DOS ATTACK PREVENTION

ON A JUNIPER M/T-SERIES ROUTER

1. Introduction

In this document , we intend to summarize the various denial of service attacks that a router is generally vulnerable to and the mechanisms that can be put in place on the Juniper M/T series routers to protect the device against such denial of service attacks. More specifically, it focuses on how the firewall functionality provided by the Juniper routers can be used to deal with denial of service attacks. For each type of the denial of service attacks, we have provided examples of how the "malicious" data packets can look like and how we can configure the router to protect against the specific type of attack. Finally, at the end, we have tried to consolidate all our protection mechanisms into a single firewall configuration and use it to protect the box against DOS attack.

The document is written only from the point of view of JunOS software and we can later investigate the equivalent features on JunOSe to provide the same DOS protection capabilities. Since JunOS and JunOSe have almost equivalent capabilities with respect to firewall features(with JunOSe even able to provide more advanced features), we can assume that we can provide the same or greater capabilities on the E-series boxes.

Lastly, the document touches upon the various architectural limitations and caveats that M/T series boxes has with respect to the DOS attack protection features and what we plan to do going ahead for overcoming these limitations.

1.1 Denial of service Attacks – Generic Definition

A router can only handle so much information coming into it at one time. Every machine has its limits and routers are no exceptions. If the total amount of traffic that a router CPU has to deal with it exceeds what it is capable of handling, then it becomes overloaded, and is not able to send or receive routing protocol messages and other important control plane packets in a timely manner. As a consequence, the control plane protocol states become inconsistent (eg, loss of router adjacencies, the routing information base becoming out of sync) and traffic flow through the router gets disrupted.

A denial of service attack is just as it sounds. It is when someone prevents the router or routers from servicing the network. The question is, how do they do this? As previously stated, a router can only handle so much information coming into it to be routed at a time. The attacker who launches a denial of service attack will deliberately send a large number of packets which will make their way up to the router CPU. If too much information starts coming in then the router gets overloaded and can't process the information fast enough. This leads to the scenario described earlier.

Eventually this will effectively shut down the network. The reason is because of the trickle down effect. Once the main routers start to get overloaded they start to send messages to the rest of the network that the connection is full, or are unable to send timely updates to the neighboring routers at all. The attacker can use methods so that this process cascades through the entire network until all the pathways in the network are full and nobody can communicate with any server on the network. Ultimately, what happens is this slows the network down to the point where nobody can access it.

There is another term that we commonly come across. This is distributed denial of service attacks, abbreviated as DDOS. Initially, the hackers sent the enormous amount of information from a single source. Routers used a safeguard mechanism where they identified a DOS attack if large no of packets were coming from a single source and rejected these as malicious packets.To circumvent around this, hackers began to send this enormous amount of information from multiple computers or IP addresses, so that the routers would have no way of knowing that a denial of service attack was in progress. This is called a distributed DOS attack.

1.2 Juniper Forwarding Model and DOS Attacks in context of Juniper M/T series routers

In case of Juniper M/T series routers, basic packet forwarding happens completely in hardware. The router consists of a routing engine(RE) where all the major daemons(including the routing daemons) run , a PFE CPU with a microkernel which receives updates from RE and programs the hardware for forwarding packets, and the PFE hardware data path which contains the forwarding functionality. With this model, only the L2/L3 control plane protocol packets and some special packets have to be processed by the RE CPU.

Since the RE CPU does not handle general packet forwarding, hence the RE CPU cannot be overwhelmed by packets that are supposed to be forwarded by the router. DOS attacks can only be launched against Juniper routers by sending a large number of locally destined packets(either control plane packets or special packets) to the router that go to the RE for processing.

1.3 Firewall Functionality usage for DOS Attack Prevention

Firewall filters and policers available in JunOS can be used to discard or rate-limit certain types of traffic that are destined to the router so that they are not able to overwhelm and bring down the router CPU.

Now, as per the Juniper forwarding architecture , the sequence of packet parsing and lookups that a RE-bound packet will undergo can be generically described as follows :

Generic → IFL → IFF → Ingress → Route → Nexthop → lo0.0 →PFE → RE
Packet Parsing Lookup Processing Entry Point Lookup Features (RPF/PDP/ Ingress IFF Filter/ RTT Filter) Lookup for local iff (NH Type : Local) CPU CPU

Please note the following points here :

- The nexthop for the loopback interface is automatically created and is used for all locally destined packets (i.e whose IP addresses match any of the addresses configured on the box) to send the packets up.
- The PFE CPU is just used for relaying the traffic(control plane packets and special packets) to the CPU via TTP channel. Only few types of packets(like ICMP echo requests and ICMP error processing happens at PFE).
- Special Packets(IP Options, TTL Expired Packets) ignore the route lookup result. They bypass the route lookup result and are sent up to the RE via PFE CPU.

Now, in order to prevent / limit certain traffic from reaching the CPU, we need to put a filter at the loopback interface that will process all CPU bound packets befoew the packet is punted up. The sequence of lookups will more or less look like this :

Generic → IFL → IFF → Ingress → Route → Nexthop → lo0.0 →lo0.0 → PFE → RE
Packet Parsing Lookup Processing Entry Point Lookup Features (RPF/PDP/ Ingress IFF Filter/ RTT Filter) Lookup for local iff (NH Type : Local) Input Filter Lookup CPU

The local nexthop for the IFF will point to the filter applied to the loopback interface. Thus, every locally destined packet, on its way to the CPU has to undergo lookup at this filter. As a consequence, we can easily use this filter to discard or rate limit packets at the hardware data plane level even before packet gets chance to reach the RE and prevent DOS attacks.

As mentioned before, for the special packets, the lookup results in the hardware data path are ignored. So, for taking care of these , we may have to apply filters at the input IFF.

Please note that the forwarding model described above applied to all the old generation platforms in M/T series. However, some of the new generation platforms like Neo are supposed to use a different forwarding architecture and we can separately look at the DOS attack prevention solution for such platforms.

Besides the filter configuration at the loopback interface, there are other features like unicast RPF checks , ifl policers for interfaces that may be used to prevent DOS attacks. Also, there are implicit filters that are configured on the box that are intended to prevent DOS attacks.

In this document, we have given examples of DOS attack prevention with only IPv4 family filters. However, similar configuration can be applied for the IPv6 traffic family also to prevent DOS attacks using RE-bound IPv6 packets.

2. Types of DOS Attacks and Prevention Strategies

2.1 Rate Limiting ARP packets

ARP Request packets are sent to broadcast MAC addresses while ARP replies are sent to the router MAC. Sometimes, there may be improper configuration in the network causing broadcast storms and lot of ARP packets might be received by the router. Also, sometimes it is possible that a customer facing port receives lot of ARP request and reply packets as part of a DOS attack. Without a protection mechanism, these will all be sent to the RE and the router will waste precious CPU cycles in trying to learn MAC addresses or resolve ARP-s.

Juniper routers have a default policer for ARP packets that is initialized at router startup time. This prevents the router from being swamped by malicious ARP requests or ARP reply packets that are received on some interface. This policer is shared by all interfaces on a linecard and rate limits the aggregate ARP traffic to a value of about 150 Kbps. However, if we want a different rate-limiting enabled for the ARP traffic, we can configure it explicitly on a per interface basis.

```
firewall {
    policer ARP-Policer {
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }
}

interfaces {
    ge-0/0/0 {
        unit 0 {
            family inet {
                policer {
                    arp ARP-Policer;
                }
            }
        }
    }
}
```

Also, we have a provision of changing the bandwidth limit and burst size of the default ARP policer that is applied on the interfaces. This helps us in doing more aggressive rate-limiting of the ARP packets on the box. Please note that these commands are hidden from the CLI and the configuration comes into effect only on reloading the configuration on the box.

```
firewall {
    arp-bandwidth-limit 32k;
    arp-burst-size-limit 2k;
}
```

2.2 Blocking Packets with Martian Addresses

Martian addresses are host or network addresses about which all routing information is ignored. They commonly are sent by improperly configured systems on the network and have destination addresses that are obviously invalid.

In IPv4, the following are the default martian addresses:

- 0.0.0.0/8
- 127.0.0.0/8
- 128.0.0.0/16
- 191.255.0.0/16
- 192.0.0.0/24
- 223.255.255.0/24
- 240.0.0.0/8

Whenever the router receives packets from these addresses, it should drop the packets as they are most likely to be malformed packets originated as part of a DOS attack. We can use a firewall filter rule to discard such packet.

The firewall configuration would look like this :

```
firewall {
   family inet {
      filter DOS-Protect {
         :
         :
         term Martian-Address-Discard {
            from {
               address {
                  0.0.0.0/8;
                  128.0.0.0/16;
                  127.0.0.0/8;
                  191.255.0.0/16;
                  192.0.0.0/24;
                  223.255.255.0/24;
                  240.0.0.0/4;
               }
            }
            then {
               discard;
            }
         }
         :
         :
      }
   }
}
```

2.3 Protecting against malicious control plane packets

All the control plane packets, which chiefly include the routing and signalling protocol packets, are sent up to the RE for processing. A router may be subject to DOS attacks by sending large number of such packets which go up to the router and are then processed by the RE. Such control packets generated by a DOS attacker will contain either invalid information that will be discarded by RE, or may even contain malicious information that can cause router's forwarding information to get corrupted.

One way to prevent DOS attacks by malicious control plane packets is by allowing packets to go up to the router only when the source of the packet is a well-known peer address for the control protocol. Only packets from well known BGP/LDP/RSVP peers, or well-known OSPF neighbors

are accepted , while the rest of the packets are dropped. This prevents unknown attackers from injecting routing protocol PDU-s or signalling protocol PDU-s into the network.

Some example configurations are shown here :

Allowing BGP/OSPF/RSVP/LDP Packets from only trusted peers

```
policy-options {
    prefix-list bgp-peer-addresses {
        178.120.10.104/30;
        178.153.111.112/28;
        181.112.10.84/30;
        200.57.21.12/30;
        200.117.183.128/28;
        201.10.12.48/28;
        201.10.15.20/30;
    }
    prefix-list ospf-peer-addresses {
        176.115.20.20/30;
        176.115.21.24/30;
        176.115.21.112/30;
        176.115.23.44/30;
        176.115.23.100/30;
        176.115.23.128/30;
        176.115.24.16/30;
        176.115.25.24/30;
   }
    prefix-list rsvp-peer-addresses {
        176.115.20.20/30;
        176.115.21.24/30;
        176.115.21.112/30;
        176.115.23.44/30;
        176.115.23.100/30;
        176.115.23.128/30;
        176.115.24.16/30;
        176.115.25.24/30;
        178.120.10.104/30;
        178.153.111.112/28;
   }
   prefix-list ldp-peer-addresses {
        178.120.10.104/30;
        178.153.111.112/28;
        181.112.10.84/30;
        200.57.21.12/30;
        200.117.183.128/28;
    }
}

firewall {
    family inet {
        filter DOS-Protect {
            :
            :
             /* Term bgp: Accept valid BGP peers. */
            term bgp {
                from {
```

```
                source-prefix-list {
                    bgp-peer-addresses;
                }
                protocol tcp;
                port bgp;
            }
            then accept;
        }
        /* Term ldp: Accept valid LDP peers. */
        term ldp {
            from {
                source-prefix-list {
                    ldp-peer-addresses;
                }
                protocol [ tcp udp ];
                port ldp;
            }
            then accept;
        }
        /* Term rsvp: Accept valid RSVP sessions. */
        term rsvp {
            from {
                source-prefix-list {
                    rsvp-peer-addresses;
                }
                protocol rsvp;
            }
            then accept;
        }
        /* Term ospf: Accept valid OSPF neighbors. */
        term ospf {
            from {
                source-prefix-list {
                    ospf-peer-addresses;
                }
                protocol [ ospf igmp ];
            }
            then accept;
        }
        :
        :
        :
    }
  }
}
```

In the above configuration, we should accept packets from all the routing/signalling protocol peers in the network, and each of them should be added to the appropriate prefix list. Also, similar rules should be applied for each control protocol configured on the box, like BFD, VRRP , RIP, PIM, IGMP and so on.

In addition to the accept action, we may seek to count the accepted control protocol packets of each type that are destined to the box. This may help either for accounting purposes or sometimes in debugging a problem in the network. In that case, our configuration should contain terms like :

```
firewall {
    family inet {
        filter DOS-Protect {
            :
            :
             /* Term bgp: Accept valid BGP peers. */
            term bgp {
                from {
                    source-prefix-list {
                        bgp-peer-addresses;
                    }
                    protocol tcp;
                    port bgp;
                }
                then {
                    accept;
                    count BGP-Count;
                }
            }
            :
            :
        }
    }
}
```

Sometimes, if we know that we will accept the control-plane packets on only certain well known interfaces , we can also apply the interface-set configuration along with this to ensure that the packets come only on interfaces on which we expect routing protocol neighbors to be present.

For example, let us assume that we have 10 network facing interfaces where BGP packets are received. In that case, we can configure something like :

```
firewall {
    interface-set BGP-Interfaces {
        ge-0/0/0.0;
        ge-0/0/1.0;
        ge-0/0/2.0;
        ge-0/0/3.0;
        ge-0/1/0.0;
        ge-0/1/1.0;
        ge-0/3/0.0;
        ge-0/3/1.0;
        ge-0/3/2.0;
        ge-0/3/3.0;
        ge-1/0/0.0;
        ge-1/0/1.0;
    }
    family inet {
        filter DOS-Protect {
            :
            :
             /* Term bgp: Accept valid BGP peers. */
            term bgp {
                from {
                    source-prefix-list {
                        bgp-peer-addresses;
```

```
                }
                protocol tcp;
                port bgp;
                interface-set BGP-Interfaces;
            }
            then accept;
        }
        :
        :
    }
  }
}
```

We can also use interface-group instead of interface-set , but interface-group configuration might not work with SCU/DCU configured.

2.4 Protecting against TCP syn/rst attacks

A very common DOS attack experienced by routers is the TCP SYN/RST attack. Here a lot of malicious TCP SYN packets or RST packets are received by a router and are sent up to the RE for processing. Now, on receiving a TCP SYN packet, the RE will attempt to complete a TCP 3-way handshake, and will queue up the TCP SYN packets in its protocol control blocks as pending connections . Similarly, when it receives a packet with RST/FIN , it will queue them up for processing and later try to look up in the database of the existing connections to reset/terminate the connection. The TCP SYN /RSTattacks thus use up the precious CPU resources like CPU memory as well as the CPU time. We must have a firewall filter rule that rate-limits the TCP SYN packets going up to the router.

The filter rule should be configured as below :

```
firewall {
   policer TCP-SYN-Policer {
      if-exceeding {
         bandwidth-limit 8k;
         burst-size-limit 1500;
      }
      then discard;
   }

   family inet {
      filter DOS-Protect {
         :
         :
         /* Term tcp-syn-fin-limit: Rate limit TCP packets with SYN/FIN/RST flags. */
         term tcp-syn-fin-limit {
            from {
               protocol tcp;
               port [bgp ldp snmp snmp-trap telnet ftp ftp-data ssh];
               tcp-flags "syn | fin | rst";
            }
            then policer TCP-SYN-Policer;
         }
         :
         :
      }
```

```
    }
}
```

Please note that in this term, in addition to the flags, we mention the port numbers. This is not compulsory, but it means that packets on only a set of well-known TCP ports (for which the router has a server or client running) would be allowed into the box. The remaining of the TCP packets would be discarded by a default rule with discard/reject action.

2.5 Protecting against IP Option attacks

Another potential source of denial-of-service-attacks would be the packets with IP Options. Since the router hardware does not have the capability to parse and process the IP Options fields in the PFE hardware, these are sent up to the RE for processing. Therefore, if due to a DOS attack, too many ip-option packets are received by RE, these might waste CPU cycles and cause some of the more legitimate control-protocol packets to be dropped.

To guard against this, the IP-Option packets going up to the RE need to be rate-limited using a filter rule configured for this purpose.

The configuration for this is shown below :

```
firewall {
    policer IP-OPT-Policer {
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }

    family inet {
        filter DOS-Protect {
            :
            :
            /* Term rsvp-option : Allow RSVP packets with RouterAlert option from trusted peers */
            term rsvp-option {
                from {
                    source-prefix-list {
                        rsvp-peer-addresses;
                    }
                    protocol rsvp;
                    ip-option any;
                }
                then accept;
            }
             /* Term ip-option-limit: Rate limit IP Option packets. */
             term ip-option-limit {
                 from ip-options any;
                 then policer IP-OPT-Policer;
             }
             :
             :
         }
    }
}
```

One point to note in this context is that the term rsvp-option must be placed in the filter before this term. This will ensure that the RSVP packets from trusted peers with Router-Alert option hit the previous rule and is not rate-limited as per this rule.

2.6 Protecting against IP Fragmentation attacks

Fragmented IP Packets can be used to launch DOS attacks on a router. Often it may happen that a router receives fragmented packets from some malicious source. The IP stack in the RE will queue up these packets for reassembly before it actually sends it to the upper layer protocol for processing. In this process, quite a lot of packets may remain buffered and the CPU might spend valuable time trying to reassemble the packets, thus impacting the RE CPU performance. To prevent this, as in the case of the IP Option or TCP SYN packets, fragmented packets must also be rate-limited before they go to the CPU.

The configuration for this is shown below :

```
firewall {
    policer IP-FRAG-Policer {
        filter-specific;
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }

    family inet {
        filter DOS-Protect {
            :
            :
            /* Term ip-fragments-limit: Rate limit IP fragment packets. */
            term ip-fragments-limit-1 {
                from first-fragment;
                then policer IP-FRAG-Policer;
            }
            term ip-fragments-limit-2 {
                from fragment-offset 64-65535;
                then policer IP-FRAG-Policer;
            }
            :
            :
        }
    }
}
```

The first rule rate limits the first fragmented packet(with fragment offset 0 and MF set) while the other rule matches the non-first-fragments. Note that the policer used is filter-specific so that both rules use a common policer to rate limit the packets.

2.7 Protecting against malicious OAM packets

There are various types of OAM traffic that a router handles. These are used to check the status of the network resources and verify reachability, error reporting etc. Chief of the OAM packets

seen in a network are the ICMP-s, the traceroutes and MPLS/L2 OAM packets. These packets are usually destined to a local address and go up to the RE for processing.

A hacker may launch a DOS attack by sending too many ICMP Pings or traceroutes to a router. Also, one common mechanism used by attackers to learn our network addresses is by sending pings to the routers and then using the learnt address to launch some other attack. Therefore we should not respond to the ICMP pings or other types of OAM packets sent by any user. To protect against such a attack, we must process such packets only if they are from trusted peer addresses.

As far as ICMP is concerned, we may also choose to process only certain types of ICMP packets and ignore the others. All this is taken care of by configuring the firewall filter rule appropriately.

The example configuration is shown below :

```
firewall {
    policer Traceroute-Policer {
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }

    family inet {
        filter DOS-Protect {
            :
            :
             /* Term accept-icmp: Accept allowed ICMP packets from trusted networks. */
            term accept-icmp {
                from {
                    source-address {
                        10.0.0.0/8;
                        172.17.0.0/16;
                        172.18.0.0/16;
                        176.115.20.20/30;
                        176.115.21.24/30;
                        176.115.21.112/30;
                        176.115.23.44/30;
                        176.115.23.100/30;
                        176.115.23.128/30;
                        176.115.24.16/30;
                        176.115.25.24/30;
                        178.120.10.104/30;
                        178.153.111.112/28;
                        181.112.10.84/30;
                        200.57.21.12/30;
                        200.117.183.128/28;
                        201.10.12.48/28;
                        201.10.15.20/30;
                    }
                    protocol icmp;
                    icmp-type [ echo-request echo-reply unreachable redirect parameter-problem ];
                }
                then {
                    count icmp-count;
```

```
                accept;
            }
        }
        /* Term allow-mpls-ping: Accept MPLS ping packets from trusted networks */
        term allow-mpls-ping {
            from {
                source-address {
                    178.120.10.104/30;
                    178.153.111.112/28;
                    181.112.10.84/30;
                    200.57.21.12/30;
                    200.117.183.128/28;
                }
                protocol udp;
                port 3503;
            }
            then {
                count mpls-ping;
                accept;
            }
        }
        /* Term traceroute-limit: Allow traceroute packets from trusted sources with rate limiting */
        term traceroute-limit {
            from {
                source-prefix-list {
                    Trusted-Addresses;
                }
                protocol udp;
                destination-port 33434-33524;
            }
            then {
                policer Traceroute-Policer;
                accept;
            }
        }
        :
        :
    }
  }
}
```

If necessary, we may also rate-limit the OAM traffic as in case of the traceroute traffic.

2.8 Dealing with management traffic

In many customer scenario-s, we do in-band management and hence management traffic destined to the router is also received on the network interfaces. This includes telnet, ssh, DNS, ftp , snmp, ntp and other types of traffic. DOS attackers may use this to send large amount of telnet/ftp traffic and unnecessarily waste router resources while denying them to more important and legitimate traffic. A good way to prevent such a scenario is to configure appropriate filter terms with policer action.

Some example configuration is shown here :

```
firewall {
    :
    :
    family inet {
        filter DOS-Protect {
            :
            :
            term telnet-ssh {
                from {
                    source-prefix-list {
                        telnet-ssh-addresses;
                    }
                    protocol tcp;
                    port [ telnet ssh ];
                }
                then {
                    policer telnet-ssh-policer;
                    accept;
                }
            }
            term snmp {
                from {
                    source-prefix-list {
                        snmp-addresses;
                    }
                    protocol udp;
                    port [snmp snmptrap];
                }
                then {
                    policer snmp-policer;
                    accept;
                }
            }
            term dns {
                from {
                    source-prefix-list {
                        dns-addresses;
                    }
                    protocol udp;
                    source-port domain;
                }
                then {
                    policer dns-policer;
                    accept;
                }
            }
            :
            :
        }
    }
}
```

In the above configuration, the prefix-lists snmp-addresses, telnet-ssh-addresses and dns-addresses , and the policers associated with each term have to be configured accordingly.

2.9 Protecting against BUM traffic for L2 networks

In case of a Juniper MX-series router, whenever the box is forwarding the packets, there is chance of DOS attacks using broadcast packets and multicast packets that generally go up to the CPU. We can avoid that using some kind of rule like :

```
firewall {
   family bridge {
      filter L2-DOS-Protect {
         :
         :
         /* Term broadcast-limit : Rate-limit L2 broadcast packet-s */
         term broadcast-limit {
            from {
               destination-mac-address {
                  ff:ff:ff:ff:ff:ff/48;
               }
             }
             then policer L2-Bcast-Policer;
         }
         /* Term broadcast-limit : Rate-limit L2 broadcast packet-s */
         term broadcast-limit {
            from {
               destination-mac-address {
                  01:00:5e:00:00:00/24;
               }
            }
            then policer L2-Mcast-Policer;
         }
         :
         :
      }
   }
}
```

Another type of packet is unknown unicast. These are packets whose destination MAC-s are not known and need to be flooded. However, in Juniper architecture, this flooding happens in hardware and has no implication on the CPU load. However, if the L2 source MAC is unknown, they generally need to be sent up to the RE for processing and causes CPU performance degradation. However, in Juniper MX routers, both the Ezchip as well as the PFE hardware rate limit the MAC learn packets (MAC-limiting enabled by default) , which protects the router against any such DOS attack.

Similarly, for all M/T-series routers, for the VPLS path too, the packets that need to be sent to the RE because of the unlearnt MAC or special MAC addresses (flood MAC etc) , are rate-limited by a policer that sits in the forwarding path of such a packet. This protects us from DOS attacks.

2.10 Concepts of trusted and untrusted interfaces and networks

We have already come across the concept of trusted networks , as defined by the prefixes (prefix-list under the policy-options configuration or the source-ip-address list attached in a term) that are allowed to send certain types of RE-bound traffic to the routers. The router considers traffic coming from these addresses as legitimate traffic that is forwarded to the RE. Trusted networks, as defined by the prefix-lists have been used in the control protocol filtering as shown in section 2.3.

Similarly, there is a concept of trusted and untrusted interfaces. Packets coming from certain interfaces(which might involve a connection to a local device managing the network) may be considered safe and all traffic from the interface can be allowed to bypass filter checks.

Thus we may have a rule like :

```
firewall {
   interface-set Trusted-Interfaces {
      fe-5/0/0.0;
      fe-5/0/1.0;
      fe-5/0/2.0;
   }

   :
   :
   family inet {
      filter DOS-Protect {
         /* Term allow-trusted-port : Allow traffic on trusted interfaces */
          term allow-trusted-port {
             from {
                interface-set Trusted-Interfaces;
             }
             then {
                accept;
             }
          }
          :
          :
      }
   }
}
```

However, we must be very careful about treating some interface as a trusted interface.

2.11 Usage of Unicast RPF for DOS attack protection

This is a very powerful feature that prevents spoofed packets from being injected into the network. Often, DOS attackers use a range of spoofed IP addresses and send them destined to the router. These packets may ultimately get processed by the CPU and cause performance issues in the network.

RPF, or Reverse Path Forwarding Check, checks whether the IP packet arrives on a interface on which the route corresponding to its source address has been learnt. Or, in other words, if a packet with a address A.B.C.D comes on an interface I, then the packet is dropped unless we have a route for the address A.B.C.D and the best matching route for the prefix A.B.C.D has a exit interface as I. Therefore, a spoofed packet is detected as arriving with a source IP which was not learnt on the interface at which it arrived and gets dropped.

When unicast RPF feature is enabled at the edges of a network, (at the PE and the CE routers) at the customer facing ports , then it reduces the probability of spoofed packets from entering the network at all.

Unicast RPF check is enabled on a per interface basis. The relevant router configuration for enabling unicast RPF checks is as follows :

```
interfaces {
    ge-0/0/0 {
        unit 0 {
            family inet {
                rpf-check;
                address 201.21.10.33/28;
            }
        }
    }
}
```

2.12 Protecting the router from Layer 2 DOS attacks

Sometimes, on the MX routers, which run L2 control protocols and implement L2 switching functionality, we might be subject to DOS attacks using spoofed L2 control protocol packets. To protect the box against such attacks, we must configure a firewall filter for the bridge family and apply it on the appropriate interfaces.

An example configuration of such a filter is shown here :

```
firewall {
    policer LACP-Policer {
        if-exceeding {
            bandwidth-limit 10k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer OAM-Policer {
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer PPP-Policer {
        if-exceeding {
            bandwidth-limit 20k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer STP-Policer {
        if-exceeding {
            bandwidth-limit 48k;
            burst-size-limit 1500;
        }
        then discard;
    }
    family bridge {
        filter L2-DOS-Protect {
            :
            :
            /* Term stp-rstp-limit : Rate-limit STP and RSTP BPDU-s */
```

```
            term stp-rstp-limit {
                from {
                    destination-mac-address {
                        80:01:43:00:00:00/48;
                    }
                }
                then policer STP-Policer;
            }
            /* Term lacp-limit : Rate-limit LACP BPDU-s */
            term lacp-limit {
                from {
                    ether-type 0x8808;
                    destination-mac-address {
                        01:80:c2:00:00:01/48;
                    }
                }
                then policer LACP-Policer;
            }
             /* Term ppp-limit : Rate-limit PPP packets */
            term ppp-limit {
                from {
                    ether-type ppp;
                }
                then policer PPP-Policer;
            }
            /* Term ether-oam-limit : Rate-limit Ethernet OAM packets */
            term ether-oam-limit {
                from {
                    ether-type oam;
                }
                then policer OAM-Policer;
            }
            :
            :
        }
    }
}
```

There might be some other L2 protocols like L2-BFD, CLNS (for the ISIS packets) for which we may put similar policing configuration.

2.13 Consolidated Firewall Configuration for DOS Attack Prevention

In this section, we have tried to consolidate all the firewall configuration for DOS attack prevention on a router and given an example configuration. The model followed by this configuration is that we have a filter to permit only ceratin types of packets (sometimes with rate-limiting) selectively based on some criterion and drop all the remaining traffic destined to the RE.

```
policy-options {
    prefix-list bgp-peer-addresses {
        178.120.10.104/30;
        178.153.111.112/28;
        181.112.10.84/30;
        200.57.21.12/30;
```

```
    200.117.183.128/28;
    201.10.12.48/28;
    201.10.15.20/30;
 }
 prefix-list ospf-peer-addresses {
    176.115.20.20/30;
    176.115.21.24/30;
    176.115.21.112/30;
    176.115.23.44/30;
    176.115.23.100/30;
    176.115.23.128/30;
    176.115.24.16/30;
    176.115.25.24/30;
}
 prefix-list rsvp-peer-addresses {
    176.115.20.20/30;
    176.115.21.24/30;
    176.115.21.112/30;
    176.115.23.44/30;
    176.115.23.100/30;
    176.115.23.128/30;
    176.115.24.16/30;
    176.115.25.24/30;
    178.120.10.104/30;
    178.153.111.112/28;
}
prefix-list ldp-peer-addresses {
    178.120.10.104/30;
    178.153.111.112/28;
    181.112.10.84/30;
    200.57.21.12/30;
    200.117.183.128/28;
 }
 prefix-list telnet-ssh-addresses {
    172.17.12.16/28;
    172.17.33.0/28;
    172.17.35.96/30;
    172.17.91.56/30;
 }
 prefix-list snmp-addresses {
    172.17.12.17;
    172.17.12..23;
    172.17.33.2;
    172.17.33.5;
    172.17.35.99;
    172.17.61.17;
    172.17.69.25;
    172.17.91.56;
 }
 prefix-list dns-addresses {
    172.17.12.19;
    172.17.33.5;
    172.17.61.11;
    172.17.69.21;
    172.17.91.57;
 }
```

```
}

firewall {
    policer dns-policer {
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer snmp-policer {
        if-exceeding {
            bandwidth-limit 12k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer telnet-ssh-policer {
        if-exceeding {
            bandwidth-limit 15k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer ARP-Policer {
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer IP-FRAG-Policer {
        filter-specific;
        if-exceeding {
            bandwidth-limit 10k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer IP-OPT-Policer {
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer Traceroute-Policer {
        if-exceeding {
            bandwidth-limit 8k;
            burst-size-limit 1500;
        }
        then discard;
    }
    policer TCP-SYN-Policer {
        if-exceeding {
            bandwidth-limit 8k;
```

```
        burst-size-limit 1500;
    }
    then discard;
}
interface-set Trusted-Interfaces {
    fe-5/0/0.0;
    fe-5/0/1.0;
    fe-5/0/2.0;
}
interface-set BGP-Interfaces {
    ge-0/0/0.0;
    ge-0/0/1.0;
    ge-0/0/2.0;
    ge-0/0/3.0;
    ge-0/1/0.0;
    ge-0/1/1.0;
    ge-0/3/0.0;
    ge-0/3/1.0;
    ge-0/3/2.0;
    ge-0/3/3.0;
    ge-1/0/0.0;
    ge-1/0/1.0;
}
family inet {
    filter DOS-Protect {
       /* Term Martian-Address-Discard : Discard packets from martian addresses */
       term Martian-Address-Discard {
           from {
              address {
                 0.0.0.0/8;
                 128.0.0.0/16;
                 127.0.0.0/8;
                 191.255.0.0/16;
                 192.0.0.0/24;
                 223.255.255.0/24;
                 240.0.0.0/4;
              }
           }
           then {
              discard;
           }
        }
        /* Term allow-trusted-port : Allow traffic on trusted interfaces */
        term allow-trusted-port {
           from {
              interface-set Trusted-Interfaces;
           }
           then {
              accept;
           }
        }
        /* Term ip-fragments-limit: Rate limit IP fragment packets. */
        term ip-fragments-limit-1 {
           from first-fragment;
           then policer IP-FRAG-Policer;
        }
```

```
 term ip-fragments-limit-2 {
     from fragment-offset 64-65535;
     then policer IP-FRAG-Policer;
 }
/* Term rsvp-option : Allow RSVP packets with RouterAlert option from trusted peers */
term rsvp-option {
    from {
        source-prefix-list {
            rsvp-peer-addresses;
        }
        protocol rsvp;
        ip-option any;
    }
    then accept;
}
 /* Term ip-option-limit: Rate limit IP Option packets. */
 term ip-option-limit {
     from ip-options any;
     then policer IP-OPT-Policer;
 }
 /* Term tcp-syn-fin-limit: Rate limit TCP packets with SYN/FIN/RST flags. */
 term tcp-syn-fin-limit {
     from {
         protocol tcp;
         port [bgp ldp snmp snmp-trap telnet ftp ftp-data ssh];
         tcp-flags "syn | fin | rst";
     }
     then policer TCP-SYN-Policer;
 }
 /* Term bgp: Accept valid BGP peers. */
 term bgp {
     from {
         source-prefix-list {
             bgp-peer-addresses;
         }
         interface-set BGP-Interfaces;
         protocol tcp;
         port bgp;
     }
     then accept;
 }
 /* Term ldp: Accept valid LDP peers. */
 term ldp {
     from {
         source-prefix-list {
             ldp-peer-addresses;
         }
         protocol [ tcp udp ];
         port ldp;
     }
     then accept;
 }
 /* Term rsvp: Accept valid RSVP sessions. */
 term rsvp {
     from {
         source-prefix-list {
```

```
            rsvp-peer-addresses;
        }
        protocol rsvp;
    }
    then accept;
}
/* Term ospf: Accept valid OSPF neighbors. */
term ospf {
    from {
        source-prefix-list {
            ospf-peer-addresses;
        }
        protocol [ ospf igmp ];
    }
    then accept;
}

/* Term accept-icmp: Accept allowed ICMP packets from trusted networks. */
term accept-icmp {
    from {
        source-address {
            10.0.0.0/8;
            172.17.0.0/16;
            172.18.0.0/16;
            176.115.20.20/30;
            176.115.21.24/30;
            176.115.21.112/30;
            176.115.23.44/30;
            176.115.23.100/30;
            176.115.23.128/30;
            176.115.24.16/30;
            176.115.25.24/30;
            178.120.10.104/30;
            178.153.111.112/28;
            181.112.10.84/30;
            200.57.21.12/30;
            200.117.183.128/28;
            201.10.12.48/28;
            201.10.15.20/30;
        }
        protocol icmp;
        icmp-type [ echo-request echo-reply unreachable redirect parameter-problem ];
    }
    then {
        count icmp-count;
        accept;
    }
}
/* Term allow-mpls-ping: Accept MPLS ping packets from trusted networks */
term allow-mpls-ping {
    from {
        source-address {
            178.120.10.104/30;
            178.153.111.112/28;
            181.112.10.84/30;
            200.57.21.12/30;
```

```
            200.117.183.128/28;
        }
        protocol udp;
        port 3503;
    }
    then {
        count mpls-ping;
        accept;
    }
}
/* Term traceroute-limit: Allow traceroute packets from trusted sources with rate limiting */
term traceroute-limit {
    from {
        source-prefix-list {
            Trusted-Addresses;
        }
        protocol udp;
        destination-port 33434-33524;
    }
    then {
        policer Traceroute-Policer;
        accept;
    }
}
/* Term telnet-ssh : Rate limit telnet and ssh traffic */
term telnet-ssh {
    from {
        source-prefix-list {
            telnet-ssh-addresses;
        }
        protocol tcp;
        port [ telnet ssh ];
    }
    then {
        policer telnet-ssh-policer;
        accept;
    }
}
/* Term snmp : Rate limit snmp traffic */
term snmp {
    from {
        source-prefix-list {
            snmp-addresses;
        }
        protocol udp;
        port [snmp snmptrap];
    }
    then {
        policer snmp-policer;
        accept;
    }
}
/* Term snmp : Rate limit dns traffic */
term dns {
    from {
        source-prefix-list {
```

```
                dns-addresses;
            }
            protocol udp;
            source-port domain;
        }
        then {
            policer dns-policer;
            accept;
        }
    }
    /* Term default : Default term to discard all other packets */
    term default {
        then {
            discard;
            log;
        }
    }
  }
 }
}
interfaces {
    lo0 {
        unit 0 {
            family inet {
                filter {
                    input Test;
                }
                address 100.100.100.100/32;
            }
        }
    }
}
```