

Chapter 26

Security Configuration Example

This chapter provides an example of a configuration that applies sound security policies so that the router can operate securely. This configuration is specific to IPv4.

The following sections explain how to configure a router securely:

Configure System Information on page 338

Configure Interfaces on page 343

Configure SNMP on page 345

Configure Protocol-Independent Routing Properties on page 345

Configure Routing Protocols on page 346

Configure Firewalls on page 349

The final section in this example, “Consolidated Security Configuration Example” on page 354, show the complete configuration example.

Configure System Information

Configure the router name and domain name:

```
[edit]
system {
  host-name Secure-Router;
  domain-name company.com;
  default-address-selection;
}
```

This section contains the following topics:

Configure RADIUS on page 338

Create Login Classes on page 339

Define User Login Accounts on page 340

Define RADIUS Template Accounts on page 340

Enable Connection Services on page 341

Configure System Logging on page 341

Configure the Time Source on page 342

Configure RADIUS

The 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+). We recommend RADIUS because it 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.

In the JUNOS model for centralized RADIUS authentication, you create one or more template accounts on the router, and the users' access to the router is configured to use the template account. In this configuration, if the RADIUS server is not reachable, the fallback authentication mechanism is through the local account set up on the router.

```
[edit]
system {
  authentication-order [ radius password ];
  root-authentication {
    encrypted-password "$9$aH1j8gqQ1gjyjjhgjgiiii"; # 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 each know, that talking to the trusted peer. Define a timeout value for each server so if there is no response within the specified number of seconds, the router can try either the next server or the next authentication mechanism.

```
[edit]
system {
  radius-server {
    10.1.2.1 {
      secret "$9$aH1j8gqQ1sdjerrhser"; # SECRET-DATA
      timeout 5;
    }
    10.1.2.2 {
      secret "$9$aH1j8gqQ1csdoiuardwefoiud"; # SECRET-DATA
      timeout 5;
    }
  }
}
```

Create Login Classes

Create several user classes, each with specific privileges. In this example, you configure 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.

The first class of users (called "observation") can only view statistics and configuration. They are not allowed to modify any configuration. The second class of users (called "operation") can view and modify the configuration. The third class of users (called "engineering") has unlimited access and control:

```
[edit]
system {
  login {
    class observation {
      idle-timeout 5;
      permissions [ view ];
    }
    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 ];
    }
    class engineering {
      idle-timeout 5;
      permissions all;
    }
  }
}
```

Define User Login Accounts

Define the local superuser account. If RADIUS fails or become unreachable, we revert to the local accounts on the router.

```
[edit]
system {
  login {
    user admin {
      uid 1000;
      class engineering;
      authentication {
        encrypted-password "<PASSWORD>"; # SECRET-DATA
      }
    }
  }
}
```

Define RADIUS Template Accounts

Define RADIUS template accounts for different users or groups of users:

```
[edit]
system {
  login {
    user observation {
      uid 1001;
      class observation;
    }
    user operation {
      uid 1002;
      class operation;
    }
    user engineering {
      uid 1003;
      class engineering;
    }
  }
}
```

Enable Connection Services

Enable connection services on the router. The secure shell program (ssh) 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. The following configuration 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 option can be useful in protecting against SYN flood DoS attacks on the ssh port.

```
[edit]
system {
  login {
    services {
      ssh connection-limit 10 rate-limit 4;
    }
  }
}
```

Configure System Logging

A file 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. You can review logs of the commands executed on the router and correlate any event in the network with changes made at a particular time. These files are stored locally on the router. Place the firewall logs in a separate system log file.

```
[edit]
system {
  syslog {
    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;
    }
    file firewall-logs {
      firewall any;
    }
  }
}
```

Configure the Time Source

Debugging and troubleshooting are 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 negative effects on system logging, make troubleshooting and intrusion detection more difficult, and impede other management functions.

The following configuration synchronizes all the routes in the network to a single time source. We recommend using authentication to make sure that the NTP peer is trusted. 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 an attacker's host posing as the time server.

```
[edit]
system {
  ntp {
    authentication-key 2 type md5 value "$9$aH1j8gqQ1gjjghjgjiiii"; # SECRET-DATA
    boot-server 10.1.4.1;
    server 10.1.4.2;
  }
}
```

Configure Interfaces

Configure the interfaces on your router. This example shows configurations for ATM, SONET, loopback, and out-of-band management interfaces. For more information about configuring interfaces, see the *JUNOS Internet Software Configuration Guide: Network Interfaces and Class of Service*.

Configure an ATM interface:

```
[edit]
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 fxp0 interface can be used for out-of-band management. However, because most service providers use inband communication for management (because of lower operating costs), you can disable this interface to make the router more secure.

```
[edit]
interfaces {
  fxp0 {
    disable;
  }
}
```

Configure the loopback interface. To protect the Routing Engine, apply a firewall filter to the router's loopback interface. This filter, which you define at the [edit firewall] hierarchy level, checks all traffic destined for the Routing Engine that enters the router from the customer interfaces. Adding or modifying filters for every interface on the router is not necessary.

```
[edit]
interfaces {
  lo0 {
    unit 0 {
      family inet {
        filter {
          input protect-routing-engine;
        }
        address 10.10.5.1/32;
      }
      family iso {
        address 48.0005.80dd.f900.0000.0001.0001.0000.0000.011.00;
      }
    }
  }
}
```

Configure a SONET interface:

```
[edit]
interfaces {
  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;
    }
  }
}
```

Configure SNMP

Configure SNMP Version 3:

```
[edit]
snmp {
  access {
    user mary {
      authentication-type md5;
      authentication password "$9$aH1j8gqQ1gjyghgjgiiii";
      privacy-type des;
      privacy-password "$9$aH1j8gqQ1gjyghgjgiiii";
    }
  }
  group admin {
    user mary;
    model usm;
  }
  context router {
    group admin {
      model usm;
      security-level privacy;
      read-view all;
      write-view all;
    }
  }
  trap-group jnx-traps {
    targets {
      10.1.6.1;
    }
  }
}
```

For more information about configuring SNMP, see the *JUNOS Internet Software Configuration Guide: Network Management*.

Configure Protocol-Independent Routing Properties

Configure a router ID and autonomous system (AS) number for BGP:

```
[edit]
routing-options {
  router-id 10.1.7.1;
  autonomous-system 222;
}
```

Configure martian addresses, which are reserved host or network addresses about which all routing information should be ignored. By default, the JUNOS software blocks the following 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, 223.255.55.0/24, and 240.0.0.0/4. It is also a good idea to block private address space (addresses defined in RFC 1918). You can add these addresses and other martian addresses to the default martian addresses.

```
[edit]
routing-options {
  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;
  }
}
```

For more information about configuring protocol-independent routing properties, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

Configure Routing Protocols

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. The 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.

In this example, we configure BGP and, as the interior gateway protocol (IGP), IS-IS. If you use OSPF, configure it similarly to the IS-IS configuration shown.

For more information about configuring BGP and IS-IS, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

This section contains the following topics:

Configure BGP on page 347

Configure IS-IS on page 348

Configure BGP

The following example shows the configuration of a single authentication key for the BGP peer group internal peers. You can also configure BGP authentication 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 (and to some extent the degree of security) and the amount of management necessary to maintain the system. This example also configures a number of tracing options for routing protocol events and errors, which can be good indicators of attacks against routing protocols. 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.

```
[edit]
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 "$9$aH1j8gqQ1gjyjghgjgiiii";
    }
    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 "$9$aH1j8gqQ1gjyjghgjgiiii";
    }
  }
}
```

Configure IS-IS

Although all JUNOS IGPs support authentication, some are inherently more secure than others. Most service providers use OSPF or IS-IS to allow fast internal convergence and scalability and to use 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. This example also configures a number of tracing options for routing protocol events and errors, which can be good indicators of attacks against routing protocols. 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.

```
[edit]
protocols {
  isis {
    authentication-key "$9$aH1j8gqQ1gjjgjhgjiiii"; # 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;
    }
  }
}
```

Configure Firewalls

To configure firewall policies, configure the trusted source addresses that each protocol or service wants to communicate with. Once you define the prefix list, you apply them in the filter definition at the [edit firewall] hierarchy level.

For more information about configuring firewalls, see the *JUNOS Internet Software Configuration Guide: Policy Framework*.

```
[edit]
policy-options {
  prefix-list ssh-addresses {
    1.1.9.0/24
  }
  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;
  }
}
```

The following firewall filter protects the Routing Engine. To protect the Routing Engine, it is important to constrain the traffic load from each of the allowed services. 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 proper functioning 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.

```
[edit]
firewall {
  filter protect-routing-engine {
    policer ssh-policer {
      if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
      }
      then discard;
    }
    policer small-bandwidth-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 following terms accept traffic only from the trusted sources. The trusted traffic is
rate-limited with the exception of the routing protocols. */
```

```
/* The following term protects against ICMP flooding attacks against the Routing Engine. */
```

```
term icmp {
  from {
    protocol icmp;
    icmp-type [ echo-request echo-reply unreachable time-exceeded ];
  }
  then {
    policer small-bandwidth-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;
  }
}
```

```
/* The following term protects SSH traffic destined for the Routing Engine. */
```

```
term ssh {
  from {
    source-prefix-list {
      ssh-addresses;
    }
    protocol tcp;
    port [ ssh telnet ];
  }
  policer ssh-policer;
  then accept;
}
```

```
/* The following term protects BGP traffic destined for the Routing Engine. */
```

```
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-address {
      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-bandwidth-policer;  
    accept;  
  }  
}
```

/* All other traffic that is not trusted is silently dropped. We recommend logging the denied traffic for analysis purposes. */

```
term everything-else {  
  then {  
    syslog;  
    log;  
    discard;  
  }  
}
```

Consolidated Security Configuration Example

Basic System Information	<pre> system { host-name Secure-Router; domain-name company.com; default-address-selection; </pre>
RADIUS	<pre> authentication-order [radius password]; root-authentication { encrypted-password "\$9\$aH1j8gqQ1gjjgjhjgjiiii"; # SECRET-DATA } name-server { 10.1.1.1; 10.1.1.2; } radius-server { 10.1.2.1 { secret "\$9\$aH1j8gqQ1sdjerrhser"; # SECRET-DATA timeout 5; } 10.1.2.2 { secret "\$9\$aH1j8gqQ1csdoiuardwefoiud"; # SECRET-DATA timeout 5; } } </pre>
Login Classes	<pre> login { class observation { idle-timeout 5; permissions [view]; } 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]; } class engineering { idle-timeout 5; permissions all; } } </pre>
User Login Accounts	<pre> user admin { uid 1000; class engineering; authentication { encrypted-password "<PASSWORD>"; # SECRET-DATA } } </pre>

```

RADIUS Template
Accounts
user observation {
  uid 1001;
  class observation;
}
user operation {
  uid 1002;
  class operation;
}
user engineering {
  uid 1003;
  class engineering;
}
}

Connection Services
services {
  ssh connection-limit 10 rate-limit 4;
}

System Logging
syslog {
  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;
  }
  file firewall-logs {
    firewall any;
  }
}

Time Source
ntp {
  authentication-key 2 type md5 value "$9$aH1j8gqQ1gjjgjhjgjiiii"; # SECRET-DATA
  boot-server 10.1.4.1;
  server 10.1.4.2;
}
}

Interfaces
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;
    }
  }
}
}

```

```

fxp0 {
  disable;
}
lo0 {
  unit 0 {
    family inet {
      filter {
        input protect-routing-engine;
      }
      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;
  }
}
}

SNMP snmp {
  access {
    user mary {
      authentication-type md5;
      authentication password "$9$aH1j8gqQ1gjyjghgjiiii";
      privacy-type des;
      privacy-password "$9$aH1j8gqQ1gjyjghgjiiii";
    }
  }
  group admin {
    user mary;
    model usm;
  }
  context router {
    group admin {
      model usm;
      security-level privacy;
      read-view all;
      write-view all;
    }
  }
  trap-group jnx-traps {
    targets {
      10.1.6.1;
    }
  }
}

```

**Protocol-Independent
Routing Properties**

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

Routing Protocols

BGP

```
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 "$9$aH1j8gqQ1gjjgjhjgiiiiii";
    }
    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 "$9$aH1j8gqQ1gjjgjhjgiiiiii";
    }
  }
}
```

```

IS-IS      isis {
              authentication-key "$9$aH1j8gqQ1gjjgjhgjiiii"; # 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;
              }
            }

```

```

Firewall Policies  policy-options {
                    prefix-list ssh-addresses {
                      1.1.9.0/24
                    }
                    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;
                    }
                  }

```

Firewall Filters

```
firewall {
  filter protect-routing-engine {
    policer ssh-policer {
      if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
      }
      then discard;
    }
  }
  policer small-bandwidth-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;
  }
}
```

```

term icmp {
  from {
    protocol icmp;
    icmp-type [ echo-request echo-reply unreachable time-exceeded ];
  }
  then {
    policer small-bandwidth-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 telnet ];
  }
  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-address {
      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-bandwidth-policer;
    accept;
  }
}
term everything-else {
  then {
    syslog;
    log;
    discard;
  }
}
}
```

