Project Report

Network Firewall Implementation

Author Name	Email ID
Abhishek Kumar Maurya	am2633@cornell.edu

Table of Contents

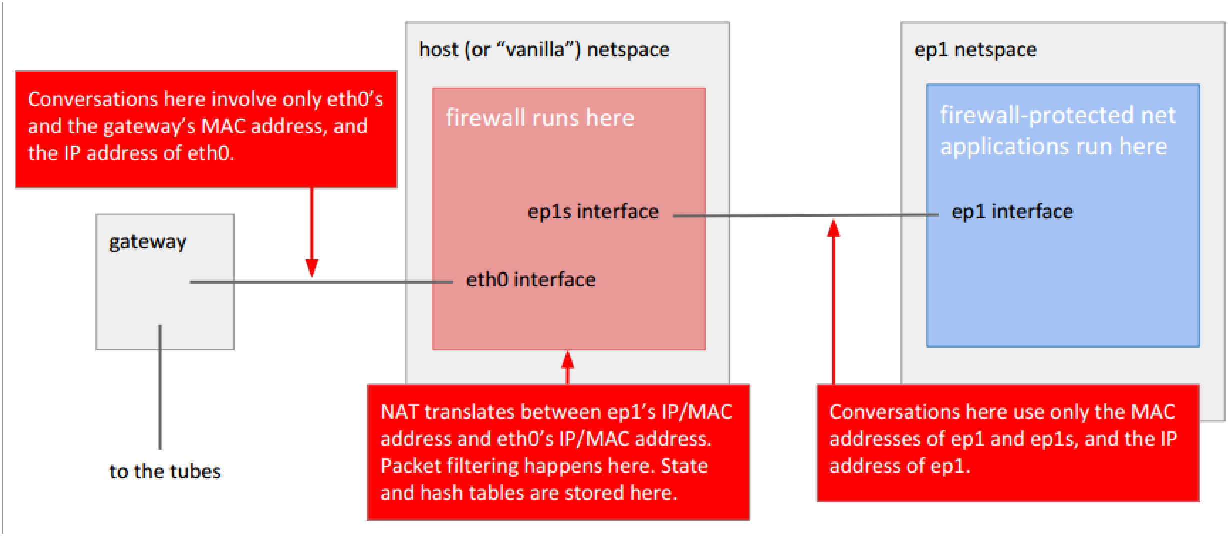
`

1. Motivation

Network segregation is vital while designing networks. Firewall provides the administrator the capability to control the packet flows across the networks. A Simple firewall Implementation can process the packet from one network, evaluate them and apply the action according to the flow policy before forwarding to other network. In this project we have implemented a state full firewall. A State Full firewall, keeps track of the state for each flow and caches the policy results, hence the policy evaluation happens only for the first flow. This provides acceleration in firewall rule processing.

2. Firewall Architecture

Placement of firewall [6]


host (or "vanilla") netspace
ep1 netspace
Conversations here involve only eth0's and the gateway's MAC address, and the IP address of eth0.
firewall runs here
firewall-protected net applications run here
ep1s interface
ep1 interface
gateway
eth0 interface
to the tubes
NAT translates between ep1's IP/MAC address and eth0's IP/MAC address. Packet filtering happens here. State and hash tables are stored here.
Conversations here use only the MAC addresses of ep1 and ep1s, and the IP address of ep1.

The firewall architecture will have following components:

2.1. Policy Engine:

Policy Engine will process rules and stored them in policy engine.

2.2. Policy Evaluator:

Policy Evaluator will evaluates the flow against a rules specified in policy database.

2.3. Action Engine:

Action Engine will carry out the actions on the flow as specified by policy.

Implementation Stages	Status
Rule Syntax	Completed
Syntax Parser	Completed
Processing rules between files	Completed
Internal and external network segregation.	Completed
Nat Implementation	Completed
ARP handling	Completed
ICMP traffic processing	Completed
UDP traffic processing	Completed
TCP traffic processing	Completed
Nat Accelerator	Completed
State-full Firewall Implementation (TCP, UDP, ICMP)	Completed
Cleaner Thread for removing stale flow.	Completed
Optimize code to minimize locks.	*In Progress

3. Network Segregation.

A Segregated network is created using network names pace to test firewall functionality. A network name space provides logically segregated networks. The firewall will spawn two threads for processing packet flows from each networks. Let's name the thread internal thread and external thread operating on interface facing inside (internal ep1s) and outside (external eth0). For state full firewall the threads will maintain a common state machine and work in synchronization with each other. The firewall uses synchronization primitive to have consistent state machine.

Command:

```
sudo ip netns add ep1

sudo ip link add name ep1 type veth peer name ep1s

sudo ip link set ep1 up netns ep1

sudo ip link set ep1s up

sudo ip netns exec ep1 ifconfig ep1 10.0.0.1 netmask 255.255.255.0 broadcast 10.0.0.255 up

sudo ip netns exec ep1 ip link set lo up

sudo ip netns exec ep1 route add default gw 10.0.0.1
```

4. Firewall components

4.1. Rule Processor

4.1.1 Rule Syntax:

A firewall rules compromises of following fields:

- **action** : The action field specify the action taken by firewall once a flow is matched against a certain rule. Action can be “block” or “pass”. The default value of action is taken as blocked.

- **proto** : The proto field specify the protocol for the rule. The rule will be applied to only those flows which will have the same proto field. A proto field can have values as “tcp”, “udp” , “icmp” or “any”. In case of 'any” the rules is matched for all flow.

- **from** : The from field specify the source ip in the flow. Apart from having a valid IP address a “from” field can have “any” wild card .A “any” wild card suggest the flow will be evaluated for all source ip address. The field is optional and have default value as “any”.

- **netmask**: The netmask field specify the netmask for the ip. An IP address is rules is applied for all the source/destination in the subnet. Default value of netmask is “32”. The field is optional.

- **port** : The port field specify the port for the rules. If port field is given before “to” it specify source port else if given after “to” it specify destination port. The port field can accept ranges of port for which the rule needs to be applied. The default value for the port field is the range from 0-MAX_PORT. Port field also supports any wild card which suggests all valid port range.

Note: MAX_PORT is 65535

- **to** : The to field specified the destination ip in the flow. Apart from having a valid IP address a flow field can have “any” wild card. A “any” wild card suggest the flow will be evaluated for all source ip address. The field is optional and have default value as “any”.

- **priority**: The priority field specify the priority order in which the rules are matched. The maximum priority is 15 and the minimum priority is 0. The default value of priority is 0. The rules are processed in the order from, higher priority to lower priority.

Rule Syntax :

```
<action> proto <proto> from <ip_address/netmask> port <port1>-<port2> to
<dest_address/netmask> port <port> priority <priority>
```

Sample Rule file:

```
#
#   Comments FIREWALL RULES
#
pass  proto tcp from 10.0.0.1  port any to 74.125.228.0/24 port 80 priority 2
pass  proto tcp from 74.125.228.0/24 port 80 to 10.0.0.1 port any priority 2
pass  proto icmp from 10.0.0.1 port any to any priority 2
pass  proto icmp from 10.0.0.0/24 to 173.194.121.17 priority 4
pass  proto icmp from 173.194.121.17 to 10.0.0.1 priority 4
block proto icmp from any to 10.0.0.1  priority 0
pass  proto udp  from 255.255.255.9/24 port 60000-65500 to 3.3.3.3/24 port 65520 priority 4
block proto any from any to any priority 1
```

Note: All lines beginning with “#” is treated as comments.
Note: Rules are case insensitive i.e. the rule params BlocK and block is considered as same.

4.1.2 Syntax Parser:

Syntax parser will parse the rules specified in “rules.txt” file which is the default rule file name. However, user can provide the rules files as follows:

Syntax:

```
        sudo ./firewall -r firewall_rules.txt
```

here “firewall_rules.txt” file is the rule file containing the rules

Firewall Rules View is as follows:

```
Priority Action   Proto         Source                    Destination

4       PASS     UDP      255.255.255.9/24[60000-65500]         3.3.3.3/24[65520-65520]

4       PASS     ICMP    173.194.121.17/32[0    -65535]        10.0.0.1/32[0    -65535]

4       PASS     ICMP         10.0.0.0/24[0    -65535]   173.194.121.17/32[0    -65535]

2       PASS     ICMP         10.0.0.1/32[0    -65535]         0.0.0.0/32[0    -65535]
```

```
2      PASS    TCP     74.125.228.0/24[80  -  80]      10.0.0.1/32[0   -65535]

2      PASS    TCP        10.0.0.1/32[0   -65535]    74.125.228.0/24[80  -  80]

1      BLOCK   ANY          0.0.0.0/32[0   -65535]       0.0.0.0/32[0   -65535]

0      BLOCK   ICMP         0.0.0.0/32[0   -65535]      10.0.0.1/32[0   -65535]
```

The rule will not be added to firewall policy engine if the priority is not within the specified range 0-15 (MAX_PRIORITY).

4.2. NAT Implementation.

The firewall supports NAT functionality which maps source port to NAT port while sending traffic from internal network to external network. The firewall allocate a free port by searching a free port in a sequentially order. The free port allocation algorithm will try to bind() sequential port, if the bind fails it will try for next sequential port, however if the bind succeeds the firewall will listen and reserved the port for network address translation proposes.

- Connection Going from Internal Network to External Network:
  - For TCP and UDP packets, NAT will changes the source port and source ip with the allocated NAT port and external facing interface IP respectively.
  - For ICMP packets NAT will change source IP with the external facing interface IP.

- Connection Going from Internal Network to External Network:
  - For TCP and UDP packets, NAT will check whether NAT port mapping exits, if mapping exits NAT will changes the destination port and destination IP with the allocated NAT port and internal facing interface IP respectively.
  - For ICMP packets NAT will change destination IP with the internal facing interface IP.

Note: Since, NAT doesn't have any NAT port mapping for ICMP packets all ICMP packets will be forwarded to internal networks.

A) ARP Handling: Since the internal network doesn't know about external network, firewall has to responds to the ARP request sent by internal network to the external network. Once, the ARP resolves, internal network will send all the packets destined towards external networks to firewall.

B) Optimized NAT port lookup :

For NAT port mapping a optimizes lookup is implemented for both the internal/external threads so that NAT port lookup can be done in O(1).

4.3. State Full Firewall:

- **TCP Flow**:

The State full firewall implementation keeps tracks of TCP state machine. The packet flow is evaluated only for the first packet in the connections. The policy evaluation results will be stored in the cache flow (hashes on destination IP). For all later subsequent flow of the same connection, the action is taken as specified in cache flow. Once the TCP closes the connection cache flow entry is removed from hash and the new connection has to be evaluated against the policy evaluation engine.

- **ICMP and UDP Flow:**

ICMP and UDP packet doesn't have state associated with them. Therefore, for the first packet policy evaluation happens and the results will be stored in cache flow hash. Later flow for the same connection will take action as specified in cache flow hash. The cache flow entries will be removed by cleaner thread (explained later) once no flow activity happens on a connections for a specified period of time.

4.4. Cleaner Thread Implementation:

State Full Firewall has implemented a cleaner thread to reclaim the memory allocated for cache flow. The cleaner thread will wakes up after certain specified time checks the time stamps for all the connection flow. If the connection is older than a certain cache flow threshold the cleaner thread will remove the connection from cache. Since the connection is removed from the cache, the NAT port mapping also will be updated and the socket connection for the reserved port will be closed. This ensures that firewall will not ran into NAT port shortage because ports are properly closed after cache flow timeout.

4.4.1 Optimize Locks Implementation (*In Progress):

The synchronization needs to be achieved when firewall access common hash cache flow table. The two threads that are operating on internal and external network will grabs locks only for hash index mapped to source ip and destination ip because the flow can affects only those cache flow. The locks are grab in increase hash index order to avoid deadlocks because of cycle. This approach improves the performance for multiple connections as the different connections can now operates and access hash cache flow simultaneously.

5. Firewall PCAP Handling

The State full firewall implementation also supports processing and applying firewall policy against specified PCAP file. In case of pcap handling firewall doesn't spawns multiple thread and

all the processing is done in single thread context. However, this functionality is still quite use full in debugging policy evaluation.
Syntax:

```
sudo ./firewall -i input.pcap -o output.pcap
```

Here, “input.pcap” is the input file on which the firewall rules processing is done and the “output.pcap” is the output pcap file generated after processing the policies specified in “rules.txt” file.

Note: Default rule file is “rules.txt”

6. References

[1] IP Checksum code taken from following link. http://www.microhowto.info/howto/calculate_an_internet_protocol_checksum_in_c.html#idp147520
[2] Network Name Spaces http://blog.scottlowe.org/2013/09/04/introducing-linux-network-namespaces/
[3] Transmission_Control_Protocol http://en.wikipedia.org/wiki/Transmission_Control_Protocol
[4] Snort Rule Language http://manual.snort.org/
[5] Firewall Diagram http://www.cs.cornell.edu/courses/CS5434/2014fa/Lecture12-nids.pdf
[6] Firewall Slides (provided by TA) https://d1b10bmlvqabco.cloudfront.net/attach/hzc0wscykmo24v/gxvxw9uyvp234h/i2zotuj9ktl3/cs5434_project_2_slides__firewall.pdf