ASSIGNMENT - CS 3052

DESIGN AND IMPLEMENT A FIREWALL

T.K.P.PERERA

170446L

Code

Github Link: https://github.com/kavindaperera/computer-security-firewall.git

```
import ipaddress
import json

#Considering the two networks 192.168.1.0 and 10.10.10.0
#interface_1 => 192.168.1.0
#interface_2 => 10.10.10.0
# creating a dictionary from ip datagram
def analyse_datagram(datagram_header):
    protocol_dict = {1: 'ICMP', 2: 'IGMP', 6: 'TCP', 17: 'UDP'}
    version_bin = datagram_header[0:4]
    ihl_bin = datagram_header[4:8]
    total_length_bin = datagram_header[16:32]
    #protocol (TCP/UDP)
    protocol_bin = datagram_header[72:80]
    #source address
    saddress_bin = datagram_header[96:128]
    #destination address
    daddress_bin = datagram_header[128:160]
    #Internet Header Length
    ihl = int(ihl_bin, 2) * 32
    protocol = int(protocol_bin, 2)
    protocol = protocol_dict.get(protocol)
    payload = datagram_header[ihl:]
    #getting the dot notations for addresses
    saddress = ipaddress.ip_address(
        int('.'.join(str(int(x, 2)) for x in saddress_bin.split())))
    daddress = ipaddress.ip_address(
        int('.'.join(str(int(x, 2)) for x in daddress_bin.split())))
    sport = int(payload[:16], 2)
    dport = int(payload[16:32], 2)
    headers = {'saddress': saddress, 'daddress': daddress,
               'sport': sport, 'dport': dport, 'protocol': protocol}
    return (headers)

# ip firewall filter
def filter(headers, interface):
    saddress = str(headers.get("saddress")).split(".")
    daddress = str(headers.get("daddress")).split(".")
    sport = str(headers.get("sport"))
    dport = str(headers.get("dport"))
    protocol = headers.get("protocol")
    print(headers)

    with open('config.json') as f:
        rules = json.load(f)

    for key in rules:
        if (rules[key][0].get("interface") == interface):
            if (rules[key][0].get("saddress") != 'any'):
                r_saddress = rules[key][0].get("saddress").split('.')
                if (r_saddress[0] == saddress[0] and r_saddress[1] == saddress[1] and r_saddre
ss[2] == saddress[2]):
                    if (rules[key][0].get("daddress") != 'any'):
                        r_daddress = rules[key][0].get("daddress").split('.')
                        if (r_daddress[0] == daddress[0] and r_daddress[1] == daddress[1] and
r_daddress[2] == daddress[2]):
                            if (rules[key][0].get("dport")!= 'any'):
```

```
                                r_dport = rules[key][0].get("dport")
                                if (int(dport)==int(r_dport)):
                                    print(interface)
                                    print(rules[key][0].get("action"))
                                    break
                                else:
                                    continue
                            else:
                                print(interface)
                                print(rules[key][0].get("action"))
                                break
                        else:
                            continue
                    else:
                        print(interface)
                        print(rules[key][0].get("action"))
                        break
                else:
                    continue
            else:
                print(interface)
                print(rules[key][0].get("action"))
                break
        else:
            continue


def firewall(interface):
    with open(interface+'.json') as f:
        tests = json.load(f)

    for key in tests:
        headers = analyse_datagram(tests[key])
        filter(headers, interface)
        print('================================================================================')

firewall('interface_1')
firewall('interface_2')
```

Firewall filtering Rules


```
{
    "1": [
        {
            "saddress": "192.168.1.0",
            "daddress": "10.10.10.0",
            "sport": "any",
            "dport": "43",
            "interface": "interface_1",
            "action": "ACCEPT"
        }
    ],
    "2": [
        {
            "saddress": "192.168.1.0",
            "daddress": "10.10.10.0",
            "sport": "any",
            "dport": "80",
            "interface": "interface_1",
            "action": "ACCEPT"
        }
    ],
    "3": [
        {
            "saddress": "192.168.1.0",
            "daddress": "10.10.10.0",
            "sport": "any",
            "dport": "any",
            "interface": "interface_1",
            "action": "DENY"
        }
    ],
    "4": [
        {
            "saddress": "10.10.10.0",
            "daddress": "192.168.1.0",
            "sport": "any",
            "dport": "43",
            "interface": "interface_2",
            "action": "ACCEPT"
        }
    ],
    "5": [
        {
            "saddress": "10.10.10.0",
            "daddress": "192.168.1.0",
            "sport": "any",
            "dport": "80",
            "interface": "interface_2",
            "action": "ACCEPT"
        }
    ],
    "6": [
        {
            "saddress": "10.10.10.0",
            "daddress": "192.168.1.0",
            "sport": "any",
            "dport": "any",
```

```
            "interface": "interface_2",
            "action": "DENY"
        }
    ],
    "7": [
        {
            "saddress": "any",
            "daddress": "any",
            "sport": "any",
            "dport": "any",
            "interface": "interface_1",
            "action": "DROP"
        }
    ],
    "8": [
        {
            "saddress": "any",
            "daddress": "any",
            "sport": "any",
            "dport": "any",
            "interface": "interface_2",
            "action": "DROP"
        }
    ]
}
```


Interface 1 – Test Datagrams


```
{
    "datagram_1": "00100101000000000000000000000000000000000000000000000000000000000000000000000000000000000000
001100000000000000000011000000101010000000000100000001000010100000101000001010000000010000000000
101000000000000001010110000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000",
    "datagram_2": "00100101000000000000000000000000000000000000000000000000000000000000000000000000000000000000
100010000000000000000101011000001000000000010000000100001010000010100000101000000001000000011
001001100000000010100000000000000000000000000000000000000",
    "datagram_3": "00100101000000000000000000000000000000000000000000000000000000000000000000000000000000000000
100010000000000000000101011000001000000000010000000100001010000010100000101000000001000000011
001001100000000010101100000000000000000000000000000000000",
    "datagram_4": "00100101000000000000000000000000000000000000000000000000000000000000000000000000000000000000
001100000000000000000110000001010100000000010000001000001010000010100000101000000010000000000
101000000000000010101100000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000",
    "datagram_5": "00100101000000000000000000000000000000000000000000000000000000000000000000000000000000000000
100010000000000000000110000001010100000000010000001100001010000010100000101000000010000000000
010101100000000010100000000000000000000000000000000000000",
    "datagram_6": "00100101000000000000000000000000000000000000000000000000000000000000000000000000000000000000
001100000000000000000110000001010100000000010000001100001010000010100000101000000010000000000
101000000000011001001100000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000"
}
```

Interface 2 – Test Datagrams


```
{
    "datagram_1": "00100101000000000000000000000000000000000000000000000000000000000000000000000000
00110000000000000000000001010000101000001010000000101100000010101000000000010000000100000011
00100110000000001010000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000",
    "datagram_2": "00100101000000000000000000000000000000000000000000000000000000000000000000000000
10001000000000000000010101100000100000000000100000011100000010101000000000010000000100000000
10100000000000000101011000000000000000000000000000000000",
    "datagram_3": "00100101000000000000000000000000000000000000000000000000000000000000000000000000
00110000000000000000000001010000101000001010000010101100000010101000000000010000001000000000
01000000000000110010011000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000",
    "datagram_4": "00100101000000000000000000000000000000000000000000000000000000000000000000000000
10001000000000000000000101000001010000010100000001011000000101010000000000100000010000000011
00100110000000000101011000000000000000000000000000000000",
    "datagram_5": "00100101000000000000000000000000000000000000000000000000000000000000000000000000
10001000000000000000010101100000100000000000100000010110000001010100000000001000000010000000
10100000000000000101011000000000000000000000000000000000"
}
```