ARCHITECTURE TECHNOLOGY CORPORATION

SPECIALISTS IN COMPUTER ARCHITECTURE

Machete Router

Common Criteria Operational Guidance

VERSION 1.6

DECEMBER 14, 2023

Version History:

Version	**Date**	**Change Description**
0.8	6/28/2023	Initial draft.
0.9	7/19/2023	Updates Guidance for rc13 firmware Updates Model Identification table Adds NIAP compliant parameter documentation Adds sections 6.3-6.5, 8.6.5, 11.1, 11.6.3, 11.8-END
1.0	11/01/2023	Updates Guidance to rc21 command set Updates NIAP compliant parameter documentation Updates sections 8.6, 11 Adds sections 8.6.5, 8.6.6, 9.3, 11.2, and 11.24.2
1.1	11/13/2023	Updates Guidance to rc22 command set Adds section 6.1.3, 8.3.5.1, 11.3
1.2	11/21/2023	Updates Guidance to rc23 command set
1.3	11/22/2023	Updates section 8.3.5 “SSH Client” Moves parameter documentation to new document
1.4	12/08/2023	Updates sections 1, 2.1, 4.2.2.1, 7.3, 9, 11 Adds sections 1.4.2, 7.1, 8.3.6, 8.3.7, 8.4.4.3, 10.3.1, 11.7
1.5	12/12/2023	Updates sections 7.4.2, 8.4.4.2, 8.4.4.3, 10.2, 10.3 Adds section 7.4.2.1.1
1.6	12/14/2023	Updates section 6.2 (allowable characters table), 8.4.3

Table of Contents

Table of Figures

1 Introduction

The Machete router is a ruggedized, compact, secure and high-performance router that also provides VPN gateway functionality. The functions of Machete are implemented in a software suite called ATCorp Routing and Encryption Suite (ARES).

This document provides operational guidance to install and configure the Machete Router to operate in the Common Criteria-approved mode of operation in accordance with collaborative Protection Profile for Network Devices version 2.0 (NDcPP) and NDcPP/Stateful Traffic Filter Firewall Collaborative Protection Profile (FWcPP) Extended Package VPN Gateway version 2.1 (VPNGWcEP).

1.1 Reference Documents

Document ID	Document Name	Version	Date
Machete Security Target	Architecture Technology Corporation Machete Router Security Target	1.0	TBD
Machete Configuration Parameters	Architecture Technology Corporation Machete Router NIAP Configuration Parameters	1.1	12/8/2023

Figure 1: Reference Documents

1.2 Audience

This document is written for administrators configuring the Machete router. This document assumes the user is familiar with networks and network terminology and is a trusted individual.

1.3 Overview of the Machete Router Hardware Platforms

Model Identification	Platform	CPU Architecture	CPU Part Number
MACHETE-FIT2	Fitlet2	Intel Apollo Lake	Atom x7-E3950
MACHETE-OTN4	OnTime 4000 Series	Intel Apollo Lake	Atom x7-E3950

Model Identification	Platform	CPU Architecture	CPU Part Number
MACHETE-OTN6	OnTime 6000 Series	Intel Apollo Lake	Atom x7-E3950
MACHETE-OTN7	OnTime 7000 Series	Intel Apollo Lake	Atom x7-E3950
MACHETE-DCS2	DCS003289	Intel Apollo Lake	Atom x7-E3950
MACHETE-V1	Virtual	AMD Ryzen 4000	Ryzen 4600G
MACHETE-AMD-R1	OL-ML100 Series	AMD Ryzen V1000	V1605B
MACHETE-WL1	BKNUC8V5PNB	Intel Whiskey Lake	Core i5-8365U
MACHETE-FIT3	Fitlet3	Intel Elkhart Lake	Atom x6425E

Figure 2: Machete Router Model Identifications

1.4 Operational Environment Components

The administrator must ensure the guidance items within this document are enforced.

1.4.1 Technical Requirements

The Machete Router is compatible with the following protocol versions:

- Virtual Private Network (VPN) Client
  - Internet Key Exchange version 1 (IKEv1)
  - Internet Key Exchange version 2 (IKEv2)
  - Encapsulating Security Payload (ESP)
- Syslog
  - Version 1 – RFC 5424
- Network Time Protocol (NTP)
  - Network Time Protocol version 4 (NTPv4) Server
  - Network Time Protocol version 4 (NTPv4) Client

The Machete router provides additional network routing features and protocols which are outside the scope of the NIAP Common Criteria validation, such as PIMv2, IGMPv1-3, GRE, and OSPF. Administrators are advised to use best practices when enabling and configuring these features.

1.4.2 Services

The Machete router provides a number of services. The services available include:

- ares - Manage ARES service (is expected to always be running)
- dhcp-server – Manage DHCP server service
    - v4 – Manage DHCPv4 server service
    - v6 – Manage DHCPv6 server service
- fake-hwclock - Control fake hardware clock (for systems with no real-time clock)
- firewall - Manage firewall service
- frr - Manage FRR service
- ipsec - Manage IPsec service
- lldp - Manage LLDP service
- ntpd - Manage NTP service
- sshd - Manage SSHD service
- vpn - Manage VPN service

These services are managed and controlled using the commands listed in the figure below. Each command use the *service <SERVICE>* where <SERVICE> is the service you wish to manage. The disable and enable commands mentioned in the figure are only available for some services, whereas the remaining commands are available for all of the services. Restarting is necessary to apply any changes made to the service configuration files.

Command	*Description*
service <SERVICE> disable	Disables the service
service <SERVICE> enable	Enables the service
service <SERVICE> restart	Restarts the service
service <SERVICE> start	Starts the service
service <SERVICE> status	Displays the status of the service
service <SERVICE> stop	Stops the service

Figure 3: Service control commands

The firewall is one exception in that it has four commands, disable, enable, status, and reload. The firewall rules can be reloaded after the packet filtering rules have been modified by entering the CLI command:

```
admin@Router> service firewall reload
[admin] Attempting to restart service firewall-post
[admin] firewall-post restart successful
admin@Router>
```

The VPN service has a more extensive command set and thus the biggest exception to the rule. The VPN service offers several options for displaying, loading, and managing connections, pools, and certificates. These commands can be useful for managing VPN as well as displaying the currently set configurations for VPN connections.

2 Getting Started

2.1 Operational Environment

Proper operation of the Machete router in its Common Criteria evaluated configuration requires that some security objectives be satisfied by the operational environment. It is the responsibility of the authorized administrator of the Machete router to ensure that the Operational Environment provides the necessary functions, and adheres to the environmental security objectives listed below. The environmental security objective identifiers map to the environmental security objectives as defined in the Security Target documentation. There is no difference in operational environment requirements to support the different hardware versions of the TOE.

Environment Security Objective	Operational Environment Security Objective Definition	Administrator Responsibility
OE.PHYSICAL	Physical security, commensurate with the value of the Machete router and the data it contains, is provided by the environment.	Administrators must ensure the Machete router is installed and maintained within a secure physical location which may include a secured building with key card access or within the physical control of an authorized administrator in a mobile environment.
OE.NO_GENERAL_PURPOSE	There are no general-purpose computing capabilities (e.g., compilers or user applications) available on the Machete router, other than those services necessary for the operation, administration and support of the Machete router.	Administrators must not add any general-purpose computing capabilities (e.g., compilers or user applications) to the Machete router.
OE.NO_THRU_TRAFFIC_ PROTECTION	The Machete router does not provide any protection of pass-thru traffic that traverses it. It is assumed that protection of pass-thru traffic is covered by other security and assurance measures in the operational environment.	Administrators are trusted to ensure that appropriate security and assurance measures are in place in the operational environment.
OE.TRUSTED_ADMIN	Security Administrators are trusted to follow and apply all guidance documentation in a trusted manner.	Administrators must be properly trained in the usage of the Machete router and all the enabled functionality. These administrators must follow the provided guidance.

OE.UPDATES	The Machete router firmware and software is updated by an Administrator on a regular basis in response to the release of product updates due to known vulnerabilities.	Administrators must regularly update the Machete router to address any known vulnerabilities.
OE.ADMIN_CREDENTIALS_ SECURE	The Administrator's credentials (private key) used to access the Machete router must be protected on any other platform on which they reside.	Administrators must protect their access credentials where ever they may reside.
OE.RESIDUAL_INFORMATION	The Security Administrator ensures that there is no unauthorized access possible for sensitive residual information (e.g. cryptographic keys, keying material, PINs, passwords etc.) on networking equipment when the equipment is discarded or removed from its operational environment.	Administrators must follow key destruction and data sanitation procedures.
OE.CONNECTIONS	The Machete router is connected to distinct networks in a manner that ensures that its security policies will be enforced on all applicable network traffic flowing among the attached networks.	Administrators must ensure that the Machete router enforces security policies for the distinct networks it is connected to.

Figure 4: Environment Security Objective Definitions

All external servers must be able to communicate with the Machete router. The following resources should be available in the operational environment as depicted in Figure 2.

- Syslog Server
- NTP Server
- Trusted Host with SSHv2 support
- Local Console with RS-232 support

The operational environment includes the Machete router, one or more VPN connections to provide secure administrative services and protected communications. A trusted channel connection is used to provide a secure connection for communications between the Machete router and remote administrative services; such as Syslog and NTP. SSHv2 is used to provide remote login and Trusted Uploads (SFTP). A connection between the router and a VPN gateway is referred to as a Data VPN Channel and can be used for protected communications other than remote administrative services. Local administration is provided by a local terminal connected to the router through a RS-232 connection.

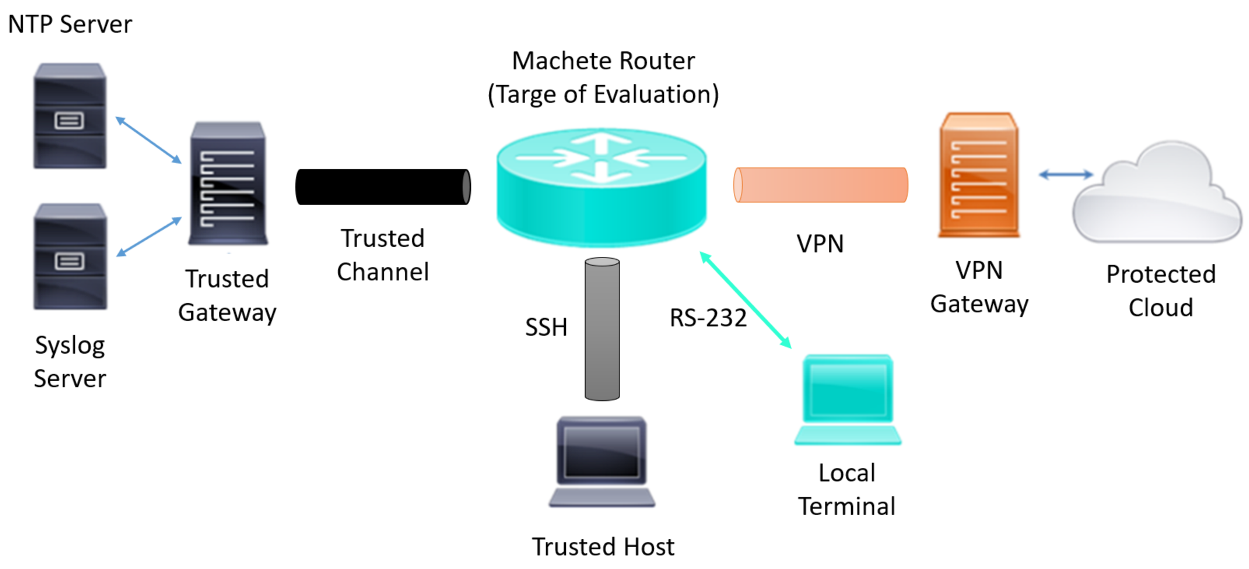

Figure 5: Operational Environment

2.2 Evaluated Configuration

Only the hardware platforms described in Figure 2 can be used to implement the evaluated configuration. Changing the software to use a different version invalidates the evaluated status of a particular hardware platform.

2.2.1 Features Prohibited from use

There are no explicitly prohibited services in the Common Criteria evaluated configuration.

2.2.2 Procedural Requirements

The following protections must be provided by the operational environment in which the Machete router is deployed:

- The Administrator must not install any additional software on the router. The router is intended to only be operated with the software provided by ATCorp.
- The router is physically protected from unauthorized access. These countermeasures should be commensurate with the value of the data being protected by the router.
- The administrators must be trusted, and they must follow and apply all administrator guidance.

2.3 CLI Overview

The CLI is the software interface used to access the Machete router. The CLI is used to configure the router, monitor its operations, and access and modify the configurations as needed. Accessing the CLI is done through a console via serial interface or through a network connection using SSH.

The CLI prompt has the following format:

```
<username>@<router-name>>
```

Where <username> is the name of the account of the current logged in user and <routername> is the hostname and router name of the Machete router.

The CLI provides help for the available input option by entering '?' at any point of the command input. The CLI also supports tab completion when a partial command is entered. For more details on the CLI see the 'Introduction to the Command-Line Interface' section of the user-guide.

2.3.1 Configuration Methods

The Machete router CLI allows modification of configuration of the router through two methods. Some commands modify configuration through direct input on the CLI and other commands open an editor that allows the user to edit a configuration file. The detailed sections for each command indicate which configuration method is implemented by that command. Only configurations made in files are persistent through power cycles, configurations made by commands will be reset to their original configuration.

2.3.2 The Text Editor

The text editing environment used is the standard NANO text editor. For those unfamiliar with the NANO editor more information can be found online at https://www.nano-editor.org/.

Configuration commands that do not edit files are transient and will not survive a device reboot. Editing configuration files with the text editor is the only way to make configurations persistent.

Many configuration files editable by the user are in JSON format. The following section provides an introduction to the JSON format. The format of any other configuration files that are not in JSON format is documented in their related sections of the user-guide.

2.3.3 JSON Format

JavaScript Object Notation (JSON) is a format for sharing data. A JSON object is a key-value data format that is rendered in curly braces. All data used in JSON ends up being encapsulated in a JSON object. Key-value pairs have a colon between them, as in "key": "value". Each key-value pair is separated by a comma. JSON keys are on the left side of the colon. Keys must be wrapped in double quotation marks, as in "key". JSON values are found to the right of the colon. At the granular level, these need to be one of 6 simple data types:

- string
- number
- object
- array
- boolean (true or false)
- null

Array values are delimited by square brackets. An example JSON configuration is shown below. This is the configuration file for a very simple ARES service setup.

```
{
  "RouterId" : 10,
  "Links" : [
    {
      "Name" : "link1",
      "Type" : "simple",
      "Interface" : {
        "Name" : "eth0",
        "Inet4Addresses" : ["10.10.10.1", "10.10.10.2"]
      }
    }
  ]
}
```

Figure 6: Example JSON Format

The JSON format is a concise and structured method for describing complex configurations. Note that whenever the user is editing a JSON structured configuration a JSON syntax check will be performed when the user ends the editing session and syntax errors, if any, will be displayed. It is not possible to save a configuration containing syntax errors, but the user is given the option to return to the editing session and correct any errors, so no configuration work is lost when this occurs.

For extensive information on JSON, see https://www.json.org/.

2.4 Initial Setup

The following sections details steps required to configure the Machete router in to the evaluated mode.

2.4.1 Identification of hardware

Ensure the evaluated Machete router device is of one of the supported hardware platforms listed in Figure 2.

2.4.2 Connect to the Device

The Machete router can be accessed locally through the RS-232 port or remotely via SSHv2.

The Machete router command-line interface (CLI) can be accessed from a remote PC by either a serial console connection or by an SSH connection. When using the serial connection, set the parameters to 115200 baud, asynchronous mode with 8 data bits, no parity bit, and 1 stop bit, and with HW and SW flow control disabled. Model-specific port identification information is provided on the Model data sheet.

SSH parameters:

```
IP Address: 192.168.100.254

Username: admin
```

```
Password: admin123
```

Example:

```
ssh admin@192.168.100.254
```

After successfully connecting, the remote login banner is displayed. Refer to *Figure 7* for the default banner.

2.4.3 Log in

When you first access the CLI, you will see the following prompt:

```
              *
              **             *
              ****           **
              *****          ****
              ******         *****
      **************         ******         Architecture
       *************     **********         Technology
        *******          **********         Corporation
      *  ******          **********
      **                 **********
      ***    **********************
***********   *****************
 ***********   ****************  *
  ***********    **************  **
   **********     *************  ****       This is a Protected System.
    *********                    *****
      *******                    *****      No Unauthorized Access.
              ************************
               ***********************
                **********************
r0 login:
```

Figure 7: Default login screen

```
Note: This is the default banner and login screen. This screen can be
customized for the local console as well as for the remote by using the
banner command in the configuration mode.
```

Once at this screen, you will enter the account username and password to login. If this is your first-time logging in or have not set up or changed the account information yet, the default username and password is as follows:

Default Username: admin

Default Password: admin123

To ensure proper security of the TOE, the default username and password for this account must be changed after initial login.

2.4.4 Identification of Firmware

To determine the Machete firmware version, enter either of the CLI commands:

```
admin@r0> show version
```

Or

```
admin@r0> firmware info
```

2.4.5 Configuration

The Machete router requires initial configurations for each of its components in order to put it into the evaluated configuration. The router has multiple components, each with their own configuration file. ARES is the main component on the Machete router. ARES is configured through a JSON formatted file, called ares.json. For more information about the Machete router components, command line interface(CLI), and the JSON format, refer to the Machete Router User Guide.

To change the router configuration, the user has to enter the “configure” mode by running the following CLI command:

```
admin@r0> configure
admin@r0(config)>
```

To configure the ARES service, while in the configuration mode, run the CLI command:

```
admin@r0(config)> ares
```

The command above will enter into a text editing session for the ares.json file. When the Machete router is started up for the first time, the ares.json file is populated with a few pre-configured parameters in order for valid operation of the ares service. These pre-configured parameters will be:

```
{
   "RouterId" : 10,
   "RouterName" : "r0",
   "SecurityMode" : "normal",
   "Session" : { "Timeout" :  900 },
   "Log" : { "Verbosity": "warning" },
   "Plugins": [ "sysconfig" ],
   "Multicast" : {
      "Enabled" : true,
      "PIM" : {
         "RP-List" : [
            {
               "RP" :  "127.0.0.1",
               "Groups" : ["224.0.0.0/4" ]
            }
         ]
      }
   },
   "Links" : [
      {
         "Name" : "LAN",
         "Type" : "simple",
         "Interface" : {
            "Name" : "eth0",
            "Inet4Addresses" : [ "192.168.10.254/24" ]
         }
      },
      {
         "Name" : "WAN",
         "Type" : "simple",
         "Interface" : {
            "Name" : "eth1",
            "Inet4Addresses" : [ "192.168.100.254/24" ]
         }
      }
   ],
}
```

```
Note: Configurations that affect the shell environment, such as the
"RouterName", will not take effect until after logging out of the
current session.
```

In order to put the router security mode in the evaluated configuration, set the "SecurityMode" parameter to "niap" as seen below.

```
{
     …
     "SecurityMode" : "niap",
     …
}
```

To exit the current editing session, press "ctrl-x", the router will prompt the user to save the changed configuration and exit the session.

2.4.6 Restart ARES

Whenever configurations have been changed under ARES, the service must be restarted in order to apply the configuration changes.

Upon exiting the text editor session for the ares.json file, the Machete router will offer the prompt to restart the service at that time or not.

```
   admin@r0(config)> ares
   Configuration changed, service ares must be restarted to apply changes.
Restart service now? (Y/n)
```

If the changes made are not meant to be immediately, the network administrator can manually restart the service by running the command while in configuration mode:

```
   admin@r0(config)> do service ares restart
   [admin] Attempting to restart service ares
   [admin] ares restart successful
   admin@r0(config)>
```

The ares service can also be restarted while in the default mode:

```
   admin@r0> service ares restart
   [admin] Attempting to restart service ares
   [admin] ares restart successful
   admin@r0>
```

3 Trusted Path/Channels

The Machete router supports either SSH or IPsec to provide a trusted communication channel between itself and all authorized IT entities that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data. The router uses SSH or IPsec to provide the trusted path with remote administrative users as well.

3.1 Recovery of Trusted Path/Channels

For automatic SSH tunnel connections, refer to the *Automatic Connections* section. Once the trusted communication channel is configured and a connection is successfully established, there are no additional configurations or commands required to re-establish these connections after a sudden power or connection loss. The router will continue to retry connecting forever until the service has been stopped.

3.2 Trusted Uploads

Trusted Uploads allow for the uploading and downloading of data to the Machete router via a SSH from a trusted remote destination established by a security administrator. Trusted Uploads can only be performed by an account in the transport or administrator groups. Trusted Uploads allow for the following functionality:

- Backup and restore configurations, including certificate and key material.
- Upgrade Firmware.

Files that are backed up/restored to/from external devices may get saved to a staging directory under /tmp/uploads.

3.3 Configuration Management

The Machete router allows saving multiple copies of configurations under different names. This allows easy switching between the different configurations. Saved configurations can also be back up configuration to a different machine so that such configuration can be copied to another router or restored later.

To back up configuration to a different machine so that such configuration can be copied to another router or restored later, run the CLI command in default mode:

```
cfg push <config> <uri>
```

Where:

<config> is the desired configurations to be pushed.

<uri> is the URI of the location to back up the configurations. This location must be one of the following:

- http(s)://
- scp://
- file://

```
Note: When pushing to an external source, the firewall may require
additional rules to allow traffic through.
```

To restore a previously backed up configuration, run the CLI command in default mode:

`cfg fetch <file|uri>`

Where <file|uri> is the location of a tar.gz file. This location must be one of the following:

- http(s)://
- scp://
- file://

```
Note: When fetching from an external source, the firewall may require
additional rules to allow traffic through.
```

Once you have fetched the desired configurations, they will have to be copied into the startup-config directory and rebooted in order to fully restore the configurations.

Example:

```
   admin@r0> copy backup-config startup-config

   Are you sure you want to overwrite the existing configuration: startup-
config? (y/N) y
   [admin] Copying files from backup-config to startup-config
   ares.json      OK
   firewall.nft   OK
   banner         OK
   banner.net     OK
   frr.conf       OK
   daemons        OK

   Restarting ARES required to apply restored configuration. Restart now?
(y/N) y
   [admin] Attempting to restart service ares
   [admin] ares restart successful
   admin@r0>
```

Figure 8: *copy* usage

3.4 Import and Export of Certificates and Key Material

All PKI material on the router can be found by using the *show pki list* command. This command will display all of the PKI related directories on the router as well as the PKI material that exists in their respective directories.

```
admin@r0> show pki list

------- x509 -------

------- x509ca -------

------- x509aa -------

------- x509ocsp -------

------- x509crl -------

------- x509ac -------

------- rsa -------

------- ecdsa -------

------- private -------

------- pkcs8 -------

------- pkcs12 -------

------- pubkey -------

------- psks -------
Jun  15   20:56    111      root             ntp.keys

------- pkcs10 -------

------- template -------
Jun  15   19:39    1.9k     admin            user-cert.cmd
Jun  15   19:39    407      admin            attrib-cert.cmd
Jun  15   19:39    1.6k     admin            ca-cert.cmd
Jun  15   19:39    400      admin            cert-signing-request.cmd
Jun  15   19:39    1.8k     admin            ica-cert.cmd
Jun  15   19:39    2.1k     admin            server-cert.cmd
Jun  15   19:39    1.1k     admin            sign-crl.cmd
admin@r0>
```

Figure 9: *show pki list* command

Alternatively, the command *show pki list* can be appended with a specified <*directory*> to view all of the PKI materials in a specific directory on the router.

```
admin@r0> show pki list template
Jun  15   19:39    1.9k     admin            user-cert.cmd
Jun  15   19:39    407      admin            attrib-cert.cmd
Jun  15   19:39    1.6k     admin            ca-cert.cmd
Jun  15   19:39    400      admin            cert-signing-request.cmd
Jun  15   19:39    1.8k     admin            ica-cert.cmd
Jun  15   19:39    2.1k     admin            server-cert.cmd
Jun  15   19:39    1.1k     admin            sign-crl.cmd
admin@r0>
```

Figure 10: *show pki list template* command

3.4.1.1 Import PKI material to Machete:

The Machete router allows the use of externally created certificates and key material, which can be imported by following the steps below.

1. Prepare a package with the right format using:

```
admin@r0(config)> pki package ?
Available commands:
   pki package <tag>   - A string tag to be added to the package name. Allowed
characters A-Z a-z 0-9 _ -
admin@r0(config)> pki package r1-certs
Packaging staged files to r1-certs.pki.data.tar.gz
r1-certs.pki.data.tar.gz ready for transfer
admin@r0(config)>
```

2. Copy the package to another machine

```
admin@r0(config)> pki push
Available commands:
   pki push <filename> url - The address or FQDN of the remote host
admin@r0(config)> pki push r1-certs.pki.tar.gz scp://admin@192.168.229.120:~/
Pushing pki file r1-certs.pki.tar.gz to admin@192.168.229.120:~/ ...
admin@192.168.229.120's password:
r1-certs.pki.data.tar.gz
[admin] pushed pki file r1-certs.pki.data.tar.gz to admin@192.168.229.120:~/
admin@r0(config)>
```

3. Extract the tar file. The directories available for PKI files are listed above. If you have a .p12 bundling all the CA cert ,the Machete cert end entity cert and private key you can copy it to the "private" directory. If you have the individual files, you can copy them to their respective directories. Once you are done. Tar/zip the files again and have the newly created package ready for the next step.

4. Fetch the package with the new martial to Machete

```
admin@r0(config)> pki fetch
Available commands:
   pki fetch <filename> <user> <host> - The address or FQDN of the remote host
admin@r0(config)> pki fetch scp://admin@192.168.229.120:~/r1-certs.pki.tar.gz

fetching pki filel admin@192.168.229.120:~//certs.router.pki.data.tar.gz ...
admin@192.168.229.120's password:
certs.router.pki.data.tar.gz
User admin fetched pki material file certs.router.pki.data.tar.gz from
admin@192.168.229.120:~/
```

5. Confirm that the newly fetched files exists in the pki staged area. If you placed a file named router.p12 under the directory "pkcs12", you will see:

```
admin@r0(config)> do show pki staged pkcs12
total 8
-rwxr-xr-x 1 7858 Aug 14 21:18 router.p12
```

3.5 Install/Upgrade Firmware

The Machete router only accepts firmware digitally signed by ATCorp. Digitally signed firmware can be acquired from ATCorp via product support channels. Firmware is uploaded to the router and placed in the updates directory. Once uploaded, the router verifies the digital signature. If the digital signature is verified, the router checks the new firmware version confirming it is able to be installed. If the check is successful the firmware is placed in a staging area for delayed activation by a security administrator.

To query the currently executing version of the Machete router firmware, enter the CLI command:

`firmware info`

OR

`show version`

To fetch and stage new firmware, use the CLI command:

`firmware fetch <uri>`

Such that <uri> is the URI of the firmware package. This package must be one of the following:

- http(s)://
- scp://
- file://

`Note: Fetching from http(s):// sources must be done over a VPN trusted channel.`

Example:

```
admin@r0> firmware fetch scp://foo@192.168.10.2:/home/foo/crrfw_2.0.0.ipks
```

The above example shows a `firmware fetch` using the scp protocol (“`scp://`”). The new firmware file is located in the /home/foo/ directory of the host 192.168.10.2 (which has an ssh server running). The firmware filename is “crrfw_2.0.0.ipks”. The username “foo” is specified in the “`foo@`” portion of the uri. Note carefully the following delimiters

- `://`
- `@`
- `:/`

in the preceding `firmware fetch` command. The most common reason for this command failing is omission of a required delimiter.

Once the new firmware is staged, it is ready to be installed by a security administrator. Only a single firmware image is able to be staged at a time. To check if the router has a staged firmware image ready to be installed, enter the CLI command:

```
admin@r0> firmware list
```

OR, for more information about the firmware use the CLI command:

```
admin@r0> firmware list detail
```

Staged firmware images are only installed after a security administrator initiates the install by entering the CLI command:

```
admin@r0> firmware apply <version>
```

the router prompts to confirm installation. The router indicates success or failure of the installation of the new firmware. Loss of power or interruption of the firmware update process puts the Machete router in an unusable state. Any attempt to update the firmware generates a syslog message indicating success or failure of the update.

Additionally, staged firmware can be deleted from the list by using the CLI command:

```
admin@r0> firmware delete <version>
```

Such that, the <version> is the version of the desired firmware to be removed.

Finally, the integrity of the firmware version can be checked by using the CLI command:

```
admin@r0> firmware verify
```

The Machete router may fail to process a firmware update for the following reasons:

- Invalid file format. The router expects a compressed tar file ending in the extension ".gz".
- Invalid digital signature.
- New firmware does not meet dependency requirements.

Should the router fail to process a firmware update for any reason, the failure and reason for failure is recorded in syslog. Successful processing of the firmware update is also recorded in syslog.

4 Protection of the Machete Router

The Machete router performs power-on self-tests to ensure correct operation. It also supports time synchronization with an external time source in order to ensure correct time, used for audit log timestamps and validation of certificates.

4.1 TSF Self-Test

The router runs a suite of the following power-on self-tests to ensure correct operation:

- Entropy source health tests
- RAM health tests
- Firmware integrity test
- Cryptographic algorithm tests

If any of the power-on self-tests fail, the router aborts the normal boot process and boots into a restricted diagnostic mode with no networking services. In this mode, the administrator can only log into the router via the serial connection with limited CLI capabilities including:

- view the self-test results to determine the cause of the error
- reboot/shutdown

Entropy health tests may fail if the entropy source is not producing quality entropy. This may be resolved by rebooting the router. RAM health test failure as well may be resolved by rebooting the router. If the self-test error persists then contact ATCorp's customer support.

The administrator can view the self-test results with the following command:

```
admin@r0> show self-test
```

Examples of self-test success and failure can be found in section 11.23.

4.2 Time Stamps UTC

The Machete router provides the ability to set the time manually as well as automatically using an external time source. Time is kept and displayed in UTC and the time zone is not user configurable. Having incorrect time could result in 'bad time' issues when working with certificates, but will not hinder the performance of the Machete router otherwise. Configuration of the external time source is restricted to NTPv4, hence no configuration is available to set the NTP version.

To view the system time, in default mode, run the CLI command:

```
admin@r0> show time
Thu Mar 30 17:00:00 UTC 2023
```

4.2.1 Manual Configurations

To set the time manually, enter configuration mode and use the CLI command:

```
admin@r0(config)> time [MMDDhhmm[[CC]YY][.ss]]
```

where:

```
MM = Month (01, 02…12)
DD = Day of month (01, 02…31)
hh = hour
mm = minute
CCYY = Year
ss = second
```

Example:

```
admin@r0(config)> time 033012002023.00
Thu Mar 30 12:00:00 UTC 2023
[admin] System time updated successfully
[admin] Hardwware Clock updated successfully
```

4.2.2 External Time Source Configuration

To synchronize time with external NTP servers or peers, sources can be listed as part of the ARES configuration after first enabling the SysConfig plugin. Authentication keys for the servers or peers can be configured if desired but are not required.

The NTP service only synchronizes the local clock if the local clock time is within 1000 seconds of the reference time. If the time difference is greater than 1000 seconds, the NTP service exits and logs the error. If the time difference is less than 1000 seconds, the NTP service steps the local clock until it matches the reference time and creates a log message.

To configure the NTP service for the Machete router, first manually set the system time to within 1000 seconds of UTC using the time command shown above.

To configure the Machete router to connect to an external NTP server, enter the following CLI commands to modify the ARES configuration:

```
admin@R1> configure
admin@R1(configure)> ares
```

The following example configuration will synchronize the Machete router with a remote NTP server with an IP of 192.168.0.1:

```
{
   "RouterId" : 10,
   "Plugins" : [ "sysconfig" ],
   "Links" : [
      ...
   ],
   "NTP": {
      "Sources": [
         {
            "Address", "192.168.0.1",
            "Type": "server"
         }
      ]
   }
}
```

The NTP service will be automatically restarted after the configuration has been applied. Additional configuration options may be used to associate an NTP key with the source or to designate the source as preferred. Note that “sysconfig” must be listed in the Plugins array for the NTP service to function properly.

NTP authentication keys may also be managed through the ARES configuration. A “Keys” array should be added to the NTP block with each Key object containing a key ID, key type, and raw key value or file containing the key as shown in the following example:

```
{
   "RouterId" : 10,
   "Plugins" : [ "sysconfig" ],
   "Links" : [
      ...
   ],
   "NTP": {
      "ControlKey": 1,
      "Keys": [
         {
            "ID": 1,
            "Type": "md5",
            "Key": "file://simplekey1"
         },
         {
            "ID": 2,
            "Type": "md5",
            "Key": " file://ntpkey.psk"
         }
      ],
      "Sources": [
         {
            "KeyID": 1,
            "Address", "192.168.0.1",
            "Type": "server",
            "Prefer": true
         },
         {
            "KeyID": 2,
            "Address", "192.168.10.1",
            "Type": "peer"
         }
      ]
   }
}
```

Figure 11: Example NTP Block Configuration

The control key is used to authenticate the user for some “show ntp ...” commands and should generally be provided in human-readable format (ASCII vs. hexadecimal).

No further configuration is needed to prevent accepting broadcast and multicast NTP packets that would result in the timestamp being updated, as this is the default behavior.

4.2.2.1 NTP Protection via IPsec

The NTP source can be configured in conjunction with an IPsec channel to protect the communication. To protect NTP traffic to a specific peer, make sure that the peer’s IP address (NTP::Sources::Address “10.0.3.1” in the example below) falls within the remote traffic selector of a VPN security association (VPN::Profiles::connections::children::remote_ts, “10.0.3.0/24” in the example below).

```
{
  "RouterId" : 10,
  "SecurityMode" : "niap",
  "Plugins": [  "sysconfig", "vpn" ],
  "NTP": {
    "ControlKey": 1,
    "Keys": [
      {
        "ID": 1,
        "Type": "md5",
        "Key": "file://simplekey1"
      }
    ],
    "Sources": [
      {
        "KeyID": 1,
        "Address", "10.0.3.1",
        "Type": "server",
        "Prefer": true
      }
```

```
          ]
        },
        "VPN": {
          "Profiles": {
            "connections": {
              "r3": {
                "version" : 2,
                "proposals": {
                  "list": [
                    {
                      "encryption_algorithms": ["aes128"],
                      "integrity_algorithms": ["sha256"],
                      "pseudo_random_functions": ["prfsha256"],
                      "ke_groups": ["modp2048"]
                    }
                  ]
                },
                "local_addrs": [ "10.0.1.1" ],
                "remote_addrs": [ "10.0.2.3" ],
                "local": {
                  "auth": "pubkey",
                  "certs" :  ["file://router.crt"],
                  "id": "SAN field from the certificate such as
     IP address, FQDN, email, etc"
                },
                "remote": {
                  "auth": "pubkey"
                },
                "children": {
                  "red": {
                    "esp_proposals": {
                      "list": [
                        {
                          "encryption_algorithms": [ "aes128" ],
                          "integrity_algorithms": [ "sha256" ]
                        }
                      ]
                    },
                    "local_ts": [ "10.0.0.0/24" ],
                    "remote_ts": [ "10.0.3.0/24" ],
                    "start_action": "trap|start"
                  }
                }
              }
            }
          }
        }
        }
```

4.2.3 NTP Status

Once the router has been configured appropriately to synchronize with an external NTP source, the show ntp command can be used to query the system for the current time and NTP status. Some of the most notable are show ntp peers, show ntp config, and show ntp time, which are listed with their associated help text along with the rest of the NTP commands in the CLI commands section of this document.

5 Machete Router Access

The Machete router is accessed locally through the RS-232 port or remotely via SSHv2.

5.1 Session Inactivity Timeout

The Machete router provides the ability to independently configure local and remote session inactivity timeouts. After the specified period of inactivity, the router blanks the screen and disables all user actions except re-authentication for local sessions, or closes the remote connection, requiring the user to reconnect and re-login.

To view the current inactivity timeouts for local and remote sessions outside of the ARES configuration file, enter the following CLI commands:

```
admin@r0> show timeout console
admin@r0> show timeout remote
```

5.1.1 Local Inactivity Timeout

To configure the inactivity timeout for local console sessions, add the “Session” block to the ARES configuration as seen here:

```
{
   "Plugins" : [ "sysconfig" ],
   "Session" : {
      "Timeout" : { "Console" : 900 }
   }
}
```

where the console timeout can be set to an integer value between 5 and 86,400 seconds (24 hours). The default console timeout value is 900 seconds (15 minutes). The new timeout value only affects new local sessions and requires a re-login in order to see the change.

5.1.2 Remote Inactivity Timeout

To configure the inactivity timeout for remote console sessions, add the SSH::Server block, if not already present, to the ARES configuration as seen here:

```
{
   "Plugins" : [ "sysconfig" ],
   "SSH" : {
      "Server" : {
         "ChannelTimeout": "100"
      }
   }
}
```

where the channel timeout specifies whether and how quickly sshd should close inactive channels. The timeout value is specified in seconds by default or may use any of the time units denoted by appending the first letter; either lower-case or uppercase ‘s’ for seconds, ‘m’ for minutes, ‘h’ for hours, ‘d’ for days, or ‘w’ for weeks. Removing this parameter or specifying a zero value disables the inactivity timeout.

5.2 Administrator-Initiated Termination

To terminate the current administrative session, either local or remote, enter CLI command:

```
admin@r0> logout
```

5.3 Banners

The Machete router provides the ability to configure two banners: one that is displayed before establishing a local session and one that is displayed before establishing a remote session.

To view a current banner, enter the *show banner* command, with either *console* or *remote* appended at the end to see the desired banner.

```
admin@r0> show banner console
              *
              **             *
              ****           **
              *****          ****
              ******         *****
      **************         ******          Architecture
       *************     **********          Technology
        *******          **********          Corporation
      *  ******          **********
      **                 **********
      ***    **********************
***********   *****************
 ***********   ****************  *
  ***********    **************  **
   **********     *************  ****        This is a Protected System.
    *********                    *****
      *******                    *****       No Unauthorized Access.
              ************************
               ***********************
                **********************
admin@r0>
```

To configure a banner, enter configuration mode and enter the *banner* command, with either *console* or *remote* appended at the end to configure the desired banner. Once this command is entered, a text editor session is started which allows the modification of the local or remote banner file. The same output as seen when running the *show banner* command will be displayed in the text editor. For more information on how to use the text editor, refer to *The Text Editor* section.

6 Identification and Authentication

The Machete Router restricts the ability to manage security functions to authenticated Security Administrators while allowing limited access to trusted users. The administration of identification and authentication configurations as well as available unauthenticated services is documented below.

6.1 Accounts

The Machete router restricts the ability to manage security functions to Administrators. All accounts on the router belong to one of the five account groups that have varying levels of access to the router's functionalities described in this document.

6.1.1 Account Groups

In additional to these different hierarchy levels, the available commands in each level are dependent on the privilege level of the user. The table below displays the three different privilege levels defined by the group a user account is in, as well as what the group has access to and the command prompt associated with each level.

- User: Accounts in this group can only monitor the functionalities of the router and manage own password.
- Transport: Accounts that are used only for pushing resources to the router. Additionally, these accounts may not log into the router.
- Security: Performs all the functionalities of the administrators group with the exceptions to performing trusted uploads, account management and group management.
- Config: Performs all of the functionalities of the administrators group with the exceptions to performing trusted uploads, account management, and group management.
- Administrators: Performs all of the functionalities described in this administrative guidance document.

To see the user accounts that belong to a specific group, enter the CLI command in default mode (or configuration mode appended to the *do* command):

```
show group <group>
```

where:

```
<group> = (administrators|config|security|transport|users)
```

6.1.1.1 Switching Groups

Account groups can be changed by issuing the CLI command in configuration mode:

```
group <group> add <username>
```

where:

```
<username> = a valid account on the router
```

```
<group> = (administrators|config|security|transport|users)
```

```
WARNING: Care should be taken when elevating an account to the administrators
group because of security privileges given to those accounts.
```

6.1.2 Account Management

In most cases, we will want to have multiple user accounts to allow multiple users to monitor and configure a device. This topic describes how to use an admin account to log in to your Machete router and manage user accounts. You can manage an account for your own use or create a test account.

To add a new user account on the device:

1. Log into an account with administrative privileges and enter configuration mode.

```
admin@r0> configure
admin@r0(config)>
```

2. Type the *user add* command and **DOUBLE-TAP TAB** to see the list of groups. Refer to section 6.1.1 for more information about groups.

```
admin@r0(config)> user add
administrators config   security         transport        users
admin@r0(config)> user add
```

3. Append to the previously typed command the desired group, followed by the new user's username. You will then be prompt to enter the new user's password, do so and then you will be asked to re-enter the password to confirm. For password specifications, refer to section 6.2.

```
admin@r0(config)> user add users testuser
[admin] User testuser created and added to group users
[admin] Attempting to change password for user testuser
Changing password for testuser
New password:
Retype password:
passwd: password for testuser changed by root
[admin] Password for user testuser successfully changed
admin@r0(config)>
```

4. Now you can use the *do show user list* command to display the list of users to verify that the new user account has been added.

```
admin@r0(config)> do show user list
User List:
      admin
      testuser
admin@r0(config)>
```

To delete a user account on the device:

1. Log into an account with administrative privileges and enter configuration mode.

```
admin@r0> configure
admin@r0(config)>
```

2. Type the *user delete* command, followed by the account username you wish to delete.

```
admin@r0(config)> user delete testuser
Are you sure you wish to delete user testuser? (y/N)y
[admin] User testuser was deleted
admin@r0(config)>
```

3. Now you can use the *do show user list* command to display the list of users to verify that the user account has been deleted.

```
admin@r0(config)> do show user list
User List:
      admin
admin@r0(config)>
```

To update a password for a user:

1. Log into an account with administrative privileges and enter configuration mode.

```
admin@r0> configure
admin@r0(config)>
```

2. Type the *user password* command, followed by the username of the account for which you wish to change the password. You will then be prompt to enter the new password and retype the password again before it is accepted. For password specifications, refer to section 6.2.

```
admin@r0(config)> user password testuser
[admin] Attempting to change password for testuser
Changing password for testuser
New password:
Retype password:
passwd: password for testuser changed by root
Password for testuser successfully changed by admin
admin@r0(config)>
```

`Note: The` *`user password`* `command is accessible by other privilege levels, but only changes the currently logged in accounts password.`

To change a user account's privilege level:

1. Log into an account with administrative privileges and enter configuration mode.

```
admin@r0> configure
admin@r0(config)>
```

2. Type the *group* command and **DOUBLE-TAP** TAB to see the list of available groups.

```
admin@r0(config)> group
administrators config   security         transport        users
admin@r0(config)> group
```

3. Append to the previously typed command the desired group, followed by the *add* command and then the user's username.

```
admin@r0(config)> group administrators add testuser
User testuser changed to group administrators by admin
admin@r0(config)>
```

4. Now you can use the *do show group administrators* command to display the list of users that belong to the administrator group privilege level.

```
admin@r0(config)> do show group administrators
Users in group administrators:
      admin
      root
      testuser
admin@r0(config)>
```

6.1.2.1 Account Management with ARES Configuration

The Machete router supports configuring user accounts in ARES configuration. This account management option will override the existing state of user account settings that have been set from the CLI but is limited to editing non-admin users and can be used to streamline deployment.

To configure user accounts with ARES, enter configuration mode.

```
admin@Router> configure
admin@Router(config)> ares
```

In order to configure user accounts, add a top-level JSON block named “Accounts” to the configuration file. The Accounts object entry must include at minimum a Name and Password.

```
{
  "Accounts": [
    {
      "Name": "testuser",
      "Group": "config",
      "Password": "mysecurepassword"
    }
  ]
}
```

Optionally, SSH keys and authorized pairs can be added to accounts here as well. Here is an example for SSH authorized keys with the text for the key replaced with <key text>.

```
{
  "Accounts": [
    {
      "Name": "testuser",
      "Group": "config",
      "Password": "mysecurepassword",
      "SSHAuthorizedKeys": [
        "ssh-rsa <key text>== testuser@testhost",
        "ssh-rsa <key text>"
      ]
    }
  ]
}
```

6.1.3 Account SSH Key management

SSH keys and authorized pairs can be added using the `user ssh authorized-keys` command in configuration mode. This command will enter a text-editing session for the ./ssh/authorized_keys file.

```
admin@Router(config)> user ssh authorized-keys
```

Authorized keys may be generated locally on the Machete router using the `user ssh keygen` command in configuration mode. This command supports the generation of ECDSA and RSA keypairs in the desired bit lengths.

The following is an example usage of the keygen command to generated an RSA keypair with length of 2048 bits.

```
admin@Router(config)> user ssh keygen rsa 2048
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/users/admin/.ssh/id_rsa
Your public key has been saved in /etc/users/admin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:9xA2ypGBtRkCxYnIcilbRftTHUiNJwZx5eaUVEIS8CQ admin@Router
The key’s randomart image is:
+---[RSA 2048]---+
| . =+=EOXB*..   |
|o * ..+*=%o+    |
| *  .  .Bo%     |
|.    . o 0 +    |
|      o S . .   |
|       .        |
|                |
|                |
+----[SHA256]----+
[admin] Created SSH keypair ‘id_rsa/id_rsa.pub’ for user ‘admin’
admin@Router(config)>
```

6.2 Password Management

Passwords are composed of printable characters which are defined as the following four class types:

- upper case characters
- lower case characters
- digits
- and the following symbols:

!	"	#	$	%	&	'	(	)
*	+	,	-	.	/	:	;	<
=	>	?	@	[	\	]	^	_
`	{	|	}	~	<space>			

6.2.1 Strong Passwords

A password is considered strong if it is difficult to discover using intelligent guessing. Common guidelines for the creation of strong passwords consist of a combination of the following:

- The password should contain at least one character from each of the four types listed above.
- The password should have a minimum length of 12 characters.
- The password should avoid patterns, dictionary words, or personal identifiers such as:
  - pet names
  - family/friend names
  - significant dates
- The password should not be the same as one used on another system.

- When changing passwords, change at least five characters. Avoid passwords created by rotating one or more characters.

6.2.2 Strong Password Enforcement

Basic rules for password creation are enforced. Administrators can configure additional rules to enforce stronger password creation. The basic password rules are:

- Dictionary Word – A password based on a dictionary word is rejected.
- Minimum Length – A password must contain at least 6 - 100 characters. The use of configurable credit rules does not reduce the required number of characters below 6.
- Palindrome – A password that is a palindrome is rejected.
- Case Change – A password that is identical to a previously used password with the exception of a change from upper case to lower case or vice versa is rejected. This only applies when changing the current account password.
- Rotated – A password that is a rotated version of a previously used password is rejected. This only applies when changing the current account password.
- Invalid Characters – A password containing non-printable characters or characters not listed in the defined character classes is rejected.
- Sequential Characters – A password that contains too many instances of sequential characters is rejected. A sequential character is when the current character is preceded by a character with an adjacent ascii value (+/- 1), case insensitive (i.e. upper case characters assume the ascii value of its lower case counterpart). This is allowed to occur in the password at most 3 times, with 1 additional allowance for every 11 characters in the password.
- Unique – A password that contains fewer than 5 unique characters is rejected.
- Whitespace – A password containing only whitespace characters is rejected.
- Format – A password in the format “aaddddda” (a=alpha, d=digit) is rejected.
- User name – A password based on, related to, or derived from the user name is rejected.

Configurable rules to enforce stronger password creation are discussed next. To configure these rules, set the top-level ARES parameter:

```
"Session" : {
      "Password" : {
         "<rule>" : <value>
      }
   }
```

Where <rule> and it’s correlating <value> is described below.

- **minlength** – The minimum number of characters required for the creation of a new password. The range is 6 to 100 characters. The default <minlength> is 8. It is recommended <minlength> be set to 12 or higher.
- **difok** – New passwords must differentiate from old passwords by <difok> number of characters. The range is 0 to 20 characters. The default <difok> is 0, disabled. It is recommended <difok> be set to 5 or higher.

- **dcredit** - For values 0-4, the number of extra credit received for using digits in a password up to <dcredit> (e.g. If <dcredit> is set to 3, the use a single digit in a password counts as 2 characters, the use 3 digits counts as 6 characters, the use of 4 digits counts as 7 characters). For negative values, passwords are required to contain the minimum number of digits. The range is -4 to 4. The default <dcredit> is 0, disabled. It is recommended <dcredit> be set to a negative value to force the inclusion of one or more digits in new passwords.
- **lcredit** – For values 0-4, the number of extra credit received for using lower case characters in a password up to <lcredit> (e.g. If <lcredit> is set to 3, the use a single lower case character in a password counts as 2 characters, the use 3 lower case characters counts as 6 characters, the use of 4 lower case characters counts as 7 characters). For negative values, passwords are required to contain the minimum number of lower case characters. The range is -4 to 4. The default <lcredit> is 0, disabled. It is recommended <lcredit> be set to a negative value to force the inclusion of one or more lower case characters in new passwords.
- **ocredit** – For values 0-4, the number of extra credit received for using other characters in a password up to <ocredit> (e.g. If <ocredit> is set to 3, the use a single other character in a password counts as 2 characters, the use of 3 other characters counts as 6 characters, the use of 4 other characters counts as 7 characters). For negative values, passwords are required to contain the minimum number of other characters. The range is -4 to 4. The default <ocredit> is 0, disabled. It is recommended <ocredit> be set to a negative value to force the inclusion of one or more other characters in new passwords.
- **minclass** – The minimum number of required password classes required for new passwords. The range is 0 to 4. The default <minclass> is 0, disabled. It is recommended <minclass> be increased to force the inclusion of multiple character classes in password if the credit options are not set to negative values.
- **maxsequence** – Reject new passwords which contain <maxsequence> number of monotonic characters (i.e. 3456 or abcde). The range is 0 to 20. The default <maxsequence> is 0, disabled.

The following example lists each property with their default value:

```
{
   "Plugins" : [ "sysconfig" ],
   "Session" : {
      "Password" : {
         "DifOk" : 0,
         "MaxClassRepeat" : 0,
         "MaxRepeat" : 0,
         "MaxSequence" : 0,
         "MinClass" : 0,
         "MinLength" : 8,
         "DCredit" : 0,
         "LCredit" : 0,
         "OCredit" : 0,
         "UCredit" : 0
```

```
                }
            }
        }
```

Figure 12: Example Password Rules Configuration

The password enforcement rules can be viewed by using the CLI command while in default mode:

```
admin@r0> show password
```

6.3 Administrative Session Establishment

The Machete Router can be administered both locally and remotely through CLI. Console access is always available for authorized local administrative users through an RS-232 connection. To establish a remote administrative session, SSH is required. No additional configuration is required to obscure password data.

6.4 Authentication Failure Management

Accounts which fail to authenticate remotely are locked after a configurable integer between 1 and 20, inclusively, of unsuccessful attempts. Once an account is locked, the account remains locked for a configurable integer between 1 and 3600, inclusively, duration of seconds. By default, accounts have 3 attempts to successfully authenticate before being locked for the duration of 300 seconds (5 minutes). An example of this configuration can be seen in the following subsection.

Any attempt to remotely authenticate a locked account before the account lock time has expired results in a failure and the timer is reset to the full account lock time. The account is not provided any feedback from the system that the account is locked or unlocked.

Each initiation of a remote session will prompt the user for a password 3 times, regardless of the lock attempts configured. For example, if the number of lock attempts is set to 2 and a user initiates a connection, the user will be prompted to enter the password three times, even if authentication failed the first two times resulting in the account already being locked when prompted the third time.

Additionally, any prompt for the password that occurs while the account is locked will result in a failed login attempt, even if the lock time has elapsed and the correct password is provided. For example, if the number of lock attempts is set to 2 and the lock time set to 30 seconds, when a remote session is initiated and authentication fails the first 2 password prompts and the user wait 30 seconds to enter the 3rd attempt, the 3rd attempt will still fail even though the account lock time has elapsed and the correct password is given because the prompt for the password occurred while the account was still locked.

Remotely locked accounts can always authenticate through the local console.

6.4.1 Remote Account Lockout

The administrator may configure the parameters of a remote account lockout in the Session::Lock JSON block. These parameters are described below.

- **Attempts** - The number of failed remote login attempts before the account is locked. Value must be greater or equal to 1 and lesser or equal to 20; default is 3.
- **Time** - The remote account lockout time (in seconds). Value must be greater or equal to 1 and lesser or equal to 3600; default is 300.

Alternatively, the number of attempts is default to 3 attempts before lockout and 300 seconds (5 minutes) for the account lock time.

To configure the parameters for remote account lockout, add the Session::Lock block to the ARES configuration as seen here:

```
{
   "Plugins" : [ "sysconfig" ],
   "Session" : {
      "Lock" : {
         "Attempts" : 3,
         "Time" : 300
      }
   }
}
```

This configuration would lock an account out from being accessed remotely for 300 seconds after 3 failed login attempts. Outside of the ARES configuration file, these settings can be shown by the following commands:

```
admin@r0> show lock attempts
admin@r0> show lock time
```

6.5 Unauthenticated Services

Prior to authentication, the Machete Router provides only the following unauthenticated services:

- ICMP echo
- Display of the local or remote login banner

The Machete router replies to ICMP echo requests in accordance with relevant standards.

7 Certificate And PKI Management

The user is assumed to have basic knowledge about certificate management and public key infrastructure (PKI). The user has to have a “Server” Certificate to be used on the Machete router and a “Client” certificate to be used by the Windows machine. Both sides have to have a valid and a complete certificate chain of trust at both sides for them to authenticate each other correctly. That means both sides have to be configured to trust the root certificate and any intermediate certificates for VPN to work. If your organization already has a PKI in place, you can request to have certificates to be used by the router and the Windows VPN client. You can also use one of the freely available PKI tools to generate all of the required certificates, such as:

Command line tools:

- openssl: https://www.openssl.org/docs/manmaster/man1/openssl.html
- Strongswan pki: https://wiki.strongswan.org/projects/strongswan/wiki/IpsecPKI

Graphical user interface tools:

- X - Certificate and Key management (XCA) : https://hohnstaedt.de/xca/

Regardless of which tool you use, the certificate final setup would be the same. We will not cover the details of how each tool works or how to use it but the steps and certificate requirements are covered in the subsections below. At each step you will generate a private key and a certificate. No configuration is necessary for the creation or destruction of PKI material described in this section.

7.1 Certification Validation

The Machete router performs the following checks for all types of certificates:

- Proper encoding
- Valid signature
- Current time is after the Not Valid Before value
- Current time is before the Not Valid After value
- CRL Check if configured and enabled

The router also performs the following additional check for CA certificates:

- Basic Constraints: CA=true

If the certificate is configured with extended key usage, the following checks are performed:

- Certificate Sign
- CRL Sign (for CA certificates used to verify a CRL)
- TOE and Peer Certificates

- Basic Constraints: CA=false

When it comes to revocation policy, the administrator has the option of choosing whether or not to accept the certificate if the CRL check cannot be performed. An example configuration of the revocation policy can be seen in section 8.4.4.2.

7.2 Certificate Authority (CA)

1. Generate a 4096-bit RSA private key.
2. Generate a self-signed certificate using the private key from step 1.

   Signature: SHA256 or better

   Required Certificate Extension:

   ```
   subjectAltName=email:CA-email@company.com
   keyUsage=digitalSignature, keyCertSign
   authorityKeyIdentifier=keyid, issuer
   subjectKeyIdentifier=hash
   basicConstraints=CA:TRUE
   ```

Example

X509v3 Basic Constraints:
```
CA:TRUE
```
X509v3 Subject Key Identifier:
```
96:71:CF:89:CC:18:85:5F:0B:C4:5E:4F:94:C7:29:2C:5A:E4:
7D:60
```
X509v3 Authority Key Identifier:
```
keyid:96:71:CF:89:CC:18:85:5F:0B:C4:5E:4F:94:C7:29:2C:
5A:E4:7D:60
DirName:/C=US/ST=Minnesota/L=Eden
Prairie/O=ATCorp/OU=ROUTER/CN=ATC ROUTERVPN
CA/emailAddress=routervpn@atcorp.com
serial:01
```
X509v3 Key Usage:
```
Digital Signature, Certificate Sign
```
X509v3 Subject Alternative Name:
```
email:routervpn@atcorp.com
```

7.2.1 Machete Router End Entity Server Certificate

1. Generate a 4096-bit RSA private key.
2. Generate a certificate using the private key from step 1 signed by the CA certificate you generated previously. Make sure the certificate includes the “serverAuth” extendedKeyUsage extensions (TLS Web Server Authentication).

   Signature: SHA256 or better

Required Certificate Extensions/Fields:

subjectAltName=DNS:<FQDN>, DNS:<IP>
extendedKeyUsage=serverAuth, ipsecTunnel
keyUsage=digitalSignature
authorityKeyIdentifier=keyid
subjectKeyIdentifier=hash
basicConstraints=CA:FALSE

Example

X509v3 Basic Constraints:
```
CA:FALSE
```
X509v3 Subject Key Identifier:
```
64:FE:14:D0:12:66:8E:1C:66:26:F9:B7:EE:21:99:91:7F:AF:
5E:56
```
X509v3 Authority Key Identifier:
```
keyid:16:DB:F6:36:27:6A:86:27:4A:16:EE:2C:D6:E8:E5:93:
B4:C8:B5:07
```

X509v3 Key Usage:
```
Digital Signature
```
X509v3 Extended Key Usage:
```
TLS Web Server Authentication, IPSec Tunnel
```
X509v3 Subject Alternative Name:
```
DNS:routervpn.atcorp.com, email:routervpn@atcorp.com,
DNS:204.72.168.66, DNS:routervpn2.atcorp.com,
DNS:50.255.113.91
```
Important note:

```
Windows verifies the identity of the server using the server's FQDN or
its IP address. For this to work, the server (the router in this case)
certificate must include SAN fields that match the server FQDN and/or
its IP address.  If the server has multiple FQDNs or IP addresses,
repeat the SAN DNS field as needed. Windows does not trust the server
if the server certificate does not include the TLS Web Server
Authentication extension.
```

7.2.2 Windows End Entity Client Certificate

1. Generate a 4096-bit RSA private key.
2. Generate a certificate using the private key from step 1 signed by the CA certificate you generated previously. Make sure the certificate includes the “clientAuth” extendedKeyUsage extensions (TLS Web Client Authentication).

   Signature: SHA256 or better

   Required Certificate Extenstion/Fields:

   subjectAltName=DNS:<USER NAME>, email:user@email.com
   extendedKeyUsage=clientAuth
   keyUsage=digitalSignature

```
authorityKeyIdentifier=keyid
subjectKeyIdentifier=hash
basicConstraints=CA:FALSE
```

Example

X509v3 Basic Constraints:
```
CA:FALSE
```
X509v3 Subject Key Identifier:
```
03:F5:CF:59:3C:5F:3A:7A:78:60:02:BB:A4:41:9D:0A:3C:D7:
C1:91
```
X509v3 Authority Key Identifier:
```
keyid:16:DB:F6:36:27:6A:86:27:4A:16:EE:2C:D6:E8:E5:93:
B4:C8:B5:07
```

X509v3 Key Usage:
```
Digital Signature
```
X509v3 Extended Key Usage:
```
TLS Web Client Authentication
```
X509v3 Subject Alternative Name:
```
DNS:Bob Trusty, email:btrusty@company.com
```

7.3 PKI Directories on the Machete Router

ecdsa	- Plain ECDSA private keys
pkcs10	- PKCS#10 certificate signing requests
pkcs12	- PKCS#12 containers
pkcs8	- PKCS#8 encoded private keys of any type
private	- Private keys in any format
psks	- Pre-shared keys and passwords
pubkey	- Raw public keys
rsa	- PKCS#1 encoded RSA private keys
x509	- End entity certificates
x509aa	- Attribute Authority certificates
x509ac	- Attribute certificates

x509ca	- Certificate Authority certificates
x509crl	- Certificate revocation lists
x509ocsp	- OCSP signer certificates

7.4 Generating Certificates Using the Machete Router

The Machete router supports exporting private keys allowing for PKI management using the router without relying on external resources. Here are the steps to generate a simple setup including a certificate authority, a server certificate for the router, and a client certificate to be used with a client connecting to the router.

7.4.1 Generate Private Keys

Private key generation offers a variety of options through the use of the command:

```
pki gen <key-name> <key-type> <key-length>
```

Where

`<key-name>` - The key file with extension "key", "pem", or "der". Prefix with "exported-" to auto export the key.

`<key-type>` `<key-length>` - can be either a key type of "rsa" with key length of 2048, 3072, 4096, or 8192, or a key type of "ecdsa" with a key length of 256, 384, or 521. If the key type and key length are not specified, "rsa" with the key length of 4096 are used by default.

To generate the certificate authority private key, run the *pki gen* command with the key name "ca.key". This can be seen in the example printout below.

```
admin@r0(config)> pki gen ca.key rsa 4096
Generating new 4096-bit rsa key ca.key
pki gen success:
-rw-r--r-- 1 3247 Jan 20 11:45 private/ca.key
```

To generate VPN Server Private Key

```
admin@r0(config)> pki gen router.key rsa 4096
Generating new 4096-bit rsa key router.key
pki gen success:
-rw-r--r-- 1 3243 Jan 20 11:45 rsa/router.key
```

7.4.1.1 VPN Client Private Key

Notice that the name must start with “exported-“ to allow the router to auto export the key so that it can be transferred to a client later.

```
admin@r0(config)> pki gen exported-user.key
Generating new 4096-bit rsa key exported-user.key
pki gen success:
-rw-r--r-- 1 3243 Jan 20 11:45 rsa/exported-user.key
Auto export rsa/exported-user.key
rsa/exported-user.key' -> 'staged/rsa/exported-user.key'
```

If you have more than one client, generate a new key for each client.

7.4.2 Certificate Generation

The Machete router uses certificate templates when generating certificates. There is a template for certificate authorities (CA) and another template for end entity certificates. When invoking a command to generate a certificate, the command takes a few options and then opens a certificate template in an editor to allow the user to customize the certificate. The fields that typically needs to be changed for each new certificate is the Distinguished Name, the Subject Alternative Name which can be repeated multiple times, the Flag/Attributes fields, and the validity period.

For the random bit generation (RNG Functionality), the CPU RDrand instruction is utilized which uses a hardware thermal entropy source. No configuration is necessary to enable the RNG functionality.

7.4.2.1 CA Certificate

CA certificate issuing offers a variety of options through the use of the command:

```
      pki issue <cert-name> [cert-type] [key-type] <key-file> [digest]
<template>
```

Where

`<cert-name>` - The certificate or signing request file with extension “pem”, “dir”, “crt”, or “csr”.

`[cert-type]` – Option of “ca”, “csr”, “ica”, “server”, “user”, or “x509ac” where:

- “ca” – Create a self signed certificate authority.
- “csr” – Create a PKCS#10 certificate signing request.
- “ica” – Create an intermediate certificate authority.
- “server” – Create an end entity certificate to be used for a server.
- “user” – Create an end entity certificate to be used for a user.
- “x509ac” – Create an attribute certificate.

`[key-type]` – Option of “ecdsa”, “pkcs10”, “priv”, “rsa”, or <holder-cert-file> where:

- "ecdsa", "priv", and "rsa" are available for all cert-types except x509ac.
- "pkcs10" is only available for cert-types "ica", "server", and "user".
- <holder-cert-file> is the only option available for x509ac cert-type.

`<key-file>` - The private key file with the PKI directory prefixed.

`[digest]` – Option of "sha256", "sha384", or "sha512".

`<template>` - The template to use. These templates can be found in the `template/` directory, as well as listed here:

- template/attrib-cert.cmd
- template/ca-cert.cmd
- template/cert-signing-request.cmd
- template/ica-cert.cmd
- template/server-cert.cmd
- template/sign-crl.cmd
- template/user-cert.cmd

To generate a self-signed CA certificate, run the *pki issue* command with the certificate name "ca.crt" and the previously generated private key. This can be seen in the example printout below.

```
   admin@r0(config)> pki issue ca.crt ca rsa rsa/ca.key sha256 template/ca-
cert.cmd
   pki self success:
   -rw-r--r-- 1 1842 Jan 20 11:53 x509ca/ca.crt
   Created a Self-Signed CA Certificate
```

Here are some of the fields that need to be changed to meet your application requirements including Common Name (CN), Organization (O), and Country (C):

```
# distinguished name to include as subject
--dn "C=US, O=ROUTER, CN=ROUTER CA"


# Set CA basicConstraint
--ca


# CA certificate validity period,15/01/2015 to 15/01/2035
# Date/Time format DD.MM.YY HH:MM:SS


--not-before 15.01.15 14:30:00
--not-after  15.01.35 14:30:00
```

7.4.2.1.1 Example Generation of Certificate Signing Request (CSR)

To generate a certificate signing request follow the example below. When the command brings up the template for editing, the Common Name, Organization, and Country can be edited as well as additional CSR fields.

```
admin@r0(config)> pki issue example.csr csr rsa rsa/router.key sha256
template/cert-signing-request.cmd
   [admin] pki req success
   [admin] -rw-r--r--    1 admin    admin         1805 Dec 12 19:27
```

7.4.2.2 VPN Server Certificate

VPN server certificate issuing is done through the use of the command:

```
      pki issue <cert-name> server [key-type] <key-file> <ca-cert> <ca-key>
[digest] template/server-cert.cmd
```

Where

`<cert-name>` - The certificate or signing request file with extension "pem", "dir", "crt", or "csr".

`[key-type]` – Option of "ecdsa", "pkcs10", "priv", or "rsa".

`<key-file>` - The private key or certificate signing request file with the PKI directory prefixed.

`<ca-cert>` - The CA certificate, with the `x509ca/` directory prefixed, to use.

`<ca-key>` - The CA private key, with the PKI directory prefixed, to use.

`[digest]` – Option of "sha256", "sha384", or "sha512".

To generate a server certificate, run the *pki issue* command as described above. An example of this can be seen here:

```
   admin@r0(config)> pki issue router.crt server rsa rsa/router.key
x509ca/ca.crt rsa/ca.key sha256 template/server-cert.cmd
   pki issue success:
   -rw-r--r-- 1 2009 Jan 20 12:05 x509/router.crt
```

A Machete server certificate example:

```
# distinguished name to include as subject
--dn "C=US, O=ROUTER, CN=router.net"


#
# Servers must have fqdn or IP address configured to confirm their identity
```

```
--san router.net
--san dns:192.168.10.254
--san dns:192.168.100.254
--san dns:192.168.255.1

#
# serverAuth and clientAuth flags should be enabled
#
--flag serverAuth
--flag clientAuth
--flag ikeIntermediate



# certificate validity period, default to 3 years, 15/01/2021 to 15/01/2024
# Date/Time format DD.MM.YY HH:MM:SS
#
--not-before 15.01.21 14:30:00
--not-after  15.01.24 14:30:00
```

7.4.2.3 VPN Client Certificate

VPN client certificate issuing is done through the use of the command:

```
      pki issue <cert-name> user [key-type] <key-file> <ca-cert> <ca-key>
[digest] template/user-cert.cmd
```

Where

`<cert-name>` - The certificate or signing request file with extension “pem”, “dir”, “crt”, or “csr”.

`[key-type]` – Option of “ecdsa”, “pkcs10”, “priv”, or “rsa”.

`<key-file>` - The private key or certificate signing request file with the PKI directory prefixed.

`<ca-cert>` - The CA certificate, with the `x509ca/` directory prefixed, to use.

`<ca-key>` - The CA private key, with the PKI directory prefixed, to use.

`[digest]` – Option of “sha256”, “sha384”, or “sha512”.

To generate a client certificate, run the *pki issue* command as described above. An example of this can be seen here:

```
   admin@r0(config)> pki issue user.crt user rsa rsa/exported-user.key
x509ca/ca.crt rsa/ca.key sha256 template/user-cert.cmd
   pki issue success:
   -rw-r--r-- 1 1907 Jan 20 00:19 x509/user.crted
```

An example client template:

```
# distinguished name to include as subject
--dn "C=US, O=ROUTER, CN=Joe Smith"


# Clients typically have names and email addresses


 --san dns:jsmith@email.com
 --san "dns:Joe Smith"


# By default add clientAuth flag
# Some clients also require ikeIntermediate flag
#
--flag clientAuth
--flag ikeIntermediate


# certificate validity period, default to 3 years, 15/01/2021 to 15/01/2024
# Date/Time format DD.MM.YY HH:MM:SS
#
--not-before 15.01.21 14:30:00
--not-after  15.01.24 14:30:00
```

7.4.3 Certificate Packaging

Depending on the VPN client and the platform used it might be easiest to package the client certificate and the private key along with the CA certificate in a standard pkcs12 file with a .p12 extension. This can be achieved by using the *pki pkcs12* command.

```
Pki pkcs12 <file-name> [encrypt] <password> <x509> <key-file> [chain]
```

Where

`<file-name>` - File name to save with .p12 extension. The file will be created in the staging ares.

`[encrypt]` – Option of “aes256” or “legacy” where:

- o “aes256” – Use AES256 to encrypt the file. [Recommended]
- o “legacy” – Use legacy (3des) to encrypt the file. Backward compatible with older clients.

`<password>` - Password to use to protect the file.

`<x509>` - Certificate to bundle.

`<key-file>` - The key file to bundle from the staging area (key must be exported already).

`[chain]` – Option of “chain” which will add the CA certificate chain to the bundle if available. [Optional]

To packet the VPN client PKI material, run the *pki pkcs12* command described above. An example of this can be seen here:

```
   admin@r0(config)> pki pkcs12 user.p12 aes256 password x509/user.crt
rsa/exported-user.key chain
   pki pkcs12 success:
   -rw-r--r-- 1 5485 Jan 20 00:21 staged/pkcs12/user.p12
```

7.4.4 Copying Certificate to Clients

PKI materials on the router are kept in two places, the deployed or operational area and the staged area. The staged area is used for transferring data on/to from Machete. Data can be exported to the staged area from the operational area, or imported from the staged area to the operational area. **pki import** and **pki export** commands can be used for this purpose and both are described in Chapter 6 in Machete User Manual.

Any certificate or .p12 files created with a name that starts with “exported-” will be automatically exported to the staging area. The easiest way to download certificates to client is to use Machete’s on-demand http server. To run the http server to serve the PKI material, do:

```
   admin@r0> httpserver pki
   Starting web server at http://0.0.0.0:80. press <Ctrl-C> to stop
```

This runs the server on the standard http port 80. This port is blocked by the Machete’s firewall by default. To allow access to it add a rule to allow TCP on port 80 to the firewall configuration file. Alternatively, you can set the filter policy for input and output to “allow”:

```
admin@r0(config)> firewall policy input accept
User admin attempting to restart service firewall-post
firewall-post restart successful

admin@r0(config)> firewall policy output accept
User admin attempting to restart service firewall-post
firewall-post restart successful

admin@r0> show firewall policy
Firewall Mode: Custom

Current Policy Settings
   INPUT -   ALLOW
   OUTPUT -  ALLOW
   FORWARD - ALLOW
admin@r0>
```

If you stopped the http server to run the above command make sure to run it again. You can run a web browser on your clients to download their certificates or the .p12 files.

7.5 Machete configuration

Machete supports certificates in text and binary formats. You can generate, stage, import, and export individual PKI material using the pki command from the configuration mode. The following section show how to generate an asymmetric key and export its public key

7.5.1 Steps to generate and export keys**:**

1. Generate the key:

```
admin@r0(config)> pki gen r2.key
Generating new 4096-bit rsa key r2.key
[admin] pki gen success
[admin] -rw-r--r--    1 admin    admin         3243 May  9 16:00 rsa/r2.key
admin@r0(config)>
```

2. Extract the public key of the newly generated key

```
admin@r0(config)> pki pubkey r2.pub rsa rsa/r2.key
[admin] pki pub success
[admin] -rw-r--r--    1 admin    admin          800 May  9 16:04 pubkey/r2.pub
admin@r0(config)>
```

3. Export the public key to the staging ares:

```
admin@r0(config)> pki export
Available commands:
   pki export <file-name> - File name
   pki export all      - Export all certificates to the PKI staging area
   pki export pkcs10   - PKCS#10 certificate signing requests
   pki export pkcs8    - PKCS#8 encoded private keys of any type
   pki export pubkey   - Raw public keys
   pki export template - Template files
   pki export x509     - End entity certificates
   pki export x509aa   - Attribute Authority certificates
   pki export x509ac   - Attribute certificates
   pki export x509ca   - Certificate Authority certificates
   pki export x509crl  - Certificate revocation lists
   pki export x509ocsp - OCSP signer certificates
admin@r0(config)> pki export pubkey
   [admin] 'pubkey/r2.pub' -> 'staged/pubkey/r2.pub'
admin@r0(config)>
```

4. Copy the key to another machine

```
admin@r0(config)> pki push
Available commands:
   pki push <filename> url - The address or FQDN of the remote host
admin@r0(config)> pki push pubkey/r2.pub scp://admin@192.168.229.120:~/
Pushing pki file r2.pub to admin@192.168.229.120:~/ ...
admin@192.168.229.120's password:
r2.pub
[admin] pushed pki file staged/pubkey/r2.pub to admin@192.168.229.120:~/
admin@r0(config)>
```

7.5.2 Checking Authentication Status

If authentication with certificates fails during operation the details of the validity check can be found by checking the vpn log.

```
admin@ro show log vpn.log
```

7.6 Key Destruction

The Machete router destroys keys in RAM by writing a dynamic value to the memory address(es) containing the key being destroyed. The TSF destroys keys in Flash by logically addressing the storage address and overwriting the key with four different static patterns (i.e. 0x00, 0xFF, 0xAA, and 0x55). The router performs a read-verify of the 0x55 pattern after the final write. The router logs a key destruction error if the read-verify fails. Examples of successfully key destruction can be found in section 11.4.2.

8 ARES Router Management Service

The ARES service is one of the main configuration services that makes up the Machete router. All ARES configurations are written in the JSON format and stored in the ares.json file. For more information on the JSON syntax, refer to the section 2.3.3.

In order to enter into a text editing session for the ares.json file and configure the ARES service, while in configuration mode, run the CLI command:

```
admin@r0(config)> ares
```

When done, use "ctrl-x" to exit the text editing session. If changes have been made, the user will be prompted to save the modified buffer and be given the opportunity to rename the file or keep the same name. The router will then validate the configurations and JSON schema.

If validation fails, the prompt will request to either return to the text editing session or discard the changes. This validation process ensures proper and secure functionality of the router. There are two primary reasons for validation to fail:

- JSON Syntax Errors - These errors are related to the JSON format of brackets, commas, and key-values. If a JSON file is missing one of these notational characters the validator will notify the user of the type of syntax error and where in the file the error occurred.
- Schema Errors - These errors happen when a user enters an invalid key-value pair into the json structure. The validator will notify the user which values are invalid and also show the format of valid entries as an example.

If validation passes, the prompt will request to restart the service. ARES configuration changes do not take effect until the ARES service is restarted. Also note that some configurations may require the user to log out and log back in before the changes will be able to take effect.

To restart the ARES service while in configuration mode, enter the CLI command:

```
admin@r0(config)> do service ares restart
```

To view the current running status of the ARES service while in configuration mode, enter the CLI command:

```
admin@r0(config)> do service ares status
```

`Note: The above two commands can also be executed from the default mode in the same manner without the prefixed` *`do`* `command.`

During an ARES service restart, the default firewall rules are enabled and all network traffic is dropped until the ARES service has returned to a running state and the user-defined firewall rules are re-enabled.

This section focuses heavily on the configurations relating to ARES and setting up primary functionalities of the Machete router, such as VPN connections and the Wi-Fi capabilities, to run in the evaluated configuration. For more information on the ARES service, refer to the ARES Configurations chapter of the Machete User Guide.

8.1 General Configurations

The configurations in this section are the general configurations for the Machete router. These parameters are directly associated to the device itself, such as the router's id, name, and the security mode that the router will run in.

```
{
   "RouterId" : 10,
   "RouterName" : "r0",
   "SecurityMode" : "niap",
   "Log" : { "Verbosity": "warning" },
   "Plugins" : [ "sysconfig" ],
   …
}
```

These configurations are parameters that are configured at the top-level of the JSON structure in the ares.json file. Top-level refers to the first key-value pairs within the first bracket.

8.1.1 Router ID

Each Machete router must have a unique router ID associated with it within the network. To configure the router ID, set the following ARES configuration parameter:

```
RouterId
```

The router ID can be any unsigned integer in the range 0-4294967295, inclusive.

Example:

```
{
   "RouterId" : 42
}
```

8.1.2 Router Name

The Machete router name/hostname can be configured by setting the following ARES configuration parameter:

```
RouterName
```

The router name can be any Alpha Numeric string and may include underscores (_) and/or dashes (-).

Example:

```
{
   "RouterName" : "machete-router-test"
}
```

The system hostname is synced with the configured router name. If no router name is configured, then the pre-configured hostname is used as the router name.

```
Note: In order see some changes to take effect, you will
have to log out and log back in.
```

8.1.3 Security Mode

ARES has preset validation schemas that adhere to the security mode set by the following parameter:

```
SecurityMode
```

The Machete router will utilize the schema associated to the selected security mode when validating the configurations set in the ares.json file.

When you first start using the Machete router, no security mode is specified so the "normal" mode is selected by default. To switch the security mode used, add the parameter mentioned above to the ARES configuration file with one of the available modes listed here:

- Normal
- NIAP
- CSFC

Example:

```
{
     "SecurityMode" : "niap"
}
```

8.1.4 Plugin Configuration

The ARES service supports dynamically loadable plugins. These plugins are:

- "wlanmanager"
- "vpn"
- "flowredirector"
- "freeboard"
- "mantra"
- "mre"
- "radiocontroller"
- "reconfiguration"
- "visualizer"
- "tscanr"
- "bit"
- "zoom"

- “grekeepalive”
- “zoomclient”
- “sysconfig”

When configuring the ARES service, some configurations will require one of the plugins mentioned above to be loaded in to enable the related features. ARES plugins are configured as an array of strings under the ARES configuration parameter:

```
Plugins
```

Example:

```
{
   "Plugins" : [ "…", "…" ]
}
```

Where “…” is replaced with the desired plugin from the list above.

Some plugins will require setting the SecurityMode parameter in order to use. In these cases, refer to the *Security Mode* section.

8.1.5 Log

The Machete router offers logging options that are configurable through ARES by setting the following ARES configuration parameter:

```
Log
```

The non-audit log messages are split among the 4 files listed below:

- kern.log – Stores messages generated by the kernel and the ARES kernel module
- user.log – Stores messages from the network/routing services
- general.log – Stores messages from any non-audit services
- syslog – Stores all other system messages

To view all the audit messages stored in a log or audit file, while in default mode, enter CLI command:

```
admin@r0> show log <filename>
```

Once this command is entered, the given log or audit file is opened using the UNIX program “less” in secure mode. The *less* program is a simple to use and powerful text file viewer.

Enter ‘h’ or ‘H’ from within the less viewer to view a summary of available features, and the key commands to activate them. Below is a condensed list of the most useful features of *less*.

UpArrow/DownArrow – Scroll up/down one line at a time

PgUp/PgDown – Scroll up/down one page at a time

/pattern – Search forward for the next line that contains the given pattern

?pattern – Search backward for the previous line that contains the given pattern

n – Repeat the previous search

N – Repeat the previous search, searching in the opposite direction

&pattern – Display only the lines that contain the given pattern

F – Display new messages written to the file, as they are written. Use ctrl+c to stop.

q – Exit the log viewer

The "show log <filename>" command automatically opens the audit file scrolled to the end of the file rather than at the beginning of the file. When searching from the end of the file, use the '?pattern' and 'n' commands in order to search backward through the file.

The following subsections can be used to specify the types of activities the router logs and where to send them.

For more information on the Machete router's logging behavior, refer to section 10.

8.1.5.1 Dest

The "Dest" ARES configuration parameter can be used to specify the destination of logs. This parameter can be set using one of the following strings.

- "StdOut" - print logs to standard output
- "Syslog" - send to remote destination
- <File_Dir> - must be a string /^[a-zA-Z0-9_/\. \-\\]{2,}$/

The strings "StdOut" and "Syslog" are special destinations, while all others are interpreted as a file name.

The following is an example of setting the destination to the "temp.log" file:

```
{
     "Log" : {
          "Dest" : [ "temp.log" ]
     }
}
```

8.1.5.2 Buckets

Array of log bucket names enabled for logging. Some buckets have sub-bucket options, if the top-level bucket is enabled, that enables all the sub-buckets as well. Default is all enabled.

The following is the list of all log buckets:

- general
- packet
- packeticmp
- packetigmp
- packetunicast
- packetmulticast
- packetudp
- packettcp
- packetme
- tunnel
- neighbor
- frr
- frrospf
- frrpim
- frrstatic
- frrzebra
- link
- linktable
- linkstate
- linkquality
- ipc
- ipclink
- ipcoption
- ipcpolicy
- ipcpacket
- aresh
- system
- plugin
- wlanmanager
- vpn
- flowredirector
- freeboard
- mantra
- mre
- radiocontroller
- reconfiguration
- visualizer
- tscanr
- bit
- zoom
- zoomclient
- dhcp

If only one or two of these buckets are needed, the following can be implemented in the ARES configuration:

```
Log::Buckets::[ "…", "…" ]
```

Where "…" are the desired log buckets to be enabled.

8.1.5.3 Log Verbosity and Kernel Log Verbosity

The Verbosity and Kernel Verbosity parameters nested under Log allows control over the level of verbosity when writing log messages. For both Verbosity and Kernel Verbosity, only error log messages are written by default. The ARES configuration only accepts one of the strings from the following list:

- "critical"
- "error"
- "warning"
- "notice"
- "info"
- "debug"
- "uber"

To set the verbosity, add either or both of the following configurations below with the desired verbosity:

```
{
     "Log": {
          "Verbosity" : "info"
          "KernelVerbosity" : "debug"
     }
}
```

8.1.5.4 Audit

The Machete router stores log messages which are generated by auditable events in a local storage. The local audit event storage is monitored and maintained to ensure that new audit events are always logged. These local audit event messages are split into four types logged to the files listed below:

- vpn.log – Stores all VPN related events
- firewall.log – Stores all Firewall related events
- auth.log – Stores all authentication related events
- general.log – Stores all other audit events

Audit event logs may also be sent to external audit event storage through the remote syslog protocol. Implementation of this can be found in section 10.3.

To see the remote audit log settings, in default mode, run the CLI command:

```
admin@r0> show audit remote
Server         TCP|UDP  Port     Trusted Channel
A.A.A.A        tcp      514      default
Admin@r0>
```

The *show audit remote* command displays a table for the current remote audit log settings. The information found here will reflect what was set up for the remote syslog.

8.1.5.5 Logging Configuration Parameters

The Logging configuration is a top-level JSON object as below. A detailed list of child parameters can be found in section 2.3 “Log” of the NIAP Configuration Parameters document.

```
{
   "Other-top-level-blocks": {…},
   "Log":{
      "…"
   }
}
```

8.2 Links

Every network interface on the Machete router that needs to be managed must be configured as an ARES link in the ARES configuration file. Each ARES link has an associated link type and maps to a single network device on the system. ARES links are configured as an array of JSON objects under the ARES configuration parameter:

```
Links
```

Example:

```
{
   "Links" : [ {…}, {…} ]
}
```

Where "…" is replaced with the JSON object configuration for each managed interface.

8.2.1 Setting up a Link

When configuring a link, there must be a name, type, and interface specified in order for the link to be valid.

8.2.1.1 Link Name

Each ARES link must be given a descriptive name. The link name is set by configuring the following parameter:

```
Links::<link>::Name
```

The ARES link name can be any alpha numeric string and may include underscores and/or dashes.

Example:

```
{
   "Links" : [
      {
         "Name" : "ITS_Link"
      }
   ]
}
```

8.2.1.2 Link Type

Each ARES link must be assigned a link type. There are several types of links corresponding to different connections. A link is where IP configuration takes place: it's where you set addresses and subnet masks and specify how the interface interacts with chosen routing protocols. The table below shows the link types.

Type	Description
`Simple`	Manages the device but no extra services by default
`Wan`	Adds Source NAT feature
`Host`	Adds advertise over OSPF
`Boundary`	Adds advertise and performs OSPF
`Core`	Connection to another Machete router. Performs OSPF and adds Neighbor Discovery and Link Quality control.
`Edge`	Connection to another Machete router with no OSPF.

Figure 13: Link "Type" Table

The link type is set by configuring the following parameter:

```
Links::<link>::Type
```

Example:

```
{
  "Links" : [
    {
      "Name" : "LAN",
      "Type" : "simple"
    }
  ]
}
```

8.2.1.3 Link Interface

Each ARES link must map to a network interface present on the system. To map an ARES link to a network interface, the following configuration parameter must be set:

```
Links::<link>::Interface
```

This parameter is a JSON object that contains several parameters that define the network interface and the information needed to manage it.

Example:

```
{
  "Links" : [
    {
      "Name" : "LAN",
      "Type" : "simple",
      "Interface" : {…}
    }
  ]
}
```

Where "…" is replaced with the JSON object parameters which define the "Interface" configuration.

8.2.1.3.1 Link Interface Hardware Address

The Hardware Address for the network device is used as the primary identifier for the device and is configured by setting the following parameter:

```
Links::<link>::Interface::HardwareAddress
```

Example:

```
{
  "Links" : [
    {
      "Name" : "LAN",
      "Type" : "simple",
      "Interface" : {
        "HardwareAddress" : "00:11:22:33:44:55"
      }
    }
  ]
}
```

8.2.1.3.2 Link Interface Name

The name of the mapped network device can be changed by setting the following parameter:

```
Links::<link>::Interface::Name
```

If the interface name is not configured, the existing name of the device identified by the hardware address is used. The interface name must be between 1 and 15 characters in length, inclusive, must begin with a letter (case insensitive) or number, and may also contain dashes, semicolons, underscores, and/or periods.

Example:

```
{
  "Links" : [
    {
      "Name" : "LAN",
      "Type" : "simple",
      "Interface" : {
        "HardwareAddress" : "00:11:22:33:44:55"
        "Name" : "eth0"
      }
    }
  ]
}
```

8.2.1.4 Link IPv4 and IPv6 Static Routes

IP (version 4 and 6) subnets which are reachable via the link interface may optionally be configured per link by setting the following parameter:

```
Links::<link>::Routes
```

The Routes parameter is a JSON array of JSON objects, where each object must define the reachable subnet and may optionally define the next hop address. A default route may also be configured by setting the “Network” parameter to “default”. There may only be one default route defined for IPv4 and only one default route for IPv6 per link configuration.

```
{
  "Links" : [
    {
      "Name" : "LAN",
      "Type" : "simple",
      "Interface" : {
        "HardwareAddress" : "00:11:22:33:44:55"
        "Name" : "eth0"
      }
      "Routes" : [
        { "Network" : "192.168.1.0/24"},
        { "Network" : "172.168.2.0/24", "Via" :
"192.168.1.1"},
        { "Network" : "default", "Via" : "192.168.1.100"},
        { "Network" : "2002::0/64"},
        { "Network" : "2200::0/64", "Via" : "2002::1"},
        { "Network" : "default", "Via" : "2002::100"}
      ]
    }
  ]
}
```

Figure 14: Example Static Routes Configuration

8.2.2 Link Configuration Parameters

The Link configuration is a top-level JSON object as below. A detailed list of child parameters can be found in section 2.6 “Links” of the NIAP Configuration Parameters document.

```
{
   "Other-top-level-blocks": {…},
   "Links": [
      "… "
   ]
}
```

8.3 SSH Configuration

This section describes configuration of SSH server and SSH client. SSH is used to access the Machete router remotely and to transfer firmware updates to the device. The confirmation options closely follow the OpenSSH documentation found at https://www.openssh.com/manual.html. Some of the OpenSSH options are not configurable on the Machete router.

8.3.1 SysConfig Plugin

Both the SSH server and the SSH client are enabled by default in the Machete configuration, however it is likely that there are options, especially relating to keys/certificates the user would like to configure. In order to configure SSH server or Client the ARES SysConfig plugin must be loaded. To load the SysConfig plugin, add it to the list of enabled ARES plugins:

```
{
  "Plugins": ["sysconfig"]
}
```

8.3.2 Security Mode

Both SSH server and SSH client adhere to the Machete security mode setting configured as follows as a top-level option in the ARES configuration file:

```
{
  "SecurityMode": "niap"
}
```

8.3.3 Certificate and PKI Requirements

Certificates and keys are generated and stored to be used for SSH. To generate keys and certificates, see ***Certificate And PK*** Chapter.

8.3.4 SSH Server

When the sysconf plugin is enabled, the SSH server is able to be configured. The following is a skeleton configuration block to configure SSH server:

```
{
  "Plugins": ["sysconfig"],
  "SSH": {
    "Server": {
      "Enable": true
    }
  }
}
```

The “Enabled” option defaults to true and setting this option to false will disable the SSH server. The following is a more complicated example that sets the Security Mode to NIAP, defines the ports the SSH server is listening on as well as private keys and certificate the server is to use. The keys and certificates are assumed to be generated prior to configuring ares.

```
{
  "Plugins": ["sysconfig"],
  "SecurityMode": "niap",
  "SSH": {
    "Server": {
      "Enable": true,
      "HostKey": ["file://hello.psk", "file://world.psk"],
      "HostCertificate": "file://helloworld.crt",
      "Ports": [2222, 2223]
    }
  }
}
```

8.3.5 SSH Client

The SSH client configuration allows different settings to be configured for multiple hosts. The following is an example that sets the Security Mode to "niap", the remote host to 192.168.1.1:2222, and the user account to log in as to "testuser".

```
    {
      "Plugins": ["sysconfig"],
      "SecurityMode": "niap",
      "SSH": {
        "Client": [
          {
            "Host": "192.168.1.1",
            "Port": 2222,
            "User": "testuser"
          }
        ]
      }
    }
```

SSH client can then be used to connect to the host by using the following command:

```
admin@r0> net ssh connect 192.168.1.1
```

Alternatively, the user to log in as on the remote host can be defined on the command line:

```
admin@r0> net ssh connect 192.168.1.1 testuser
```

Global defaults for the SSH client configuration can be configured by setting `"Host":"*"`, and if configured should be the last host in the array. If not configured, a small set of global default settings will be automatically applied.

8.3.5.1 Remote SSH Management

Connections to remote SSH hosts rely on host keys stored in the system-wide known_hosts file. A set of commands under `net ssh known-hosts` can be used to display the contents of the known_hosts file or remove a stored host key. The following are the command help outputs:

```
admin@Router> net ssh known-hosts ?
Available commands:
net ssh known-hosts delete - Delete stale host fingerprint
net ssh known-hosts print - Print known_hosts file
```

The `net ssh copy-id` command is used to copy a local SSH ID to a remote host in order to enable public key authentication to that host. The ID copied should then be listed in the `IdentityFile` array for the specified host in the SSH client configuration.

```
admin@Router> net ssh ?
Available commands:
net ssh connect     - Connect to a remote host via SSH
net ssh copy-id     - Copy SSH ID to a remote host
net ssh known-hosts – Manage known hosts
```

8.3.5.2 Automatic Connections

The Machete router may be configured to create persistent SSH tunnels that automatically connect after service restarts. This requires the generation of an RSA or ECDSA private key as shown in section 7.4.1, for an SSH ID to be copied from that key to the remote host at the other end of the tunnel, and for that `IdentityFile` to be listed in the SSH Client configuration for that host. First, generate a private key:

```
admin@r0(config)> pki gen tunnel-ssh.key ecdsa
Generating new 384-bit ecdsa key tunnel-ssh.key
pki gen success:
-rw-r--r-- 1 admin admin 288 Nov 22 21:36 ecdsa/tunnel-ssh.key
```

Copy the identity from that key to the remote host:

```
admin@r0> net ssh copy-id 192.168.1.1 testuser global ecdsa ecdsa/tunnel-ssh.key
```

After the previous steps have been completed, the following configuration creates a simple persistent tunnel that forwards traffic from the local port 10514 to port 514 on the remote host:

```
{
  "Plugins": [ "sysconfig" ],
  "SecurityMode": "niap",
  "SSH": {
    "Client": [
      {
        "AutoConnect": true,
        "Host": "192.168.1.1",
        "User": "testuser",
        "IdentityFile": [ "file://ecdsa/tunnel-ssh.key" ],
        "LocalForward": [
          {
            "Port": 10514,
            "Host": "127.0.0.1",
            "HostPort": 514
          }
        ]
      }
    ]
  }
}
```

The `DynamicForward` option may be configured to make the SSH client act as a SOCKS proxy. Tunnels may also be set up to forward traffic from the remote host by configuring the `RemoteForward` option. Multiple forwards of each type may be configured for a single host.

Active SSH tunnels can be viewed using the command `show ssh tunnel`.

8.3.6 SSH Session Rekey Enforcement

The router may be configured to enforce SSH session rekeying events either after a time limit, of 1 second up to 1 hour, or a data limit, of 1K up to 1G of data, has been reached. The rekey limit parameters are described in greater detail below.

> **RekeyLimit::MaxTime** - Maximum amount of time that may pass before the session key is renegotiated. Specified in seconds, minutes, or hours with a corresponding suffix of 's', 'S', 'm', 'M', or 'h', 'H'. If no suffix is provided, the default format is seconds. This can be configured to anything from 1 second up to 1 hour.
>
> **RekeyLimit::MaxData** - Maximum amount of data transmitted before the session key is renegotiated. Specified in kilobytes, megabytes, or gigabytes with a corresponding suffix of 'k', 'K', 'm', 'M', or 'g', 'G'. This can be configured to anything from 1 kilobyte up to 1 gigabyte of data.

Rekeying can be initiated by both the SSH client and SSH server by adding the “RekeyLimit” JSON block to the SSH::Client and SSH::Server JSON blocks. The following is an example implementation of this configuration for an SSH client.

```
{
  "Plugins": [ "sysconfig" ],
  "SecurityMode": "niap",
  "SSH": {
    "Client": [
      {
        "Host": "192.168.1.1",
        "Port": 12345,
        "User": "testuser",
        "RekeyLimit": {
          "MaxData": "50K",
          "MaxTime": "5m"
        }
      }
    ]
  }
}
```

The configuration of the "RekeyLimit" for the SSH::Server can be achieved by adding the same "RekeyLimit" JSON block as seen in the example above.

8.3.7 SSH Cryptographic algorithms

The Machete router supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The router supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521.

The following configurations contain all of the available cryptographic algorithms for SSH described in this section. The "Ciphers", "KexAlgorithms", and "MACs" parameters may contain a subset of these algorithms. There are no additional configurations necessary to make use of these algorithms.

Server Configuration:

```
      {
        "Plugins": [ "sysconfig" ],
        "SecurityMode": "niap",
        "SSH": {
          "Server": {
            "Ciphers": ["aes128-cbc", "aes256-cbc", "aes128-ctr",
      "aes256-ctr", "aes128-gcm@openssh.com", "aes256-
      gcm@openssh.com"],
            "KexAlgorithms": ["ecdh-sha2-nistp256", "diffie-
      hellman-group14-sha256", "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512", "ecdh-sha2-nistp384",
      "ecdh-sha2- nistp521", "diffie-hellman-group14-sha1"],
            "MACs": ["hmac-sha1", "hmac-sha2-256", "hmac-sha2-
      512", "hmac-sha1-96" ]
          }
        }
      }
```

Client Configuration:

```
{
  "Plugins": [ "sysconfig" ],
  "SecurityMode": "niap",
  "SSH": {
    "Client": [
      {
        "Ciphers": ["aes128-cbc", "aes256-cbc", "aes128-
ctr", "aes256-ctr", "aes128-gcm@openssh.com", "aes256-
gcm@openssh.com"],
        "KexAlgorithms": ["ecdh-sha2-nistp256", "diffie-
hellman-group14-sha256", "diffie-hellman-group16-sha512",
"diffie-hellman-group18-sha512", "ecdh-sha2-nistp384",
"ecdh-sha2- nistp521", "diffie-hellman-group14-sha1"],
        "MACs": ["hmac-sha1", "hmac-sha2-256", "hmac-sha2-
512", "hmac-sha1-96" ],
        "Host": "192.168.1.1",
        "Port": 12345,
        "User": "testuser"
      }
    ]
  }
}
```

Note that neither configuration uses "none" as one of the MAC algorithms, as it is not allowed in the NIAP security mode.

8.3.8 SSH Configuration Parameters

The SSH configuration is a top-level JSON object that contains the Client and Server configuration blocks as below. A detailed list of child parameters may be found in section 2.39 "SSH" of the NIAP Configuration Parameters document.

```
{
   "Other-top-level-blocks": {…},
   "SSH":{
      "Client": […],
      "Server": {…}
   }
}
```

8.4 VPN Configurations

The Machete router offers configurable Virtual Private Network connections to provide a secure communication between remote computers across a public Wide Area Network (WAN). The router implements strongSwan which is a modular and portable IPsec-based VPN solution that supports IKEv1, IKEv2, and ESP to establish secure connection.

The Machete router stores the VPN configurations in the ares.json file that maps one-to-one with the strongSwan parameters for configuring the VPN. For more information on the strongSwan parameters, refer to https://docs.strongswan.org/docs/6.0/swanctl/swanctlConf.html. Many of the most commonly used parameters are also covered in this section through example Machete router configurations.

To edit the VPN configurations, enter configuration mode and run the CLI command:

```
admin@r0(config)> ares
```

Once the *ares* command is entered, an editor session is started allowing the user to edit the ares.json file.

8.4.1 VPN Plugin

In order to configure the VPN, the ARES VPN plugin must be loaded. To load the VPN plugin, add it to the list of enabled ARES plugins:

```
{
  "Plugins": ["vpn"]
}
```

8.4.2 Security Mode

VPN connections adhere to the Machete security mode setting configured as follows as a top-level option in the ARES configuration file:

```
{
  "SecurityMode": "niap"
}
```

8.4.3 Connections

VPN configuration is centered on the configuration of connections. To add new VPN connections, configure the following parameter:

```
VPN::Profiles::onnections::<connection-name>
```

Connections is a JSON object that lists JSON objects.

Example:

```
{
  "VPN" : {
    "Profiles": {
      "connections" : { "conn1":{…}, "conn2":{…} }
    }
  }
}
```

Where "…" is replaced with the JSON object configuration for each connection with the name associated.

Each connection configuration must be given a descriptive name. The name can be any alpha numeric string and may include underscores and/or dashes. The connection name is used to identify individual VPN connections in the configuration file.

8.4.4 Example VPN Configurations

The Machete router can be configured to only allow VPN client connections from specific IP addresses by specifying them in the "remote_addrs" field. Further restrictions for time of day and week can be specified in the firewall. See section 9.3 to setup the denial of session establishment.

8.4.4.1 VPN Configuration using Pre-shared Keys

The following ARES example shows the configuration for the connection r3's local, remote, and child connections. Additionally, the secrets parameter is configured to use a weak password for "ike-r3". The router enforces an 8 to 130 character password length with any uppercase lowercase, number, or special symbol: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')', '+', '/', '-', '_', '=', '?', '<space>.

```
      {
         "RouterId" : 10,
         "SecurityMode" : "niap",
         "Plugins": [  "sysconfig", "vpn" ],
         "VPN": {
           "Profiles": {
             "connections": {
               "r3": {
                 "local_addrs" : ["10.0.1.1"],
                 "remote_addrs" : ["10.0.2.3"],
                 "local": {
                   "auth": "psk",
                   "id" : "10.0.1.1"
                 },
                 "remote": {
                   "auth": "psk",
                   "id" : "10.0.2.3"
                 },
                 "children": {
                   "red": {
                     "local_ts": [ "10.0.0.0/24" ],
                     "remote_ts": [ "10.0.3.0/24" ],
                     "start_action" : "trap|start"
                   }
                 }
               }
             },
             "secrets": {
               "ike-r3": {
                 "id": "10.0.2.3",
                 "secret": "my-not-very-strong-password!"
               }
             }
           }
         }
      }
```

8.4.4.2 VPN Configuration using Certificates

Certificates can be generated using the Certificate and PKI management tools on the router. Refer to section 7 for more information on how to generate and modify the desired certificates and keys. The Machete router supports the use of SAN or Domain name to reference the connection id. The “proposals” JSON block may be configured as desired using any of the NIAP compliant algorithms listed in section 8.4.5.

The Machete router automatically performs certificate validation for VPN connections with certificate authentication by default. This behavior can be configured in the VPN::Modules::charon-systemd::plugins::revocation object. The revocation parameter is also configurable for a desired location, local or remote of a configured connection (VPN::Profiles::connections::r3::remote::revocation, in the example below). Certificate revocation policy for CRL revocation can be one of the following string options.

- **strict** - Fails if no revocation information is available, i.e. the certificate is not known to be unrevoked
- **ifuri** - Fails only if a CRL URI is available but certificate revocation checking fails, i.e. there should be revocation information available but it could not be obtained
- **relaxed** - Fails only if a certificate is revoked, i.e. it is explicitly known that it is bad

These configurations can be seen in the example below.

```
      {
        "RouterId" : 10,
        "SecurityMode" : "niap",
        "Plugins": [  "sysconfig", "vpn" ],
        "VPN": {
          "Modules": {
            "charon-systemd": {
              "signature_authentication": true,
              "plugins": {
```

```
          "revocation": {
            "load": true,
            "enable_crl": true
          },
        }
      }
    },
    "Profiles": {
      "connections": {
        "r3": {
          "version" : 2,
          "proposals": {
            "list": [
              {
                "encryption_algorithms": ["aes128"],
                "integrity_algorithms": ["sha256"],
                "pseudo_random_functions": ["prfsha256"],
                "ke_groups": ["modp2048"]
              }
            ]
          },
          "local_addrs": [ "10.0.1.1" ],
          "remote_addrs": [ "10.0.2.3" ],
          "local": {
            "auth": "pubkey",
            "certs" :  ["file://router.crt"],
            "id": "SAN field from the certificate such as
IP address, FQDN, email, etc"
          },
          "remote": {
            "auth": "pubkey",
            "revocation": "ifuri"
          },
          "children": {
            "red": {
              "esp_proposals": {
                "list": [
                  {
                    "encryption_algorithms": [ "aes128" ],
                    "integrity_algorithms": ["sha256",
"sha384", "sha512"]
                  }
                ]
              },
              "local_ts": [ "10.0.0.0/24" ],
              "remote_ts": [ "10.0.3.0/24" ],
              "start_action": "trap|start"
```

```
                        }
                    }
                }
            }
        }
    }
}
```

8.4.4.3 VPN Configuration with Additional Options

This configuration includes options with some default fields included to illustrate the full configuration option. The “tunnel-conn” connection drops SSH traffic from specific addresses, bypasses ICMP traffic, and uses tunnel mode for its child SA of the same name. IKEv1 and IKEv2 can be explicitly configured using the version field in the connection_base object. A value of 1 uses IKEv1 (aka ISAKMP), 2 uses IKEv2. A connection using the default of 0 accepts both IKEv1 and IKEv2 as a responder and initiates the connection actively with IKEv2. The SA lifetime of a connection can be configured by using the “rekey_time” parameter. In addition to time the child SA lifetime can be configured to rekey after transmitting a data limit. This can be configured with the “rekey_bytes” parameter of the child connection with a minimum of 10 million and maximum of 4 billion bytes. Child connections can also be configured to be terminated after a period of inactivity. This can be configured by adding the “inactivity” parameter to the child connection. The Machete router supports using Distinguished Name (DN) and Subject Alternate Name (SAN) for connection identifiers as well as IP addresses with their respective local, remote, and child fields.

The Security Policy Database (SPD) is also configured here by setting the “mode” parameter for each of the child configurations. By setting the “mode” to “pass”, packets will bypass the tunnel. Setting it to “drop” will drop the packets. To encrypt traffic for a connection, set the “mode” parameter to “tunnel” or “transport”.

Note that the same format that is used for the default field can be used for each connection and will override the default when there is a difference. This enables an administrator to ensure security at the tunnel policy level while keeping the verbosity of the configuration file low. Additionally, this requires pass/drop policies to have a higher priority (lower numeric value) than the tunnel policies, otherwise they will not be applied. To verify these priorities, use the command `show vpn policy-db`. To manually set the priority, add the “priority” parameter to the child policy and set it to the desired integer. By default, the “priority” is set to 0, which dynamically calculates priorities based on the size of the traffic selectors. Policies must be configured on all peers.

In order for these policies to take effect and child SAs to be established, they will need to be initiated based on the “start_action” parameter. Pass and drop policies should have a “start_action” of “trap”. If “start_action” is set to “none” for a child SA configuration (“mode” of “tunnel” or “transport”), it will have to be started manually through the use of the following command:

```
service vpn initiate [child|ike] <child_name|ike_name>
```

Initiating an IKE SA manually will not automatically install any child SAs but initiating a child SA manually will first establish its parent IKE SA.

```
{
  "RouterId" : 10,
  "RouterName" : "r0",
  "SecurityMode" : "niap",
  "Session" : { "Timeout" :  900 },
  "Log" : { "Verbosity" : "warning" },
  "Plugins": [ "sysconfig", "vpn" ],
  "Multicast": {
    "Enabled" : true,
    "PIM" : { "RP-List": [{"RP": "127.0.0.1", "Groups":
["224.0.0.0/4"]}]}
  },
  "Links" : [
    {
      "Name" : "LAN",
      "Type" : "simple",
      "Interface" : {
        "Name" : "eth0",
        "Inet4Addresses" : "auto",
        "KeepDHCP4Gateway": true
      }
    },
    {
      "Name" : "WAN",
      "Type" : "simple",
      "Interface" : {
        "Name" : "eth1",
        "Inet4Addresses" : [ "192.168.100.254/24" ]
      }
    }
  ],
  "VPN": {
    "Modules": {
      "charon-systemd": {
        "signature_authentication": true,
        "plugins": {
          "bypass-lan": {
            "load": false
          }
        }
      }
    },
    "Profiles": {
      "IKEStrongerThanESP": true,
      "connections": {
        "tunnel-conn": {
          "local_addrs": [ "192.168.100.254" ],
          "remote_addrs": [ "192.168.100.253" ],
          "local": {
            "auth": "psk",
            "id": "192.168.100.254"
          },
          "remote": {
            "auth": "psk",
            "id": "192.168.100.253"
          },
          "rekey_time": "4h",
          "children": {
            "pass-icmp": {
              "local_ts": [ "192.168.100.254[1]" ],
              "remote_ts": [ "192.168.100.253[1]" ],
              "mode": "pass",
              "rekey_time": "4h",
              "rekey_bytes": 10000000,
              "inactivity": "2h",
              "start_action": "trap"
            },
            "drop-ssh-in": {
              "local_ts": [ "192.168.100.254[tcp/ssh]" ],
              "remote_ts": [ "192.168.100.253" ],
              "mode": "drop",
              "start_action": "trap"
            },
            "drop-ssh-out": {
              "local_ts": [ "192.168.100.254" ],
              "remote_ts": [ "192.168.100.253[tcp/ssh]" ],
              "mode": "drop",
              "start_action": "trap"
            },
            "tunnel-conn": {
              "local_ts": [ "192.168.100.254" ],
              "remote_ts": [ "192.168.100.253" ],
              "mode": "tunnel",
              "start_action": "trap|start"
            }
          }
        },
      },
      "secrets": {
        "ike-r3": {
          "secret": "Abc123!@",
          "id": "192.168.100.253"
        }
      }
    }
  }
}
```

8.4.5 VPN Cryptographic Algorithms

The VPN cryptographic algorithms described in this section can be applied to the "proposals" and "esp_proposals" configuration parameters. These parameters take a "list" of sets of algorithms or the special value "default", which forms a default proposal of supported algorithms considered safe and is usually a good choice for interoperability. There are no additional configurations necessary to make use of these algorithms.

Order for non-AEAD algorithms: encryption algorithm, integrity algorithm, (optional) pseudo-random function, Diffie-Hellman key exchange group.

Order for AEAD algorithms: combined algorithm (instead of encryption and integrity), pseudo-random function, Diffie-Hellman key exchange group.

Example non-AEAD:

```
    "r1" : {
      "proposals": {
        "add_default": true, #or false
        "list": [
          {
            "encryption_algorithms": ["aes128", "aes256",
    "aes128ctr", "aes256ctr"],
            "integrity_algorithms": ["sha256", "sha384",
    "sha512"],
            "pseudo_random_functions": ["prfsha256",
    "prfsha384", "prfsha512"],
            "ke_groups": ["modp2048", "modp3072", "modp4096",
    "modp6144", "modp8192", "modp2048s256", "ecp256", "ecp384",
    "ecp521"]
          }
        ]
      }
    }
```

Example AEAD:

```
    "r1" : {
      "proposals": {
        "list": [
          {
            "aead_algorithms": ["aes128gcm8", "aes256gcm8",
    "aes128gcm12", "aes256gcm12", "aes128gcm16",
    "aes256gcm16"],
            "pseudo_random_functions": [ "prfsha256",
    "prfsha384", "prfsha512" ],
            "ke_groups": ["modp2048", "modp3072", "modp4096",
    "modp6144", "modp8192", "modp2048s256", "ecp256", "ecp384",
    "ecp521"]
          }
        ]
      }
    }
```

8.4.5.1 Encryption Algorithms Supported

Keyword	**Description**
aes128	128 bit AES-CBC
aes256	256 bit AES-CBC
aes128ctr	128 bit AES-COUNTER
aes256ctr	256 bit AES-COUNTER

8.4.5.2 Integrity Algorithms Supported

Keyword	**Description**
sha256	SHA2_256_128 HMAC (128 bit)
sha384	SHA2_384_192 HMAC (192 bit)
sha512	SHA2_512_256 HMAC (256 bit)

8.4.5.3 Authenticated Encryption (AEAD) Algorithms Supported

AEAD (Authenticated Encryption with Associated Data) algorithms can’t be combined with classic encryption ciphers in the same proposal. No separate integrity algorithm must be proposed and therefore Pseudo-Random Functions (PRFs) have to be included explicitly in such proposals.

Keyword	**Description**
aes128gcm8	128 bit AES-GCM with 64 bit ICV
aes256gcm8	256 bit AES-GCM with 64 bit ICV
aes128gcm12	128 bit AES-GCM with 96 bit ICV

aes256gcm12	256 bit AES-GCM with 96 bit ICV
aes128gcm16	128 bit AES-GCM with 128 bit ICV
aes256gcm16	256 bit AES-GCM with 128 bit ICV

8.4.5.4 Pseudo-Random Functions Supported (Optional)

PRF algorithms can optionally be defined in IKEv2 proposals. In earlier releases or if no pseudo-random functions are configured, the proposed integrity algorithms are mapped to pseudo-random functions.

If AEAD ciphers are proposed there won't be any integrity algorithms from which to derive PRFs. Thus, PRF algorithms have to be configured explicitly.

Keyword	**Description**
prfsha256	SHA2_256 PRF
prfsha384	SHA2_384 PRF
prfsha512	SHA2_512 PRF

8.4.5.5 Diffie Hellman Key-Exchange Groups Supported

8.4.5.5.1 Regular Modular Prime Groups

Keyword	**Modulus**
modp2048	2048 bits
modp3072	3072 bits
modp4096	4096 bits
modp6144	6144 bits
modp8192	8192 bits

8.4.5.5.2 Modular Prime Groups with Prime Order Subgroup

Keyword	**Modulus**	**Subgroup**
modp2048s256	2048 bits	256 bits

8.4.5.5.3 NIST Elliptic Curve Groups

Keyword	**Prime Size**
ecp256	256 bits
ecp384	384 bits
ecp521	521 bits

8.4.6 VPN Configuration Parameters

The VPN configuration is a top-level JSON object containing the "Modules" and "Profiles" configuration blocks as below. A detailed list of child parameters can be found in section 2.24 "VPN" of the NIAP Configuration Parameters document.

```
{
   "Other-top-level-blocks": {…},
   "VPN":{
      "Modules": {…},
      "Profiles": {…}
   }
}
```

8.5 DHCP Configuration

The Dynamic Host Configuration Protocol (DHCP) allows the automatic assignment of IP addresses and distribution of other network information from a DHCP server to clients on the same network. This section describes the configuration of both global and per-interface DHCP settings.

8.5.1 DHCP Client

The Machete router may be configured as a client of another DHCP server, requesting an IP address on an interface by way of the following configuration:

```
{
   "RouterId" : 10,
   "Links" : [
      {
         "Name" : "eth1",
         "Type" : "simple",
         "Interface" : {
            "Name" : "eth1",
            "Inet4Addresses" : "auto",
            "KeepDHCP4Gateway" : true
            }
         }
      }
   ]
}
```

Setting Inet4Addresses to "auto" will perform DHCP to request an address, while setting KeepDHCP4Gateway to true will keep the default route added by DHCP. The same options are available for IPv6.

8.5.2 DHCP Server

The Machete router may also be configured to act as a DHCP server for both IPv4 and IPv6 networks. This requires enabling the SysConfig plugin by adding it to the list of enabled plugins in the ARES configuration:

```
"Plugins" : [ "sysconfig" ]
```

There are a number of configuration options that may be set both globally and per-subnet/pool/reservation. When those options are set in more than one location, reservation options override pool options, which override subnet options, which override global options.

An example DHCP server configuration is as follows:

```
{
  "RouterId": 10,
  "SecurityMode": "niap",
  "Plugins": [ "sysconfig" ],
  "Links": [
    {
      "Name": "link1",
      "Type": "simple",
      "Interface": {
        "Name": "eth0",
        "Inet4Addresses": [ "10.0.1.1/24" ],
        "DHCP": {
          "IPv4": {
            "Enable": true,
            "Subnet4": [
              {
                "Subnet": "10.0.1.0/24",
                "ID": 75,
                "Pools": [
                  {
                    "Pool": "10.0.1.67-10.0.1.95"
                  }
                ],
                "ValidLifetime": 300,
                "MaxValidLifetime": 1800,
                "OptionData": {
                  "Routers": [ "10.0.1.1" ]
                }
              }
            ]
          }
        }
      }
    },
    {
      "Name": "link2",
      "Type": "simple",
      "Interface": {
        "Name": "eth1",
        "Inet4Addresses": [ "10.0.2.127/24" ],
        "Inet6Addresses": [ "2001:db8::1/64" ],
        "DHCP": {
          "IPv4": { "Enable": true },
          "IPv6": { "Enable": true }
        }
      }
    }
  ],
  "DHCP": {
    "IPv4": {
      "ValidLifetime": 1800,
      "MaxValidLifetime": 43200
    },
    "IPv6”: {
      "ValidLifetime": 1800,
      "MaxValidLifetime": 43200
    }
  }
}
```

In this example configuration, the DHCP server on eth0 is manually configured with a subnet ID of 75 and will serve a pool of addresses from 10.0.1.67-10.0.1.95, with a default lease time of 300 seconds and a maximum lease time of 1800 seconds. These lease times override the lease times set in the top-level DHCP block. The DHCP server also distributes the IP address of its own interface as the default route.

The DHCP server on eth1 has no manual configuration, and so will automatically generate a pool of addresses to distribute based on the IP addresses assigned to the interface. In this case, those pools would be “10.0.1.128-10.0.2.254” and “2001:db8::2-2001:db8::ffff:ffff:ffff:fffe”. The generated IPv4 subnet also includes the interface’s IP address as the default route. Both of the subnets will inherit the global lease time configuration.

8.5.3 DHCP Configuration Parameters

The DHCP configuration consists of either or both of a Link::Interface JSON object and a global top-level JSON object that contains the "IPv4" and "IPv6" configuration blocks as below. A detailed list of child parameters can be found in sections 2.6.3.20 (link-specific) and 2.34 (global) of the NIAP Configuration Parameters document.

```
      {
         "Other-top-level-blocks": {…},
         "Links" : [
            {
               "Name" : "…",
               "Type" : "…",
               "Interface" : {
                  "Name" : "…",
                  "DHCP": {
                     "IPv4": {…},
                     "IPv6": {…}
                  }
               }
            }
         ],
         "DHCP":{
            "IPv4": {…},
            "IPv6": {…}
         }
      }
```

8.6 Sysctl Configuration

The Machete router allows the configuration of a number of network-related kernel parameters at runtime. Most of these parameters are set in a "Sysctl" block at the top level of the ARES configuration; some may be configured per-interface or in both locations. In order for these kernel parameters to be applied, the SysConfig plugin must be enabled by adding it to the Plugins array.

A limited example configuration for the purpose of demonstrating the layout is as follows:

```
   {
      "RouterId" : 10,
      "SecurityMode" : "niap",
      "Plugins" : [ "sysconfig" ],
      "Links" : [
         {
            "Name" : "eth0",
            "Type" : "simple",
            "Interface" : {
               "Name" : "eth0",
               "Inet4Addresses" : [ "10.0.0.1/24" ],
               "Inet6Addresses" : [ "2001:db8::1/32" ],
               "Sysctl" : {
                  "Net" : {
                     "IPv4" : {
                        "Conf" : {
                           "Forwarding" : 1,
                           "SendRedirects" : 1
                        }
                     },
                     "IPv6" : {
                        "Conf" : {
                           "Forwarding" : 1,
                           "IgnoreRoutesWithLinkdown" : 1
                        }
                     }
                  }
               }
            }
         }
      ],
      "Sysctl" : {
         "Net" : {
            "Core" : {
               "DefaultQdisc" : "pfifo_fast"
            },
            "IPv4" : {
               "Conf" : {
                  "Forwarding" : 1,
                  "SendRedirects" : 1
               },
               "Neigh" : {
                  "BaseReachableTimeMs" : 14400000
               },
               "Route" : {
                  "MinPmtu" : 552
               },
               "Misc" : {
                  "IpForward" : 1
               }
            },
            "IPv6" : {
               "Conf" : {
                  "Forwarding" : 1,
                  "IgnoreRoutesWithLinkdown" : 1
               }
               "Neigh" : {
                  "McastSolicit" : 10
               },
               "Route" : {
                  "SkipNotifyOnDevDown" : 1
               },
               "Misc" : {
                  "MldMaxMsf" : 64
               }
            }
            }
         }
      }
```

Top-level settings apply to all interfaces; in the case that a kernel parameter is set at both the top level and inside an interface configuration, the behavior depends on the specific parameter and is beyond the scope of this document.

Note that in most cases manually setting these parameters is unnecessary. The router has a minimal set of these parameters configured by default, those being:

```
Sysctl::Net::IPv4::Conf::RpFilter = 0
Sysctl::Net::IPv4::Misc::IpForward= 1
Sysctl::Net::IPv4::Misc::IpForwardUsePmtu = 1
Sysctl::Net::IPv4::Misc::IpNonlocalBind = 1
Sysctl::Net::IPv6::Conf::Forwarding = 1
```

8.6.1 Sysctl Network Configuration Parameters

The Sysctl configuration can be configured for a specific network with the Link::Interface JSON object that contains the "IPv4" and "IPv6" configuration blocks. Additionally, Sysctl configuration can be made for the kernel at runtime with the top-level JSON object that contains the "Core", "IPv4" and "IPv6" configuration blocks. A detailed list of child parameters can be found in sections 2.6.3.21 (link-specific) and 2.37 (global) of the NIAP Configuration Parameters document.

```
{
   "Other-top-level-blocks": {…},
   "Links" : [
      {
         "Name" : "…",
         "Type" : "…",
         "Interface" : {
            "Name" : "…",
            "Sysctl": {
               "Net": {
                  "IPv4": {…},
                  "IPv6": {…}
               }
            }
         }
      }
   ],
   "Sysctl": {
      "Net": {
         "Core": {…},
         "IPv4": {…},
         "IPv6": {…}
      }
```

```
          }
      }
```

9 Firewall

The Machete router provides a robust firewall implemented using nftables and the linux kernel. The firewall provides packet filtering, packet mangling, NAT, and VPN Connection access control through a single consolidated configuration file. The firewall configuration format follows the nftables scripting format.

The Machete router supports packet filtering for the following protocols:

- IPv4 (RFC 791)
  - Source address
  - Destination address
  - Protocol
- IPv6 (RFC 8200)
  - Source address
  - Destination address
  - Next header (protocol)
- TCP (RFC 793)
  - Source port
  - Destination port
- UDP (RFC 768)
  - Source port
  - Destination port

Each packet filtering rule is configured as a permit (pass packets), deny (drop packets silently), reject (drop packets and issue and ICMP response), or log rule. The Machete router applies the packet filtering rules in the order they are configured by the administrator. If a packet does not match any of the configured rules, the Machete router drops and logs the packet without sending an ICMP response.

The Machete router allows packet filter rules to be associated with one or more interfaces. The Machete router does not define any interface groups; however, the Machete router allows wildcards to be used when specifying the interfaces, a rule applies to. INPUT and OUTPUT rules can be assigned to physical or virtual interfaces. FORWARD rules allow packets to be routed from one interface to another.

With the exception of IKE and ESP packets, all received packets destined for the Machete router (based on IP address) are first subjected to the INPUT rules associated with the receiving interface. All routed packets are subjected to the FORWARD rules to determine if the packet should be passed on. All packets generated locally by the Machete router are subjected to the OUTPUT rules associated with the sending interface.

9.1.1 Firewall Modes

The Machete router firewall supports 2 predefined modes for packet filtering: Router or VPN Gateway. These modes set non-persistent configurations that can be overwritten by other CLI commands or saved configurations.

In Router mode, the firewall will allow all forwarded and outgoing traffic, but only allows specific incoming traffic to the router.

In VPN Gateway mode, the firewall blocks all traffic incoming, outgoing, or forwarded unless specifically allowed.

To switch between modes, enter the configuration mode and run the following CLI command where <mode> is the one of the two predefined modes:

```
admin@Router(config)> firewall mode <mode>
```

Figure 15: *firewall mode* command

```
Note: The default mode of the firewall is Router mode.
```

To see what the mode and policies are set to, in default mode, run the following CLI command:

```
Admin@Router> show firewall policy
Firewall Mode: Router

Current Policy Settings
     Input:  drop
    Output:  accept
   Forward:  accept
```

Figure 16: *show firewall policy* command

The next section details the default rules to allow traffic in Router and/or VPN Gateway mode.

9.1.2 Default/Automatic Packet Filter Rules

The Machete router has a set of default rules that are added to the firewall and may be modified in the firewall configuration file; however, it is not recommended to edit these rules without knowledge of how it will affect the routing and control protocols utilized by the OS routing software.

Rules to allow traffic that is sourced or destined to the localhost address (127.0.0.1 or ::1) are always allowed on input and output. These rules are important to the correct operation of the Machete router and should not be removed.

Rules to allow traffic matching the following specifications are also installed in the firewall by default. As an administrator, you may wish to edit and/or disable but do so only when necessary as it may adversely affect the ability of the router to perform certain features.

- Traffic allowed in input:
  - Established or related connections (connections originated by the router)
- Traffic allowed in output:
  - Established connections
- Traffic allowed in input and output:

- IP Protocols: 1 (icmp), 2 (igmp), 47 (gre), 50 (esp), 51 (ah), 55 (me), 58 (icmpv6), 89 (ospf), 103 (pim), 155, 156
- TCP (proto 6) ports: 22 (ssh), 12344 (dport only)
- UDP (proto 17) dest ports: 500 (ike), 4500 (mobike), 12345, 12346, 12347, 32768

The Machete router supports non-persistent rules for these firewall policies. The input, output, and forward policies can be set to either accept or drop packets for each policy. To set these rules, switch to the configuration mode in your router and run the CLI command:

```
   admin@Router(config)> firewall policy <policy> <action>
```

Figure 17: *firewall policy* command

Where:

<policy> is the desired firewall policy to effect.

```
   admin@Router(config)> firewall policy ?
   Available commands:
      firewall policy forward     - Firewall policy for forwarded traffic
      firewall policy input       - Firewall policy for inbound traffic
      firewall policy output      - Firewall policy for outbound traffic
   admin@Router(config)> firewall policy
```

And:

<action> is the desired action over packets in the selected policy above.

```
   admin@Router(config)> firewall policy forward ?
   Available commands:
      firewall policy forward accept      - Allow all traffic if no rules are
matched to drop it.
      Firewall policy forward drop        - Deny traffic if no rules are
matched to allow it.
   admin@Router(config)> firewall policy
```

To completely disable the firewall, simply set all of these policies to allow traffic.

9.1.3 NFT Firewall Configuration

The default firewall configuration has all the nftable rules required for the router to operate but if desired, the firewall rules can be configured more specifically and be applied persistently by editing the routers nftables configuration file. This can be done with the following command to open the text editor

```
admin@Router(config)> firewall
```

Details on the nftables scripting format can be found at: https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes#Tables and the main documentation page for nftables: https://wiki.nftables.org/wiki-nftables/index.php/Main_Page

9.1.4 Automatic VPN Connection Firewall Rules

The Machete router also does automatic injection of packet filtering rules for VPN connections.

VPN connections that run-in transport mode, which is intended for host-to-host communication, have rules automatically added which allow all incoming traffic that is encrypted or was decrypted, and all outgoing traffic that is encrypted or will be encrypted, by the VPN connection.

VPN connections that run-in tunnel mode, which is intended for network-to-network communication, have rules automatically added which allow all forwarded traffic that will be encrypted or was decrypted, by the VPN connection.

The primary difference between the two VPN connection modes, from the perspective of the Firewall, is that tunnel mode is routed, transport mode is not routed.

Additionally, VPN connections with Child_SA mode set to either pass or drop will have an effect on the packet ruling. With pass mode set, the VPN connection is assumed to have no Ipsec processing and all packets are passed as-is, while in drop mode, all packets are discarded.

If the user experiences connectivity issues with too-restrictive intermediary firewalls blocking ESP packets, the following parameter can be set to true to enforce UDP encapsulation of ESP packets by manipulating the NAT detection payloads of the connection as shown below.

```
Connections::<conn>::children::<child>::encap = true
```

However, for traffic to actually flow through that VPN connection, the user is required to manually add the appropriate Firewall rules to allow the appropriate traffic through the VPN connection. It is highly recommended to use the dynamic rules added via use of the VPN connection firewall option.

9.2 Firewall Configuration

To add/edit Firewall rules, first go into configure mode with the following CLI command:

```
admin@r0> configure terminal
```

Then enter the CLI command:

```
admin@r0> firewall
```

Once entered, an editor session is started allowing the user to modify the current firewall configuration script. Once the file is saved, the configuration is validated. If invalid, the user is prompted to fix the error or discard changes. If valid, the user is prompted to restart necessary services in order to apply the new configuration.

The rest of this section will outline the basic configuration of the firewall using the nftables script format. This section will only cover the basics. Please refer to the nftables wiki web site (https://wiki.nftables.org/wiki-nftables/index.php/Main_Page) for a complete reference.

9.2.1 Basic format

The nftables scripting format is read directly by the nftables executable and has support for comments, variables, and commands.

You can add comments to your file by using the '#' character. Everything after the '#' will be ignored up to a newline.

You can use the define keyword to define variables. Variables may be defined as a single item, or as a set.

```
#!/usr/sbin/nft -f

# A singleton variable definition
define google_dns = 8.8.8.8
define web_ports = { 80, 443 } # A set definition
```

The brackets have a special semantics when used from rules, since they indicate that this variable represents a set. Therefore, avoid things like:

```
define google_dns = { 8.8.8.8 }
```

It is simply overkill to define a set that only stores a single element, instead use the singleton definition as used in the first example above.

Definitions may then be referenced in the rest of the firewall configuration using the syntax *$google_dns* or *$web_ports*

The nftables firewall script is executed from the current state of the firewall, therefore, in order to avoid duplicating firewall rules every time a change is made to the firewall, the script must contain the line *"flush ruleset"* at the top of the firewall script in order to apply a fresh rule set.

9.2.2 Tables

Tables are the top-level containers within an nftables ruleset; they hold chains, sets, maps, flowtables, and stateful objects.

Each table belongs to exactly one family. So, your ruleset requires at least one table for each family you want to filter.

A table is defined as follows:

```
table inet global {
}
```

This follows the syntax of *table <family> <name>*.

The following are the supported table families.

- **ip**: Tables of this family see IPv4 traffic/packets only.
- **ip6**: Tables of this family see IPv6 traffic/packets only.
- **inet**: Tables of this family see both IPv4 and IPv6 traffic/packets, simplifying dual stack support. Within a table of inet family, both IPv4 and IPv6 packets traverse the same rules. Rules for IPv4 packets do not affect IPv6 packets and vice-versa. Rules for both layer 3 protocols affect both. Use meta l4proto to match on the layer 4 protocol regardless the packet is either IPv4 or IPv6.
- **arp**: Tables of this family see ARP level (i.e. L2) traffic, before any L3 handling is done by the kernel.
- **bridge**: Tables of this family see traffic/packets traversing bridges (i.e. switching). No assumptions are made about L3 protocols.
- **netdev**: The netdev family is different from the others in that is it used to create base chains attached to a *single network interface*. Such base chains see *all* network traffic on the specified interface, with no assumptions about L2 or L3 protocols. Therefore, you can filter ARP traffic from here. The principal use for this family is for base chains using the ingress hook. Such ingress chains see network packets just after the NIC driver passes them up to the networking stack. This very early location in the packet path is ideal for dropping packets associated with DDoS attacks. Dropping packets from an ingress chain is twice as efficient as doing so from a prerouting chain.

9.2.3 Chains

Specific rules are attached to a chain. There are no predefined chains like INPUT, OUTPUT, etc. Instead, to filter packets at a particular processing step, you explicitly create a base chain with a name of your choosing, and attach it to the appropriate Netfilter hook. This allows very flexible configurations without slowing Netfilter down with built-in chains not needed by your ruleset. Not all chains require a Netfilter hook, as you may use a rule to jump or goto a different chain.

```
table inet global {
     chain myinput {
# This base chain defines a Netfilter hook, so it will #
automatically be entered for packets processed by
# the kernel, as defined by the hook.
          type filter hook input priority filter; policy
drop;
     }
     chain anotherchain {
# This regular chain has no Netfilter hook, so it will #
only be used if entered from a jump or goto rule
     }
}
```

9.2.3.1 Netfilter Hooks

The format for defining a Netfilter hook is as follows:

```
type <chain type> hook <hook> priority <priority>; [policy
<policy>;]
```

The possible **chain types** are:

- **filter**: Used to filter packets. This is supported by the arp, bridge, ip, ip6 and inet table families.
- **route**: Used to reroute packets if any relevant IP header field or the packet mark is modified. This is supported by the ip, ip6 and inet table families.
- **nat**: Used to perform Networking Address Translation (NAT). Only the first packet of a given flow hits this chain; subsequent packets bypass it. Therefore, never use this chain for filtering. The nat chain type is supported by the ip, ip6 and inet table families.

The possible **hooks** that you can use when you configure your base chain are:

- **ingress**: Sees packets immediately after they are passed up from the NIC driver, before even prerouting.
- **prerouting**: Sees all incoming packets, before any routing decision has been made. Packets may be addressed to the local or remote systems.
- **input**: Sees incoming packets that are addressed to and have now been routed to the local system and processes running there.
- **forward**: Sees incoming packets that are not addressed to the local system.
- **output**: Sees packets that originated from processes in the local machine.
- **postrouting**: Sees all packets after routing, just before they leave the local system.

Each nftables base chain is assigned a **priority** that defines its ordering among other base chains, flowtables, and Netfilter internal operations at the same hook. For example, a chain on the prerouting hook with priority -300 will be placed before connection tracking operations. The priority may be a positive or negative integer, or it may use a keyword name for the default priority for that chain type.

NOTE: If a packet is accepted and there is another chain, bearing the same hook type and with a later priority, then the packet will subsequently traverse this other chain. Hence, an accept verdict - be it by way of a rule or the default chain policy – is not necessarily final. However, the same is not true of packets that are subjected to a drop verdict. Instead, drops take immediate effect, with no further rules or chains being evaluated.

Each nftables base chain has a **policy**. This is the default verdict that will be applied to packets reaching the end of the chain (i.e, no more rules to be evaluated against).

Currently there are 2 policies: **accept** (default) or **drop**.

The **accept** verdict means that the packet will keep traversing the network stack (default).

The **drop** verdict means that the packet is discarded if the packet reaches the end of the base chain.

> **NOTE:** If no policy is explicitly selected, the default policy accept will be used.

9.2.4 Rules

Rules in nftables are built through expersions to match packets based on specific fields and/or meta-data about the packet, if the packet matches, then statements are executed.

```
<match(es)> <statement(s)>
```

Match expressions support the following operations:

- **eq** which stands for equal. Alternatively you can use **==**.
- **ne** which stands for not equal. Alternatively you can use **!=**.
- **lt** which stands for less than. Alternatively you can use <.
- **gt** which stands for greater than. Alternatively you can use >.
- **le** which stands for less than or equal to. Alternatively you can use <=.
- **ge** which stands for greater than or equal to. Alternatively you can use >=.

Example:

```
define web_ports = { 80, 443 }
table inet global {
 chain myinput {
   type filter hook input priority filter; policy accept;
   # Block access to TCP web ports if not coming from an
   # internal address
   ip saddr ne 192.168.0.0/24 tcp dport $web_ports drop

 }
}
```

A user can add additional rules within the input, output, and forward chains by following the format below:

```
<source> <destination> ip protocol icmp [defined-protocol] <accept|drop>
```

where:

<source> is the source ip address denoted as _ip saddr x.x.x.x_

<destination> is the destination ip address denoted as _ip daddr x.x.x.x_

[defined-protocol] is where the protocol type and code may be defined.

<accept|drop> firewall rule to accept or drop packets from that fits the defined rule here.

Example:

```
chain input {
  ip saddr 192.168.191.10 tcp sport 2500 ip daddr 10.91.1.10 tcp dport 8080
accept
    ip6 saddr 2001:192:168:191::10 tcp sport 2500 ip6 dadr 2001:10:91:1::10
tcp dport 8080 accept

    ip saddr 192.168.191.10 ip daddr 10.91.1.10 meta l4proto { tcp,udp } th
sport 2500 th dport 8080 accept
    ip6 saddr 2001:192:168:191::10 ip6 daddr 2001:10:91:1::10 meta l4proto {
tcp,udp }th sport 2500 th dport 8080 accept

    ip saddr 192.168.191.10 ip daddr 10.91.1.10 icmp type router-advertisement
icmp code 0 accept
    ip saddr 192.168.191.10 ip daddr 10.91.1.10 ip protocol icmp icmp type 9
icmp code 0 accept

    ip saddr 172.16.8.253 tcp dport { 1024-65535 } drop
    ip daddr 172.16.8.253 ip daddr 0.0.0.0/0 tcp dport 1024-65535 drop
}
```

9.2.4.1 Matches

This section contains a quick reference of some common match statements used in nftable rules.

ip match		
dscp <value>		ip dscp cs1 ip dscp != cs1 ip dscp 0x38 ip dscp != 0x20 ip dscp {cs0, cs1, cs2, cs3, cs4, cs5, cs6, cs7, af11, af12, af13, af21, af22, af23, af31, af32, af33, af41, af42, af43, ef}
length <length>	Total packet length	ip length 232 ip length != 233 ip length 333-435 ip length != 333-453 ip length { 333, 553, 673, 838}
id <id>	IP ID	ip id 22 ip id != 233 ip id 33-45 ip id != 33-45 ip id { 33, 55, 67, 88 }
frag-off <value>	Fragmentation offset	ip frag-off 222 ip frag-off != 233 ip frag-off 33-45 ip frag-off != 33-45 ip frag-off { 33, 55, 67, 88 }

ttl <ttl>	Time to live	ip ttl 0 ip ttl 233 ip ttl 33-55 ip ttl != 45-50 ip ttl { 43, 53, 45 } ip ttl { 33-55 }
protocol <protocol>	Upper layer protocol	ip protocol tcp ip protocol 6 ip protocol != tcp ip protocol { icmp, esp, ah, comp, udp, udplite, tcp, dccp, sctp }
checksum <checksum>	IP header checksum	ip checksum 13172 ip checksum 22 ip checksum != 233 ip checksum 33-45 ip checksum != 33-45 ip checksum { 33, 55, 67, 88 } ip checksum { 33-55 }
saddr <ip source address>	Source address	ip saddr 192.168.2.0/24 ip saddr != 192.168.2.0/24 ip saddr 192.168.3.1 ip daddr 192.168.3.100 ip saddr != 1.1.1.1 ip saddr 1.1.1.1 ip saddr & 0xff == 1 ip saddr & 0.0.0.255 < 0.0.0.127
daddr <ip destination address>	Destination address	ip daddr 192.168.0.1 ip daddr != 192.168.0.1 ip daddr 192.168.0.1-192.168.0.250 ip daddr 10.0.0.0-10.255.255.255 ip daddr 172.16.0.0-172.31.255.255 ip daddr 192.168.3.1-192.168.4.250 ip daddr != 192.168.0.1-192.168.0.250 ip daddr { 192.168.0.1-192.168.0.250 } ip daddr { 192.168.5.1, 192.168.5.2, 192.168.5.3 }
version <version>	Ip Header version	ip version 4
hdrlength <header length>	IP header length	ip hdrlength 15

ip6 match		
dscp <value>		ip6 dscp cs1 ip6 dscp != cs1

		ip6 dscp 0x38 ip6 dscp != 0x20 ip6 dscp {cs0, cs1, cs2, cs3, cs4, cs5, cs6, cs7, af11, af12, af13, af21, af22, af23, af31, af32, af33, af41, af42, af43, ef}

flowlabel <label>	Flow label	ip6 flowlabel 22
		ip6 flowlabel != 233
		ip6 flowlabel { 33, 55, 67, 88 }
		ip6 flowlabel { 33-55 }

length <length>	Payload length	ip6 length 232 ip6 length != 233 ip6 length 333-435 ip6 length != 333-453 ip6 length { 333, 553, 673, 838}
nexthdr <header>	Next header type (Upper layer protocol number)	ip6 nexthdr {esp, udp, ah, comp, udplite, tcp, dccp, sctp, icmpv6} ip6 nexthdr esp ip6 nexthdr != esp ip6 nexthdr { 33-44 } ip6 nexthdr 33-44 ip6 nexthdr != 33-44
hoplimit <hoplimit>	Hop limit	ip6 hoplimit 1 ip6 hoplimit != 233 ip6 hoplimit 33-45 ip6 hoplimit != 33-45 ip6 hoplimit {33, 55, 67, 88} ip6 hoplimit {33-55}
saddr <ip source address>	Source Address	ip6 saddr 1234:1234:1234:1234:1234:1234:1234:1234 ip6 saddr ::1234:1234:1234:1234:1234:1234:1234 ip6 saddr ::/64 ip6 saddr ::1 ip6 daddr ::2
daddr <ip destination address>	Destination Address	ip6 daddr 1234:1234:1234:1234:1234:1234:1234:1234 ip6 daddr != ::1234:1234:1234:1234:1234:1234:1234-1234:1234::1234:1234:1234:1234:1234
version <version>	IP header version	ip6 version 6

tcp match		

dport <destination port>	Destination port	tcp dport 22
		tcp dport != 33-45
		tcp dport { 33-55 }
		tcp dport {telnet, http, https }
		tcp dport vmap { 22 : accept, 23 : drop }

		tcp dport vmap { 25:accept, 28:drop }

sport < source port>	Source port	tcp sport 22
		tcp sport != 33-45
		tcp sport { 33, 55, 67, 88}
		tcp sport { 33-55}
		tcp sport vmap { 25:accept, 28:drop }
		tcp sport 1024 tcp dport 22

sequence <value>	Sequence number	tcp sequence 22 tcp sequence != 33-45

ackseq <value>	Acknowledgement number	tcp ackseq 22
		tcp ackseq != 33-45
		tcp ackseq { 33, 55, 67, 88 }
		tcp ackseq { 33-55 }

flags <flags>	TCP flags	tcp flags { fin, syn, rst, psh, ack, urg, ecn, cwr} tcp flags cwr tcp flags != cwr

window <value>	Window	tcp window 22
		tcp window != 33-45
		tcp window { 33, 55, 67, 88 }
		tcp window { 33-55 }

checksum <checksum>	IP header checksum	tcp checksum 22
		tcp checksum != 33-45
		tcp checksum { 33, 55, 67, 88 }
		tcp checksum { 33-55 }

urgptr <pointer>	Urgent pointer	tcp urgptr 22 tcp urgptr != 33-45 tcp urgptr { 33, 55, 67, 88 }
doff <offset>	Data offset	tcp doff 8

udp match		

dport <destination port>	Destination port	udp dport 22
		udp dport != 33-45
		udp dport { 33-55 }
		udp dport {telnet, http, https }
		udp dport vmap { 22 : accept, 23 : drop }
		udp dport vmap { 25:accept, 28:drop }

sport < source port>	Source port	udp sport 22
		udp sport != 33-45
		udp sport { 33, 55, 67, 88}
		udp sport { 33-55}

		udp sport vmap { 25:accept, 28:drop } udp sport 1024 tcp dport 22

length <length>	Total packet length	udp length 6666
		udp length != 50-65
		udp length { 50, 65 }
		udp length { 35-50 }

checksum <checksum>	UDP checksum	udp checksum 22
		udp checksum != 33-45
		udp checksum { 33, 55, 67, 88 }
		udp checksum { 33-55 }

ah match		
hdrlength <length>	AH header length	ah hdrlength 11-23 ah hdrlength != 11-23 ah hdrlength {11, 23, 44 }

reserved <value>		ah reserved 22
		ah reserved != 33-45
		ah reserved {23, 100 }
		ah reserved { 33-55 }

spi <value>		ah spi 111 ah spi != 111-222 ah spi {111, 122 }
sequence <sequence>	Sequence Number	ah sequence 123 ah sequence {23, 25, 33} ah sequence != 23-33

esp match		
spi <value>		esp spi 111 esp spi != 111-222 esp spi {111, 122 }
sequence <sequence>	Sequence Number	esp sequence 123 esp sequence {23, 25, 33} esp sequence != 23-33

meta match		
iifname <input interface name>	Input interface name	meta iifname "eth0" meta iifname != "eth0" meta iifname {"eth0", "lo"} meta iifname "eth*"

oifname <output interface name>	Output interface name	meta oifname "eth0" meta oifname != "eth0" meta oifname {"eth0", "lo"} meta oifname "eth*"
iif <input interface index>	Input interface index	meta iif eth0 meta iif != eth0
oif <output interface index>	Output interface index	meta oif lo meta oif != lo meta oif {eth0, lo}
iiftype <input interface type>	Input interface type	meta iiftype {ether, ppp, ipip, ipip6, loopback, sit, ipgre} meta iiftype != ether meta iiftype ether
oiftype <output interface type>	Output interface hardware type	meta oiftype {ether, ppp, ipip, ipip6, loopback, sit, ipgre} meta oiftype != ether meta oiftype ether
length <length>	Length of the packet in bytes	meta length 1000 meta length != 1000 meta length > 1000 meta length 33-45 meta length != 33-45 meta length { 33, 55, 67, 88 } meta length { 33-55, 67-88 }
protocol <protocol>	ethertype protocol	meta protocol ip meta protocol != ip meta protocol { ip, arp, ip6, vlan }
nfproto <protocol>		meta nfproto ipv4 meta nfproto != ipv6 meta nfproto { ipv4, ipv6 }
l4proto <protocol>		meta l4proto 22 meta l4proto != 233 meta l4proto 33-45 meta l4proto { 33, 55, 67, 88 } meta l4proto { 33-55 }
mark [set]	Packet mark	meta mark 0x4 meta mark 0x00000032 meta mark and 0x03 == 0x01 meta mark and 0x03 != 0x01 meta mark != 0x10 meta mark or 0x03 == 0x01 meta mark or 0x03 != 0x01

		meta mark xor 0x03 == 0x01 meta mark xor 0x03 != 0x01 meta mark set 0xffffffc8 xor 0x16 meta mark set 0x16 and 0x16 meta mark set 0xffffffe9 or 0x16 meta mark set 0xffffffde and 0x16 meta mark set 0x32 or 0xfffff meta mark set 0xfffe xor 0x16
pkttype <type>	Packet type	meta pkttype broadcast meta pkttype != broadcast meta pkttype { broadcast, unicast, multicast}

9.2.4.2 Statements

A statement is the action performed when the packet matches the rule. It could be *terminal* or *non-terminal*. In a certain rule we can consider several non-terminal statements but only a single terminal statement.

The **verdict statement** alters control flow in the ruleset and issues policy decisions for packets. The valid verdict statements are:

- *accept*: Accept the packet and stop the remain rules evaluation.
- *drop*: Drop the packet and stop the remain rules evaluation.
- *queue*: Queue the packet to userspace and stop the remain rules evaluation.
- *continue*: Continue the ruleset evaluation with the next rule.
- *return*: Return from the current chain and continue at the next rule of the last chain. In a base chain it is equivalent to accept
- *jump <chain>*: Continue at the first rule of <chain>. It will continue at the next rule after a return statement is issued
- *goto <chain>*: Similar to jump, but after the new chain the evaluation will continue at the last chain instead of the one containing the goto statement

log statement		
level [over] <value> <unit> [burst <value> <unit>]	Log level	log log level emerg log level alert log level crit log level err log level warn log level notice log level info log level debug

group <value> [queue-threshold <value>] [snaplen <value>] [prefix "<prefix>"]		log prefix aaaaa-aaaaaa group 2 snaplen 33 log group 2 queue-threshold 2 log group 2 snaplen 33

The default **reject** will be the ICMP type **port-unreachable**. The **icmpx** is only used for inet family support.

reject statement		
with <protocol> type <type>		reject reject with icmp type host-unreachable reject with icmp type net-unreachable reject with icmp type prot-unreachable reject with icmp type port-unreachable reject with icmp type net-prohibited reject with icmp type host-prohibited reject with icmp type admin-prohibited reject with icmpv6 type no-route reject with icmpv6 type admin-prohibited reject with icmpv6 type addr-unreachable reject with icmpv6 type port-unreachable reject with icmpx type host-unreachable reject with icmpx type no-route reject with icmpx type admin-prohibited reject with icmpx type port-unreachable ip protocol tcp reject with tcp reset

counter statement		
packets <packets> bytes <bytes>		counter counter packets 0 bytes 0

limit statement		
rate [over] <value> <unit> [burst <value> <unit>]	Rate limit	limit rate 400/minute limit rate 400/hour limit rate over 40/day limit rate over 400/week limit rate over 1023/second burst 10 packets limit rate 1025 kbytes/second limit rate 1023000 mbytes/second

		limit rate 1025 bytes/second burst 512 bytes limit rate 1025 kbytes/second burst 1023 kbytes limit rate 1025 mbytes/second burst 1025 kbytes limit rate 1025000 mbytes/second burst 1023 mbytes

nat statement		
dnat to <destination address>	Destination address translation	dnat to 192.168.3.2 dnat to ct mark map { 0x00000014 : 1.2.3.4}
snat to <ip source address>	Source address translation	snat to 192.168.3.2 snat to 2001:838:35f:1::-2001:838:35f:2:::100
masquerade [<type>] [to :<port>]	Masquerade	masquerade masquerade persistent,fully-random,random masquerade to :1024 masquerade to :1024-2048

9.2.5 Examples

Simple IP/IPv6 Workstation

```
flush ruleset
table inet global {
chain incoming {
type filter hook input priority 0; policy drop;

# established/related connections
ct state established,related accept

# loopback interface
iifname lo accept

# icmp
icmp type echo-request accept

# icmp6
# routers may also want:
# mld-listener-query, nd-router-solicit
icmpv6 type {echo-request,nd-neighbor-solicit} accept
# open tcp ports: sshd (22), httpd (80)
tcp dport {ssh, http} accept
  }
}
```

Simple IP/IPv6 Web Server

```
flush ruleset
table inet global {
   chain inbound_ipv4 {
      # accepting ping (icmp-echo-request) for diagnostic
      # purposes.
      # However, it also lets probes discover this host is
alive.
      # This sample accepts them within a certain rate
limit
      icmp type echo-request limit rate 5/second accept
   }

   chain inbound_ipv6 {
      # accept neighbor discovery otherwise connectivity
breaks
      icmpv6 type { nd-neighbor-solicit, nd-router-advert,
nd-neighbor-advert } accept

      # accepting ping (icmpv6-echo-request) for diagnostic
      # purposes. However, it also lets probes discover
this host
      # is alive. This sample accepts them within a certain
rate
      # limit
      icmpv6 type echo-request limit rate 5/second accept
   }

   chain inbound {
      # By default, drop all traffic unless it meets a
filter
      # criteria specified by the rules that follow below.
      type filter hook input priority 0; policy drop;

      # Allow traffic from established and related packets,
drop
      # invalid
      ct state vmap { established : accept, related :
accept, invalid : drop }

      # Allow loopback traffic.
      iifname lo accept
      # Jump to chain according to layer 3 protocol using a
      # verdict map
      meta protocol vmap { ip : jump inbound_ipv4, ip6 :
jump inbound_ipv6 }

      # Allow SSH on port TCP/22 and allow HTTP(S) TCP/80
and
      # TCP/443 for IPv4 and IPv6.
      tcp dport { 22, 80, 443} accept

      log prefix "[nftables] Inbound Denied: " counter drop
   }

   chain forward {
      # Drop everything (assumes this device is not a
router)
      type filter hook forward priority 0; policy drop;
   }

   # no need to define output chain, default policy is
accept if
   # undefined.
}
```

Simple Home Router

```
flush ruleset

define DEV_PRIVATE = eth1
define DEV_WORLD = ppp0
define NET_PRIVATE = 192.168.0.0/16

table ip global {
   chain inbound_world {
      # accepting ping (icmp-echo-request) for diagnostic
      # purposes. However, it also lets probes discover
this host
      # is alive. This sample accepts them within a certain
rate
      # limit
      icmp type echo-request limit rate 5/second accept

      # allow SSH connections from some well-known internet
host
      ip saddr 81.209.165.42 tcp dport ssh accept
   }
```

```
    chain inbound_private {
       # accepting ping (icmp-echo-request) for diagnostic
       # purposes.
       icmp type echo-request limit rate 5/second accept

       # allow DHCP, DNS and SSH from the private network
       ip protocol . th dport vmap { tcp . 22 : accept, udp
 . 53 : accept, tcp . 53 : accept, udp . 67 : accept}
    }

    chain inbound {
       type filter hook input priority 0; policy drop;

       # Allow traffic from established and related packets,
 drop
       # invalid
       ct state vmap { established : accept, related :
 accept, invalid : drop }

       # allow loopback traffic, anything else jump to chain
 for
       # further evaluation
       iifname vmap { lo : accept, $DEV_WORLD : jump
 inbound_world, $DEV_PRIVATE : jump inbound_private }

       # the rest is dropped by the above policy
    }

    chain forward {
       type filter hook forward priority 0; policy drop;

       # Allow traffic from established and related packets,
 drop
       # invalid
       ct state vmap { established : accept, related :
 accept, invalid : drop }

       # connections from the internal net to the internet
 or to
       # other internal nets are allowed
       iifname $DEV_PRIVATE accept

       # the rest is dropped by the above policy
    }

    chain postrouting {
          type nat hook postrouting priority 100; policy
accept;

          # masquerade private IP addresses
          ip saddr $NET_PRIVATE oifname $DEV_WORLD masquerade
     }
}
```

9.3 Deny Session Establishment

The denial of session establishment gives administrators control over when an interface can be used. This configuration is specified under the input chain in the firewall file. To configure this feature, enter a text editing session for the firewall by executing the *firewall* command while in configuration mode.

```
admin@r0(config)> firewall
```

The following configuration should be added to the input chain before all of the accept statements:

`iifname <interface> <day|hour|time> drop`

Where:

<interface> is the name of the interface that should drop packets.

<day|hour> can be either the word "day" followed by the day of the week, or "hour" followed by the range of hours to drop packets. Note that the "meta" keyword is required after the interface for the time specifier.

The following example will deny session establishment on the weekend and throughout the week between the hours 6:00PM and 7:00AM for the eth0 interface and also reject packets between the dates "2026-01-01 00:00:00" and "2038-01-19 03:14:07"

```
chain input {
   # The default policy can be changed here
   # The log for dropped packets at the end of the table
   # should be updated as well to reflect if the policy is
   # to drop (uncommented) or accept (comment out)
   type filter hook input priority filter; policy accept;

   iifname eth0 hour 18:00-07:00 drop
   iifname eth0 day Saturday drop
   iifname eth0 day Sunday drop
   iifname eth0 meta time "2026-01-01 00:00:00"-"2038-01-19 03:14:07" counter
reject
   …
```

10 Security Audit

The Machete router stores log messages which are generated by auditable events in a local storage. The local audit event storage is monitored and maintained to ensure that new audit events are always logged. Audit event logs may also be sent to external audit event storage through the remote syslog protocol. The router's CLI provides the ability for an administrator to view and search the audit event logs locally.

10.1 Local Audit Event Log Access

To view all the audit messages stored in an audit file, enter the CLI command:

```
show log <auditfile>
```

Once this command is entered, the given audit file is opened using the UNIX program "less" in secure mode. The *less* program is a simple to use and powerful test file viewer. Refer to *Audit Logging Messages* section to determine the correct audit log file for the audit event messages that need to be reviewed.

Enter 'h' or 'H' from within the less viewer to view a summary of available features, and the key commands to activate them. Below is a condensed list of the most useful features of *less*.

UpArrow/DownArrow – Scroll up/down one line at a time

PgUp/PgDown – Scroll up/down one page at a time

/pattern – Search forward for the next line that contains the given pattern

?pattern – Search backward for the previous line that contains the given pattern

n – Repeat the previous search

N – Repeat the previous search, searching in the opposite direction

&pattern – Display only the lines that contain the given pattern

F – Display new messages written to the file, as they are written. Use ctrl+c to stop.

q – Exit the log viewer

The "show log <auditfile>" command automatically opens the audit file scrolled to the end of the file rather than at the beginning of the file. When searching from the end of the file, use the '?pattern' and 'n' commands in order to search backward through the file.

Refer to *Audit Logging Messages* section for message formats, titles and bodies for all the auditable events.

10.1.1 Local Log Access

The Machete router also stores non-audit logging separately from the audit event logging. The non-audit logging will include logging related to the network and routing services, as well as any additional system logging that is not covered by the audit event logs.

To view all the non-audit messages stored in a log file, enter CLI command:

```
show log <logfile>
```

10.2 Local Audit Event Storage

The Machete router maintains 4 separate log types and up to 10 files for each log type, one active file and 9 archive files. The current file for each log type is allowed to reach 50MB in size before it is rotated, when the router attempts to write an audit log message to an audit file which would increase the size beyond 50MB then that audit log type is rotated. Audit log rotation will delete the oldest archive file, if archiving the current file will create more than 9 archive files. Each archive file is renamed with a suffix .1.gz through .9.gz, indicating the age of the archive file, 1 being the newest and 9 being the oldest. Then the current audit log is compressed and renamed with the suffix .1.gz. Once rotation is completed, a new empty current file is created and the queued audit message is written to the file. This approach ensures that no audit log messages are discarded during the log rotation process and that the allocated storage capacity of 2GB is never exceeded.

The Machete router CLI does not implement any commands that allow the modification or deletion of audit records. No configuration is needed or possible for local storage of audit data or audit log rotation.

Audit event messages are split into four types logged to the files listed below:

- auth.log – Stores all authentication related events
- vpn.log – Stores all VPN related events
- firewall.log – Stores all Firewall related events
- general.log – Stores all other audit events

10.2.1 Local Log Storage

The Machete router implements the same partition size and log rotation scheme for the non-audit event logging as well. However, since the standard log location for non-audit event logging is not as tightly controlled as the audit log storage, an additional service is run which monitors the available partition space for non-audit log files and deletes all history files (anything with the .gz file extension) when the available space in the non-audit log partition is less than 10%.

The non-audit log messages are split among 4 files as well, listed below:

- kern.log – Stores messages generated by the kernel and the ARES kernel module
- user.log – Stores messages from the network/routing services

- local.log – Stores messages from any non-audit services
- syslog – Stores all other system messages

10.3 External Audit Event Storage

The Machete router supports the transmission of audit event data to an external audit server using the syslog protocol, RFC 3164. The router transmits audit records to the syslog server in real-time, i.e. when an audit event is generated, is it simultaneously sent to both the external server and the local store. The router does not have the ability to cache or retransmit audit records if the syslog server is unavailable. If the connection to the syslog server is lost, audit logs continue to be stored locally on the router. When the connection is restored, the router automatically resumes sending new audit log messages, requiring no further action from the operator

To view the external audit server settings, enter the *show audit remote* command:

```
admin@r0> show audit remote
Server         TCP|UDP  Port
A.B.C.D        tcp      12345
Admin@r0>
```

To Configure an external audit server, enter configuration mode and run the *ares* command. This command will enter a text editor session for the ares.json file. For more information on how to use the text editor, refer to *The Text Editor* section.

Once in the text editor session, add or modify the existing "Log" parameter to the following JSON object:

```
"Log": {
   "Audit": {
      "Syslog": {
         "IPAddress": "A.B.C.D",
         "Port": 12345,
         "Protocol": "tcp"
      }
   }
}
```

```
Note: The IP address in the example above should be modified to match
that of the target server, while the port and protocol should be
updated as desired.
```

10.3.1 Protected Syslog via Trusted Channel

The transmission of audit event data to an external audit server may be protected via the use of one of the Machete router's trusted channels. An SSH tunnel can be configured for this purpose by setting the SSH::Client::LocalForward configuration block. The "Host" should be set to the loopback address "127.0.0.1" and the "Port" should be set to an unused port, on which the SSH client will listen.

```
"SSH": {
   "Client" : {
      "LocalForward": {
         "Port": 10514,
         "Host": "127.0.0.1",
         "HostPort": 514
      }
   }
}
```

Note that the "HostPort" is set to 514, a well-known UDP port for syslog services. This port is where the audit event data will be transmitted when connected to the external audit server.

Then you can configure the Log::Audit::Syslog configuration block to point the "Port" where the port matches the one set in the SSH::Client::LocalForward and the "IPAddress" is set to the loopback address.

```
"Log" : {
  "Audit": {
    "Syslog" : {
      "IPAddress" : "127.0.0.1",
      "Port" : 10514,
      "Protocol" : "tcp"
    }
  }
},
```

Then you can use the `net ssh connect <host-ip> <host-user>` command to create the tunnel.

The syslog can be configured in conjunction with an IPsec channel to protect the communication. To protect the transmission of audit log data to a syslog server, make sure that the IP address (Log::Audit::Syslog::IPAddress "10.0.3.3" in the example below) falls within the remote traffic selector of a VPN security association (VPN::Profiles::connections::children::remote_ts, "10.0.3.0/24" in the example below).

```
{
  "RouterId" : 10,
  "SecurityMode" : "niap",
  "Plugins": [  "sysconfig", "vpn" ],
  "Log" : {
    "Audit": {
      "Syslog" : {
        "IPAddress" : "10.0.3.3",
        "Port" : 514,
        "Protocol" : "tcp"
```

```
          }
        }
      },
      "Links" : [
        {
          "Name" : "LAN",
          "Type" : "simple",
          "Interface" : {
            "Name" : "eth0",
            "Inet4Addresses" : [ "10.0.1.0/24" ]
          }
        },
        {
          "Name" : "WAN",
          "Type" : "simple",
          "Interface" : {
            "Name" : "eth1",
            "Inet4Addresses" : [ "10.0.10.0/24" ]
          }
        }
      ],
      "VPN": {
        "Profiles": {
          "connections": {
            "r3": {
              "version" : 2,
              "proposals": {
                "list": [
                  {
                    "encryption_algorithms": ["aes128"],
                    "integrity_algorithms": ["sha256"],
                    "pseudo_random_functions": ["prfsha256"],
                    "ke_groups": ["modp2048"]
                  }
                ]
              },
              "local_addrs": [ "10.0.1.1" ],
              "remote_addrs": [ "10.0.2.3" ],
              "local": {
                "auth": "pubkey",
                "certs" :  ["file://router.crt"],
                "id": "SAN field from the certificate such as
IP address, FQDN, email, etc"
              },
              "remote": {
                "auth": "pubkey"
              },
              "children": {
                "red": {
                  "esp_proposals": {
                    "list": [
                      {
                        "encryption_algorithms": [ "aes128" ],
                        "integrity_algorithms": [ "sha256" ]
                      }
                    ]
                  },
                  "local_ts": [ "10.0.0.0/24" ],
                  "remote_ts": [ "10.0.3.0/24" ],
                  "start_action": "trap|start"
                }
              }
            }
          }
        }
      }
}
```

10.4 Exporting Logs

You can export logs from the router with the command line interface by using the logs command. The following command is an example of how to push logs to another host.

```
admin@Router> logs push scp://<username>@<IPv4 address>:~
Pushing logs Router.log.zip to '<username>@<IPv4 address>'
```

The URI can be one of: `file://`, `http://`, `https://`, or `scp://`

The zip file includes the following logs:

```
Audit:
        Auth.log
        Firewall.log
        General.log
Log:
        Kern.log
        Syslog
        User.log
```

11 Audit Logging Messages

The following table lists the auditing requirements. If there is a requirement it is linked to the respective section in this chapter which contains the logs. No additional configuration is required for protection of the locally stored audit data against unauthorized modification or deletion.

All audit messages use one of the following message formats:

<date/time> <host name> <user name>: <message title>: <message body>

where:

<date/time> is MMM DD hh:mm:ss
MMM = Month (Jan, Feb, etc.)
DD = Day of month (01, 02…31)
hh:mm:ss = hour:min:sec
<host name> is the name of the host that generates the audit log message
<user name> is the name of the user or process that generates the audit log message
<message title> is the title of the message, a descriptive name of the event type
<message body> is the content of the message

<date/time> <host name> <user name>: <message body>

where:

<date/time> is MMM DD hh:mm:ss
MMM = Month (Jan, Feb, etc.)
DD = Day of month (01, 02…31)
hh:mm:ss = hour:min:sec
<host name> is the name of the host that generates the audit log message
<user name> is the name of the user or process that generates the audit log message
<message body> is the content of the message

The audit messages for the administrative actions are listed below.

```
Note: All actions performed below are done by an administrative
account, on one device denoted by default as ‘r0’.
```

Requirement	Auditable Events	Additional Content
NDcPP22e/VPNGW12:FAU_GEN.1	None	None
NDcPP22e:FAU_GEN.2	None	None
NDcPP22e:FAU_STG.1	None	None
NDcPP22e:FAU_STG_EXT.1	None	None
NDcPP22e:FCS_CKM.1	None	None

VPNGW12:FCS_CKM.1/IKE	None	None
NDcPP22e:FCS_CKM.2	None	None
NDcPP22e:FCS_CKM.4	None	None
NDcPP22e:FCS_COP.1/DataEncryption	None	None
VPNGW12:FCS_COP.1/DataEncryption	None	None
NDcPP22e:FCS_COP.1/Hash	None	None
NDcPP22e:FCS_COP.1/KeyedHash	None	None
NDcPP22e:FCS_COP.1/SigGen	None	None
NDcPP22e/VPNGW12:FCS_IPSEC_EXT.1	Failure to establish an IPsec SA.	Reason for failure.
	Protocol failures.	Reason for failure. Non-TOE endpoint of connection.
	Establishment or Termination of an IPsec SA.	Non-TOE endpoint of connection.
NDcPP22e:FCS_NTP_EXT.1	Configuration of a new time server Removal of configured time server	Identity if new/removed time server
NDcPP22e:FCS_RBG_EXT.1	None	None
NDcPP22e:FCS_SSHC_EXT.1	Failure to establish an SSH session.	Reason for failure.
NDcPP22e:FCS_SSHS_EXT.1	Failure to establish an SSH session.	Reason for failure.
NDcPP22e:FIA_AFL.1	Unsuccessful login attempt limit is met or exceeded.	Origin of the attempt (e.g., IP address).
NDcPP22e:FIA_PMG_EXT.1	None	None
VPNGW12:FIA_PSK_EXT.1	None	None
VPNGW12:FIA_PSK_EXT.2	None	None
VPNGW12:FIA_PSK_EXT.3	None	None
NDcPP22e:FIA_UAU.7	None	None
NDcPP22e:FIA_UAU_EXT.2	All use of identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
NDcPP22e:FIA_UIA_EXT.1	All use of identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
NDcPP22e:FIA_X509_EXT.1/Rev	Unsuccessful attempt to validate a certificate. Any addition, replacement or removal of trust anchors in the TOE's trust store	Reason for failure of certificate validation Identification of certificates added, replaced or removed as trust anchor in the TOE's trust store
NDcPP22e/VPNGW12:FIA_X509_EXT.2	None	None

NDcPP22e:FIA_X509_EXT.3	None	None
NDcPP22e:FMT_MOF.1/Functions	None	None
NDcPP22e:FMT_MOF.1/ManualUpdate	Any attempt to initiate a manual update.	None
NDcPP22e:FMT_MOF.1/Services	None	None
NDcPP22e:FMT_MTD.1/CoreData	None	None
NDcPP22e:FMT_MTD.1/CryptoKeys	None	None
VPNGW12:FMT_MTD.1/CryptoKeys	None	None
NDcPP22e:FMT_SMF.1	All management activities of TSF data.	None
FMT_SMF.1.1/VPN	All administrative actions	No additional information.
NDcPP22e:FMT_SMR.2	None	None
VPNGW12:FPF_RUL_EXT.1	Application of rules configured with the 'log' operation	Source and destination addresses Source and destination ports Transport Layer Protocol
NDcPP22e:FPT_APW_EXT.1	None	None
VPNGW12:FPT_FLS.1/SelfTest	None	None
NDcPP22e:FPT_SKP_EXT.1	None	None
NDcPP22e:FPT_STM_EXT.1	Discontinuous changes to time - either Administrator actuated or changed via an automated process. (Note that no continuous changes to time need to be logged. See also application note on FPT_STM_EXT.1)	For discontinuous changes to time: The old and new values for the time. Origin of the attempt to change time for success and failure (e.g., IP address).
NDcPP22e:FPT_TST_EXT.1	None	None
VPNGW12:FPT_TST_EXT.1	None	None
NDcPP22e/VPNGW12:FPT_TST_EXT.1	Execution of TSF self-test.	None.
	Detected integrity violations.	The TSF code file that caused the integrity violation.
VPNGW12:FPT_TST_EXT.3	None	None
NDcPP22e/VPNGW12:FPT_TUD_EXT.1	Initiation of update; result of the update attempt (success or failure).	None

NDcPP22e:FTA_SSL.3	The termination of a remote session by the session locking mechanism.	None
NDcPP22e:FTA_SSL.4	The termination of an interactive session.	None
NDcPP22e:FTA_SSL_EXT.1	(if 'lock the session' is selected) Any attempts at unlocking of an interactive session. (if 'terminate the session' is selected) The termination of a local session by the session locking mechanism.	None
NDcPP22e:FTA_TAB.1	None	None
VPNGW12:FTA_TSE.1	None	None
VPNGW12:FTA_VCM_EXT.1	None	None
NDcPP22e:FTP_ITC.1	Initiation of the trusted channel. Termination of the trusted channel. Failure of the trusted channel functions.	Identification of the initiator and target of failed trusted channels establishment attempt.
VPNGW12:FTP_ITC.1/VPN	Termination of the trusted channel Failure of the trusted channel functions	Identification of the initiator and target of failed trusted channel establishment attempt
NDcPP22e:FTP_TRP.1/Admin	Initiation of the trusted path. Termination of the trusted path. Failure of the trusted path functions.	None

Table 1 Auditable Events

11.1 IPsec Session Establishment

The following message titles and bodies are generated for this event and logged to vpn.log.

11.1.1 Successful Session Establishment with Peer

11.1.1.1 Status Message Format

[DATE][HOSTNAME] [PROCESS] [INSTANCE] <status message>

11.1.1.2 Successful Establishment with Peer

```
<date/time> r1 charon-systemd: 16[IKE] initiating IKE_SA r13[1] to 10.0.0.13
<date/time> r1 charon-systemd: 16[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP)
N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
<date/time> r1 charon-systemd: 16[NET] sending packet: from 10.0.0.11[500] to 10.0.0.13[500] (1080 bytes)
<date/time> r1 charon-systemd: 09[NET] received packet: from 10.0.0.13[500] to 10.0.0.11[500] (280 bytes)
<date/time> r1 charon-systemd: 09[ENC] parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP)
N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(CHDLESS_SUP) N(MULT_AUTH) ]
```

```
<date/time> r1 charon-systemd: 09[CFG] selected proposal:
IKE:AES_CBC_128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/ECP_256
<date/time> r1 charon-systemd: 09[IKE] authentication of `10.0.0.11` (myself) with pre-shared key
<date/time> r1 charon-systemd: 09[ENC] generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr
AUTH SA TSi TSr N(MOBIKE_SUP) N(ADD_4_ADDR) N(MULT_AUTH) N(EAP_ONLY)
N(MSG_ID_SYN_SUP) ]
<date/time> r1 charon-systemd: 09[NET] sending packet: from 10.0.0.11[4500] to 10.0.0.13[4500] (400 bytes)
<date/time> r1 charon-systemd: 06[NET] received packet: from 10.0.0.13[4500] to 10.0.0.11[4500] (224 bytes)
<date/time> r1 charon-systemd: 06[ENC] parsed IKE_AUTH response 1 [ IDr AUTH SA TSi TSr
N(MOBIKE_SUP) N(ADD_4_ADDR) ]
<date/time> r1 charon-systemd: 06[IKE] authentication of `10.0.0.13` with pre-shared key successful
<date/time> r1 charon-systemd: 06[IKE] peer supports MOBIKE
<date/time> r1 charon-systemd: 06[IKE] IKE_SA r13[1] established between
10.0.0.11[10.0.0.11]…10.0.0.13[10.0.0.13]
<date/time> r1 charon-systemd: 06[IKE] scheduling rekeying in 14274s
<date/time> r1 charon-systemd: 06[IKE] maximum IKE_SA lifetime 15714s
```

11.1.2 Failure to Establish IPsec SA

11.1.2.1 Failure due to timeout/dead-peer

```
<date/time> r1 charon-systemd: [IKE] initiating IKE_SA r3[3] to 10.0.0.3
<date/time> r1 charon-systemd: [ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP)
N(FRAG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
<date/time> r1 charon-systemd: [NET] sending packet: from 10.0.0.1[500] to 10.0.0.3[500] (1080 bytes)
<date/time> r1 charon-systemd: [IKE] retransmit 1 of request with message ID 0
<date/time> r1 charon-systemd: [NET] sending packet: from 10.0.0.1[500] to 10.0.0.3[500] (1080 bytes)
<date/time> r1 charon-systemd: [IKE] retransmit 2 of request with message ID 0
<date/time> r1 charon-systemd: [NET] sending packet: from 10.0.0.1[500] to 10.0.0.3[500] (1080 bytes)
<date/time> r1 charon-systemd: [IKE] retransmit 3 of request with message ID 0
<date/time> r1 charon-systemd: [NET] sending packet: from 10.0.0.1[500] to 10.0.0.3[500] (1080 bytes)
<date/time> r1 charon-systemd: [IKE] retransmit 4 of request with message ID 0
<date/time> r1 charon-systemd: [NET] sending packet: from 10.0.0.1[500] to 10.0.0.3[500] (1080 bytes)
<date/time> r1 charon-systemd: [IKE] retransmit 5 of request with message ID 0
<date/time> r1 charon-systemd: [NET] sending packet: from 10.0.0.1[500] to 10.0.0.3[500] (1080 bytes)
<date/time> r1 charon-systemd: [IKE] giving up after 5 retransmits
<date/time> r1 charon-systemd: [IKE] establishing IKE_SA failed, peer not responding
```

11.2 Denial of WLAN Client Session Establishment

The following are example messages generated and logged to firewall.log for Denial of WLAN Client session establishment.

11.2.1 Denial by MAC Address

```
<date/time> r0 kernel:CRRFW Input Dropped: IN=wlan0 OUT= MAC=ff:ff:ff:ff:ff:ff:da:c4:7b:fb:8b:95:08:00
SRC=0.0.0.0 DST=255.255.255.255 LEN=338 TOS=0x10 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP
SPT=68 DPT=67 LEN=318
```

11.2.2 Denial of Interface Session Establishment

The following are example messages generated and logged to kern.log for Denial of client session establishment. The example messages found here are specific to the wlan0 interface, showing dropped packets for IPv4 and IPv6.

The following firewall rules use a counter to display that number of packets and bytes dropped by that rule that can be displayed by the *show firewall* command.

```
admin@r0> show firewall
table inet global {
  chain input {
    type filter hook input priority filter; policy drop;
    iifname "wlan0" meta hour "10:00"-"11:00" counter packets 32 bytes 2108
log drop
    …
    log prefix "CRRFW Input Dropped: "
  }

  chain forward {
    type filter hook forward priority filter; policy accept;
  }

  chain output {
    type filter hook output priority filter; policy accept;
    …
  }
}
```

Note: The *show firewall monitor* command will repeat the command above to display the information in real time as packets are dropped.

11.2.2.1 Hour(s) of the Day

11.2.2.1.1 Firewall rule

```
  iifname wlan0 hour 10:00-11:00 counter log drop
```

11.2.2.1.2 Audit Record

<date> 10:15:24 r0 kernel:IN=wlan0 OUT= MAC=01:00:5e:00:00:16:da:c4:7b:fb:8b:95:08:00 SRC=10.0.0.2 DST=224.0.0.22 LEN=40 TOS=0x00 PREC=0xC0 TTL=1 ID=0 DF PROTO=2
<date> 10:15:24 r0 kernel:IN=wlan0 OUT= MAC=33:33:00:00:00:02:da:c4:7b:fb:8b:95:86:dd SRC=fe80:0000:0000:0000:d8c4:7bff:fefb:8b95 DST=ff02:0000:0000:0000:0000:0000:0000:0002 LEN=56 TC=0 HOPLIMIT=255 FLOWLBL=0 PROTO=ICMPv6 TYPE=133 CODE=0

11.2.2.2 Denial by Day of the Week

11.2.2.2.1 Firewall rule

```
  iifname wlan0 day Friday counter log drop
```

11.2.2.2.2 Audit Record

Sep 1 2023 <time> r0 kernel:IN=wlan0 OUT= MAC=01:00:5e:00:00:16:da:c4:7b:fb:8b:95:08:00 SRC=10.0.0.2 DST=224.0.0.22 LEN=40 TOS=0x00 PREC=0xC0 TTL=1 ID=0 DF PROTO=2
Sep 1 2023 <time> r0 kernel:IN=wlan0 OUT= MAC=33:33:00:00:00:02:da:c4:7b:fb:8b:95:86:dd SRC=fe80:0000:0000:0000:d8c4:7bff:fefb:8b95 DST=ff02:0000:0000:0000:0000:0000:0000:0002 LEN=56 TC=0 HOPLIMIT=255 FLOWLBL=0 PROTO=ICMPv6 TYPE=133 CODE=0

11.3 Authentication with X.509v3 Certificates

Authentication using X.509v3 events are logged to vpn.log. The following only include the parts of the logging which are different based on the authentication method.

11.3.1 Successful Authentication using a certificate

```
<date/time> r0 charon-systemd[]: received packet: from 192.168.100.253[500] to 192.168.100.254[500] (2092
bytes)
<date/time> r0 charon-systemd[]: parsed ID_PROT response 0 [ ID CERT SIG ]
<date/time> r0 charon-systemd[]: received end entity cert "C=US, O=Router, CN=user"
<date/time> r0 charon-systemd[]:   using trusted certificate "C=US, O=Router, CN=user"
<date/time> r0 charon-systemd[]:   using trusted ca certificate "C=US, O=Router, CN=router.ca"
<date/time> r0 charon-systemd[]:   reached self-signed root ca with a path length of 0
<date/time> r0 charon-systemd[]: checking certificate status of "C=US, O=Router, O=user"
<date/time> r0 charon-systemd[]: certificate status not available
<date/time> r0 charon-systemd[]: authentication of 'C=US, O=Router, CN=user' with
RSA_EMSA_PKCS1_NULL successful
```

11.3.2 Failed authentication due to an expired certificate

```
<date/time> r0 charon-systemd[]: received packet: from 192.168.100.253[500] to 192.168.100.254[500] (2092
bytes)
<date/time> r0 charon-systemd[]: parsed ID_PROT response 0 [ ID CERT SIG ]
<date/time> r0 charon-systemd[]: received end entity cert "C=US, O=Router, CN=mars"
<date/time> r0 charon-systemd[]:   using trusted certificate "C=US, O=Router, CN=mars"
<date/time> r0 charon-systemd[]:   using trusted ca certificate "C=US, O=Router, CN=router.ca"
<date/time> r0 charon-systemd[]: subject certificate invalid (valid from Jan 15 19:06:45 2023 to Jan 15 19:06:45
2023)
```

11.4 Cryptographic Keys

The following are example messages generated and logged to general.log for events relating to cryptographic keys.

11.4.1 Key Generation

11.4.1.1 User Action

```
admin@r0(config)> pki gen machete.pem rsa 2048
Generating new 2048-bit rsa key machete.pem
[admin] pki gen success
[admin] -rw-r--r-- 1 admin   admin     1679 Jan 19 19:54 rsa/machete.pem
admin@r0(config)>
```

11.4.1.2 Audit Record

```
<date/time> r0 admin: pki-gen: /usr/bin/pki --gen --outform pem --type rsa --size 2048 output-
file=rsa/machete.pem
<date/time> r0 admin: pki-gen: pki gen success
<date/time> r0 admin: pki-gen: -rw-r--r--    1 admin    admin        1679 <date/time> rsa/machete.pem
```

11.4.2 Key Deletion

11.4.2.1 User Action

```
admin@r0(config)> pki delete rsa/machete.pem

WARNING! Deleting rsa/machete.pem. Continue? (Y/n) Y
[admin] Deleting rsa /etc/swanctl/rsa/machete.pem
admin@r0(config)>
```

11.4.2.2 Audit Record

```
<date/time> r0 admin: Delete-rsa: Deleting rsa /etc/swanctl/rsa/machete.pem
<date/time>  r0 admin: Delete-rsa: scrub: using pre v1.7 scrub (skip random) patterns
<date/time>  r0 admin: Delete-rsa: scrub: padding /etc/swanctl/rsa/machete.pem with 2421 bytes to fill last fs
block
<date/time>  r0 admin: Delete-rsa: scrub: scrubbing /etc/swanctl/rsa/machete.pem 4096
<date/time>  r0 admin: Delete-rsa: scrub: 0x00     |................................................|
<date/time>  r0 admin: Delete-rsa: scrub: 0xff      |................................................|
<date/time>  r0 admin: Delete-rsa: scrub: 0xaa     |................................................|
<date/time>  r0 admin: Delete-rsa: scrub: 0x55     |................................................|
<date/time>  r0 admin: Delete-rsa: scrub: verify   |................................................|
<date/time>  r0 admin: Delete-rsa: scrub: unlinking /etc/swanctl/rsa/machete.pem
```

11.4.3 Key Export

11.4.3.1 User Action

```
admin@r0(config)> pki export all
[admin] x509 is empty
[admin] x509ca is empty
[admin] x509aa is empty
[admin] x509ocsp is empty
[admin] x509crl is empty
[admin] x509ac is empty
[admin] pubkey is empty
[admin] pkcs10 is empty
[admin] 'template/attrib-cert.cmd' -> 'staged/template/attrib-cert.cmd'
[admin] 'template/ca-cert.cmd' -> 'staged/template/ca-cert.cmd'
[admin] 'template/cert-signing-request.cmd' -> 'staged/template/cert-signing-
request.cmd'
[admin] 'template/ica-cert.cmd' -> 'staged/template/ica-cert.cmd'
[admin] 'template/server-cert.cmd' -> 'staged/template/server-cert.cmd'
[admin] 'template/sign-crl.cmd' -> 'staged/template/sign-crl.cmd'
[admin] 'template/user-cert.cmd' -> 'staged/template/user-cert.cmd'
admin@r0(config)>
```

11.4.3.2 Audit Record

```
<date/time> r0 admin: pki-export: x509 is empty
<date/time> r0 admin: pki-export: x509ca is empty
<date/time> r0 admin: pki-export: x509aa is empty
<date/time> r0 admin: pki-export: x509ocsp is empty
<date/time> r0 admin: pki-export: x509crl is empty
<date/time> r0 admin: pki-export: x509ac is empty
<date/time> r0 admin: pki-export: pubkey is empty
<date/time> r0 admin: pki-export: pkcs10 is empty
<date/time> r0 admin: pki-export: 'template/attrib-cert.cmd' -> 'staged/template/attrib-cert.cmd'
<date/time> r0 admin: pki-export: 'template/ca-cert.cmd' -> 'staged/template/ca-cert.cmd'
<date/time> r0 admin: pki-export: 'template/cert-signing-request.cmd' -> 'staged/template/cert-signing-
request.cmd'
<date/time> r0 admin: pki-export: 'template/ica-cert.cmd' -> 'staged/template/ica-cert.cmd'
<date/time> r0 admin: pki-export: 'template/server-cert.cmd' -> 'staged/template/server-cert.cmd'
<date/time> r0 admin: pki-export: 'template/sign-crl.cmd' -> 'staged/template/sign-crl.cmd'
<date/time> r0 admin: pki-export: 'template/user-cert.cmd' -> 'staged/template/user-cert.cmd'
```

11.4.4 Key Import

11.4.4.1 User Action

```
admin@r0(config)> pki import all
[admin] staged/x509 is empty
[admin] staged/x509ca is empty
[admin] staged/x509aa is empty
[admin] staged/x509ocsp is empty
[admin] staged/x509crl is empty
[admin] staged/x509ac is empty
[admin] staged/rsa is empty
[admin] staged/ecdsa is empty
[admin] staged/private is empty
[admin] staged/pkcs8 is empty
[admin] staged/pkcs12 is empty
[admin] staged/pubkey is empty
[admin] staged/psks is empty
[admin] staged/pkcs10 is empty
cp: overwrite 'template/attrib-cert.cmd'? y
[admin] 'staged/template/attrib-cert.cmd' -> 'template/attrib-cert.cmd'
cp: overwrite 'template/ca-cert.cmd'? y
[admin] 'staged/template/ca-cert.cmd' -> 'template/ca-cert.cmd'
cp: overwrite 'template/cert-signing-request.cmd'? y
[admin] 'staged/template/cert-signing-request.cmd' -> 'template/cert-signing-
request.cmd'
cp: overwrite 'template/ica-cert.cmd'? y
[admin] 'staged/template/ica-cert.cmd' -> 'template/ica-cert.cmd'
cp: overwrite 'template/server-cert.cmd'? y
[admin] 'staged/template/server-cert.cmd' -> 'template/server-cert.cmd'
cp: overwrite 'template/sign-crl.cmd'? y
[admin] 'staged/template/sign-crl.cmd' -> 'template/sign-crl.cmd'
cp: overwrite 'template/user-cert.cmd'? y
[admin] 'staged/template/user-cert.cmd' -> 'template/user-cert.cmd'
admin@r0(config)>
```

11.4.4.2 Audit Record

```
<date/time> r0 admin: pki-import: staged/x509 is empty
<date/time> r0 admin: pki-import: staged/x509ca is empty
<date/time> r0 admin: pki-import: staged/x509aa is empty
<date/time> r0 admin: pki-import: staged/x509ocsp is empty
<date/time> r0 admin: pki-import: staged/x509crl is empty
<date/time> r0 admin: pki-import: staged/x509ac is empty
<date/time> r0 admin: pki-import: staged/rsa is empty
<date/time> r0 admin: pki-import: staged/ecdsa is empty
<date/time> r0 admin: pki-import: staged/private is empty
<date/time> r0 admin: pki-import: staged/pkcs8 is empty
<date/time> r0 admin: pki-import: staged/pkcs12 is empty
<date/time> r0 admin: pki-import: staged/pubkey is empty
<date/time> r0 admin: pki-import: staged/psks is empty
<date/time> r0 admin: pki-import: staged/pkcs10 is empty
<date/time> r0 admin: pki-import: 'staged/template/attrib-cert.cmd' -> 'template/attrib-cert.cmd'
<date/time> r0 admin: pki-import: 'staged/template/ca-cert.cmd' -> 'template/ca-cert.cmd'
<date/time> r0 admin: pki-import: 'staged/template/cert-signing-request.cmd' -> 'template/cert-signing-
request.cmd'
<date/time> r0 admin: pki-import: 'staged/template/ica-cert.cmd' -> 'template/ica-cert.cmd'
<date/time> r0 admin: pki-import: 'staged/template/server-cert.cmd' -> 'template/server-cert.cmd'
<date/time> r0 admin: pki-import: 'staged/template/sign-crl.cmd' -> 'template/sign-crl.cmd'
<date/time> r0 admin: pki-import: 'staged/template/user-cert.cmd' -> 'template/user-cert.cmd'
```

11.5 Administrative login and logout

The following are example messages generated and logged to auth.log for administrative login and logout.

11.5.1 Successful login local console

```
<date/time> r0 login[19178]: pam_unix(login:session): session opened for user admin(uid=1000) by
admin(uid=0)
<date/time> r0 systemd-logind[371]: New session c3 of user admin.
```

11.5.2 Failed login local console

```
<date/time> r0 login[24019]: pam_unix(login:auth): authentication failure; logname=admin uid=0 euid=0
tty=/dev/tty1 ruser= rhost=  user=admin
<date/time> r0 login[24019]: FAILED LOGIN (1) on '/dev/tty1' FOR 'admin', Authentication failure
```

11.5.3 Logout local console

```
<date/time> r0 login[24019]: pam_unix(login:session): session closed for user admin
<date/time> r0 systemd[1]: getty@tty1.service: Deactivated successfully.
<date/time> r0 systemd[1]: session-c3.scope: Deactivated successfully.
<date/time> r0 systemd[1]: Stopped Getty on tty1.
<date/time> r0 systemd-logind[371]: Removed session c3.
```

11.5.4 Successful remote login

```
<date/time> r0 sshd[5173]: Connection from 10.0.0.228 port 57879 on 10.0.0.146 port 22 rdomain ""
<date/time> r0 sshd[5173]: Postponed keyboard-interactive for admin from 10.0.0.228 port 57879 ssh2 [preauth]
<date/time> r0 sshd[5173]: Postponed keyboard-interactive/pam for admin from 10.0.0.228 port 57879 ssh2
[preauth]
<date/time> r0 sshd[5173]: Accepted keyboard-interactive/pam for admin from 10.0.0.228 port 57879 ssh2
<date/time> r0 sshd[5173]: pam_unix(sshd:session): session opened for user admin(uid=1000) by (uid=0)
<date/time> r0 systemd-logind[365]: New session c5 of user admin.
<date/time> r0 sshd[5173]: User child is on pid 5225
<date/time> r0 sshd[5225]: Starting session: forced-command (config) '/usr/bin/atc_ssh_jail' on pts/1 for admin
from 10.0.0.228 port 57879 id 0
```

11.5.5 Failed remote login

```
<date/time> r0 sshd[20301]: Connection from 10.0.0.228 port 53934 on 10.0.0.146 port 22 rdomain ""
<date/time> r0 sshd[20301]: Postponed keyboard-interactive for admin from 10.0.0.228 port 53934 ssh2
[preauth]
<date/time> r0 sshd[20308]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser=
rhost=10.0.0.228  user=admin
<date/time> r0 sshd[20301]: error: PAM: Authentication failure for admin from 10.0.0.228
<date/time> r0 sshd[20301]: Failed keyboard-interactive/pam for admin from 10.0.0.228 port 53934 ssh2
```

11.5.6 Remote Logout

```
<date/time> r0 sshd[13747]: Received disconnect from 10.0.0.228 port 55740:11: disconnected by user
<date/time> r0 sshd[13747]: Disconnected from user admin 10.0.0.228 port 55740
<date/time> r0 sshd[13580]: pam_unix(sshd:session): session closed for user admin
<date/time> r0 systemd-logind[365]: Removed session c2.
```

11.6 Remote Session Establishment Via SSH Client

Remote sessions may be established to external entities via the use of the Machete router's SSH client and logged to auth.log.

11.6.1 Successful session establishment

```
<date/time> r0 ssh[11170]: Authenticated to 10.0.2.2 ([10.0.2.2]:22) using "password".
<date/time> r0 ssh[11170]: Transferred: sent 3776, received 5404 bytes, in 10.5 seconds
<date/time> r0 ssh[11170]: Bytes per second: sent 360.4, received 515.8
```

11.6.2 Failed session establishment

```
<date/time> r0 ssh[16703]: error: Permission denied, please try again.
<date/time> r0 ssh[16703]: error: Permission denied, please try again.
<date/time> r0 ssh[16703]: fatal: user@10.0.2.2: Permission denied (publickey,password).
```

11.7 Termination of Remote Session

When a remote account has been inactive for a specified period of time, the session is terminated and the following is logged to auth.log.

```
<date/time> r0 sshd[30269]: channel 0: closing after 300 seconds of inactivity
<date/time> r0 sshd[30269]: Close session: user admin from 10.0.0.228 port 55911 id 0
<date/time> r0 sshd[30269]: error: session_by_pid: unknown pid 30274
<date/time> r0 sshd[30269]: Received disconnect from 10.0.0.228 port 55911:11: disconnected by user
<date/time> r0 sshd[30269]: Disconnected from user admin 10.0.0.228 port 55911
<date/time> r0 sshd[30191]: pam_unix(sshd:session): session closed for user admin
<date/time> r0 systemd-logind[365]: Removed session c3.
```

11.8 Local Session Re-Authentication

The following are example messages generated for re-authentication at the local console when a configurable period of time has elapsed without user input. The account being unlocked in these example messages is the admin account. The following is logged to auth.log for every attempt.

11.8.1 Successful attempt at unlocking

```
<date/time> r0 vlock[3730]: Locked VC on tty1 for admin by (uid=1000)
<date/time> r0 admin: Attempting to unlock locked session
<date/time> r0 admin: Session successfully unlocked
<date/time> r0 vlock[3730]: Unlocked VC on tty1 for admin by (uid=1000)
```

11.8.2 Unsuccessful attempt at unlocking

```
<date/time> r0 admin: Attempting to unlock locked session
<date/time> r0 unix_chkpwd[4108]: password check failed for user (admin)
<date/time> r0 vlock[3738]: pam_unix(vlock:auth): authentication failure; logname=admin uid=1000 euid=1000
tty=tty1 rusuer= rhost= user=admin
```

11.9 Remote account locking

When an account is locked for remote login for a defined number of failed remote login-attempts for a defined number of seconds, the following audit record is logged to auth.log where tally is the current number of unsuccessful login-attempts and deny is the threshold:

```
<date/time> r0 sshd[7415]: Connection from 10.0.0.228 port 58521 on 10.0.0.146 port 22 rdomain ""
<date/time> r0 sshd[7415]: Postponed keyboard-interactive for admin from 10.0.0.228 port 58521 ssh2 [preauth]
<date/time> r0 sshd[7417]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser=
rhost=10.0.0.228  user=admin
<date/time> r0 sshd[7415]: error: PAM: Authentication failure for admin from 10.0.0.228
<date/time> r0 sshd[7415]: Failed keyboard-interactive/pam for admin from 10.0.0.228 port 58521 ssh2
<date/time> r0 sshd[7415]: Postponed keyboard-interactive for admin from 10.0.0.228 port 58521 ssh2 [preauth]
<date/time> r0 sshd[7469]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser=
rhost=10.0.0.228  user=admin
<date/time> r0 sshd[7415]: error: PAM: Authentication failure for admin from 10.0.0.228
<date/time> r0 sshd[7415]: Failed keyboard-interactive/pam for admin from 10.0.0.228 port 58521 ssh2
<date/time> r0 sshd[7415]: Postponed keyboard-interactive for admin from 10.0.0.228 port 58521 ssh2 [preauth]
<date/time> r0 sshd[7534]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser=
rhost=10.0.0.228  user=admin
<date/time> r0 sshd[7534]: pam_faillock(sshd:auth): Consecutive login failures for user admin account
temporarily locked
<date/time> r0 sshd[7415]: error: PAM: Authentication failure for admin from 10.0.0.228
<date/time> r0 sshd[7415]: Failed keyboard-interactive/pam for admin from 10.0.0.228 port 58521 ssh2
```

11.10 Create User

When a new account is created, the following is logged to general.log:

```
<date/time> r0 admin: User-Management: Contents of file /etc/passwd successfully changed by user: admin
<date/time> r0 admin: User-Management: --- /tmp/.etc.passwd
<date/time> r0 admin: User-Management: +++ /etc/passwd
<date/time> r0 admin: User-Management: @@ -30,0 +31 @@
<date/time> r0 admin: User-Management: +johndoe:x:1001:1001:Linux User,,,:/etc/users/johndoe:/bin/bash
<date/time> r0 admin: User-Management: End Changes of file /etc/passwd
<date/time> r0 admin: User-Management: Contents of file /etc/group successfully changed by user: admin
<date/time> r0 admin: User-Management: --- /tmp/.etc.group
<date/time> r0 admin: User-Management: +++ /etc/group
<date/time> r0 admin: User-Management: @@ -22 +22 @@
<date/time> r0 admin: User-Management: -sudo:x:27:admin
<date/time> r0 admin: User-Management: -sudo:x:27:admin,johndoe
<date/time> r0 admin: User-Management: @@ -41 +41 @@
<date/time> r0 admin: User-Management: -users:x:100:admin,root
<date/time> r0 admin: User-Management: +users:x:100:admin,root,johndoe
<date/time> r0 admin: User-Management: @@ -47 +47 @@
<date/time> r0 admin: User-Management: -transport:x:983:admin,root
<date/time> r0 admin: User-Management: +transport:x:983:admin,root,johndoe
<date/time> r0 admin: User-Management: @@ -53,2 +53,2 @@
<date/time> r0 admin: User-Management: -frrvty:x:989:frr,admin
<date/time> r0 admin: User-Management: -frr:x:990:admin
<date/time> r0 admin: User-Management: +frrvty:x:989:frr,admin,johndoe
<date/time> r0 admin: User-Management: +frr:x:990:admin,johndoe
<date/time> r0 admin: User-Management: @@ -65,0 +66 @@
<date/time> r0 admin: User-Management: +johndoe:x:1001:
```

```
<date/time> r0 admin: User-Management: End Changes of file /etc/group
<date/time> r0 admin: User-Management: User johndoe created and added to group users
<date/time> r0 admin: User-Management: Attempting to change password for user johndoe
<date/time> r0 passwd: password for johndoe changed by root
<date/time> r0 admin: User-Management: Password for user johndoe successfully changed
```

11.11 Change User Group

When an existing account group membership is changed, the following is logged to general.log:

```
<date/time> r0 admin: User-Management: User johndoe changed to group security
<date/time> r0 admin: User-Management: Contents of file /etc/group successfully changed by user: admin
<date/time> r0 admin: User-Management: --- /tmp/.etc.group
<date/time> r0 admin: User-Management: @@ -44,2 +44,2 @@
<date/time> r0 admin: User-Management: -config:x:980:admin,root
<date/time> r0 admin: User-Management: -security:x:981:admin,root
<date/time> r0 admin: User-Management: +config:x:980:admin,root,johndoe
<date/time> r0 admin: User-Management: +security:x:981:admin,root,johndoe
<date/time> r0 admin: User-Management: End Changes of file /etc/group
```

11.12 Delete User

When an existing account is deleted, the following is logged to general.log:

```
<date/time> r0 admin: User-Management: Contents of file /etc/passwd successfully changed by user: admin
<date/time> r0 admin: User-Management: @@ -31 +30,0 @@
<date/time> r0 admin: User-Management: End Changes of file /etc/passwd
<date/time> r0 admin: User-Management: Contents of file /etc/group successfully changed by user: admin
<date/time> r0 admin: User-Management: --- /tmp/.etc.group
<date/time> r0 admin: User-Management: @@ -22 +22 @@
<date/time> r0 admin: User-Management: +sudo:x:27:admin
<date/time> r0 admin: User-Management: -users:x:100:admin,root,johndoe
<date/time> r0 admin: User-Management: @@ -44,2 +44,2 @@
<date/time> r0 admin: User-Management: -security:x:981:admin,root,johndoe
<date/time> r0 admin: User-Management: +security:x:981:admin,root
<date/time> r0 admin: User-Management: -transport:x:983:admin,root,johndoe
<date/time> r0 admin: User-Management: @@ -53,2 +53,2 @@
<date/time> r0 admin: User-Management: -frr:x:990:admin,johndoe
<date/time> r0 admin: User-Management: +frr:x:990:admin
<date/time> r0 admin: User-Management: -johndoe:x:1001:
<date/time> r0 admin: User-Management: End Changes of file /etc/group
<date/time> r0 admin: User-Management: User johndoe was deleted
```

11.13 Change Password

When a password is changed, the event is logged to general.log:

11.13.1Successful password change

```
<date/time> r0 admin: User-Management: Attempting to change password for user johndoe
<date/time> r0 admin: User-Management: Password for user johndoe successfully changed
```

11.13.2Failed password change

```
<date/time> r0 admin: User-Management: Attempting to change password for user johndoe
<date/time> r0 admin: User-Management: Failed to change password for user johndoe
```

11.14 Changes to TSF data

The following are example messages generated and logged to general.log for changes to TSF data:

Format is always a context message followed by the file name that was modified along with the changes. The line number is logged and indicated by the numbers surrounded by '@@'. The changes are then logged and marked with – or + to indicated subtractions or additions to the effected line. The last line indicates the end of the changes to the file.

```
<date/time> r0 admin: Modify-Configuration: Contents of file /etc/crr/firewall.nft successfully changed by user:
admin
<date/time> r0 admin: Modify-Configuration: --- /etc/crr/firewall.nft
<date/time> r0 admin: Modify-Configuration: +++ /tmp/.etc.crr.firewall.nft
<date/time> r0 admin: Modify-Configuration: @@ -79 +79 @@
<date/time> r0 admin: Modify-Configuration: -
<date/time> r0 admin: Modify-Configuration: +      ip saddr 192.168.100.10 ip daddr 10.0.0.112 meta l4proto {
tcp,udp } th sport 2500 th dport 8080 accept
<date/time> r0 admin: Modify-Configuration: End Changes to file /etc/crr/firewall.nft
```

11.15 Stop/Start/Restart service

The following are example messages generated and logged to general.log for Stop/Start/Restart service by admin:

11.15.1Start

```
<date/time> r0 admin: Service: Attempting to start service ntpd
<date/time> r0 ntpd[31276]: ntpd 4.2.8p15@1.3728-o Tue 23 Jun 2020 09:22:22 AM UTC (1): Starting
<date/time> r0 ntpd[31276]: Command line: /usr/sbin/ntpd -u ntp:ntp -p /run/ntpd.pid -g
<date/time> r0 ntpd[31276]: ----------------------------------------------------
<date/time> r0 ntpd[31276]: ntp-4 is maintained by Network Time Foundation,
<date/time> r0 ntpd[31276]: Inc. (NTF), a non-profit 501(c)(3) public-benefit
<date/time> r0 ntpd[31276]: corporation.  Support and training for ntp-4 are
<date/time> r0 ntpd[31276]: available at https://www.nwtime.org/support
<date/time> r0 ntpd[31276]: ----------------------------------------------------
<date/time> r0 ntpd[31283]: proto: precision = 0.116 usec (-23)
<date/time> r0 ntpd[31283]: basedate set to 2020-06-11
<date/time> r0 ntpd[31283]: gps base set to 2020-06-14 (week 2110)
<date/time> r0 ntpd[31283]: Listen and drop on 0 v6wildcard [::]:123
<date/time> r0 ntpd[31283]: Listen and drop on 1 v4wildcard 0.0.0.0:123
<date/time> r0 ntpd[31283]: Listen normally on 2 lo 127.0.0.1:123
<date/time> r0 ntpd[31283]: Listen normally on 3 eth0 10.0.0.146:123
<date/time> r0 ntpd[31283]: Listen normally on 4 lo [::1]:123
<date/time> r0 ntpd[31283]: Listen normally on 5 eth0 [fe80::201:c0ff:fe31:90b%4]:123
<date/time> r0 ntpd[31283]: Listening on routing socket on fd #22 for interface updates
<date/time> r0 ntpd[31283]: kernel reports TIME_ERROR: 0x41: Clock Unsynchronized
<date/time> r0 ntpd[31283]: 0.0.0.0 c01d 0d kern kernel time sync enabled
<date/time> r0 ntpd[31283]: kernel reports TIME_ERROR: 0x41: Clock Unsynchronized
```

```
<date/time> r0 ntpd[31283]: 0.0.0.0 c012 02 freq_set kernel 0.000 PPM
<date/time> r0 ntpd[31283]: 0.0.0.0 c016 06 restart
<date/time> r0 admin: Service: ntpd start successful
```

11.15.2Stop

```
<date/time> r0 admin: Service: Attempting to stop service ntpd
<date/time> r0 ntpd[27201]: ntpd exiting on signal 15 (Terminated)
<date/time> r0 ntpd[27201]: 0.0.0.0 c01d 0d kern kernel time sync disabled
<date/time> r0 admin: Service: ntpd stop successful
```

11.15.3Restart

```
<date/time> r0 admin: Service: Attempting to restart service ntpd
<date/time> r0 ntpd[31283]: ntpd exiting on signal 15 (Terminated)
<date/time> r0 ntpd[31283]: 0.0.0.0 c01d 0d kern kernel time sync disabled
<date/time> r0 ntpd[31401]: ntpd 4.2.8p15@1.3728-o Tue 23 Jun 2020 09:22:22 AM UTC (1): Starting
<date/time> r0 ntpd[31401]: Command line: /usr/sbin/ntpd -u ntp:ntp -p /run/ntpd.pid -g
<date/time> r0 ntpd[31401]: ----------------------------------------------------
<date/time> r0 ntpd[31401]: ntp-4 is maintained by Network Time Foundation,
<date/time> r0 ntpd[31401]: Inc. (NTF), a non-profit 501(c)(3) public-benefit
<date/time> r0 ntpd[31401]: corporation.  Support and training for ntp-4 are
<date/time> r0 ntpd[31401]: available at https://www.nwtime.org/support
<date/time> r0 ntpd[31401]: ----------------------------------------------------
<date/time> r0 ntpd[31408]: proto: precision = 0.109 usec (-23)
<date/time> r0 ntpd[31408]: basedate set to 2020-06-11
<date/time> r0 ntpd[31408]: gps base set to 2020-06-14 (week 2110)
<date/time> r0 ntpd[31408]: Listen and drop on 0 v6wildcard [::]:123
<date/time> r0 ntpd[31408]: Listen and drop on 1 v4wildcard 0.0.0.0:123
<date/time> r0 ntpd[31408]: Listen normally on 2 lo 127.0.0.1:123
<date/time> r0 ntpd[31408]: Listen normally on 3 eth0 10.0.0.146:123
<date/time> r0 ntpd[31408]: Listen normally on 4 lo [::1]:123
<date/time> r0 ntpd[31408]: Listen normally on 5 eth0 [fe80::201:c0ff:fe31:90b%4]:123
<date/time> r0 ntpd[31408]: Listening on routing socket on fd #22 for interface updates
<date/time> r0 ntpd[31408]: kernel reports TIME_ERROR: 0x41: Clock Unsynchronized
<date/time> r0 ntpd[31408]: 0.0.0.0 c01d 0d kern kernel time sync enabled
<date/time> r0 ntpd[31408]: kernel reports TIME_ERROR: 0x41: Clock Unsynchronized
<date/time> r0 ntpd[31408]: 0.0.0.0 c012 02 freq_set kernel 0.000 PPM
<date/time> r0 ntpd[31408]: 0.0.0.0 c016 06 restart
<date/time> r0 admin: Service: ntpd restart successful
```

11.16 Setting System Time

11.16.1Manual

Setting time manually at the CLI, the following messages titles and bodies are generated for this event and logged to general.log. The old-time value is the timestamp of the initial log message and the new-time value is in the initial log value message body. All subsequent log messages show the updated time value as the timestamp.

```
<date/time> r0 admin: Set-Time: Updating system time to 062116002023.15 from local login\n'
<date/time> r0 admin: Set-Time: System time updated successfully\n'
<date/time> r0 admin: Set-Time: Hardware Clock updated successfully\n'
```

11.16.2 Automatic via NTP

NTPD won't sync if it the local clock is off by more than 1000 seconds from the reference clock(s) time. If the local clock is within 1000 seconds of the reference clock the following message is logged to general.log

```
<date/time> r0 ntpd[15691]: 0.0.0.0 c61c 0c clock_step -21.354884 s
<date/time> r0 ntpd[15691]: 0.0.0.0 c615 05 clock_sync
```

11.17 Audit

11.17.1 Modification of handling of audit data to an external IT entity

The following are example messages generated and logged to general.log for the modification of audit data to an external IT entity.

```
<date/time> r0 ARES[12965]: [SessionConfig.cpp:100] -- [AUDIT] Remote syslog server added. Server:
10.0.0.228, Protocol: tcp, Port: 8000
<date/time>  r0 ARES[12965]: [SysConfigUtil.cpp:61] -- [AUDIT] Contents of file /etc/rsyslog.d/remote.conf
successfully changed:
<date/time>  r0 ARES[12965]: [SysConfigUtil.cpp:64] -- [AUDIT] --- /tmp/session.D3SSlE
<date/time>  r0 ARES[12965]: [SysConfigUtil.cpp:64] -- [AUDIT] +++ /etc/rsyslog.d/remote.conf
<date/time>  r0 ARES[12965]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -0,0 +1 @@
<date/time>  r0 ARES[12965]: [SysConfigUtil.cpp:64] -- [AUDIT] +action(type="omfwd" target="10.0.0.228"
protocol="tcp" port="8000" queue.spoolDirectory="/var/spool/rsyslog" queue.filename="rmtsrv"
queue.maxdiskspace="5000000" queue.saveonshutdown="on" queue.type="LinkedList")
<date/time>  r0 ARES[12965]: [SysConfigUtil.cpp:66] -- [AUDIT] End of changes to file:
/etc/rsyslog.d/remote.conf.
<date/time>  r0 ARES[12965]: [SysConfigUtil.cpp:83] -- [AUDIT] Attempting to restart service rsyslog.
<date/time>  r0 ARES[12965]: [SysConfigUtil.cpp:88] -- [AUDIT] rsyslog restart successful.
```

11.18 Trusted Update

Trusted updates are a two-step process, the uploading of the firmware to the Machete router and the applying of the firmware version. The following are example messages generated and logged to general.log for this event.

11.18.1 Firmware update upload successful

```
<date/time> r0 admin: Trusted-Update: Firmware '2.0.0-rc11' staged
```

11.18.2 Firmware update upload failed digital signature validation

The digital signature validation must pass both checks for decryption and signature validation. A failure in either condition results in the following log message:

```
<date/time> r0 admin: Trusted-Update: Failed to decrypt firmware 'http://10.0.0.228:8000/crrfw_2.0.0-rc7-
FIT2.ipks'
```

11.18.3 Firmware update upload verified digital signature but failed

```
<date/time> r0 admin: Trusted-Update: Failed to stage firmware:  * Solver encountered 1 problem(s):
<date/time> r0 admin: Trusted-Update: * Problem 1/1:
<date/time> r0 admin: Trusted-Update: *   - conflicting requests
```

```
<date/time> r0 admin: Trusted-Update: *   - package crrfw-2.0.0-rc11s1.cm6000 does not have a compatible
architecture
<date/time> r0 admin: Trusted-Update: *
<date/time> r0 admin: Trusted-Update: * Solution 1:
<date/time> r0 admin: Trusted-Update: *   - do not ask to install crrfw-2.0.0-rc11s1.cm6000
<date/time> r0 admin: Trusted-Update: FAILED: No such firmware version found: '2.0.0-rc11s1'
```

11.18.4Successful firmware update

```
<date/time> r0 admin: Trusted-Update: Verifying firmware version 2.0.0-rc11
<date/time> r0 admin: Trusted-Update: Current CRR firmware '2.0.0-rc9'
<date/time> r0 admin: Trusted-Update: Updating default CRR firmware to '2.0.0-rc11'
<date/time> r0 admin: Trusted-Update: Bootloader default changed to 'crr-2.0.0-rc11.conf'
<date/time> r0 admin: Trusted-Update: Rebooting into CRR firmware '2.0.0-rc11'
```

11.19 Packet Overruns

Overrun events are logged to general.log and the title of "RX-Overrun", which can be searched for in the log.

```
<date/time> crr RX-Overrun[367]: Interface: eth0 -- 5 new packets dropped because of receive FIFO buffer
overrun (total=10)
```

11.20 Packet Filters

Packet Filter events are logged to firewall.log and have the title of “Restrictive-Firewall” and “Firewall for the firewall-pre and firewall-post service logs. Packet filter events are logged anytime the packet filter rules are applied, both when reset to deny everything and when applying the user rules.

Packet filter deny all rules are applied on system start-up, before networking is enabled. User rules are then applied after ARES has started. If ARES ever stops running, the deny-all rules are re-applied, until ARES is started again, which triggers the user rules to be re-applied. User rules can also be re-applied on command through the CLI, to apply any new rules configured by the user without needing to restart the system or ARES.

Starting the firewall-pre service:

```
<date/time> crr Restrictive-Firewall[354]: Resetting Firewall
<date/time> crr Restrictive-Firewall[354]: Firewall Rules Test...ok
<date/time> crr Restrictive-Firewall[354]: Apply Firewall Rules...ok
<date/time> crr Restrictive-Firewall[354]: /etc/crr/restrictive_firewall.nft
<date/time> crr Restrictive-Firewall[354]: # Flush everything first
<date/time> crr Restrictive-Firewall[354]: flush ruleset
<date/time> crr Restrictive-Firewall[354]: table inet global {
<date/time> crr Restrictive-Firewall[354]: chain input {
<date/time> crr Restrictive-Firewall[354]: type filter hook input priority filter; policy drop;
<date/time> crr Restrictive-Firewall[354]: # Allow all local host traffic
<date/time> crr Restrictive-Firewall[354]: ip saddr 127.0.0.1 ip daddr 127.0.0.1 accept
<date/time> crr Restrictive-Firewall[354]: ip6 saddr ::1 ip6 daddr ::1 accept
<date/time> crr Restrictive-Firewall[354]: # Log all dropped packets
<date/time> crr Restrictive-Firewall[354]: log prefix "CRRFW Input Dropped: "
<date/time> crr Restrictive-Firewall[354]: }
<date/time> crr Restrictive-Firewall[354]: chain forward {
<date/time> crr Restrictive-Firewall[354]: type filter hook forward priority filter; policy drop;
<date/time> crr Restrictive-Firewall[354]: # Log all dropped packets
<date/time> crr Restrictive-Firewall[354]: log prefix "CRRFW Forward Dropped: "
<date/time> crr Restrictive-Firewall[354]: }
<date/time> crr Restrictive-Firewall[354]: chain output {
<date/time> crr Restrictive-Firewall[354]: type filter hook output priority filter; policy drop;
<date/time> crr Restrictive-Firewall[354]: # Allow all localhost traffic
<date/time> crr Restrictive-Firewall[354]: ip saddr 127.0.0.1 ip daddr 127.0.0.1 accept
<date/time> crr Restrictive-Firewall[354]: ip6 saddr ::1 ip6 daddr ::1 accept
<date/time> crr Restrictive-Firewall[354]: # Log all dropped packets
<date/time> crr Restrictive-Firewall[354]: log prefix "CRRFW Output Dropped: "
<date/time> crr Restrictive-Firewall[354]: }
<date/time> crr Restrictive-Firewall[354]: }
```

Starting the firewall-post service:

```
<date/time> crr Firewall[5268]: Firewall Rules Test...ok
<date/time> crr Firewall[5304]: Firewall Rules Test...ok
<date/time> crr Firewall[5304]: Apply Firewall Rules...ok
<date/time> crr Firewall[5304]: /etc/crr/firewall.nft
<date/time> crr Firewall[5304]: # Flush everything first to install a fresh set of rules defined below
<date/time> crr Firewall[5304]: flush ruleset
<date/time> crr Firewall[5304]: # Define ports/protocols used by default for ease of use in the rules below
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # Define UDP Ports
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # Allow all ipsec connection attempts (IKE, MOBIKE)
<date/time> crr Firewall[5304]: define ike_udp_dports = { 500, 4500 }
<date/time> crr Firewall[5304]: # Allow ARES udp routing protocols
<date/time> crr Firewall[5304]: # 12345 - ARES neighbor discoveru
<date/time> crr Firewall[5304]: # 12346 - ARES Mobility Service
<date/time> crr Firewall[5304]: # 12347 - ARES Mobility Response
<date/time> crr Firewall[5304]: # 32768 - ARES Link Sense
<date/time> crr Firewall[5304]: define ares_udp_dports = { 12345, 12346, 12347, 32768 }
<date/time> crr Firewall[5304]: # Allow Common udp routing protocols - add port number(s) to udp_dports
definition below.
<date/time> crr Firewall[5304]: # 161 - Default port for SNMP Managers communicating with SNMP Agents.
<date/time> crr Firewall[5304]: # Set of all allowed UDP destination ports incoming and outgoing
<date/time> crr Firewall[5304]: define udp_dports = { $ike_udp_dports, $ares_udp_dports }
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # Define TCP Ports
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # Allow SSH traffic and ARES certificate exchange (12344)
<date/time> crr Firewall[5304]: define tcp_dports = { 22, 12344 }
<date/time> crr Firewall[5304]: define tcp_sports = { 22 }
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # Define Protocols
```

```
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # Allow standard routing protocols
<date/time> crr Firewall[5304]: define routing_protos = { icmp, ospf, pim, igmp }
<date/time> crr Firewall[5304]: # Allow ARES routing protocols
<date/time> crr Firewall[5304]: # gre - ARES neighbor tunnels
<date/time> crr Firewall[5304]: # 55 - ARES neighbor tunnels
<date/time> crr Firewall[5304]: # 155 - ARES ESP, cut through routing
<date/time> crr Firewall[5304]: # 156 - ARES ESP, cut through routing
<date/time> crr Firewall[5304]: define ares_routing_protos = { gre, 55, 155, 156 }
<date/time> crr Firewall[5304]: # Allow IPSEC protocols
<date/time> crr Firewall[5304]: define ipsec_protos = { esp, ah }
<date/time> crr Firewall[5304]: # Set of all allowed IP protocols incoming and outgoing
<date/time> crr Firewall[5304]: define protos = { $routing_protos, $ares_routing_protos, $ipsec_protos }
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # Start of the actual rule specifications inet global table supports both IPv4
<date/time> crr Firewall[5304]: # and IPv6
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: #                          WARNING
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # The inet global table is the default primary table of the firewall. It must
<date/time> crr Firewall[5304]: # exist for some command line functions to operate correctly. Do not remove or
<date/time> crr Firewall[5304]: # change this table name or family unless you know what you are doing.
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: table inet global {
<date/time> crr Firewall[5304]:
############################################################################
<date/time> crr Firewall[5304]: # input chain for traffic destined locally
<date/time> crr Firewall[5304]:
############################################################################
<date/time> crr Firewall[5304]: chain input {
<date/time> crr Firewall[5304]: # The default policy can be changed here
<date/time> crr Firewall[5304]: # The log for dropped packets at the end of the table
<date/time> crr Firewall[5304]: # should be updated as well to reflect if the policy is
<date/time> crr Firewall[5304]: # to drop (uncommented) or accept (comment out)
<date/time> crr Firewall[5304]: type filter hook input priority filter; policy drop;
<date/time> crr Firewall[5304]: # Allow all local host traffic
<date/time> crr Firewall[5304]: ip saddr 127.0.0.1 ip daddr 127.0.0.1 accept
<date/time> crr Firewall[5304]: ip6 saddr ::1 ip6 daddr ::1 accept
<date/time> crr Firewall[5304]: # Allow any established connections
<date/time> crr Firewall[5304]: ct state established,related accept
<date/time> crr Firewall[5304]: # Allow defined ip protocols
<date/time> crr Firewall[5304]: ip protocol $protos accept
<date/time> crr Firewall[5304]: ip6 nexthdr $protos accept
<date/time> crr Firewall[5304]: # Allow defined udp traffic
<date/time> crr Firewall[5304]: udp dport $udp_dports accept
<date/time> crr Firewall[5304]: # Allow defined tcp traffic
<date/time> crr Firewall[5304]: tcp dport $tcp_dports accept
<date/time> crr Firewall[5304]: tcp sport $tcp_sports accept
<date/time> crr Firewall[5304]: # Log all dropped packets
<date/time> crr Firewall[5304]: log prefix "CRRFW Input Dropped: "
<date/time> crr Firewall[5304]: }
```

```
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # forward chain for traffic being forwarded
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: chain forward {
<date/time> crr Firewall[5304]: # The default policy can be changed here
<date/time> crr Firewall[5304]: # The log for dropped packets at the end of the table
<date/time> crr Firewall[5304]: # should be updated as well to reflect if the policy is
<date/time> crr Firewall[5304]: # to drop (uncommented) or accept (comment out)
<date/time> crr Firewall[5304]: type filter hook forward priority filter; policy accept;
<date/time> crr Firewall[5304]: # Log all dropped packets
<date/time> crr Firewall[5304]: #log prefix "CRRFW Forward Dropped: "
<date/time> crr Firewall[5304]: }
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # output chain for locally generated traffic
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: chain output {
<date/time> crr Firewall[5304]: # The default policy can be changed here
<date/time> crr Firewall[5304]: # The log for dropped packets at the end of the table
<date/time> crr Firewall[5304]: # should be updated as well to reflect if the policy is
<date/time> crr Firewall[5304]: # to drop (uncommented) or accept (comment out)
<date/time> crr Firewall[5304]: type filter hook output priority filter; policy accept;
<date/time> crr Firewall[5304]: # Allow all localhost traffic
<date/time> crr Firewall[5304]: ip saddr 127.0.0.1 ip daddr 127.0.0.1 accept
<date/time> crr Firewall[5304]: ip6 saddr ::1 ip6 daddr ::1 accept
<date/time> crr Firewall[5304]: # Allow established connections
<date/time> crr Firewall[5304]: ct state established accept
<date/time> crr Firewall[5304]: # Allow defined ip protocols
<date/time> crr Firewall[5304]: ip protocol $protos accept
<date/time> crr Firewall[5304]: ip6 nexthdr $protos accept
<date/time> crr Firewall[5304]: # Allow defined udp traffic
<date/time> crr Firewall[5304]: udp dport $udp_dports accept
<date/time> crr Firewall[5304]: # Allow defined tcp traffic
<date/time> crr Firewall[5304]: tcp dport $tcp_dports accept
<date/time> crr Firewall[5304]: tcp sport $tcp_sports accept
<date/time> crr Firewall[5304]: # Log all dropped packets
<date/time> crr Firewall[5304]: #log prefix "CRRFW Output Dropped: "
<date/time> crr Firewall[5304]: }
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: # Example chains with hooks for packet nat rules
<date/time> crr Firewall[5304]:
################################################################################
<date/time> crr Firewall[5304]: #   chain nat_pre {
<date/time> crr Firewall[5304]: #      type nat hook prerouting priority dstnat;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]: #   chain nat_in {
<date/time> crr Firewall[5304]: #      type nat hook input priority 100;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]: #   chain nat_out {
<date/time> crr Firewall[5304]: #      type nat hook output priority -100;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]: #   chain nat_post {
```

```
<date/time> crr Firewall[5304]: #      type nat hook postrouting priority srcnat;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]:
###############################################################################
<date/time> crr Firewall[5304]: # Example chains with hooks for packet mangle rules
<date/time> crr Firewall[5304]:
###############################################################################
<date/time> crr Firewall[5304]: #   chain mangle_pre {
<date/time> crr Firewall[5304]: #      type filter hook prerouting priority mangle;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]: #   chain mangle_in {
<date/time> crr Firewall[5304]: #      type filter hook input priority mangle;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]: #   chain mangle_fwd {
<date/time> crr Firewall[5304]: #      type filter hook forward priority mangle;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]: #   chain mangle_out {
<date/time> crr Firewall[5304]: #      type route hook output priority mangle;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]: #   chain mangle_post {
<date/time> crr Firewall[5304]: #      type filter hook postrouting priority mangle;
<date/time> crr Firewall[5304]: #   }
<date/time> crr Firewall[5304]: }
```

11.21 Dropped Packets

Examples of logging messages of dropped packets: included are TCP, UDP, and one of each IPv4 and IPv6 for both incoming and outgoing. Forward packets are basically the same, just with “FIREWALL_FWD” instead. Dropped packets are logged to firewall.log

```
<date/time> r0 kernel:CRRFW Output Dropped: IN= OUT=eth0 SRC=10.0.1.1 DST=10.0.1.2 LEN=168
TOS=0x00 PREC=0x00 TTL=64 ID=57725 DF PROTO=TCP SPT=22 DPT=63624 WINDOW=501 RES=0x00
ACK PSH URGP=0
<date/time> r0 kernel:CRRFW Input Dropped: IN=eth0 OUT= MAC=ff:ff:ff:ff:ff:ff:84:a9:3e:d8:b9:c2:08:00
SRC=0.0.0.0 DST=255.255.255.255 LEN=328 TOS=0x00 PREC=0x00 TTL=64 ID=5 PROTO=UDP SPT=68
DPT=67 LEN=308
<date/time> r0 kernel:CRRFW Input Dropped: IN=eth0 OUT=
MAC=33:33:00:00:00:01:3c:82:c0:fc:64:e7:86:dd SRC=fe80:0000:0000:0000:3e82:c0ff:fefc:64e7
DST=ff02:0000:0000:0000:0000:0000:0000:0001 LEN=160 TC=0 HOPLIMIT=255 FLOWLBL=908413
PROTO=ICMPv6 TYPE=134 CODE=0
<date/time> r0 kernel:CRRFW Output Dropped: IN= OUT=eth0 SRC=0.0.0.0 DST=224.0.0.2 LEN=32
TOS=0x00 PREC=0xC0 TTL=1 ID=0 DF PROTO=2
<date/time> r0 kernel:CRRFW Output Dropped: IN= OUT=eth0
SRC=0000:0000:0000:0000:0000:0000:0000:0000 DST=ff02:0000:0000:0000:0000:0000:0000:0016 LEN=116
TC=0 HOPLIMIT=1 FLOWLBL=0 PROTO=ICMPv6 TYPE=143 CODE=0 MARK=0xd4
```

11.22 NTP

NTP is configured in ares.json by adding a top-level object. The following are log outputs for configuring NTP and are logged to general.log.

Example configuration of NTP without a key:

```
<date/time> CRR ARES[11778]: [SysConfigUtil.cpp:61] -- [AUDIT] Contents of file /etc/ntp.ares.conf
successfully changed:
<date/time> CRR ARES[11778]: [SysConfigUtil.cpp:64] -- [AUDIT] --- /tmp/session.V9uyAk
<date/time> CRR ARES[11778]: [SysConfigUtil.cpp:64] -- [AUDIT] +++ /etc/ntp.ares.conf
<date/time> CRR ARES[11778]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -7,0 +8 @@
<date/time> CRR ARES[11778]: [SysConfigUtil.cpp:64] -- [AUDIT] +server 192.168.0.1
<date/time> CRR ARES[11778]: [SysConfigUtil.cpp:66] -- [AUDIT] End of changes to file: /etc/ntp.ares.conf.
<date/time> CRR ARES[11778]: [SysConfigUtil.cpp:83] -- [AUDIT] Attempting to restart service ntpd.
```

Example configuration of NTP server using keys:

```
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:61] -- [AUDIT] Contents of file /etc/ntp.ares.conf
successfully changed:
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:64] -- [AUDIT] --- /tmp/session.24eaFt
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:64] -- [AUDIT] +++ /etc/ntp.ares.conf
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -6,0 +7 @@
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:64] -- [AUDIT] +controlkey 1
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:66] -- [AUDIT] End of changes to file: /etc/ntp.ares.conf.
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:61] -- [AUDIT] Contents of file /etc/swanctl/psks/ntp.keys
successfully changed:
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:64] -- [AUDIT] --- /tmp/session.SD6wKD
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:64] -- [AUDIT] +++ /etc/swanctl/psks/ntp.keys
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -5,0 +6 @@
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:64] -- [AUDIT] +1 MD5
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:66] -- [AUDIT] End of changes to file:
/etc/swanctl/psks/ntp.keys.
<date/time> CRR ARES[19835]: [SysConfigUtil.cpp:83] -- [AUDIT] Attempting to restart service ntpd.
```

Example configuration of NTP server and peer:

```
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:61] -- [AUDIT] Contents of file /etc/ntp.ares.conf
successfully changed:
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] --- /tmp/session.8a97n6
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] +++ /etc/ntp.ares.conf
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -9 +9,2 @@
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] -server 192.168.0.1
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] +server 192.168.0.1 key 1 prefer
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] +peer 192.168.10.1 key 2
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:66] -- [AUDIT] End of changes to file: /etc/ntp.ares.conf.
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:61] -- [AUDIT] Contents of file /etc/swanctl/psks/ntp.keys
successfully changed:
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] --- /tmp/session.K4sCkz
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] +++ /etc/swanctl/psks/ntp.keys
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -6,0 +7 @@
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:64] -- [AUDIT] +2 MD5
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:66] -- [AUDIT] End of changes to file:
/etc/swanctl/psks/ntp.keys.
<date/time> CRR ARES[28012]: [SysConfigUtil.cpp:83] -- [AUDIT] Attempting to restart service ntpd.
Example of removal of NTP server/peer from configuration
<date/time> CRR admin: Service: ares restart successful
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:61] -- [AUDIT] Contents of file /etc/ntp.ares.conf
successfully changed:
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] --- /tmp/session.RbxecE
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] +++ /etc/ntp.ares.conf
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -7 +6,0 @@
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] -controlkey 1
```

```
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -9,2 +7,0 @@
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] -server 192.168.0.1 key 1 prefer
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] -peer 192.168.10.1 key 2
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:66] -- [AUDIT] End of changes to file: /etc/ntp.ares.conf.
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:61] -- [AUDIT] Contents of file /etc/swanctl/psks/ntp.keys
successfully changed:
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] --- /tmp/session.1S2HqF
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] +++ /etc/swanctl/psks/ntp.keys
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] @@ -6,2 +5,0 @@
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] -1 MD5
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:64] -- [AUDIT] -2 MD5
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:66] -- [AUDIT] End of changes to file:
/etc/swanctl/psks/ntp.keys.
<date/time> CRR ARES[31769]: [SysConfigUtil.cpp:83] -- [AUDIT] Attempting to restart service ntpd.
```

11.23 Self-Test

11.23.1Success

The following is an example of a successful Power-On Self-Test. All self-test audit logs are tagged with the “Self-Test” identifier and logged to general.log.

```
<date/time> crr Self-Test[362]: Begin Power-on Self Tests
<date/time> crr Self-Test[385]: HMAC : (Module_Integrity) : Pass
<date/time> crr Self-Test[385]: SHA1 : (KAT_Digest) : Pass
<date/time> crr Self-Test[385]: SHA2 : (KAT_Digest) : Pass
<date/time> crr Self-Test[385]: SHA3 : (KAT_Digest) : Pass
<date/time> crr Self-Test[385]: TDES : (KAT_Cipher) : Pass
<date/time> crr Self-Test[385]: AES_GCM : (KAT_Cipher) : Pass
<date/time> crr Self-Test[385]: AES_ECB_Decrypt : (KAT_Cipher) : Pass
<date/time> crr Self-Test[385]: RSA : (KAT_Signature) : RNG : (Continuous_RNG_Test) : Pass
<date/time> crr Self-Test[385]: Pass
<date/time> crr Self-Test[385]: ECDSA : (PCT_Signature) : Pass
<date/time> crr Self-Test[385]: ECDSA : (PCT_Signature) : Pass
<date/time> crr Self-Test[385]: DSA : (PCT_Signature) : Pass
<date/time> crr Self-Test[385]: TLS13_KDF_EXTRACT : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: TLS13_KDF_EXPAND : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: TLS12_PRF : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: PBKDF2 : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: SSHKDF : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: KBKDF : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: HKDF : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: SSKDF : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: X963KDF : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: X942KDF : (KAT_KDF) : Pass
<date/time> crr Self-Test[385]: HASH : (DRBG) : Pass
<date/time> crr Self-Test[385]: CTR : (DRBG) : Pass
<date/time> crr Self-Test[385]: HMAC : (DRBG) : Pass
<date/time> crr Self-Test[385]: DH : (KAT_KA) : Pass
<date/time> crr Self-Test[385]: ECDH : (KAT_KA) : Pass
<date/time> crr Self-Test[385]: RSA_Encrypt : (KAT_AsymmetricCipher) : Pass
<date/time> crr Self-Test[385]: RSA_Decrypt : (KAT_AsymmetricCipher) :
<date/time> crr Self-Test[398]: This CPU supports the RDRAND instruction
<date/time> crr Self-Test[398]: RDRAND without retries
```

```
<date/time> crr Self-Test[398]: rand16= 1534
<date/time> crr Self-Test[398]: rand32= 550789823
<date/time> crr Self-Test[398]: rand64= 10499314898130617338
<date/time> crr Self-Test[398]: RDRAND with up to 10 retries, each
<date/time> crr Self-Test[398]: rand16= 45091
<date/time> crr Self-Test[398]: rand32= 1235473284
<date/time> crr Self-Test[398]: rand64= 11372214855264547225
<date/time> crr Self-Test[398]: Fill an array of 16 unsigned ints with random values
<date/time> crr Self-Test[398]: Got 16 rands:
<date/time> crr Self-Test[398]: val[0]= 2575015158
<date/time> crr Self-Test[398]: val[1]= 2915969854
<date/time> crr Self-Test[398]: val[2]= 2078876017
<date/time> crr Self-Test[398]: val[3]= 1647989839
<date/time> crr Self-Test[398]: val[4]= 3899019765
<date/time> crr Self-Test[398]: val[5]= 2553435331
<date/time> crr Self-Test[398]: val[6]= 1536329463
<date/time> crr Self-Test[398]: val[7]= 558226555
<date/time> crr Self-Test[398]: val[8]= 978991087
<date/time> crr Self-Test[398]: val[9]= 2592662256
<date/time> crr Self-Test[398]: val[10]= 2519019092
<date/time> crr Self-Test[398]: val[11]= 3238661039
<date/time> crr Self-Test[398]: val[12]= 28617825
<date/time> crr Self-Test[398]: val[13]= 3666081456
<date/time> crr Self-Test[398]: val[14]= 1239356168
<date/time> crr Self-Test[398]: val[15]= 3927587068
<date/time> crr Self-Test[398]: Fill unaligned buffer with 112 random bytes
<date/time> crr Self-Test[398]: Got 112 bytes:
<date/time> crr Self-Test[398]: 0x7fff60dcc0f0:  00 00 00 00 00 3B 6A 5C  B6 5E C7 E8 23 5B BA 79
<date/time> crr Self-Test[398]: 0x7fff60dcc100:  C1 07 F6 C8 45 19 B9 89  0E 69 88 7C 94 15 6D BB
<date/time> crr Self-Test[398]: 0x7fff60dcc110:  C8 1D 95 F3 4F FB 37 81  02 56 51 8A BA A3 0B 47
<date/time> crr Self-Test[398]: 0x7fff60dcc120:  0C 86 5D 12 11 DA CF 11  53 D5 04 2B 3D 56 5E 72
<date/time> crr Self-Test[398]: 0x7fff60dcc130:  0F 81 DF 54 8E 09 2E CD  1F AA 87 09 5B 71 E0 D2
<date/time> crr Self-Test[398]: 0x7fff60dcc140:  D1 AE FF 0E 4E 42 DB 63  71 0A B5 E6 FD B1 44 B1
<date/time> crr Self-Test[398]: 0x7fff60dcc150:  FC 8E E7 69 42 B3 D1 3C  90 E3 F0 E3 7C 79 82 32
<date/time> crr Self-Test[398]: Fill an aligned buffer with 128 random bytes
<date/time> crr Self-Test[398]: Got 128 bytes:
<date/time> crr Self-Test[398]: 0x7fff60dcc0f0:  31 24 3E 3A 95 34 87 20  07 48 6B E1 17 FB 18 8A
<date/time> crr Self-Test[398]: 0x7fff60dcc100:  A5 09 25 1F 7A E3 6C 33  D4 F8 30 14 7C DF 37 67
<date/time> crr Self-Test[398]: 0x7fff60dcc110:  C5 B7 9D 84 11 26 16 4D  64 EA 98 06 09 5F 3E 65
<date/time> crr Self-Test[398]: 0x7fff60dcc120:  94 B3 5D B6 C4 F2 B7 22  F1 61 EF C6 2E 13 0A D3
<date/time> crr Self-Test[398]: 0x7fff60dcc130:  BA 6C FA 2B 61 76 F4 95  C9 6B BC 76 41 CE F4 A9
<date/time> crr Self-Test[398]: 0x7fff60dcc140:  E2 90 A7 BF C4 29 1C 6C  52 B1 92 E0 7A 3F 7C 10
<date/time> crr Self-Test[398]: 0x7fff60dcc150:  22 DF 67 53 33 75 AC 60  E3 3A B5 B1 4C A7 09 60
<date/time> crr Self-Test[398]: This CPU supports the RDSEED instruction
<date/time> crr Self-Test[398]: RDSEED without retries
<date/time> crr Self-Test[398]: seed16= 44529
<date/time> crr Self-Test[398]: seed32= 3165339335
<date/time> crr Self-Test[398]: seed64= 15329113813757536795
<date/time> crr Self-Test[385]: Pass
<date/time> crr Self-Test[385]: RSA_Decrypt : (KAT_AsymmetricCipher) : Pass
<date/time> crr Self-Test[385]: INSTALL PASSED
<date/time> crr Self-Test[381]: RDRAND Health Test...ok
<date/time> crr Self-Test[362]: OpenSSL3 FIPS KATS...ok
<date/time> crr Self-Test[409]: OpenSSL3 ECDSA Key Generation...ok
<date/time> crr Self-Test[407]: OpenSSL3 RSA Key Generation...ok
<date/time> crr Self-Test[1367]: 2.0.0-rc11/bzImage_5.15.94-intel-pk-standard: OK
```

```
<date/time> crr Self-Test[1367]: 2.0.0-rc11/microcode_20230512.cpio: OK
<date/time> crr Self-Test[1367]: 2.0.0-rc11/rootfs_2.0.0-rc11.cpio.gz: OK
<date/time> crr Self-Test[410]: Firmware Integrity test...ok
<date/time> crr Self-Test[408]: OpenSSL3 DSA Key Generation...ok
<date/time> crr Self-Test[383]: Memory Test...ok
<date/time> crr Self-Test[362]: Power-on Self Test...ok
```

11.23.2Failure

The following is an example of a failure in the Power-On Self-Test. All self-test audit logs are tagged with the “Self-Test” identifier and logged to general.log.

```
<date/time> crr Self-Test[352]: Begin Power-on Self Tests
<date/time>  crr RX-Overrun[351]: Watching network device erspan0 for FIFO buffer receive overrun packets
<date/time>  crr Self-Test[364]: Can't open "/usr/lib/ossl-modules/fips-module.so" for reading, No such file or
directory
<date/time>  crr Self-Test[364]: 4047ED1B8C7F0000:error:80000002:system library:BIO_new_file:No such
file or directory:../openssl-3.0.11/crypto/bio/bss_file.c:67:calling fopen(/usr/lib/ossl-modules/fips-module.so, rb)
<date/time>  crr Self-Test[364]: 4047ED1B8C7F0000:error:10000080:BIO routines:BIO_new_file:no such
file:../openssl-3.0.11/crypto/bio/bss_file.c:75:
<date/time>  crr Self-Test[364]: Failed to open module file
<date/time>  crr Self-Test[364]: INSTALL FAILED
<date/time>  crr Self-Test[352]:  OpenSSL FIPS KATS...FAIL
<date/time>  crr Self-Test[352]:  Power-on Self Test...FAIL
```

12 Example Operational Environment Configuration

12.1 ARES Configuration

```
{
  "RouterId" : 10,
  "RouterName" : "r0",
  "SecurityMode" : "niap",
  "Session" : { "Timeout" :  86400 },
  "Log" : {
    "Verbosity" : "warning"
    "Audit": {
      "Syslog": {
        "IPAddress": "10.0.3.1",
        "Port": 514,
        "Protocol": "udp"
      }
    }
  },
  "NTP": {
    "ControlKey": 1,
    "Keys": [
      {
        "ID": 1,
        "Type": "md5",
        "Key": "file://simplekey1"
      }
    ],
    "Sources": [
      {
        "KeyID": 1,
        "Address", "10.0.3.1",
        "Type": "server",
        "Prefer": true
      }
    ]
  },
  "Plugins": [ "sysconfig", "vpn"],
  "Multicast" : {
    "Enabled" : true,
    "PIM" : { "RP-List" : [ { "RP" :  "127.0.0.1",
    "Groups" : ["224.0.0.0/4" ] } ] }
  },
  "Links" : [
    {
      "Name" : "LAN",
      "Type" : "simple",
      "Interface" : {
        "Name" : "eth0",
        "Inet4Addresses" : [ "192.168.10.254/24" ]
      }
    },
    {
      "Name" : "WAN",
      "Type" : "simple",
      "Interface" : {
        "Name" : "eth1",
        "Inet4Addresses" : [ "192.168.100.254/24" ]
      }
    }
  ],
  "SSH": {
    "Client": [
      {
        "AutoConnect": true,
        "Host": "192.168.1.1",
        "User": "testuser",
        "IdentityFile": [ "file://ecdsa/tunnel-ssh.key" ],
        "RekeyLimit": {
          "MaxData": "50K",
          "MaxTime": "5m"
        }
      }
    ],
    "Server": {
      "Ciphers": ["aes128-cbc", "aes256-cbc", "aes128-ctr",
"aes256-ctr", "aes128-gcm@openssh.com", "aes256-
gcm@openssh.com"],
      "KexAlgorithms": ["ecdh-sha2-nistp256", "diffie-
hellman-group14-sha256", "diffie-hellman-group16-sha512",
"diffie-hellman-group18-sha512", "ecdh-sha2-nistp384",
"ecdh-sha2- nistp521", "diffie-hellman-group14-sha1"],
      "MACs": ["hmac-sha1", "hmac-sha2-256", "hmac-sha2-
512", "hmac-sha1-96" ]
      "RekeyLimit": {
        "MaxData": "50K",
        "MaxTime": "5m"
      }
    }
  },
  "VPN": {
    "Profiles": {
```

```
      "connections": {
        "r3": {
          "version" : 2,
          "proposals": {
            "list": [
              {
        "encryption_algorithms": ["aes128", "aes256",
"aes128ctr", "aes256ctr"],
        "integrity_algorithms": ["sha256", "sha384",
"sha512"],
        "pseudo_random_functions": ["prfsha256",
"prfsha384", "prfsha512"],
        "ke_groups": ["modp2048", "modp3072", "modp4096",
"modp6144", "modp8192", "modp2048s256", "ecp256", "ecp384",
"ecp521"]
              }
            ]
          },
          "local_addrs": [ "10.0.1.1" ],
          "remote_addrs": [ "10.0.2.3" ],
          "local": {
            "auth": "pubkey",
            "certs" :  ["file://router.crt"],
            "id": "a san field from the cert such ip
address, fqdn, email, etc"
          },
          "remote": {
            "auth": "pubkey"
          },
          "children": {
            "red": {
              "esp_proposals": {
                "list": [
                  {
        "encryption_algorithms": ["aes128", "aes256",
"aes128ctr", "aes256ctr"],
        "integrity_algorithms": ["sha256", "sha384",
"sha512"],
        "pseudo_random_functions": ["prfsha256",
"prfsha384", "prfsha512"],
        "ke_groups": ["modp2048", "modp3072", "modp4096",
"modp6144", "modp8192", "modp2048s256", "ecp256", "ecp384",
"ecp521"]
      }
                ]
              },
              "local_ts": [ "10.0.0.0/24" ],
              "remote_ts": [ "10.0.3.0/24" ],
              "start_action": "trap|start"
            }
          }
        }
      }
    }
  }
}
```