The firewall has been divided into two modules- setup.pl and engine.pl

## Module : setup.pl

**NOTE :**

1. All inputs are in the actual IEEE and IANA bit string format.
2. The data being input in data.csv has been formatted to help in ease of packet verification by the engine.
3. Recursive calls are made to allow multiple inputs.

- adapter(P) : Asks the user the adapter(s) that the user wants active.
- ether(L) : L denotes the list been added so far. Asks for the ether type the user wants to allow - IPv4, IPv6 or both.
- blockIP(L,X,Y) : Asks the user for the IP addresses they manually want to block in IPv4 and IPv6 respectively. X denotes if IPv4 is selected and Y denotes IPv6.
- reverseList([A|B], L) : Adds the list L to a data.csv file.

**Usage :**

1.) To start the setup
start().

## Module : engine.pl

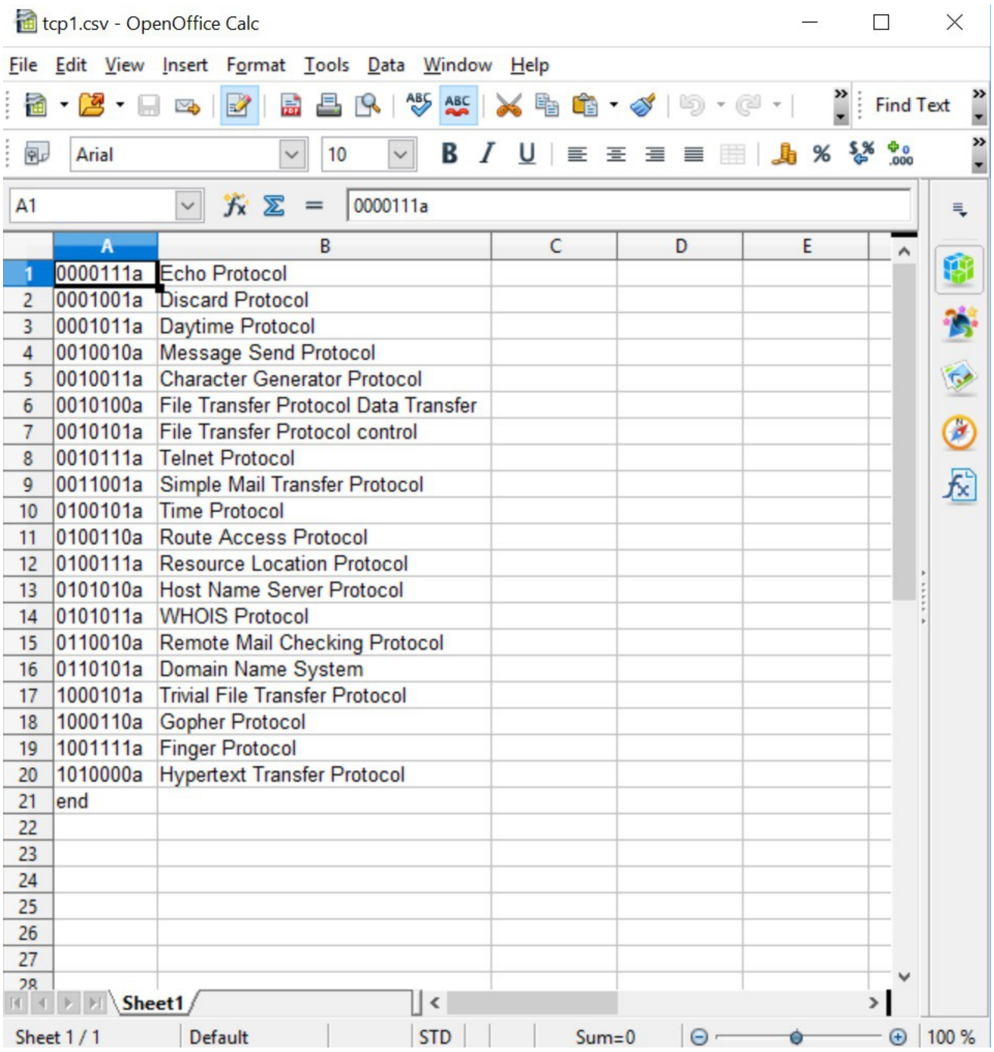
**NOTE :**

1. In the engine the input to the packet containing its details is parsed and represented closely to the real format of a header of a packet by using bit strings and ordering them accordingly.
2. The data structure being used is lists.
3. Sample packet format:
   ['Adapter','802.1q VLAN(if there is) TPID','Ether Type', 'IP address of the source','IP protocol number','Port Number(TCP/UDP)/ Type(ICMP/ICMPV6)', 'Code(ICMP/ICMV6)'].

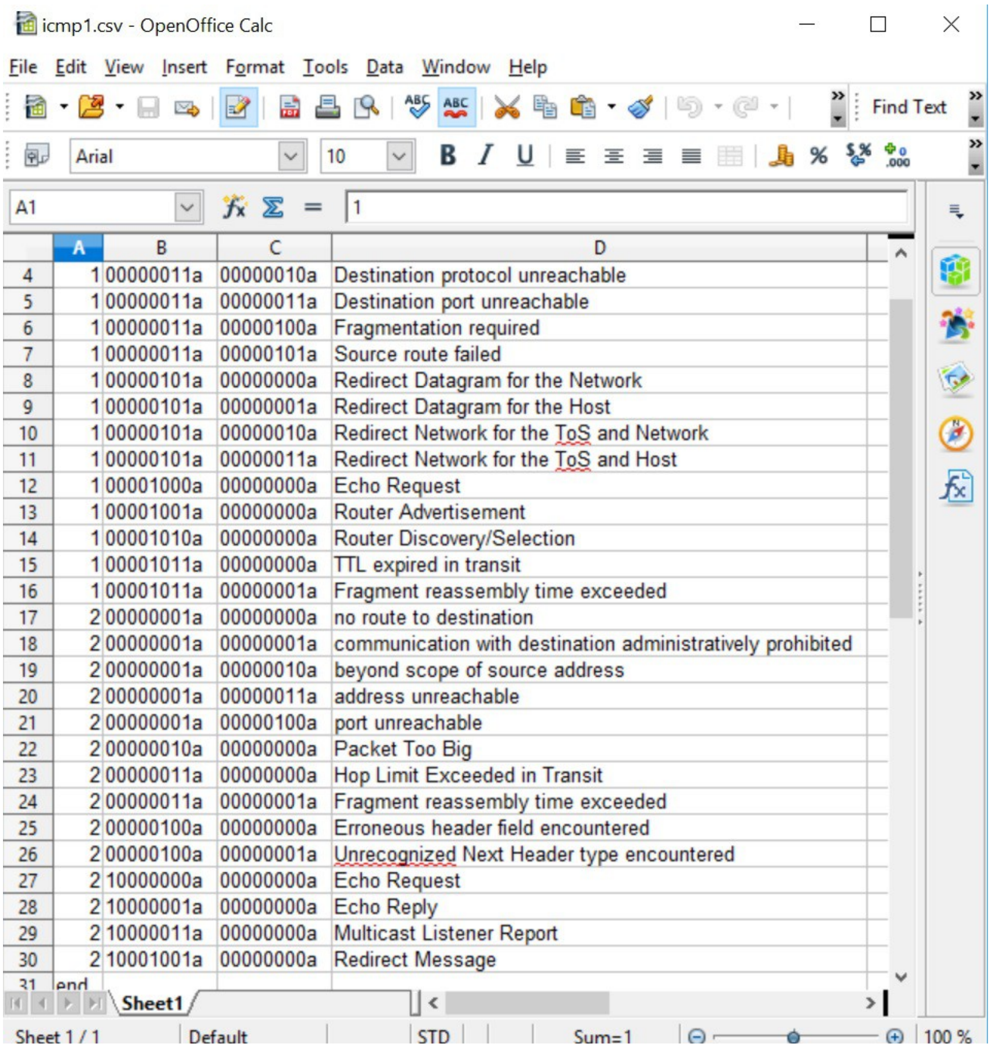
- packetChk([A|B]) : [A|B] is the list containing all details of the input and data.csv is imported to read.
- adapterChk([A|B], [E|F], D) : Checks whether the adapter requested by the packet matches with the adapter activated by the user in set up.
- vLanChk([A|B],[C|D]) : Filters out and rejects 802.1q VLAN traffic based on the presence on TPID(Tagged Protocol Identifier) in the 802.1q frame.

- etherTypeCheck([A|B],[C|D]) : Matches the Ether Type of the packet with the one(s) allowed, else rejected.
- ipAdd([A|B],[C|D],X) : Rejects the blocked IP addresses, else lets the packet pass.
- protoChk([A|B],X) : Checks the IP protocol(TCP, UDP or ICMP/ICMPv6) based on the IP header.
- tcpChk([A|B],[C|D]) : Handles TCP and UDP by checking port number in the IP header with the database csv file and accordingly returns an error, if any. If the port number in the header does not exist, it is dropped.
- icmpChk([A|B],[C|D]) : Handles ICMP and ICMPv6 according to the type and code in the IP header with the database csv file and accordingly returns an error, if any. If the type and code in the header does not exist, it is dropped.

TCP/UDP and ICMP database.
NOTE: All port numbers/type and code numbers haven't been added. Only a few have been added for example.


tcp1.csv - OpenOffice Calc
File Edit View Insert Format Tools Data Window Help
Find Text
Arial
10
A1
0000111a
Sheet1
Sheet 1 / 1
Default
STD
Sum=0
100 %

|  | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 0000111a | Echo Protocol |  |  |  |
| 2 | 0001001a | Discard Protocol |  |  |  |
| 3 | 0001011a | Daytime Protocol |  |  |  |
| 4 | 0010010a | Message Send Protocol |  |  |  |
| 5 | 0010011a | Character Generator Protocol |  |  |  |
| 6 | 0010100a | File Transfer Protocol Data Transfer |  |  |  |
| 7 | 0010101a | File Transfer Protocol control |  |  |  |
| 8 | 0010111a | Telnet Protocol |  |  |  |
| 9 | 0011001a | Simple Mail Transfer Protocol |  |  |  |
| 10 | 0100101a | Time Protocol |  |  |  |
| 11 | 0100110a | Route Access Protocol |  |  |  |
| 12 | 0100111a | Resource Location Protocol |  |  |  |
| 13 | 0101010a | Host Name Server Protocol |  |  |  |
| 14 | 0101011a | WHOIS Protocol |  |  |  |
| 15 | 0110010a | Remote Mail Checking Protocol |  |  |  |
| 16 | 0110101a | Domain Name System |  |  |  |
| 17 | 1000101a | Trivial File Transfer Protocol |  |  |  |
| 18 | 1000110a | Gopher Protocol |  |  |  |
| 19 | 1001111a | Finger Protocol |  |  |  |
| 20 | 1010000a | Hypertext Transfer Protocol |  |  |  |
| 21 | end |  |  |  |  |



|  | A | B | C | D |
|---|---|---|---|---|
| 4 | 1 | 00000011a | 00000010a | Destination protocol unreachable |
| 5 | 1 | 00000011a | 00000011a | Destination port unreachable |
| 6 | 1 | 00000011a | 00000100a | Fragmentation required |
| 7 | 1 | 00000011a | 00000101a | Source route failed |
| 8 | 1 | 00000101a | 00000000a | Redirect Datagram for the Network |
| 9 | 1 | 00000101a | 00000001a | Redirect Datagram for the Host |
| 10 | 1 | 00000101a | 00000010a | Redirect Network for the ToS and Network |
| 11 | 1 | 00000101a | 00000011a | Redirect Network for the ToS and Host |
| 12 | 1 | 00001000a | 00000000a | Echo Request |
| 13 | 1 | 00001001a | 00000000a | Router Advertisement |
| 14 | 1 | 00001010a | 00000000a | Router Discovery/Selection |
| 15 | 1 | 00001011a | 00000000a | TTL expired in transit |
| 16 | 1 | 00001011a | 00000001a | Fragment reassembly time exceeded |
| 17 | 2 | 00000001a | 00000000a | no route to destination |
| 18 | 2 | 00000001a | 00000001a | communication with destination administratively prohibited |
| 19 | 2 | 00000001a | 00000010a | beyond scope of source address |
| 20 | 2 | 00000001a | 00000011a | address unreachable |
| 21 | 2 | 00000001a | 00000100a | port unreachable |
| 22 | 2 | 00000010a | 00000000a | Packet Too Big |
| 23 | 2 | 00000011a | 00000000a | Hop Limit Exceeded in Transit |
| 24 | 2 | 00000011a | 00000001a | Fragment reassembly time exceeded |
| 25 | 2 | 00000100a | 00000000a | Erroneous header field encountered |
| 26 | 2 | 00000100a | 00000001a | Unrecognized Next Header type encountered |
| 27 | 2 | 10000000a | 00000000a | Echo Request |
| 28 | 2 | 10000001a | 00000000a | Echo Reply |
| 29 | 2 | 10000011a | 00000000a | Multicast Listener Report |
| 30 | 2 | 10001001a | 00000000a | Redirect Message |
| 31 | end |  |  |  |

Authors

Aayush Atul Verma 2017A7PS0061P
Jithin Kallukalam Sojan 2017A7PS0163P
Ramachandren Shankar 2017A7PS1171P