"). **Accepted**	Accepted packet
2.	?- firewall("A 34915 8888 198.2.18.66 173.3.2.33 TCP 8 30"). **Reject: IP Protocol is not allowed !**	Rejected
3.	?- firewall("A 34915 5000 198.2.18.66 173.3.2.33 TCP 12 30"). **Reject: Ethernet VLAN ID is wrong!** **Reject: IP Protocol is not allowed !**	Multiple rejections displayed
4.	?- firewall("A 34919 8888 198.2.18.66 173.3.2.33 IP 12 99"). **Reject: Ethernet Protocol number/code is wrong!** **Reject: ICMP Message code or protocol is denied !**	Multiple rejections displayed
5.	?- firewall("Z 34915 8888 198.2.18.66 173.3.2.33 IP 12 30"). **Reject: Adapter not Supported!**	Rejected
6.	?- firewall("A 34915 8888 200.2.18.66 173.3.2.33 IP 999 30"). **Packet Dropped**	Drop: no message
7.	?- firewall("Z 34915 8888 198.2.18.66 173.3.2.33 IP 4566 30"). **Reject: Adapter not Supported!**	Reject overrides drop
8.	?- firewall("Z 34915 8888 198.2.18.64 173.3.2.33 IP 12 30"). **Reject: Adapter not Supported!** **Reject: IP source address not allowed!**	Multiple Reject overrides drop

SCREENSHOT



[PTO]

KNOWLEDGE BASE / DATABASE

The comments in the code for knowledge base clearly mention the specific examples that we chosen to block or allow in each category.

1. **Adapter Protocols:** There is one predicate called *adapterAllow(adapter_name)*. This predicate accepts the name of the adapter has provisions for allowing the following:

- "Any" Adapter
- Range of Adapters
- Specific Adapters

This predicate will output "true" value for allowed adapters. Eg: Range of A through H is allowed in our code as an example. Specific ones can also be allowed.

2. **Ethernet Protocols**: There are two predicates called *etherproto(protocol_number)* and *ethervid(VLAN_number).* These predicates accept the number of the ethernet protocol and the VLAN respectively. The provisions are the following:

    - Specific ethernet protocols
    - Range of ethernet VLAN IDs
    - Specific VLAN IDs

These predicates return "true" if they allow the packet. Eg: 34915 is allowed.

3. **IPv4 Protocols**: There are 5 predicates here.

    - The *dotted_IP_address_to_int()* predicate takes in two a list of split IP address (split with respect to the dot delimiter) and converts it to a integer value using the *exponent constant list*. This is further used for comparing various IP address to check whether the given address falls in permissible range or not.

    - The *ip_range()* predicate takes in the IP address and with the help of the above predicate decides whether the given IP is in range or not. The user can change the values of the lower and the upper limit according to choice without affecting the working of the firewall engine. Eg: "198.168.10.0" to "198.168.10.255" is blocked.

    - The *ipv4src()* predicate takes in the source IP address and checks it against blocked IP address for source field. *It returns false in case it matches any one of them and true if not so. Eg: "198.2.18.64" is blocked*

    - The *ipv4dst()* predicate takes in the destination IP address and checks it against blocked IP address for destination field. *It returns false in case it matches any one of them and true if not so. Eg: "173.6.2.33" is blocked.*

    - The *ipproto()* predicate takes in the IP protocol type and checks it against blocked types. *It returns false in case it matches any one of them and true if not so. Eg: TCP is blocked as of now.*

4. TCP/UDP Protocols
   There are two predicates here: The *tcpudprangecheck() and tcpudpsingletoncheck()* take in the TCP/UDP Port address and verify if it lies in the range of the allowed ports. *They return true if the TCP/UDP is allowed, false otherwise. Ports 2-100 and 888 are allowed as of now (example).*

5. ICMP Protocols
   There are three predicates here. *icmp_proto_type* checks for the traceroute protocol with code 30 (taken as an example). *icmp_message_code* checks for the unreachable protocol with code 2. *They return true if they match the correct protocol and messeage codes.*

FIREWALL ENGINE

The firewall engine is directly calling independent predicates from the knowledge base which has been imported initially*. It splits the input into various fields and converts them to numbers or integers if needed, using the atom_number() function.*

It then calls the reject and deny clauses that check for rejection first and deny later, after which it proceeds to accept the packet after checking the information itself once more in the same clause. *Since the logic used is that of AND here, accept has lowest priority after everything has been checked.*

Reject clause combines the randomly chosen conditions for 6 fields like adapter, ethernet protocol number, IP source address etc.

Drop clause combines only destination address and TCP UDP Port number checks.