Application Note

JUNOS Router Security

Best Common Practices for Hardening the Infrastructure

Pejhan Peymani
Systems Engineer

Matt Kolon
Marketing Engineer

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
408 745 2000 or 888 JUNIPER
www.juniper.net

Part Number: 350013-003 0 5/03

Contents

Executive Summary 4
Perspective 4
Router Security 4
JUNOS Default Settings 5
Router Access Security 6
    Out-of-Band Management 6
    Inband Management 6
Communicating with the Router 7
    Secure Shell 7
    Secure Copy Protocol 8
Central Authentication 8
Routing Protocol Security 10
Filtering Inbound Traffic to the Routing Engine 11
    Applying Firewall Filters to Routing Engine Traffic 11
    Designing a Firewall Filter 11
    Protecting Against ICMP and SYN Floods 12
    Protecting Against Malicious Fragments 13
    Protecting Against Packets with IP Options 14
    Preventing Routers from Being Used as Attack Reflectors 16
    VPN Considerations 16
        Applying Firewall Filters for Each VPN 18
    Rate-Limiting Traffic to the Routing Engine 18
Auditing for Security 19
    Logging Authentication and Command Events 19
    Routing Protocol Events and Errors 19
    Logging of Traffic Being Denied 20
    Network Time Protocol 20
    Conclusion 20
Appendix A 21
Appendix B 21

Executive Summary

The anonymity and flexibility of IP and the Internet make it possible for attackers to remotely degrade router performance or interrupt the network communication s of large numbers of end users. The increasing frequency of attacks targeting service provider routers makes it essential to have techniques for securing them. This document introduces common security issues concerning router configuration and operation, identifies Juniper Networks router features that can be useful in preventing and combating attacks, and outlines best common practice guidelines for the secure deployment of Juniper Networks routers.

The configuration and operation techniques discussed in this paper are not the only way to accomplish router security, nor are they required by Juniper Networks. They are rather best common practices recommended by us based on our experience with our customers and partners.

Perspective

A recent increase in reported attacks against routers and other network components has coincided with the current focus on the security of all types of public infrastructure to create an extremely challenging security environment for network service providers. A number of facets of the network security question apply to the configuration of routers and to the design of routed networks. To protect their networks from attacks by those wishing to disrupt or destroy them, service providers must take an active role in identifying and utilizing network and router design features that can enhance the security of their networks.

NOTE The practices discussed in this paper help secure the router infrastructure, which is only one part of an overall network security strategy. Network services also require host systems to manage the network, provide domain name resolution, exchange e-mails with customers, and perform other functions. While some of the techniques discussed in this paper are applicable to protecting hosts as well as routers, the specific approaches in this paper focus on router security.

Router Security

Router security can be divided into three elements:

- Physical security of the router
- Operating system security
- Configuration hardening

This paper does not provide recommendations on physical security, other than to state the rather obvious fact that restricting the physical access to any network equipment is the first step in securing it. Many exploits that are trivial to prevent from remote locations are extremely difficult to prevent if the attacker is able to access the management port or console.

The inherent security of the router operating system also plays an important role in the security of the router. If the operating system itself is not secure the router can be compromised—regardless of careful and secure configuration. For instance, one of the common techniques hackers use to compromise routers is to elicit buffer overflows by finding weaknesses in router operating system code. To prevent such exploitation, the operating system must be extremely stable and robust. Juniper Networks takes extreme care to ensure that each release of software is as stable and bug-free

as possible, but a few additional techniques and guidelines will ensure that even the rare software fault will not expose the system to attack. A secure operating system also provides features to help operators protect the system from attacks, so a knowledgeable user can minimize whatever vulnerability exists as a result of necessary configuration elements.

This paper concentrates on the third aspect of router security: configuration hardening. Configuration hardening is the process of applying sound security policies using the tools that are made available by the router operating system. Given a robust set of security tools, virtually any router configuration can operate securely. Even with such tools, it is possible to misconfigure a perfectly secure operating system and thereby make a router vulnerable.

The remainder of this paper discusses four aspects of configuration hardening:

- Router access security
- Routing protocol security
- Protecting the routing engine
- Security auditing

The security recommendations and considerations in this paper are specific to IPv4. Although many of the issues and recommendations are quite similar for IPv6, they are not explicitly discussed here.

NOTE We assume for the remainder of this paper that the reader is comfortable with JUNOS software configuration features and syntax. Make sure you understand the consequences of using the commands and configuration options described here. Improper use of them can disrupt network operation by restricting traffic flow, or making it difficult to log in to the router. For complete configuration and operational commands, please see the technical documentation at www.juniper.net/techpubs/.

JUNOS Default Settings

The JUNOS software presents a hardened target to the attacker immediately after it is installed and a root account password has been configured. The following is a list of common router concerns that the JUNOS software addresses in its default configuration.

- The JUNOS software does not forward directed broadcasts. Directed broadcast services can be used to attack other users across the Internet by sending ping requests from a spoofed source address to a broadcast address such as 200.0.0.255. If such "broadcast pings" are allowed on the 200.0.0.0/24 network, a single ping request can elicit up to 254 responses, all aimed at the supposed source of the ping—which actually becomes the victim of a denial-of-service (DoS) attack. On large-scale or multiple networks, the problem becomes even worse.
- Remote management access to the router is disabled by default and must be explicitly enabled. Only console access to the router is enabled in a default configuration. This restriction applies to all management access protocols such as telnet, ftp, and even ssh.
- The JUNOS software does not support SNMP set capability for editing configuration data. While the JUNOS software does support SNMP set capability for conducting network monitoring and troubleshooting, this exposes no known security issues and can be disabled in the configuration.
- The JUNOS software by default has a rate-limiter applied on ARP packets that go to the routing engine. This prevents an "ARP storm" attack either through misconfiguration or malicious behavior. To further rate-limit ARP messages, an ARP policer can be placed on Ethernet interfaces

- Martian addresses are reserved host or network addresses about which all routing information should be ignored. To add specific addresses to the list of default martian addresses add the following to your router configuration:

```
routing-options {
      martians {
          destination-prefix  match-type;
      }
}
```

By default, JUNOS martian addresses contain the following prefixes:

```
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
240.0.0.0/4
```

Router Access Security

Secure remote access is essential to managing and maintaining a router infrastructure. As mentioned above, the JUNOS software provides initial remote access security by disabling all remote access by default. This ensures that no remote access is possible unless deliberately turned on by an authorized user. There are two ways to establish remote communication with a router: out-of-band and inband communication.

Out-of-Band Management

Out-of-band allows connection to the router through an interface dedicated to router management. Juniper Networks routers support out-of-band management with a dedicated management Ethernet interface (fxp0), as well as with EIA-232 console and auxiliary ports. The management Ethernet interface connects directly to the Routing Engine. No transit traffic is allowed through this interface, providing complete separation of customer and management traffic, and ensuring that congestion or failures in the transit network do not affect the management of the router. Having a separate management network facilitates management of the router infrastructure even during DoS attacks or other outages.

Many major ISPs maintain an out-of-band management network in order to facilitate remote management and recovery even in the face of catastrophic failures in the primary network. While it is costly and requires considerable resources, this network provides a good separation of management control traffic and transit traffic. Note, however, that an out-of-band network can itself be compromised.

Inband Management

Inband management allows connection to the routers using the same interfaces through which customer traffic flows. While this approach is simple and requires no dedicated management resources, it has some disadvantages:

- Management and transit traffic flows are mixed together. Any attack traffic that is mixed with the normal traffic can affect the ability to communicate with the router.
- The links between routers may not be totally trustworthy, leading to the possibility of wiretapping and replay attacks.

Almost all networks need inband routing protocol traffic, which requires the same type of security measures as inband management traffic. Therefore, the remainder of this document assumes that inband management is the method by which management communication is established with the router.

Communicating with the Router

For management access, there are several ways one can communicate with a router from a remote console. The most common methods are usethe applications telnet and ssh (secure shell). For reasons outlined below, we recommend using ssh rather than telnet for console access.

Secure Shell

The secure shell provides secure encrypted communications over an insecure network and is therefore useful for inband router management. Like all other types of network-based access, however, ssh access to the router is disabled by default in the JUNOS software. Ssh can be enabled with a JUNOS configuration similar to this, which enables ssh access and sets optional parameters that can be used to control the number of concurrent ssh sessions and the maximum number of ssh sessions that can be established in one minute. The `rate-limit` parameter can be useful in protecting against SYN flood DoS attacks on the ssh port.

```
system {
   services {
      ssh connection-limit 10 rate-limit 4;
   }
}
```

When ssh is enabled, all users can use it for access to the router. Access to the root account from ssh can be restricted with the following configuration:

```
system {
   services {
      ssh {
         root-login deny;
         protocol-version  v2;
      }
   }
}
```

The `deny` option disallows all root access using ssh. Another option, `deny-password,` allows root access only when no root user password is set. The latter option can be useful for initial router access during deployment and installation.

By default, the router accepts connections for both version 1 and version 2 of ssh. To restrict the router to one version the ssh version can be configured. In the configuration shown above, access using version 1 of the protocol is not allowed.

Version 2 of ssh is generally considered more secure than version 1, because it is more deliberately designed and better documented, and therefore more widely understood. We suggest the use of

version 2 when possible.

Secure Copy Protocol

The secure copy protocol (SCP) uses the ssh encryption and authentication infrastructure to securely copy files between hosts. The JUNOS software can use scp with the `file copy` operational mode command. For example, the operational mode command:

```
file copy router.config user@remoterouter.example.com/destdir/
```

securely copies the file `router.config` from the current local directory to the `destdir` directory on `remoterouter.example.com`. We recommend using scp when possible as an alternative to ftp for copying files between routers, especially when the transit path crosses untrusted networks.

Central Authentication

The management of multiple routers by many different personnel can create a significant user account management problem. This problem can be addressed by using a central authentication service, which simplifies account management. With this mechanism, account creation and deletion are performed only on one central server. This is useful, for example, to enable or disable an employee's access to all routers with a single change that does not even require committing a configuration change to any router.

A centralized authentication system also simplifies the use of one-time password systems such as SecureID, which offer good protection against password sniffing and password replay attacks (attacks in which a captured password is used to by an attacker to pose as a router administrator.) With a one-time password, every time a legitimate user accesses the system they will use a different password string. If an attacker is able to capture the password, the system does not recognize it in subsequent login attempts.

JUNOS software supports two protocols for central authentication of users on multiple routers—Remote Authentication Dial In User Service (RADIUS) and Terminal Access Controller Access Control System Plus (TACACS+).

In choosing an authentication service protocol, we recommend RADIUS. RADIUS is a multivendor IETF standard, and its features are more widely accepted than those of TACACS+ or other proprietary systems. In addition, we recommend using a one-time-password system for increased security, and all vendors of these systems support RADIUS.

The JUNOS model for centralized authentication is essentially the same for both RADIUS and TACACS+. You create one or more *template accounts* on the router, and the users' access to the router is configured to use the template account. The following shows a configuration that supports three different access levels on two different servers:

```
system {
   authentication-order [ radius ];
   radius-server {
      10.1.2.1 {
          secret "XXXXXXXXXXXXXXXXXXXX"; # SECRET-DATA
          timeout 5;
      }
      10.1.2.2{
          secret "XXXXXXXXXXXXXXXXXXXX"; # SECRET-DATA
          timeout 5;
```

```
   }
 }
}
```

The `authentication-order` command shown above enables RADIUS authentication. Only if the RADIUS server is not reachable would the fallback mechanism (a local account set up on the router for this purpose) be used.

The two server sections above enable the RADIUS protocol and define the shared secrets between the client (the router) and the servers so they each know that they are talking to the trusted peer. A timeout is also defined for each server so if there is no response within the specified time, the router can either try the next server (in the first section) or try the next authentication mechanism (in the second section).

Then, multiple user classes can be created, each with specific privileges. We also show the use of timeouts to disconnect the class members after a period of inactivity. Users' privilege levels (and therefore the classes of which they are members) should be dependent on their responsibilities within the organization, and the permissions shown here are only examples:

```
login {
/* This class of users can only view statistics and configuration. They are not
allowed to modify any configuration. */
   class observation {
      idle-timeout 5;
      permissions [ configure firewall interface network routing
      snmp system trace view];
   }
/* This class of users can view and modify configuration. */
   class operation {
      idle-timeout 5;
      permissions [admin clear configure interface
      interface-control network reset routing routing-control snmp
      snmp-control trace-control firewall-control rollback];
   }
/* This class has the unlimited access and control. */
   class engineering {
      idle-timeout 5;
      permissions all;
   }
}
```

Finally, after defining the classes and their respective permissions, users corresponding to the classes for use by the RADIUS authentication mechanism can be defined, along with a local superuser.

Public key authentication with RSA/DSA keys is much more secure than simple password exchange. Simple passwords are often susceptible to "dictionary" and other attacks that predict or guess passwords. In addition, requiring encrypted public key authentication excludes the use of telnet for management access, which is yet another advantage over simple passwords

```
login {
/* This is the local superuser account. If RADIUS fails or becomes unreachable,
revert to using the local accounts on the router. */
   user admin {
      uid 1000;
      class engineering;
      authentication {
```

```
            ssh-dsa "XXXXXXXXXXXX"; #  Secure shell (ssh) RSA public key string

         }
      }
/* This defines RADIUS templates for the three different users or groups of
users. */
      user observation {
         uid 1001;
         class observation;
      }
      user operation {
         uid 1002;
         class operation;
      }
      user engineering {
         uid 1003;
         class engineering;
      }
}
```

For complete RADIUS configuration information, see the JUNOS configuration guides.

Routing Protocol Security

The main task of a router is to use its routing and forwarding tables to forward user traffic to its intended destination. Attackers can send forged routing protocol packets to a router with the intent of changing or corrupting the contents of its routing table or other databases, which in turn can degrade the functionality of the router and the network. To prevent such attacks, routers must ensure that they form routing protocol relationships (peering or neighboring relationships) to trusted peers. One way of doing this is by authenticating routing protocol messages. We strongly recommend using authentication when configuring routing protocols. JUNOS software supports HMAC-MD5 authentication for BGP, OSPF, IS-IS, RIP, and RSVP. HMAC-MD5 uses a secret key that is combined with the data being transmitted to compute a hash. The computed hash is transmitted along with the data. The receiver uses the matching key to recompute and validate the message hash. If an attacker has forged or modified the message, the hash will not match and the data will be discarded.

The following example shows the configuration of a single key for the BGP peer group `internalpeers`. BGP authentication can also be configured at the `neighbor` or `routing-instance` levels, or for all BGP sessions. As with any security configuration, there is a tradeoff between the degree of granularity used (and to some extent the degree of security) and the amount of management necessary to maintain the system.

```
protocols {
   bgp {
      group internalpeers {
         authentication-key XXXXXXX;
      }
   }
}
```

Although all JUNOS IGPs support authentication, some IGPs are inherently more secure than others. Most service providers use OSPF or IS-IS to allow fast internal convergence, scalability and use of traffic engineering capabilities with MPLS. Because IS-IS does not operate at the network layer, it is more difficult to spoof than OSPF, which is encapsulated in IP and is therefore subject to remote

spoofing and DOS attacks.

Filtering Inbound Traffic to the Routing Engine

In the previous section, use of authentication was discussed to make sure that routers form routing relationships only with trusted peers. Although this helps protect routing protocols, it still does not completely prevent malicious or untrusted packets from reaching a particular process on the Routing Engine (RE). For example, an attacker could still launch an attack against a router by targeting a particular protocol with forged packets. Although these packets will fail the authentication check, the attack may still consume router resources (CPU cycles and communication queues) on the RE, therefore making it to some extent successful. In order to avoid this, only protocol and control packets from trusted sources should be allowed to reach the RE. Juniper Networks highly recommends the use of firewall filters to ensure this. Since all Juniper Networks routers implement firewalls in hardware without compromising forwarding speed, firewall filters can provide very strong and flexible protection against attacks without limiting performance.

Applying Firewall Filters to Routing Engine Traffic

To protect the Routing Engine, a firewall filter needs to be applied only to the router's loopback interface. Adding or modifying filters for every interface on the router is not necessary, which is a significant departure from equivalent procedures on legacy routers.

The following example shows how to apply the `protect-RE` firewall filter as an input filter on the lo0 interface to protect the Routing Engine. This single filter checks all traffic destined for the Routing Engine that enters the router from the customer interfaces. The configuration of the firewall filter itself is discussed in the next section.

```
interfaces {
   lo0 {
      unit 0 {
         family inet {
            filter {
               input protect-RE;
            }
            address 10.10.5.1/32;
         }
       }
    }
}
```

Designing a Firewall Filter

When constructing a firewall to admit only friendly traffic to the Routing Engine, the following factors must be taken into account:

- Services and protocols that need to be running on the router
  - Routing protocols (BGP, OSPF, RSVP, etc.)
  - Management services (SSH, DNS, NTP, etc.)
  - Diagnostic and troubleshooting protocols (ICMP, traceroute, etc.)
- What are the destination addresses and ports of the above services and protocols on the RE?

- What are the trusted source addresses of the peers with whom I intend to communicate regarding the above services?
- What is the traffic rate that can be allocated to each of the services mentioned in 1?

Once the necessary services and protocols are determined, each of them is combined with the trusted source addresses from which each service is expected to originate to create a match condition in the firewall definition. The corresponding action is to accept packets, and to discard any other traffic.

The example configuration below shows three filter terms that can be used in a firewall to protect the Routing Engine. The example configuration file in Appendix B contains a more complete listing of what such a firewall might contain. The policer definitions are omitted from the example below and are discussed later, under the section heading entitled "Rate-limiting Traffic Reaching the Routing Engine." The policer action itself, however, is shown in the following configuration.

```
firewall {
   filter protect-RE {
/* Policer configuration omitted */
      term ssh {
         from {
            source-prefix-list {
               ssh-addresses;
            }
            protocol tcp;
            port [ ssh telnet ];
            }

         then {
            policer ssh-policer;
            accept;
         }
      }
        term bgp {
            from {
                source-prefix-list {
              bgp-sessions-addresses;
                }
                protocol tcp;
                port bgp;
            }
            then accept;
        }
/* Everything else that is not trusted is silently dropped and logged for
analysis purposes. */
        term everything-else {
            then {
                 syslog;
                log;
                discard;
            }
         }
      }
   }
```

Protecting Against ICMP and SYN Floods

To protect against ICMP floods and similar attacks against the Routing Engine, we suggest rate-

limiting ICMP traffic destined for the router. An attacker can use several different types of ICMP messages to either degrade router functionality or scan machines. We therefore recommend allowing only those types of ICMP messages that are required for proper network operation and troubleshooting.

Another common form of attack is a TCP SYN flood, in which the attacker uses a script or program to create TCP connection requests (SYN messages) at a rate faster than the victim releases them. For this reason, we recommend rate-limiting TCP SYN messages. Because establishing a TCP connection requires only a three-way handshake, limiting the rate of incoming SYN packets to 500 Kbps can be safely done.

Below are listed two examples of firewall filter terms that that limit SYN and ICMP rates (in the `protect-RE` filter), as well as a term that rejects ICMP redirect messages.

```
firewall {
   filter protect-RE {
     term police-init {
         from {
            source-prefix-list {
               ssh-addresses;
               bgp-addresses;
                }
                protocol tcp;
                tcp-flags "(syn & !ack) | fin | rst ";
            }
            then {
                policer tcp-init-policer;
                accept;
            }
       }
      term icmp {
            from {
                protocol icmp;
           icmp-type [ echo-request echo-reply unreachable time-exceeded ];
            }
            then {
                policer small-policer;
                accept;
            }
         }
      }
```

The details of the rate-limiting policer statements are given below in the section “Rate Limiting Traffic Reaching the Routing Engine.” For more information about SYN and ICMP attacks, see the references listed in Appendix A.

Protecting Against Malicious Fragments

In an attack, IP fragmentation can be used to disguise TCP packets from IP filters used in routers and hosts. RFC 1858 talks about using small offsets in potential DOS attacks, it also recommends general algorithms to protect against such attacks. JUNOS provides the ability to implement the algorithms defined in the RFC.

Perhaps unknowingly, the firewall configuration that many users implement will drop fragments. Consider the following filter:

```
term bgp{
    from {
        protocol tcp;
        destination-port bgp;
        source-address <trusted peer>;
    }
    then {
        accept;
    }
term default {
    then discard;
    }
```

The first packet fragment will contain destination port (and other match criteria), and they will be accepted and allowed to pass. Subsequent fragments of the same packet will not have the destination port information, will not match the firewall term, and will be discarded (because of the default term).

If you expect to have fragmented packets destined to the RE (for example, bgp packets) then you need to add the following to your lo0 firewall filter:

```
term small-fragments {
    from {
        fragment-offset [1-5];
        }
    then {
        syslog;
        discard;
        }

term fragments {
       from {
           source-prefix-list {
               trusted-sources
                }
           is-fragment;
            }
       then {
           accept;
           }
```

The term “small-fragments” will drop any small fragments. Since fragments that small are not valid, they should be discarded. In the second term, all fragments from trusted sources are accepted.

Note: you need to deal with all fragments before you deal with any packets that have layer-4 attributes (TCP or UDP ports).

Protecting Against Packets with IP Options

IP options provide control functions needed in some special situations, but for most practical communication they are not necessary. Part of the functionality that options provide include timestamps and special routing. Options tell the transit routers to examine the packets that are not destined to them more closely, hence these optioned packets will go up to the Routing Engine for processing. A flooding of such packets could therefore overwhelm the routing engine and create a

successful denial of service condition. In a service provider environment where backbone routers simply transport customer traffic, the transit routers should not see any packets with IP options, unless they are part of control traffic that exists within the backbone only. Therefore it is a good idea to implement a filter to drop optioned packets so they won't reach the Routing Engine.

If a service provider is running MPLS in the core, then they would expect certain amount of options in control traffic. Such control traffic includes RSVP packets which have the "Router Alert" options set. Instead of discarding optioned packets, a provider may wish to rate-limit them.

Since optioned packets are not destined to the router itself, a lo0 firewall filter will not be able to intercept them. One can apply the recommended filter to an incoming interface, but the management of such a filter is cumbersome - especially for a router with many interfaces or sub-interfaces. Alternatively, one can apply a filter to the router forwarding table.

Because of the burstiness of the control traffic, an important value in the rate-limiter is the burst-size-limit. To come up with a value for the burst-size-limit for RSVP packets, one should take into account

1. The RSVP packet size
2. The number of messages received by a router during one second

The first variable depends on the traffic engineering options that are enabled on the MPLS backbone. One can look at the RSVP packet sizes by turning on logging and viewing packet size directly. The second variable depends on the number of LSPs that go through the router. A transit router will have the most RSVP messages received by it, hence we'll use the transit router scenario in calculating the burst-size-limit. Lets say 400 bytes for the RSVP packets (including the IP header). The transit router will be the worst-case scenario, because it will receive RSVP messages from both sides (assuming bidirectional LSPs) so the formula to use will be:

```
400 bytes per packet * no of LSPs * 4
```

with 200 transit LSPs this will be 320000 bytes for the burst size:

```
policer option-policer {
        if-exceeding {
                bandwidth-limit 3m;
                burst-size-limit 320000;
            }
        then discard;
```

So the filter to rate-limit the optioned packet will be:

```
filter filter-optioned {
        term one {
            from {
                 ip-options any;
             }
            then {
                 count option-packets;
                 policer option-policer;
                 }
        }
     term default {
         then accept;
        }
    }
```

Which can then be applied to the forwarding table under the forwarding-options configuration

hierarchy.

Preventing Routers from Being Used as Attack Reflectors

Routers by default will respond with ICMP "destination unreachable" error messages to the sources of packets for whom they have no route. This behavior can be exploited by attackers, who can turn routers into reflectors by spoofing requests from the victim to a large set of routers that will in turn send their combined replies to the victim. An attacker can send packets to non-existing destinations with the source address set to the victim's address and thereby flood a victim with ICMP error messages. To prevent routers to be used as reflectors one can silently discard packets where there is no destination for. One can combat reflection attacks by configuring a default route which points to the "discard" interface. This way packets with no valid route in the forwarding table will be silently discarded, and by installing a filter which logs these discards the network administrator can analyze this attack traffic. An example configuration is shown below:

First, configure the "discard" interface with a private IP address:

```
dsc {
    unit 0 {
        family inet {
            filter {
                input log-discard;
            }
            address 10.1.1.1/32 {
                destination 10.1.1.2;
            }
        }
    }
}
```

A filter is placed on the "discard" interface to monitor packets being discarded:

```
filter log-discard {
    term one {
        then {
            syslog;
            discard;
        }
    }
}
```

Set a default route so packets without a route in the forwarding table are directed to the discard interface:

```
static {
    route 0.0.0.0/0 next-hop  10.1.1.2 ;
}
```

VPN Considerations

VPNs allow service providers to offer their customers the benefits of private networks, while reaping the cost savings benefit of a common IP infrastructure. VPNs act like private networks by using private forwarding tables. The separation of the forwarding in conjunction with an MPLS transport provides a clean and secure separation of traffic across an IP backbone. But creation and

management of these forwarding tables occurs via one common control plane, and because they use the same control plane and share its resources, extra protection needs to be in place to protect VPNs from their compromised or a misbehaving peers.

The recommendations discussed here apply to each virtual network, and will be either for the CE (customer edge) equipment or the PE (provider edge) equipment.

In one scenario, a compromised host or multiple hosts belonging to one VPN could attack a PE's control plane, which could affect other VPNs on the same PE. To protect against a misbehaving VPN, one can consider a number of safety precautions:

Distribution of routes between PE and a CE can be via different routing protocols such as BGP, OSPF, and RIP. Since BGP and RIP each use a designated layer 4 port that they communicate through, it is easier for the compromised host to flood those ports from outside the network and create a denial of service attack. On the other hand, OSPF operates at layer 3, sending and receiving information as the IP payload itself, so the compromised host would need to have access to raw socket services on the PE-CE link to create spoofed OSPF packets – a much more difficult job. Therefore, when choosing a PE-CE routing protocol one might want to consider OSPF.

Any valid routing updates that reach a PE device must come from a corresponding CE router. Hence one can place a inbound firewall filter on the CE's interface facing the corporate LAN to discard all the routing or control updates destined to the PE device. An example of such filter is shown below

```
filter protect-PE {
    term one {
        from {
            destination-address {
                192.1.1.1/32;     # PE's address
            }
            protocol bgp;         # Routing protocol between PE and a CE
        }
        then {
            log;
            discard;
        }
    }
}
```

Each VPN provides a level of virtualization for each private network. That is, each customer has its own routing-table and routing-instance. So in order to protect against resource abuse, it is highly recommended to set an upper boundary for resources that can be consumed by each private network.

An example of such a resource is the routing table memory. Each VPN contains its own routing / forwarding table on a PE router. A CE router sends the necessary routes toward a PE to be placed in the proper routing/forwarding table. A misbehaving CE router can send more routes than it is supposed to towards a PE, thereby consuming memory on the PE, and perhaps using memory that should have been reserved for the routes belonging to another VPN. One may keep this from occurring with the following configuration:

```
[edit routing-instances]
vpna {
    routing-options {
        maximum-routes {
            100;
            threshold 80;
        }
```

```
    }
}
```

Setting `maximum-routes` for each VPN will protect the PE router against unnecessary memory consumption by a single VPN. The `threshold` value indicates the percentage of the limit at which warning messages will begin being generated.
In a truly bad situation, the misbehaving VPN could also send floods of routing updates to the PE router, which could still take away CPU resources away from the other VPNs. To combat this, one may rate-limit the protocol updates coming from the CE router.
I deally, other control traffic destined to a PE should just be discarded by the PE. In some cases, however, the provider must allow other control traffic to exist (such as ICMP traffic used for troubleshooting) The small degree of risk that these protocols carry may be limited by policing or rate-limiting them to appropriate values.

Applying Firewall Filters for Each VPN

One can use the filters defined for each VPN and incorporate them all in a single firewall filter applied to the main loopback interface of the PE router. But it is also possible to define a logical loopback filter for each VRF which allows for easy identification of each VRF. This is useful for troubleshooting, allowing the user to ping a remote CE for a local PE routers (for more details on this feature, refer to the JUNOS software configuration manual). But this also has other advantages:

It makes for an easier firewall filter management. Every time a VPN is added, a separate firewall filter is applied to the corresponding loopback interface

Launching an attack on a particular VPN from the Internet will be much harder, since a filter for each VPN discards everything that is not coming to it from the trusted CE or PE.

Rate-Limiting Traffic to the Routing Engine

To protect the Routing Engine, it is important to constrain the traffic load from each of the allowed services to a rate determined during the analysis mentioned above in "Filter Configuration." Rate-limiting control traffic helps protect the routing engine from attack packets that are forged such that they appear to be legitimate traffic, and are then sent at such a high rate as to cause a DoS attack.

Routing and control traffic are essential to the proper functionality of the router, and rapid convergence of routing protocols is crucial for stabilizing the network during times of network instability. While it might seem desirable to limit the amount of routing protocol traffic to protect against various types of attacks, it is very difficult to determine a fixed maximum rate for protocol traffic, because it depends upon the number of peers and adjacencies, which varies over time. Therefore, it is best not to rate-limit routing protocol traffic.

By contrast, because management traffic is less essential and more deterministic than routing protocol traffic, it can be policed to a fixed rate, to prevent it from consuming resources necessary for less flexible traffic. We recommend allocating a fixed amount of bandwidth to each type of management traffic, so that an attacker cannot consume all the router's CPU if an attack is launched using any single service.

The policer configurations shown below are examples of potential values that might be used to prevent such attacks.

```
firewall {
    filter protect-RE {
        policer ssh-police {
            if-exceeding {
```

```
                bandwidth-limit 1m;
                burst-size-limit 15k;
            }
            then discard;
        }
policer tcp-init-policer {
     if-exceeding {
         bandwidth-limit 500k;
         burst-size-limit 15k;
     }
     then discard;
 }
/* Firewall terms omitted* /
```

Appendix B contains additional policer configurations and applications.

Auditing for Security

Network Operators are responsible for monitoring the significant events that occur on their networks, and system event logs are usually the primary sources of this information. Although the logging of events and actions does not itself increase router security, logs can be used to monitor the effectiveness of current security policies and router configurations. They can also be useful when reacting to a continued and deliberate attack, by identifying the source address, router, or port of the attacker's traffic. Finally, logs can also provide important forensic evidence for prosecuting an attacker. We recommend logging all significant protocol and router events to a remote system log (syslog) server. Syslog files can then either be parsed in real time to identify suspicious behavior or be archived for later review.

Determining which router and protocol events need to be logged depend son many factors and differs for each network. Generally, these events can be divided into three major categories:

- Authentication and command events
- Routing protocol events and errors
- Traffic being denied

Logging Authentication and Command Events

A file which that records when authentication and authorization is granted and rejected, as well as all user commands, provides an excellent way to track all management activity on the router. Checking these files for failed authentication events can help identify attempts to hack into the router. These files can also provide logs of all the command executed on the router and who has performed them. Operators can review logs of the commands executed on the router and correlate any event in the network with changes made at a particular time.

Routing Protocol Events and Errors

Routing protocol events and errors can be good indicators of attacks against routing protocols. For example, these events include protocol authentication failures, which might point to an attacker that is sending spoofed or otherwise malformed routing packets to the router in an attempt to elicit a particular behavior.

Logging of Traffic Being Denied

As mentioned in the section "Filter Configuration," above, logging all traffic that does not pass the router's firewall filters can provide an excellent way of identifying attackers. Scripts can parse the syslog files in real time and send an alarm to designated engineers for further investigation, or they can preserve these logs for analysis at a later time. Not only can such a log indicate if and when there is an attack on the router, but they can also help determine the source and type of packets used in the attack. A configuration to send emergency information to any active users, and other, different types of syslog information to two different syslog servers, is shown below.

```
syslog {
        user * {
            any emergency;
        }
        host 10.1.3.1  {
            authorization any;
            daemon info;
            kernel notice;
            interactive-commands any;
        }
        host 10.1.3.2 {
            authorization any;
            daemon info;
            kernel notice;
            user notice;
            interactive-commands any;
}
```

Network Time Protocol

Debugging and troubleshooting is much easier when the timestamps in the log files of all routers are synchronized, because events that span the network can be correlated with synchronous entries in multiple logs. We strongly recommend the using the Network Time Protocol (NTP) to synchronize the system clocks of routers and other network equipment.

By default, NTP operates in an entirely unauthenticated manner. If a malicious attempt to influence the accuracy of a router's clock succeeds, it could have potentially negative affects on system logging, making troubleshooting and intrusion detection more difficult, and impeding other management functions.

The following is a typical NTP configuration that enables authentication.

```
ntp {
    authentication-key 2 type md5 value "XXXXXXXXXXXX"; # SECRET-DATA
        boot-server 10.1.4.1;
        server 10.1.4.2;
}
```

The `boot-server` statement identifies the server from which the initial time of day and date is obtained when the router boots. The **`server`** statement identifies the NTP `server` used for periodic time synchronization. The `authentication-key` statement specifies that an HMAC-MD5 scheme is used to hash the key value for authentication, which will prevent the router from synchronizing with a attacker's host posing as the time server.

Conclusion

Implementing an effective security plan of any type requires difficult choices between preserving ease of use and obtaining impregnability. In today's network service provider environment, security involves finding a middle ground that allows for relatively simple management while providing a hardened target to the potential attacker. The tools and techniques mentioned in this paper are some of the ways to execute a security policy to meet the needs of both your network infrastructure and your customers.

A security policy and the configurations that result from it are not static entities, they must evolve over time. As attackers develop new methods to penetrate network infrastructures, you must re-evaluate your concept of what constitutes a "secure" network and change your configurations to suit. Even considerations external to the network threat itself may require you to perform this re-evaluation, such as the requirements of a new customer who places a particular emphasis on protection from a specific type of attack, or your own implementation of a new service or protocol.

Therefore, in addition to considering the items in this paper in light of your network requirements, and implementing those that suit your needs, we also suggest that you periodically review your security policy to ensure that it addresses new threats and other potential hazards to your network. Combining such a periodic review with a similar review of operating system updates (particularly those that identify a specific security threat) will ensure that your router infrastructure is ready to meet the security challenges of the New Public Network.

Appendix A

Following are some references that provide history and tutorial information about DoS (and DDoS) attacks.

- http://rr.sans.org/threats/DDoS.php
- http://rr.sans.org/securitybasics/dos.php
- http://www.juniper.net/techcenter/app_note/350001.html

Appendix B

```
system {

    host-name   Secure-Router;

    domain-name    company.com;

    default-address-selection;

/* The authentication method is set to only RADIUS so the RADIUS policy is
enforced. Only if the RADIUS server is not reachable is the local account used. */

    authentication-order [ radius ];

    root-authentication {

        encrypted-password " XXXXXXXXXXXXXXXXXXXXX"; # SECRET-DATA

    }
```

```
    name-server {

        10.1.1.1;

        10.1.1.2;

}

/* Enable RADIUS authentication and define the shared secret between the client
and the server so they each know that they are talking to the trusted peer.  A
timeout can be defined so a backup server or the next authentication mechanism can
be tried after the specified number of second has elapsed. */

   radius-server {

        10.1.2.1 {

            secret "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"; # SECRET-DATA

            timeout 5;

        10.1.2.2 {

            secret "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"; # SECRET-DATA

            timeout 5;

   }

/* Create multiple login accounts each with specific privileges.  It is a good
idea to define a timeout to disconnect the user with after there has been no
activity on his account for a specified period of time.  Users privilege levels
should be a function of their responsibilities within the organization. */

login {

/* This class of users can only view statistics and configuration, they are not
allowed to modify any configuration. */

    class provisioning {

            idle-timeout 5;

            permissions [ configure firewall interface network routing snmp system
trace view];

    }

/* This class of users can view and modify configuration. */

     class operation {

            idle-timeout 5;
```

```
        permissions [admin clear configure interface interface-control network
reset routing routing-control snmp snmp-control trace-control firewall-control
rollback];

     }

/* This class has the unlimited access and control. */

     class engineering {

            idle-timeout 5;

        permissions all;

     }

/* This is the local superuser account. If RADIUS fails or the server becomes
unreachable, try the local login accounts on the router. */

    user admin {

            uid 1000;

            class engineering;

            authentication {

            encrypted-password "<PASSWORD>"; #  SECRET-DATA

        }

    }

/*Define RADIUS template for different users or group of users. */

    user observation {

            uid 1001;

            class observation;

    }

    user operation {

            uid 1002;

            class operation;

    }

    user engineering {

            uid 1003;
```

```
            class engineering;

    }

}

/* Enable connection services on the router.  Specify how many concurrent
connection and number of connection attempts per minute can be made on the router.
*/

    services {

        ssh connection-limit 10 rate-limit 4;

}

/* Define which messages are placed in syslog files and sent to a remote server.
User can send different messages to different syslog server and in case of an
emergency message it will send it to all users who are logged in.  Having multiple
syslog servers is a good idea in case of failures. */

    syslog {

        user * {

            any emergency;

        }

        host 10.1.3.1  {

            authorization any;

            daemon info;

            kernel notice;

            interactive-commands any;

        }

        host 10.1.3.2 {

            authorization any;

            daemon info;

            kernel notice;

            user notice;

            interactive-commands any;

      }
```

```
/* We also recommend storing the syslog files locally. */
        file messages {
            any notice;
            authorization info;
            daemon any;
            kernel any;
            archive size 10m files 5 no-world-readable;
        }
        file authorization-commands  {
            authorization any;
            interactive-commands any;
        }
/* Place firewall logs are put in a separate syslog file. */
        file firewall-logs {
            firewall any;
        }
}
/* Synchronize all the routers in the network to a single time source.  We
recommend using authentication to make sure the NTP peer is trusted. */
ntp {
    authentication-key 2 type md5 value "XXXXXXXXXXXXXX"; # SECRET-DATA
    boot-server 10.1.4.1;
    server 10.1.4.2;
}
interfaces {
    at-4/0/0 {
        description  core router;
        atm-options {
```

```
            vpi 0 maximum-vcs 1024;
            ilmi;
        }
        unit 131 {
            description to-other-core-router;
            encapsulation atm-snap;
            point-to-point;
            vci 0.131;
            family inet {
                address 12.1.1.1/30;
            }
            family iso;
        }
/* The management Ethernet interface can be use for out-of-band management.
However, since most service providers will use inband communication for management
(because of lower operating costs) we will disable this interface to make the
router more secure. */
    fxp0 {
        disable;
    }
    lo0 {
        unit 0 {
            family inet {
                filter {
/* This filter selectively accepts and denies traffic going to the Routing Engine.
It needs to be applied only to the loopback interface of the router. */
                    input protect-RE;
                }
                address 10.10.5.1/32;
            }
```

```
                family iso {
                    address 48.0005.80dd.f900.0000.0001.0001.0000.0000.011.00;
                }
            }
        }
        so-2/0/0 {
            description To-other-router;
            clocking external;
            sonet-options {
                fcs 32;
                payload-scrambler;
            }
            unit 0 {
                family inet {
                    address 10.1.5.1/30;
                }
                family iso;
            }
        }
    }
    /* The community defined here as “com1” is just an example.  We recommend
    configuring authorization as read-only to minimize exploitations by hackers. */
    snmp {
        community com1 {
            authorization read-only;
        }
        trap-group jnx-traps {
            version v1;
```

```
        targets {
            10.1.6.1;
        }
    }
}
routing-options {
    router-id 10.1.7.1;
    autonomous-system 222;

/* We recommend blocking private address space as defined in RFC1918.  These
addresses and other martian addresses can be added to the default martian
addresses. */
martians {
    1.0.0.0/8 exact;
    10.0.0.0/8 exact;
    19.255.0.0/16 exact;
    59.0.0.0/8 exact;
    129.156.0.0/16 exact;
    172.16.0.0/12 exact;
    192.0.2.0/24 exact;
    192.5.0.0/24 exact;
    192.9.200.0/24 exact;
    192.9.99.0/24 exact;
    192.168.0.0/16 exact;
    224.0.0.0/3 exact;
 }
}
protocols {
```

```
    bgp {
        group ibgp {
            type internal;
            traceoptions {
                file bgp-trace size 1m files 10;
                flag state;
                flag general;
            }
            local-address 10.10.5.1;
            log-updown;
            neighbor 10.2.1.1;
                authentication-key "XXXXXXXXXXXXXXXXXXXXXXXXXXX";
        }
        group ebgp {
            type external;
            traceoptions {
                file ebgp-trace size 10m files 10;
                flag state;
                flag general;
            }
            local-address 10.10.5.1;
log-updown;
            peer-as 2;
            neighbor 10.2.1.2;
                authentication-key "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
        }
    }
```

```
    isis {
            authentication-key "XXXXXXXXXXXXXXXXXXXXXX"; # SECRET-DATA
            authentication-type md5;
            traceoptions {
                file isis-trace size 10m files 10;
                flag normal;
                flag error;
            }
            interface at-0/0/0.131 {
                lsp-interval 50;
                level 2 disable;
                level 1 {
                    metric 3;
                    hello-interval 5;
                    hold-time 60;
                }
            }
           interface lo0.0 {
                passive;
}

/* The following addresses can be thought of as the trusted source addresses that
each protocol or service wants to communicate with.  For the ease of
configuration and readability we define prefix-list here and then refer to them in
the filter definition. Also note that we are using private address space in the
examples below, if it is desired to use private address space then they should be
deleted from the martian address list*/

   prefix-list ssh-addresses {
        192.168.122/24
    }
    prefix-list bgp-addresses {
```

```
        10.2.1.0/24;
    }
    prefix-list ntp-addresses {
        10.1.4.0/24
    }
    prefix-list snmp-addresses {
        10.1.6.0/24;
    }
    prefix-list dns-address {
        10.1.1.0/24;
    }
    prefix-list radius-address {
        10.1.2.0/24;
    }


/* This is the filter that protects the Routing Engine.*/
firewall {
    filter protect-RE {
/* Allocate a certain amount of  bandwidth or speed to certain services to protect
protocol traffic in case an attacker passes through the filter (source address
spoofing). */
        policer ssh-policer {
            if-exceeding {
                bandwidth-limit 1m;
                burst-size-limit 15k;
            }
            then discard;
        }
```

```
policer small-bw-policer {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
    }
    then discard;
}
policer snmp-policer {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
    }
    then discard;
}
policer ntp-policer {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
    }
    then discard;
}
policer dns-policer {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
    }
    then discard;
```

```
        }
        policer radius-policer {
            if-exceeding {
                bandwidth-limit 1m;
                burst-size-limit 15k;
            }
            then discard;
        }
        policer tcp-policer {
            if-exceeding {
               bandwidth-limit 500k;
               burst-size-limit 15k;
            }
                then discard;
 }
/* The terms below accept traffic only from trusted sources.  The trusted traffic
is rate-limited with the exception of the routing protocols.  For an explanation,
see the “Designing a Firewall Filter” section of the paper*/.
        term icmp {
            from {
                protocol icmp;
                     icmp-type [ echo-request echo-reply unreachable time-exceeded
];
            }
            then {
                policer small-bw-policer;
                accept;
            }
        }
```

```
        term tcp-connection {
        from {
            source-prefix-list {
                                            ssh-addresses;
                                            bgp-addresses;
            }
            protocol tcp;
            tcp-flags "(syn & !ack) | fin | rst";
        }
        then {
            policer tcp-policer;
            accept;
        }
        term ssh {
        from {
            source-prefix-list {
                                            ssh-addresses;
            }
            protocol tcp;
            port [ ssh ];
        }
            policer ssh-policer;
        then accept;
    }
    term bgp {
        from {
            source-prefix-list {
```

```
                                        bgp-sessions-addresses;
            }
            protocol tcp;
            port bgp;
        }
        then accept;
    }
    term snmp {
        from {
            source-prefix-list {
                                        snmp-addresses;
            }
            protocol udp;
            port snmp;
        }
        then {
            policer snmp-policer;
            accept;
        }
    }
    term ntp {
        from {
            source-prefix-list {
                                        ntp-addresses;
            }
            protocol udp;
            port ntp;
```

```
        }
        then {
            policer ntp-policer;
            accept;
        }
    }
    term dns {
        from {
            source-prefix-list {
                dns-addresses;
            }
            protocol udp;
            port domain;
        }
        then {
            policer dns-policer;
            accept;
        }
    }
    term radius {
        from {
            source-address {
                radius-addresses;


            }
            protocol udp;
            port radius;
```

```
            }
            then {
                policer radius-policer;
                accept;
            }
        }
term trace-route {
     from {
         protocol udp;
         destination-port 33434-33523;
     }
     then {
         policer small-bw-policer;
         accept;
     }
 }
/* Everything that is not trusted is silently dropped.  We recommend logging the
denied traffic for analysis purposes.*/
        term everything-else {
            then {
                    syslog;
                log;
                discard;
            }
        }
    }
}
```