Common Criteria Configuration Guide for Regulus VPN with SpaceX OS v1.0

PP_Network Device

NDcPP VPN v2.2e

MOD_VPNGW v1.1

VID 11327

March 31, 2023

Revision

Version	Date	Title or Brief Description
0.1	3/31/2023	Initial Release with NIAP Check-Out package for Regulus

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This document covers the Regulus Virtual Private Network (VPN) component, how to configure and administer the component to be in accordance with the National Information Assurance Partnership (NIAP) validated configuration.

1.1 Purpose

This document details the configuration guidance to put the Regulus VPN, or Target of Evaluation (TOE), into the Common Criteria evaluated configuration. This guidance will inform users how to configure a Regulus VPN to be in compliance with Network Device Collaborative Protection Profile v2.2e (NDcPP v2.2e) and the VPN Gateway Protection Profile v1.1 (MOD_VPNGW 1.1).

1.2 Operational Environment

The required hardware and software for TOE functionality is listed below in Table 1.

Table 1 : Required Test Equipment

Component	Description
Audit Log Server	This stores audit logs generated by the TOE
Certificate Authority	This includes the HW and SW required to generate and sign certificates for use on the TOE. This can be used during certificate enrollment.
Trivial File Transfer Protocol server	This is required to perform the software update functionality

2 Secure Acceptance of the TOE

Inspection of the physical package to ensure that tampering has not occurred is required upon receipt of the TOE.

Ensure the tamper seals are still in-tact and no physical evidence of tampering is present.

Only allow the management interface to connect to a secure internal management network that is protected from unauthorized access.

Ensure all administrators and users of this device are properly trained and knowledgeable for secure and proper set-up and configuration.

2.1 TOE Rack Unit Specifications

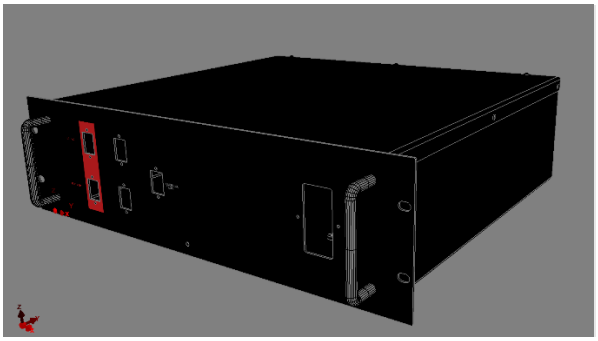

Figure 1 : TOE Rack-Mount Case (Isometric Render)

2.2 External Dimensions

Dimension	Value
Size Class	3U
Height	5.1"
Width	17.8" (face plate is 19")
Depth	25.5"

Table 2 : Dimensions of the TOE

2.3 Other Specifications

Power Source	120V AC (cable included)
Ethernet Ports	5x RJ45
Serial Ports	1x UART per SoC (headers and cables included)

Table 3 : Device Specifications

2.4 Hardware included in Delivery

2.4.1 Included Quantities

Quantity	Item	Notes
1	SpaceX TOE Rack Unit	Contains the VPN PCBA & 2 Whisper RJ45 Adapter boards
1	120V AC Power Supply Cable	Power source for all internal components

Quantity	Item	Notes
2	FTDI UART to USB Adapters	Used for serial console local access point
2	F1AL250VP Fuses	Power supply input fuses

Table 4: Delivered Hardware

*Note: Customer is expected to provide the management workstation for TFTP updates to the device. Details about this can be found under "Software Update Instructions" below.

2.5 Interfaces and Functional Diagrams

Figure 2 : Assembled front panel interfaces

2.5.1 Front Panel Port Mappings

The leftmost ports are the 'trusted' interfaces to the TOE. These ports should only connect to devices on the trusted side of the network. The bottom ethernet port supports up to 100Mbps throughput, and the top supports up to 10G. They present as interfaces eth_prime and eth_whisper by default on the TOE, respectively. The intended configuration of this device in a CSfC solution allows the integrator to connect these ports to red networks or directly to red devices as needed. The ports on the right are all functionally equivalent, but support various throughput speeds from 100M to 10G. Each interface connects to the other SoC on the other half of the board, labeled below as 'Linux Node' in Figure 4. This compute unit is constructed identically to VPN1, but is external to the boundary of the TOE for Regulus. With the provided software images it is named 'ovpn1' and is configured to simply pass data through to the black interfaces from the eth_peer interface to VPN1.

Note: Depending on the testing needs, SpaceX can provide a software image for the configurable Linux compute node that directly bridges its SGMII interface to VPN1 to any of the 3 black ethernet ports (100M Black, 10G Black, or 1G UMB).

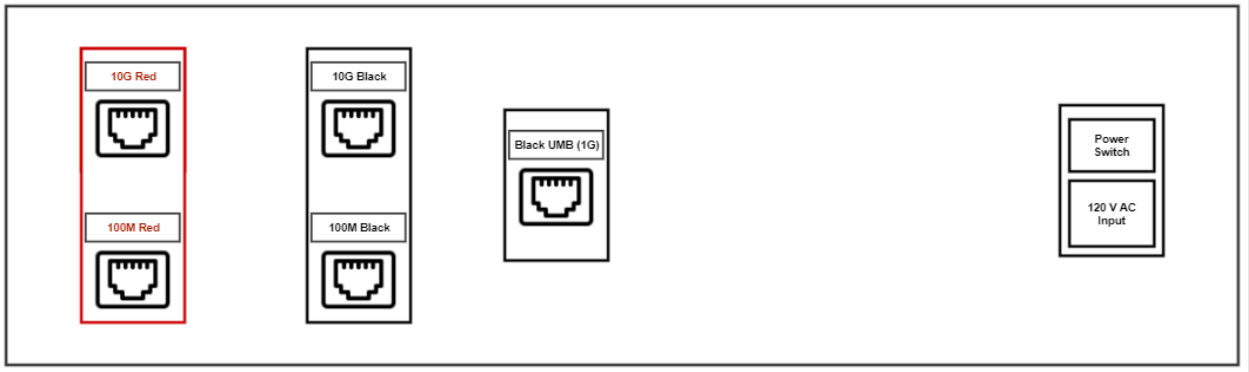

Figure 3 : Front panel port labels

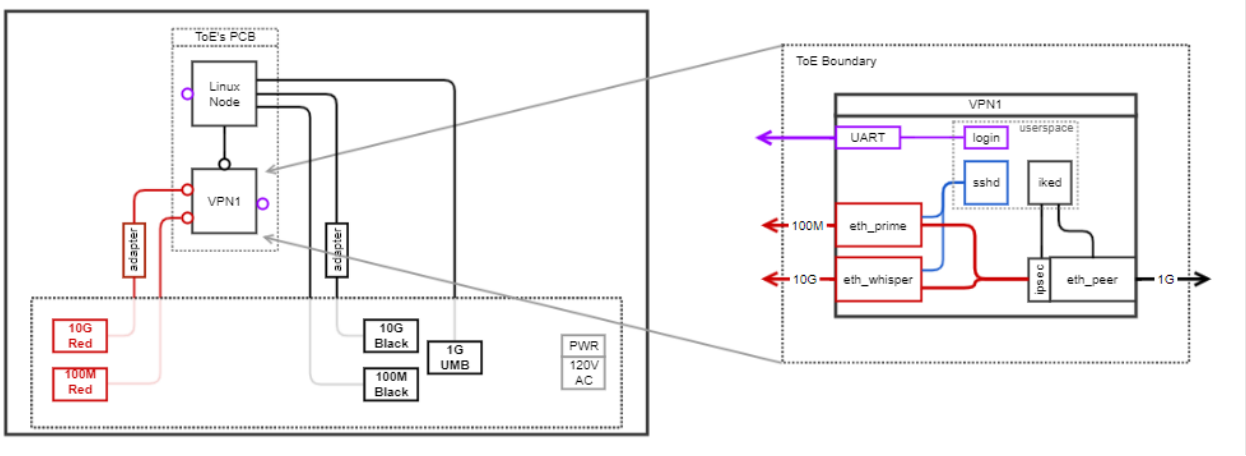

Figure 4 : Functional diagram of front port connectivity to the TOE and supporting hardware

Interface Details			
TOE - VPN1 node name: node1	Ethernet	eth_prime	default IP: 10.0.41.1/32
		eth_whisper	default IP: 10.0.41.1/32
		eth_peer	default IP: 10.0.41.1/32
	Serial	UART	config: 115200-8-N-1 login: root default password: acumenA

Table 5 : Interface Details

2.6 UART Headers

The UART header pins for serial access are labeled as TX, RX, and GND for TLM1 and TLM2. TLM2 on the silkscreen corresponds to the VPN1 TOE. TLM1 Corresponds to the linux compute node. USB to UART FTDI headers will be provided that can connect here. This closeup is called out in the purple box in Figure 4.

2.7 Secure Installation and Configuration

The TOE may be administered using standard linux commands.

To create new users, use the command:

```
useradd <username>
```

To set or change a user account password, use the command:

```
passwd <username>
```

This will prompt the administrator to write in a new password. Users may change their own passwords using this command, and administrative users may change any account password.

The TOE is one part of a compound linux system. The IP addresses for eth_prime, eth_whisper, and eth_peer must always be the same; the TOE presents a choice of multiple physical interfaces suitable the mission requirements, but logically has only one interface.

To configure the IP address of the system, use the following commands:

```
vi /etc/network/interfaces
```

This will open the vi editor, which allows the administrator use standard vi commands to edit a text file on the system. The “interfaces” file contains blocks of configuration data for a number of interfaces. Each interface is identified by “iface <name>”, where <name> is one of eth_prime, eth_whisper, or eth_peer.

```
auto eth0

iface eth_whisper inet static

        address <ip address>

        netmask 255.255.255.0

        gateway <address>

        dns-nameservers 8.8.8.8 8.8.4.4
```

Change the <ip address> of each interface to the desired IP address. Change the gateway address to the desired gateway address. Change the netmask, if necessary, to the required IP address subnet mask.

2.8 Software Update Instructions

In order to reconfigure the VPN device, a TFTP (Trivial File Transfer Protocol) server is required for hosting four artifacts: Image.fit.ecc, boot.bin, and fpga.fit.ecc for the initial platform boot, and runtime.sxv for runtime software. The first three of these files contain the Linux kernel and rootfs, bootloader, and fpga

image, respectively, and all are required for proper device functionality. Once the device reaches the U-Boot stage of the bootloader, it will search for a tftp server on 10.1.1.1 by default. The node will only boot images that it can verify are signed with one of the secure-boot public keys installed in the platform. These key slots can be configured by SpaceX to only trust designated software signing keys. The board provided for evaluation trusts both a development key pair, and a separate 'validation' key pair created for test purposes of non-development signed software. This is true for both VPN1 and the additional external linux compute node on the target PCB.

The node will expect to find a TFTP server with the following directory tree and contents to boot properly.

```
/ROOT_OF_YOUR_TFTP_SERVER/swupdate/ivpn1

   boot.bin

   fpga.fit.ecc

   Image.fit.ecc

   runtime.sxv
```

2.8.1 TFTP Server Set-up example

An example of how to set up this TFTP server on a standard Ubuntu Linux machine is provided below. Organizations typically uses the /opt/[organization]/tftp_server/bin directory as a root directory for tftp updating nodes, but any other root directory should work fine if desired.

2.8.1.1 Install tftp-hpa service and set up directory structure

```
# First install the tftp server (tftpd-hpa.service)

sudo apt install tftpd-hpa


# Create required directory structure at /opt/organization/tftp_server/bin

mkdir -p /opt/organization/tftp_server/bin/swupdate/ivpn1

mkdir -p /opt/organization/tftp_server/bin/swupdate/ovpn1

sudo chown -R tftp:tftp /opt/organization/tftp_server/bin
```

2.8.1.2 Set up default configuration

Add the contents of the below code block to the tftp server. Place the following file contents in your tftpd-hpa config file, (default **/etc/default/tftpd-hpa**)

```
TFTP_USERNAME="tftp"

TFTP_DIRECTORY="/opt/organization/tftp_server/bin"

TFTP_ADDRESS=":69"

TFTP_OPTIONS="--secure"
```

2.8.1.3 Start/Restart the tftp-hpa service

```
# Restart the server and check status

sudo /etc/init.d/tftpd-hpa restart
```

```
sudo systemctl status tftpd-hpa.service
```

Something similar to the following output is expected:

```
$ sudo /etc/init.d/tftpd-hpa restart
[ ok ] Restarting tftpd-hpa (via systemctl): tftpd-hpa.service.



$ sudo systemctl status tftpd-hpa.service
● tftpd-hpa.service - LSB: HPA's tftp server
   Loaded: loaded (/etc/init.d/tftpd-hpa; bad; vendor preset: enabled)
   Active: active (running) since Mon 2021-12-27 13:41:16 PST; 1min 15s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 2832 ExecStop=/etc/init.d/tftpd-hpa stop (code=exited, status=0/SUCCESS)
  Process: 2844 ExecStart=/etc/init.d/tftpd-hpa start (code=exited, status=0/SUCCESS)
    Tasks: 1
   Memory: 2.0M
      CPU: 12ms
   CGroup: /system.slice/tftpd-hpa.service
           └─2870 /usr/sbin/in.tftpd --listen --user tftp --address :69 --
secure /opt/spacex/tftp_server/bin


Dec 27 13:41:16 localhost tftpd-hpa[2844]:  * Starting HPA's tftpd in.tftpd
Dec 27 13:41:16 localhost tftpd-hpa[2844]:    ...done.
```

2.8.1.4 Configure device-facing IP address

Add the expected IP address. Replace <ETHERNET_INTERFACE> below with the name of the ethernet interface on your machine that is connected to either red port of the TOE front panel.

```
# The interface connected to either red port should have ip addr show <IF> ==>
inet 10.1.1.1/12 scope global <ETHERNET_INTERFACE> ... If not, then run

sudo ip addr add 10.1.1.1/12 dev <ETHERNET_INTERFACE>

sudo ip addr add 10.0.1.1 dev <ETHERNET_INTERFACE>

sudo ip addr add 10.0.2.1 dev <ETHERNET_INTERFACE>

sudo ip addr add 10.0.3.1 dev <ETHERNET_INTERFACE>
```

2.8.1.5 Version Verification

To query the currently running version, run the command “version info” from the command line. The version information will be displayed. The highlighted line indicates the “version” of the TOE software, while the “platform version” line above it indicates the specific buildID of firmware.

```
[root@ivpn1 ~]# version_info

Assembly Serial Number, 00000

Board Serial Number, error_no_nxid_device_found

MAC Addresses,  02:00:00:00:00:00 26:12:ac:1a:29:01 16:8d:f4:3e:8a:a3

Platform Signature, 0xd34534fed9c938de

OS Part Number,

Platform Version, e0dc4b895134eee3c5d65f64da41b84e932c270d

Platform Branch, niap_evaluation_243

Platform Bundle Hash, 74c1c6b0385752cd139023ea7aac3bb7de282131

<snip>

...

[root@ivpn1 ~]#
```

3 Audit Data

This section details the required Auditable Events per the Common Criteria requirements.

3.1 Audit Data Generation - Examples

Many events of the TOE will be automatically audited. Here are examples of what those audit logs might look like and the type of event that triggers an audited event.

3.1.1 Failure to Establish an IPSec SA

The TOE logs messages from attempted session establishments to /var/log/messages. When an IPsec session establishment fails, the reason will be provided via an iked message log, like the following:

```
2022-12-22T01:55:02.347Z node1 <7> iked[25847]: ikev2_pld_notify: protoid
IKE spisize 0 type AUTHENTICATION_FAILED
```

3.1.2 Session establishment with peer

If an IPsec session is established successfully, openiked will indicate success with a message like the following:

```
2022-12-22T01:52:52.799Z node1 <7> iked[23989]: spi=0x2ef2cdcb7f36d61d: sa_state:
VALID -> ESTABLISHED from 10.1.1.1:500 to 10.0.41.1:500 policy 'csfc'
```

During this session establishment, the TOE logs all details of the exchange with the peer including the contents of received packets. For example, here is the part of the audit log that shows the TOE decrypting the packets containing the negotiated SA parameters:

```
2022-12-22T01:52:52.755Z node1 <7> iked[23989]: ikev2_pld_payloads: decrypted payload
SA nextpayload TSi critical 0x00 length 44
```

```
2022-12-22T01:52:52.755Z node1 <7> iked[23989]: ikev2_pld_sa: more 0 reserved 0 length
40 proposal #1 protoid ESP spisize 4 xforms 3 spi 0xce5b142e

2022-12-22T01:52:52.755Z node1 <7> iked[23989]: ikev2_pld_xform: more 3 reserved 0
length 12 type ENCR id AES_CBC

2022-12-22T01:52:52.755Z node1 <7> iked[23989]: ikev2_pld_attr: attribute type
KEY_LENGTH length 256 total 4

2022-12-22T01:52:52.755Z node1 <7> iked[23989]: ikev2_pld_xform: more 3 reserved 0
length 8 type INTEGR id HMAC_SHA2_384_192

2022-12-22T01:52:52.756Z node1 <7> iked[23989]: ikev2_pld_xform: more 0 reserved 0
length 8 type ESN id NONE
```

3.1.3 Failure to establish an SSH Session

The TOE provides an SSH server for remote administration. When an ssh session fails to establish, the reason for the failure is recorded in the audit logs. Here is an example of a public key algorithm failure:

```
2022-12-22T02:12:44.307Z node1 <6> sshd[6658]: Failed publickey for root
from 10.0.99.1 port 43540 ssh2: ECDSA
SHA256:HPwz1URZOSloVqc9c3mjehwDJqVDO6+tcQq0eQDKIRM
```

3.1.4 Unsuccessful login attempts limit is met or exceeded

The TOE will lock out an administrator who exceeds the configured number of unsuccessful logout attempts, and the origin of this attempt will be logged to the audit records. Here is what it looks like when a remote admin exceeds the lockout limit:

```
2022-12-10T01:54:31.402Z node1 <6> sshd[3044]: pam_faillock(sshd:auth):
Consecutive login failures for user root account temporarily locked

2022-12-10T01:54:31.403Z node1 <5> auditd: type=USER_AUTH
msg=audit(1670637271.402:2674):
pid=3044 uid=0 auid=4294967295 ses=4294967295 subj==unconfined
msg='op=PAM:authentication grantors=? acct="root" exe="/usr/bin/sshd"
hostname=10.1.1.1 addr=172.1610.1erminal=ssh res=failed'
```

3.1.5 Use of identification and authentication mechanism

The TOE will log any attempt to authenticate as a user of the system in the audit records. Here is an example of a successful authentication:

```
2022-12-10T01:54:33.113Z node1 <5> auditd: type=CRED_REFR
msg=audit(1670637273.112:2676):
pid=3045 uid=0 auid=4294967295 ses=4294967295 subj==unconfined
msg='op=PAM:setcred grantors=pam_faillock,pam_faillock acct="root"
exe="/usr/bin/sshd" hostname=10.1.1.1 addr=172.1610.1erminal=ssh
res=success'
```

3.1.6 Use of certificates in the TOE trust store

When attempting to establish an IPsec session, the TOE must load CA certificates from disk into the trust store, so they can be used during the IKE negotiation. The details of this usage will be logged to the audit records.

3.1.6.1 Unsuccessful attempt to validate a certificate

Unsuccessful attempts to validate a certificate are logged to the audit records by openiked and include a label of the certificate and an explanation of the failure:

```
2023-02-18T00:52:37.225Z node1 <7> iked[2956]: ca_validate_cert:
/C=US/O=Organization/CN=node.org.corp certificate revoked
```

3.1.6.2 Any addition, replacement or removal of trust anchors in the TOE's trust store

The TOE loads trust anchors when openiked starts at boot time, from a location on disk. It logs to the audit records anything found on disk that it is attempting to load:

```
02-18T00:52:37.123Z node1 <7> iked[2956]: ca_getreq: found CA /C=US/O=Org/CN=node-
revoked

2023-02-18T00:52:37.124Z node1 <7> iked[2956]: ca_getreq: found CA
/C=US/O=Org/CN=RootCA

2023-02-18T00:52:37.124Z node1 <7> iked[2956]: ca_getreq: found local certificate
/C=US/O=Org/CN=node-test.org.corp
```

3.1.7 Any attempt to initiate a manual update

Initiating a manual update on the TOE requires a power cycle to the TOE, as reboots from within the TOE boundary are disabled. Because of this, the TOE is incapable of auditing such an attempt itself, thus the audit server must be configured to store record of an administrator action to cycle power on the TOE whether remotely or locally, to provide a message like the following:

```
2023-02-18 00:52:38,203 - Device power was disconnected
```

3.1.8 Application of rules configured with the ‘log’ operation

If the TOE has been configured with traffic handling rules that include the `log` operation, the logs generated by the application of these rules will be stored in the audit record. iptables will log information about the traffic that matched the rule as well as the configured log message associated with that rule. Here is an example of such a message:

```
2022-12-28T18:42:46.941Z node1 <5> kernel: [41132.000000]
IPTABLES_ACCEPTED_PACKETIN=eth_prime OUT=
MAC=26:12:ac:1a:29:01:42:db:13:0e:f0:0b:08:00 SRC=4.0.0.4 DST=2.0.0.1 LEN=
28 TOS=0x00 PREC=0x00 TTL=64 ID=28900 PROTO=ICMP
TYPE=8 CODE=0 ID=35995 SEQ=1
```

```
2022-12-28T18:42:46.941Z node1 <5> kernel: [41132.000000]
IPTABLES_ACCEPTED_PACKETIN= OUT=eth_prime
SRC=2.0.0.1 DST=4.0.0.4 LEN=28 TOS=0x00 PREC=0x00 TTL=64 ID=42875 PROTO=ICM
P TYPE=0 CODE=0 ID=35995 SEQ=1
```

Here, the rule was configured to log with the message IPTABLES_ACCEPTED_PACKET

3.1.9 Discontinuous changes to time

Administrators of the TOE are able to manually set the time. Changes to the time are logged to the audit records encoded in hex as a security measure. The time-change command begins with date -s, which is 64617465202D73 when encoded in hex.

```
2022-12-22T22:03:10.090Z node1 <5> auditd: type=EXECVE
msg=audit(1671746590.087:19131): argc=3 a0="sh" a1="-
c" a2=64617465202D732040313637313134313373930
```

The old time is visible as time of execution and the command, when unencoded, reveals the new time:

```
date -s @1671141790
```

3.2 Protected Audit Event Storage

In order to configure remote storage of audit records, an IPsec session must be established between the TOE and the remote audit server.

Once the session has been established, Add the audit server as a remote syslog destination to the TOE's syslog config (/etc/syslog.conf) and reload syslogd

```
$ echo '*.*  @{AUDIT_SERVER_IP}:{TEST_SYSLOG_PORT}' >> {SYSLOG_CONFIG};

$ killall -SIGHUP syslogd
```

Assuming the audit server is running and properly configured on the audit server, and the ipsec tunnel is established, audit logs will flow over the trusted ipsec channel to the audit server.

4 Cryptographic Key Generation

Cryptographic keys for Regulus are generated inside the hardware security module. No other cryptographic engines were tested during the evaluation, and the use of any other cryptographic module is out of scope for the evaluation. The functionality for generating a new cryptographic key can be accessed via a character device in the driver path. While logged into the TOE as a security administrator, the following steps can be performed to generate a new cryptographic key:

```
Go to the driver directory for the first SE050 device on the system

$ SE050_DIR=$(find /sys/bus/i2c/drivers/se050 -type l -print | sort | head -n 1)

$ cd ${SE050_DIR}
```

Here, we look at the high-level directory structure and files (where applicable).

ec_curves/ includes create and delete files for elliptic curve definitions along with id and created attributes for every known SE050 elliptic curve.

```
$ find ec_curves -mindepth 1 | sort
ec_curves/Brainpool160
ec_curves/Brainpool160/created
ec_curves/Brainpool160/id
ec_curves/Brainpool192
ec_curves/Brainpool192/created
ec_curves/Brainpool192/id
ec_curves/Brainpool224
ec_curves/Brainpool224/created
ec_curves/Brainpool224/id
ec_curves/Brainpool256
ec_curves/Brainpool256/created
ec_curves/Brainpool256/id
ec_curves/Brainpool320
ec_curves/Brainpool320/created
ec_curves/Brainpool320/id
ec_curves/Brainpool384
ec_curves/Brainpool384/created
ec_curves/Brainpool384/id
ec_curves/Brainpool512
ec_curves/Brainpool512/created
ec_curves/Brainpool512/id
ec_curves/NIST_P192
ec_curves/NIST_P192/created
ec_curves/NIST_P192/id
ec_curves/NIST_P224
ec_curves/NIST_P224/created
ec_curves/NIST_P224/id
ec_curves/NIST_P256
ec_curves/NIST_P256/created
```

```
ec_curves/NIST_P256/id
ec_curves/NIST_P384
ec_curves/NIST_P384/created
ec_curves/NIST_P384/id
ec_curves/NIST_P521
ec_curves/NIST_P521/created
ec_curves/NIST_P521/id
ec_curves/Secp160k1
ec_curves/Secp160k1/created
ec_curves/Secp160k1/id
ec_curves/Secp192k1
ec_curves/Secp192k1/created
ec_curves/Secp192k1/id
ec_curves/Secp224k1
ec_curves/Secp224k1/created
ec_curves/Secp224k1/id
ec_curves/Secp256k1
ec_curves/Secp256k1/created
ec_curves/Secp256k1/id
ec_curves/TPM_ECC_BN_P256
ec_curves/TPM_ECC_BN_P256/created
ec_curves/TPM_ECC_BN_P256/id
ec_curves/create
ec_curves/delete
```

secure_objects/ includes subdirectories for each known SE050 Secure Object type

```
$ /bin/ls -1 secure_objects
aes_keys
binary_files
counters
curves
des_keys
ec_key_pairs
ec_priv_keys
```

```
ec_pub_keys
hmac_keys
pcrs
rsa_key_pair_crts
rsa_key_pairs
rsa_priv_key_crts
rsa_priv_keys
rsa_pub_keys
userids
```

Before creating a cryptographic key, you must create the cryptographic curve definition for that key, if it's not already created.

4.1 Check for an existing definition

Here, we check for an existing curve definition for NIST_P256:

```
$ if [ $(cat ec_curves/NIST_P256/created) == 1 ]; then echo CREATED; fi
CREATED
$ cat ec_curves/NIST_P256/id
0x03
```

Here we find that the curve exists, and has id 0x03

If the curve did not exist, we could create it, like this:

```
$ echo 0x03 > ec_curves/create
$ cat ec_curves/NIST_P256/created
1
```

The id for each curve can be found in the ec_curves directory, but the two curves supported by the TOE p-256 and p-384 have ids 0x03 and 0x04

```
$ cat /sys/devices/platform/amba/ff020000.i2c/i2c-0/0-
0048/ec_curves/NIST_P384/id
0x04
```

4.2 Creating an EC Key Pair

EC key pairs can be created by echoing a 4 byte id to the intended elliptic curve directory inside ec_key_pairs/create , which lives within secure_objects

```
$ echo -n TEST > secure_objects/ec_key_pairs/create/NIST_P384
```

Here TEST is the id of the new key pair. And once a key pair is created, there will be a new directory in ec_key_pairs which would allow signing messages using the private key. Applications such as openiked will use these directories to sign messages during normal TOE operation.

```
$ /bin/ls secure_objects/ec_key_pairs/TEST

bin   sign
```

4.3 Reading the public key

The private key is stored only on the secure element and is not accessible, but is made available for signing. However, the public key can be read from the bin device in the key pair directory:

```
$ hexdump -Cv secure_objects/ec_key_pairs/TEST/bin
```

The file always starts with 0x04 , meaning it is an uncompressed public key pair (X,Y).

4.4 Cryptographic Key Destruction

Cryptographic keys stored in volatile memory are overwritten with zeroes as soon as they are no longer needed. During normal operation of the TOE, there should not be a situation where the keys are not successfully destroyed. Any keys present in volatile memory are ephemeral, and would have no use if the TOE were to experience a power failure that would prevent the proper destruction of the keys. New ephemeral keys would be created during proper runtime of the TOE after a successful restart.

Keys stored in non-volatile storage live in the secure element, and are inaccessible except to be used for signing operations. However, keys can be deleted in the same manner as they can be created.

4.4.1 Deleting an EC Key Pair

Perform the following command to delete an EC key pair from the TOE:

```
$ echo -n TEST > secure_objects/ec_key_pairs/delete
```

Here TEST is the 4 byte identifier of the key pair you wish to delete. If the key pair does not have an id made of printable bytes, it can be destroyed using the raw byte values, like this:

```
$ echo -n -e '\x01\x02\x03\x04' > secure_objects/ec_key_pairs/delete

$ /bin/ls -R secure_objects/ec_key_pairs/0x01020304/

ls: secure_objects/ec_key_pairs/0x01020304/: No such file or directory
```

4.4.2 Encrypting/decrypting packets

Packets will be encrypted/decrypted if:

The source and destination of the packet are within the local_subnet and remote_subnet specified in iked.conf.

4.4.3 Dropping packets

Packets are dropped by default. You can additionally add rules for dropping packets by following the steps in section 5 “Packet Filtering Rules” and using either DROP or LOG_DROP as the target.

5 Configuring packet filtering rules

Supported Protocols

The following protocols are supported for packet filtering:

- Internet Protocol version 4 (IPv4)
  - All protocols under this family are supported
- Internet Protocol version 6 (IPv6)
  - All protocols under this family are supported

5.1 Supported Attributes for Packet Filtering

- IPv4
  - Source address
  - Destination address
  - Protocol number
- IPv6
  - Source address
  - Destination address
  - Protocol number
- TCP
  - Source port
  - Destination port
- UDP
  - Source port
  - Destination port

5.2 Supported Actions for Packet Filtering

- Permit (ACCEPT)
- Permit and log (LOG_ACCEPT)
- Discard (DROP)
- Discard and log (LOG_DROP)
- Log (LOG)
  - Note: this is a non-terminating rule. If a packet matches this rule, the chain of rules will continue to be traversed.

5.3 Configuration Guidance

Packet filtering rules are configured using iptables commands. In general, the syntax is:

```
# For IPv4 rules.iptables --insert <chain> <rulenum> <rule-specifications>

# For IPv6 rules.ip6tables --insert <chain> <rulenum> <rule-specifications> --jump
<action>
```

- Chain can be one of the following:
  - FORWARD, for traffic not destined for the TOE.
  - INPUT, for traffic destined for the TOE.
  - OUTPUT, for traffic sourced from the TOE.
- Rulenum can be any positive integer N. this rule will then be the Nth rule to be checked.
- Rule specifications can be any of the following flags:
  - --source <src>, for matching the packet's source address.
  - --destination <dst>, for matching the packet's destination address.
  - --in-interface <iface>, for matching the interface the packet arrived on.
  - --out-interface <iface>, for matching the interface the packet is destined for.
  - --protocol <proto>, for matching the packet's protocol number.
  - --source-port <port>, for matching a packet's source port
    - Note: this rule specification is only available when --protocol is 6 (TCP) or 17 (UDP)
  - --destination-port <port>, for matching a packet's destination port
    - Note: this rule specification is only available when --protocol is 6 (TCP) or 17 (UDP)
- Action can be any of the supported actions for packet filtering:
  - ACCEPT, LOG_ACCEPT, DROP, LOG_DROP, LOG

5.4 Example Configuration

An example rule could look like:

```
iptables --insert FORWARD 1 --source 1.2.3.4 --destination 5.6.7.8 --protocol 6 --
destination-port 22 --in-interface eth0 --jump ACCEPT
```

This rule ACCEPTs packets that:

- Have a source address of 1.2.3.4
- Have a destination address of 5.6.7.8
- Is a TCP packet (protocol number 6)
- Has a destination port of 22
- Entered on the eth0 interface

5.5 Inspecting and Ordering the Packet filtering Rules

Run the following command to inspect the packet filtering rules:

```
# For IPv4 rules.iptables -vnL

# For IPv6 rules.ip6tables -vnL
```

This will show the rules in order. They are checked from top-to-bottom; the first rule that matches is used and the action associated with that rule applies.

- To add rules to a specific position in the list, use the appropriate Rulenum (explained in "Configuration Guidance")

5.6 Example Ordering of the Packet Filtering Rules

```
   [user@node1]# iptables –vnL

Chain INPUT (policy ACCEPT 5 packets, 644 bytes)

pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

pkts bytes target     prot opt in     out     source               destination    0
0 LOG_ACCEPT  all  --  *      *       1.0.0.1             3.0.0.1      0     0
LOG_DROP   all  --  *      *       1.0.0.1             3.0.0.1        0     0
LOG_DROP   all  --  *      *       0.0.0.0/0           0.0.0.0/0

Chain OUTPUT (policy ACCEPT 7 packets, 908 bytes)

pkts bytes target     prot opt in     out     source               destination

Chain LOG_ACCEPT (1 references)

pkts bytes target     prot opt in     out     source               destination    0
0 LOG        all  --  *      *       0.0.0.0/0           0.0.0.0/0            LOG
flags 0 level 5 prefix "IPTABLES_ACCEPTED_PACKET:"

   0     0 ACCEPT     all  --  *      *       0.0.0.0/0           0.0.0.0/0

Chain LOG_DROP (2 references)

pkts bytes target     prot opt in     out     source               destination    0
0 LOG        all  --  *      *       0.0.0.0/0           0.0.0.0/0            LOG
flags 0 level 5 prefix "IPTABLES_DROPPED_PACKET:"

   0     0 DROP       all  --  *      *       0.0.0.0/0           0.0.0.0/0
```

5.7 Behavior If No Rule Matches

If there are no matching user-added rules in a chain, the default is to LOG_DROP the packet. Each chain (FORWARD, INPUT, OUTPUT) comes equipped with a rule that matches all packets is associated with the LOG_DROP action. This rule must always be the last rule in the FORWARD, INPUT, and OUTPUT chains in order to be compliant with the protection profile.

5.8 Supported Configuration Information

Packet filtering rules are configured using iptables commands. In general, the syntax is:

```
# For IPv4 rules.

iptables --insert <chain> <rulenum> <rule-specifications>

# For IPv6 rules.

ip6tables --insert <chain> <rulenum> <rule-specifications> --jump <action>
```

- Chain can be one of the following:
  - FORWARD, for traffic not destined for the TOE.

    - INPUT, for traffic destined for the TOE.
    - OUTPUT, for traffic sourced from the TOE.
- Rulenum can be any positive integer N. this rule will then be the Nth rule to be checked.
- Rule specifications can be any of the following flags:
    - --source <src>, for matching the packet's source address.
    - --destination <dst>, for matching the packet's destination address.
    - --in-interface <iface>, for matching the interface the packet arrived on.
    - --out-interface <iface>, for matching the interface the packet is destined for.
    - --protocol <proto>, for matching the packet's protocol number.
    - --source-port <port>, for matching a packet's source port
        - Note: this rule specification is only available when --protocol is 6 (TCP) or 17 (UDP)
    - --destination-port <port>, for matching a packet's destination port
        - Note: this rule specification is only available when --protocol is 6 (TCP) or 17 (UDP)
- Action can be any of the supported actions for packet filtering:
    - ACCEPT, LOG_ACCEPT, DROP, LOG_DROP, LOG
- Example Configuration

An example rule could look like:

```
iptables --insert FORWARD 1 --source 1.2.3.4 --destination 5.6.7.8 --protocol 6 --
destination-port 22 --in-interface eth0 –jump ACCEPT
```

This rule ACCEPTs packets that:

- Have a source address of 1.2.3.4
- Have a destination address of 5.6.7.8
- Is a TCP packet (protocol number 6)
- Has a destination port of 22
- Entered on the eth0 interface
- Inspecting and Ordering the Packet filtering Rules

Run the following command to inspect the packet filtering rules:

```
# For IPv4 rules.

iptables -vnL

# For IPv6 rules.

ip6tables -vnL
```

This will show the rules in order. They are checked from top-to-bottom; the first rule that matches is used and the action associated with that rule applies.

- To add rules to a specific position in the list, use the appropriate Rulenum T

Example Ordering of the Packet Filtering Rules

```
[root@node1]# iptables -vnL

Chain INPUT (policy ACCEPT 5 packets, 644 bytes)

pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

 pkts bytes target     prot opt
in     out     source               destination

    0     0 LOG_ACCEPT  all  -
-  *      *       1.0.0.1              3.0.0.1

    0     0 LOG_DROP   all  -
-  *      *       1.0.0.1              3.0.0.1

    0     0 LOG_DROP   all  -
-  *      *       0.0.0.0/0            0.0.0.0/0


Chain OUTPUT (policy ACCEPT 7 packets, 908 bytes)

 pkts bytes target     prot opt
in     out     source               destination


Chain LOG_ACCEPT (1 references)

 pkts bytes target     prot opt
in     out     source               destination

    0     0 LOG        all  -
-  *      *       0.0.0.0/0            0.0.0.0/0           LOG
flags 0 level 5 prefix "IPTABLES_ACCEPTED_PACKET:"

    0     0 ACCEPT     all  -
-  *      *       0.0.0.0/0            0.0.0.0/0


Chain LOG_DROP (2 references)

 pkts bytes target     prot opt
in     out     source               destination

    0     0 LOG        all  -
-  *      *       0.0.0.0/0            0.0.0.0/0           LOG
flags 0 level 5 prefix "IPTABLES_DROPPED_PACKET:"

    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

- FPF_RUL_EXT.1.6
- Behavior If No Rule Matches

If there are no matching user-added rules in a chain, the default is to LOG_DROP the packet. Each chain (FORWARD, INPUT, OUTPUT) comes equipped with a rule that matches all packets is associated with the LOG_DROP action. This rule must always be the last rule in the FORWARD, INPUT, and OUTPUT chains in order to be compliant with the protection profile.

5.8.1 Bypassing packets

You can additionally add rules for bypassing packets by following the guide in <VPN FILTER guide for FPF_RUL_EXT.1.4> and using either ACCEPT or LOG_ACCEPT as the target.

Note: the source and destination of the packets must not be within the local_subnet and remote_subnet specified in iked.conf (otherwise, they will be encrypted)

6 Configuring IPsec Parameters

IPsec parameters are configured in an iked.conf file. The general form looks like this:

```
set nomobike

ikev2 "csfc" {mode} tunnel esp \

        from {local_subnet} to {remote_subnet} \

        local {local_ip} peer {remote_ip} \

        ikesa \

                enc aes-256 \

                group {ecp384|ecp256} \

                auth hmac-sha2-384 \

                prf hmac-sha2-384 \

        childsa \

                enc aes-256 \

                group {ecp384|ecp256} \

                auth hmac-sha2-384 \

                prf hmac-sha2-384 \

                noesn \

        srcid {local_fqdn} \

        dstid {remote_fqdn} \

        ikelifetime {phase_1_lifetime} lifetime {phase_2_lifetime} [bytes
{bytes_lifetime}]\

        ecdsa384
```

6.1 IKE version

The TOE only supports IKEv2.

This is specified on line 2 of the iked.conf output in Section 5 by the keyword "ikev2".

6.2 IKE mode

The TOE only supports tunnel mode.

This is specified on line 2 of the iked.conf output in Section 5 by the keyword "tunnel".

6.3 IKE algorithms

The TOE only supports the AES-CBC-256 cryptographic algorithm together with HMAC-SHA-384.

This is specified for the Phase 1 SA on lines 6, 8 and 9 of the iked.conf output in Section 5.

This is specified for the Phase 2 SA on lines 11, 13, and 14 of the iked.conf output in Section 5.

The TOE supports the use of either Group 19 (ecp256) or Group 20 (ecp384) as the DH Group.

This is specified for the Phase 1 SA on line 7 of the iked.conf output in Section 5 by specifying either "ecp256" or "ecp384".

This is specified for the Phase 2 SA on line 12 of the iked.conf output in Section 5 by specifying either "ecp256" or "ecp384".

6.4 IKE lifetimes

The Phase 1 SA lifetime can be set to any value within 5 minutes up to 24 hours (specified in seconds).

This value should be specified where "{phase_1_lifetime}" is on line 18 of the iked.conf output in Section 5.

Example: A lifetime of 24 hours would be specified by 86400 (86400 seconds = 24 hours)

The Phase 2 SA lifetime can be set to any value within 5 minutes up to 8 hours (specified in seconds).

This value should be specified where "{phase_2_lifetime}" is on line 18 of the iked.conf output in Section 5.

Example: A lifetime of 24 hours would be specified by 28800 (28800 seconds = 24 hours).

The Phase 2 SA lifetime can also be set by number of bytes.

This value should be specified where "{bytes_lifetime}" is on line of the iked.conf output in Section 5.

7 Cryptographic Operation (Random Bit Generation)

No configuration of the RNG is functionally possible. It is in a NIAP compliant state as delivered.

8 Authentication Failure Management

The TOE's security administrator has the ability to configure the details of lockouts resulting from successive unsuccessful login attempts, including the amount of time the account will be locked, as well as the number of successive unsuccessful attempts required for the account to be locked out.

8.1 Configuring lockout time

To configure the lockout time, a security administrator must log in as the administrator account 'root', and edit the unlock_time in the faillock.conf file.

The file is located at /etc/security/faillock.conf

The default lockout time is 900 seconds. An administrator can use any of the tools available on the TOE to edit the faillock file. Here's an example of how you might change the lockout time from 600 to 1200 seconds.

```
Before:


unlock_time = 600


After:


unlock_time = 1200
```

If the unlock time is set to 0 or never , the lockout will never time out, so security administrators must not set the value to 0 or never in order to always have remote access to the TOE.

If the value is set to never allow time-based account unlocks, security administrators will still have access to the TOE over the local interface.

8.2 Configuring the number of successive unsuccessful authentication attempts

To configure number of successive unsuccessful authentication attempts, a security administrator must log in as the administrator account 'root', and edit the deny in the faillock.conf file. The file is located at /etc/security/faillock.conf

The default number of attempts is 3. An administrator can use any of the tools available on the TOE to edit the faillock file. Here's an example of how you might change the number of unsuccessful attempts from 3 to 5:

```
Before:


deny = 3
```

```
After:

deny = 5
```

8.3 Password Management

8.3.1 Configuring the minimum password length

The security administrator has the ability to configure the minimum length required for new passwords, between 15 and 128 characters. To edit the minimum password length, administrators must log in as the administrator account 'root', and edit /etc/passwd_minlen

For example, here is how an administrator might set the minimum password length to 32 characters:

```
$ echo 32 > /etc/passwd_minlen
```

8.3.2 Setting administrator password

The security administrator has the ability to modify the administrative account password on the TOE. There are requirements that a password must meet in order to be set successfully.

Passwords shall be able to be composed of any combination of upper and lower case letters, numbers, and the following special characters: [“!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)”];

The password must also be at least as long as the minimum length required by /etc/passwd_minlen .

To change the administrator password, an administrator must first log in with the current administrator password. Then the administrator can issue the following command:

```
$ passwd root
```

The admin will then be prompted for the new password twice. If the passwords match and meet the required complexity, composition, and minimum length requirements, the password will be changed.

9 User Identification and Authentication

9.1 Remote Access

9.2 Establishing Remote Administration Sessions

SSH is the only supported mechanism for remotely administering the device over a network. For SSH access, public key authentication as well as password authentication are supported. The following provides an example command that can be executed on a Linux system connected to the same network as the VPN device.

123.45.67.89 is the IP address of the device in this example, and the administrative user is root.

For public key authentication, an optional -i flag can be used to specify the path to an ecdsa private key that has a corresponding ssh public key loaded into /root/.ssh/authorized_keys.

Otherwise, if no private key argument is specified, a password will be prompted.

```
ssh root@10.0.1.15. -i path/to/private_key.pem
```

SSH public keys of the following format can be loaded by an administrator on the device. The following shows the contents of the authorized key file for this example.

```
[root@node1]# cat /root/.ssh/authorized_keys


ecdsa-sha2-nistp384
AAAAE2VjZHNhLXNoYTItbmlzdHAzODQAAAAIbmlzdHAzODQAAABhBHvtsEVs7pfjFedxBubxuS6aU41ypi4Ghn
82tedUqbXbZvoQ3u8uFiIwsW9q4sHdsNTk0JBh3sesl0JDtp/lDCNz2OMFoyzOzI6RxiiRwAseaY9sy56S/S2w
/DEnT1LRNA== evaluation-key
```

Once logged in, the following shell prompt will be displayed, along with the banner message configured in /etc/issue.

```
This is a configurable banner message!

[root@node1 ~]#

# The following file contents can be changed to display a different banner message

[root@node1 ~]# cat /etc/issue

This is a configurable banner message!
```

By default, modifications of the **authorized_keys** file are restricted to root administrative users. No configuration is necessary to restrict access.

9.3 Establishing Local Administrative Sessions

To establish a local administrative session, the UART serial port can be used. If this port is connected to a linux machine, any terminal emulator can be used. The baud rate configuration settings are `115200-8-N-1`

```
$ minicom -D /dev/ttyUSB0 -b 115200

Sample Banner Message.

node1 login:
```

Authentication will be performed via the password prompt. Changing passwords is restricted to the root administrative user, and no additional configuration is required to restrict access to the ability to modify passwords.

9.4 Accessing the TOE locally.

Local access to the TOE is available over UART serial headers. The UART header pins for serial access are labeled as TX, RX, and GND for TLM1 and TLM2. TLM2 on the silkscreen corresponds to the VPN1 TOE. TLM1 Corresponds to the linux compute node. USB to UART FTDI headers will be provided that can connect here. Once connected, the interface can be accessed using a serial console with a baud rate of 115200. For example:

```
$ minicom -D /dev/tty3 -b 115200
```

9.5 Logging into the TOE

To log in as any user, simply enter the username at the login: prompt and the password at the password: prompt. To log in as the security administrator, use the root login. The default root password will be provided along with the TOE.

9.6 Protected Authentication Feedback

By default, all passwords are obscured and no user configuration is required in order to be NIAP compliant.

10 Manual Update Mode

10.1 The following sections detail the process for software updates on the TOE. Software Update Instructions

See Section 2.7 for detailed instructions on TFTP Server set up and Software Update details and steps.

10.1.1 Step 1: Install tftp-hpa service and set up directory structure

```
# First install the tftp server (tftpd-hpa.service)
sudo apt install tftpd-hpa


# Create required directory structure at /opt/Organization/tftp_server/bin
mkdir -p /opt/OrganizationName/tftp_server/bin/swupdate/node1
sudo chown -R tftp:tftp /opt/OrganizationName/tftp_server/bin
```

10.1.2 Step 2: Set up default configuration

Add the contents of the below code block to the tftp server. Place the following file contents in your tftpd-hpa config file, (default `/etc/default/tftpd-hpa`)

```
/etc/default/tftpd-hpa
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/opt/OrganizationName/tftp_server/bin"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

10.1.3 Step 3: Start/Restart the tftp-hpa service

```
# Restart the server and check status

sudo /etc/init.d/tftpd-hpa restart


sudo systemctl status tftpd-hpa.service
```

Something similar to the following output is expected:

```
$ sudo /etc/init.d/tftpd-hpa restart

[ ok ] Restarting tftpd-hpa (via systemctl): tftpd-hpa.service.


 $ sudo systemctl status tftpd-hpa.service

● tftpd-hpa.service - LSB: HPA's tftp server

   Loaded: loaded (/etc/init.d/tftpd-hpa; bad; vendor preset: enabled)

   Active: active (running) since Mon 2021-12-27 13:41:16 PST; 1min 15s ago

     Docs: man:systemd-sysv-generator(8)

  Process: 2832 ExecStop=/etc/init.d/tftpd-hpa stop (code=exited, status=0/SUCCESS)

  Process: 2844 ExecStart=/etc/init.d/tftpd-hpa start (code=exited, status=0/SUCCESS)

    Tasks: 1

   Memory: 2.0M

      CPU: 12ms

   CGroup: /system.slice/tftpd-hpa.service

           └─2870 /usr/sbin/in.tftpd --listen --user tftp --address :69 --
secure /opt/OrganizationName/tftp_server/bin


Dec 27 13:41:16 localhost tftpd-hpa[2844]:  * Starting HPA's tftpd in.tftpd

Dec 27 13:41:16 localhost tftpd-hpa[2844]:    ...done.
```

10.1.4 Step 4: Configure device-facing IP address

Add the expected IP address. Replace <ETHERNET_INTERFACE> below with the name of the ethernet interface on your machine that is connected to either red port of the TOE front panel.

```
# The interface connected to either red port should have ip addr show <IF> ==>
inet 10.0.1.1/12 scope global <ETHERNET_INTERFACE> ... If not, then run

sudo ip addr add 10.0.1.1/12 dev <ETHERNET_INTERFACE>

sudo ip addr add 10.0.1.1 dev <ETHERNET_INTERFACE>

sudo ip addr add 10.0.2.1 dev <ETHERNET_INTERFACE>
```

```
sudo ip addr add 10.0.3.1 dev <ETHERNET_INTERFACE>
```

10.2 Administrator Usage

Once logged into the device, run `set_update_mode <MODE>` to set the update mode to either `UPDATE_DISABLED`, `MANUAL_UPDATE`, or `AUTO_UPDATE` (default). Run with the help argument (-h) for more information.

```
[root@node1 ~]# set_update_mode -h
This program sets the update mode for this device as
specified via a command line argument.
Syntax: set_update_mode [AUTO_UPDATE|MANUAL_UPDATE|UPDATE_DISABLED]


Currently, 3 options are supported:
   (1.) AUTO_UPDATE
         - Device will check for updates on reboot and
           update if one is available. This behavior will
           persist across reboots as the default.
   (2.) MANUAL_UPDATE
         - Device will check for updates once on reboot and
           update if one is available. Once rebooted, the
           device will no longer check for updates unless
           this mode is set again explicitly.
   (3.) UPDATE_DISABLED
         - Device will not check for updates or attempt to
           update when rebooted.
```

10.2.1 Set MANUAL UPDATE mode

```
[root@node1 ~]# set_update_mode MANUAL_UPDATE
Update checking enabled on next boot.
```

10.3 Console Messages

During boot, the following relevant messages can be seen from the boot console:

```
# Manual Update Expected Output
| Checking value of update code...
| SF: Detected mt25qu02g with page size 256 Bytes, erase size 64 KiB, total 256 MiB
```

```
| Manual Update Code: 0x1
| State MANUAL_UPDATE detected. Proceeding with update...
| Resetting Update Flag to 0x0 (UPDATE_DISABLED)


# Update Disabled Expected Output
| Checking value of update code...
| SF: Detected mt25qu02g with page size 256 Bytes, erase size 64 KiB, total 256 MiB
| Manual Update Code: 0x2
| State UPDATE_DISABLED detected. Skipping to boot.


# Auto Update Expected Output
| Checking value of update code...
| SF: Detected mt25qu02g with page size 256 Bytes, erase size 64 KiB, total 256 MiB
| Manual Update Code: 0x0
| State UPDATE_AUTO detected. Proceeding with update...
```

If the Update candidate is successfully verified, the TOE will execute the update and reboot as described. If the update candidate is not successfully verified, the TOE will log an error message to the audit trail and the local console, and will not update to the new version. If the TOE update candidate does not verify, the administrator should contact SpaceX support for assistance.

11 TSF Testing

11.1 Update Tests

For the self-tests performed as part of trusted update in Xilinx UltraScale+ MPSOC and U-Boot, there are two failure errors possible.

Nominally, the following messages will be seen in the boot console from the Xilinx RSA signature validation and the U-Boot integrity check, respectively, if the boot is successful:

```
# Xilinx Signature Verification (example)


======= In Stage 3, Partition No:1 =======
...
.Authentication Enabled
XFsbl_Authentication: SHA2-384 Selected by Hash Override Flag
Auth: Partition Offset 3B00000, PartitionLen 25700, AcOffset FFFDF4C0, HashLen 30
```

```
Doing Partition Sign verification

XFsbl_PartitionVer: SHA2-384 Selected by Hash Override Flag

Partition Verification done

Partition 1 Load Success

======= In Stage 3, Partition No:2 =======

...

.Authentication Enabled

XFsbl_Authentication: SHA2-384 Selected by Hash Override Flag

Auth: Partition Offset 8000000, PartitionLen 66480, AcOffset FFFDF4C0, HashLen 30

Doing Partition Sign verification

XFsbl_PartitionVer: SHA2-384 Selected by Hash Override Flag

Partition Verification done

Partition 2 Load Success

All Partitions Loaded


# U-Boot Integrity Verification (example)


fit_hash_verify_sha384: Verifying FIT Integrity for pinned_fit_image_hash_fpga

fit_hash_node_offset: 20704

Computed SHA384
hash: 0x607e84aba8658f751897aa382a9e0bf80e755a720531e75ffbac9d08fd878858e554c2510cc3df
f9d3dc3e8a06974db2

Device-tree expected SHA384 hash
#0: 0x607e84aba8658f751897aa382a9e0bf80e755a720531e75ffbac9d08fd878858e554c2510cc3dff9
d3dc3e8a06974db2

Hashes match.
```

Failure in the Xilinx signature check will result in a failure to boot with the following message in the boot console. The device will update back to the old image. The Administrator will be able to detect using version_info that the update was not performed, and can verify this via the boot logs. The suggested action is to discard the update image that caused the failure, and attempt the update again with a new image.

```
RSA decrypted Hash END

XFsbl_PartVer: XFSBL_ERROR_PART_SIGNATURE

Partition 2 Load Failed, 0x32
```

Failure in the U-Boot integrity check will result in a failure to boot. The device will not boot linux and will instead present the U-Boot console. The suggested action is to discard the update image that caused the failure and attempt the update again with a new image by rebooting the device.

```
fit_hash_verify_sha384: Verifying FIT Integrity for pinned_fit_image_hash_kernel

fit_hash_node_offset: 312

Computed SHA384
hash: 0x6e8d2fa829da4218a9d5b41109c7f07544aa2afdf4cffaac122f66779e2cf65dd0b3aefb5e52c9
1332b3adbde9546229

Device-tree expected SHA384 hash
#0: 0x12ab25d7541901abc552d6b22de9a0df6fe4ee82268ce55b61d0a0f6e5aaa1af2d0f68189d93e826
35a7d7e2dde3293c

No matching hash found.

fit_hash_verify_sha384: Verification failed. No FIT hash match found.Failed to verify
sha384 hash for kernel

ERROR: can't get kernel image!


  Run 'load_fdt' to configure the FPGA(s) (if applicable).

  Run 'load_fpgabs' to bootup the FPGA(s) (if applicable).

  Run 'load_nxid' to load the board identification (including MAC addresses).


ZynqMP>
```

11.2 BoringSSL

The BoringSSL cryptographic library performs self-tests both at startup and during runtime depending on the algorithm under test.

The following algorithm self-tests are performed during initialization of the library:

AES KAT

AES-GCM KAT

ECDSA KAT

HMAC KAT

SHA KAT

The following algorithm self-tests are performed during operation of the library:

ECDSA pairwise consistency test

If any of the Known Answer Tests (KAT) were to fail, the TOE would fail to perform a function as soon as the TOE initialized the cryptographic library. In the case of an SSH algorithm failure, this would result in the TOE being unreachable over SSH. In the case of OpenIKED, the failure would occur as soon as the daemon initialized.

If the pairwise consistency test failed, the shutdown of TOE functionality would occur during an ECDSA negotiation, such as when trying to establish a new SSH connection (whether a successful one has already been performed or not) or when performing an IKEv2 SA or child SA negotiation.

Some clarity as to which of the self-tests is failing could be obtained by running the IKE or SSH applications in debug mode.

In the event of such a failure, the TOE should first be rebooted to check if the problem persists, and if it does, a software update should be performed.

11.3 SE050F (Secure Element)

The secure element contains the private key(s) for host identification, along with the corresponding certificate(s) and the random bit generator (RBG). The TOE utilizes the secure element for signature operations using the private key (for host authentication) and randomness (generated using the internal RBG).

The SE050 performs both boot-time and runtime self-checks.

11.3.1 Boot-time checks

If the SE050 boot-time checks fail, the devices will fail to fully enumerate to the driver. In that case, the driver will back out not attach to the device at all, meaning all run-time operations will fail (both signature and randomness). While the services may be running, no connections will succeed.

11.3.2 Runtime checks

Similarly, if runtime checks fail, no connections will succeed:

If the internal signature self-consistency checks fail, the host-authentication operation will fail.

If the RBG health check fails, the key exchange step will fail.

11.4 Linux

The Linux kernel performs self-tests on the cryptographic algorithms used for IPsec. If these self-tests fail, data will not be encrypted by the Linux kernel for the purpose of securing IPsec traffic. To determine whether a failed kernel crypto self-test is the cause of the loss of communication, the administrator should examine the `/proc/crypto` file to look for failures.

```
$ cat /proc/crypto | head -n 14

name         : echainiv(authenc(hmac(sha384),cbc(aes)))

driver       : echainiv(authenc(hmac(sha384-arm64),cbc-aes-ce))

module       : kernel
```

```
priority     : 3150
refcnt       : 1
selftest     : passed
internal     : no
type         : aead
async        : yes
blocksize    : 16
ivsize       : 16
maxauthsize  : 48
geniv        : <none>
```

The presence of failed in the output will reveal to the administrator which of the kernel self-tests failed. If a reboot of the TOE does not solve the problem, a software update should be performed on the TOE to ensure the implementation is correct.

Details on the time at which the failure occurred can be found by examining `/var/log/messages`

12 Reliable Time Stamps

In order to ensure proper function of the TOE, the device must have proper timekeeping. In order to keep the TOE up to date, administrators can adjust the TOE time using the date command:

```
$ date -u -s "2015-10-22 02:28"
Thu Oct 22 02:28:00 UTC 2015
```

13 TSF-initiated Session Locking

13.1 Configuring session inactivity timeout for administrators

The TOE supports both local and remote session locking for security administrators with a configurable timeout value. The timeout value is determined by the `TMOUT` environment variable. To check the current value for session timeout, the following command can be performed:

```
$ echo $TMOUT
900
```

The `TMOUT` environment variable can be changed for the current administrative session by performing the following command:

```
$ export TMOUT=<value>
```

Where <value> is the number of seconds of inactivity required for session timeout. In order to set a timeout value that will persist across multiple sessions, the value must be changed in the /etc/profile .

13.2 TSF-initiated Session Termination

13.2.1 Configuring session inactivity timeout for administrators

The TOE supports both local and remote session locking for security administrators with a configurable timeout value. The timeout value is determined by the TMOUT environment variable. To check the current value for session timeout, the following command can be performed:

```
$ echo $TMOUT

900
```

The TMOUT environment variable can be changed for the current administrative session by performing the following command:

```
$ export TMOUT=<value>
```

Where <value> is the number of seconds of inactivity required for session timeout. In order to set a timeout value that will persist across multiple sessions, the value must be changed in the /etc/profile .

13.3 User-initiated Termination

13.3.1 Terminating an administrative session

A security administrator can manually terminate an administrative session by issuing either of the following commands in the shell:

```
$ exit
```

```
$ logout
```

14 Default TOE Access Banner

14.1 Configuring default access banners

The default access banner that will be displayed upon attempting to establish a local or remote administrative session can be viewed or changed by editing the /etc/issue file:

```
$ cat /etc/issue

This is a configurable banner message!
```

15 X509 Certificate Validation

15.1 General Information on X.509 Usage

X.509 certificates are only used by the device for authentication during Internet Key Exchange v2 (IKEv2) for IPSec tunnel establishment. They are validated in two distinct ways:

1. Certificates loaded into the device's trust store are validated when the VPN application and IKEv2 daemon are started. This includes validation of the device's own identity leaf certificate, as well as the full certificate chain and certificate authorities (CAs)
2. Certificates presented by a peer during the IKE_AUTH phase of tunnel establishment are validated against trusted chains in the trust store once the VPN IKE daemon is running.

Certificates are checked for validity based on the RFC 5280 specification, which include some of the following conditions for reference:

- A full chain is present that terminates in a self-signed root of trust
- No certs have expired based on the date at the time the check is performed
- No certs in the chain are present in the Certificate Revocation Lists (CRLs) issued by trusted CAs
- Certificate Revocation Lists are valid for each CA in the chain of trust
- The certificate has not been modified in any way in transit (verified by digital signature)
- Curve parameters must be named, not defined explicitly
- basicConstraints extensions on CAs must be present and mark CA=TRUE
- CRLs must not be expired and must be up to date

15.2 Generating New Keys

Two key types are supported by the device's SE050 secure element: **ecdsa-sha2-nistp384** and **ecdsa-sha2-nistp256**. For first time setup, the two curves (p256 and p384) must be activated with the following steps. This only needs to be performed once. This example shows activation for **NIST_P384**, but the same steps should be replicated for **NIST_P256:**

```
# Navigate to the following directory

cd /sys/bus/i2c/drivers/se050/0-0048/ec_curves



# Verify the curve has not already been activated (returns 1 if activated, 0 if not).
If 1 is returned, the following steps are not needed, as the curve is active already.

cat NIST_P384/created



# Get the ID of the curve you want to generate (should be performed for NIST_P384 and
NIST_P256). This should return a byte value (e.g. 0x04)

cat NIST_P384/id



# Echo the ID into `create` file to activate the curve.

echo 0x04 > create
```

```
# Verify success

echo $?

cat NIST_P384/created
```

Once the curves are activated, the following steps can be followed to generate a new key pair

```
# CD into the following directory:

cd /sys/bus/i2c/drivers/se050/0-0048/secure_objects/ec_key_pairs


# Echo the a 4-byte ASCII name for the new key into the create/NIST_P256 or
create/NIST_P384 file for an ecdsa nistp256 or 384 key, respectively.

# EXAMPLE: Create a new NIST_P384 ecdsa key named "key0"

echo -n key0 > create/NIST_P384

# EXAMPLE: Create a new NIST_P256 ecdsa key named "key1"

echo -n key1 > create/NIST_P256


# Expected output of the ec_key_pairs/ directory post-creation

[root@node1 ec_key_pairs]# ls -l

total 0

drwxr-xr-x    3 root     root             0 Feb  8 23:16 0x6b657930/

drwxr-xr-x    3 root     root             0 Feb  8 23:27 0x6b657931/

drwxr-xr-x    3 root     root             0 Feb  8 23:18 0x7fff0201/

drwxr-xr-x    3 root     root             0 Feb  8 23:18 0x7fff0202/

drwxr-xr-x    3 root     root             0 Feb  8 23:18 0xf0000000/

drwxr-xr-x    3 root     root             0 Feb  8 23:18 0xf0000002/

drwxr-xr-x    3 root     root             0 Feb  8 23:18 0xf0000012/

drwxr-xr-x    3 root     root             0 Feb  8 23:18 0xf0000100/

drwxr-xr-x    3 root     root             0 Feb  8 23:18 0xf0000102/

drwxr-xr-x    2 root     root             0 Feb  8 23:18 create/

--w-------    1 root     root          4096 Feb  8 23:18 delete

lrwxrwxrwx    1 root     root             0 Feb  8 23:16 key0 -> 0x6b657930/

lrwxrwxrwx    1 root     root             0 Feb  8 23:27 key1 -> 0x6b657931/


# Hexdump or xxd the named key file directory 'bin' file to see the public key file in
hex format. The private key is not viewable. Example:

[root@node1 ec_key_pairs]# hexdump -C key1/bin
```

15.3 Creating X.509 Certificate Signing Requests (CSR) for a Key Pair

Once a key pair has been created, a certificate signing request or CSR can be generated with the following **generate_new_csr** utility. The tool has 1 required argument, which is the 4-byte name of the key pair to generate the CSR for (see the above section on generating new key pairs). An example is provided for a key named "key0". The tool will prompt for entering the country name (**C**), organization name (**O**) and common name (**CN**) fields to use in the cert and CSR. Certs are exported in DER format.

```
[root@node1 ~]# generate_new_csr key0

Generating new CSR for key key0. Please enter the following information:

   > Country Name (2 letter code): US

   > Organization Name (e.g., company): Example

   > Common Name: example-cert.company.com

Generating CSR for key-type P-384

Signature completed successfully.

Constructing final CSR in DER format...

CSR generation complete. Output can be found in new_csr.der
```

15.4 Loading CRLs into the Trust Store

CA certs, including the self-signed root CA, must be loaded into the following directories. The device expects all certs in PEM format. Certificates loaded into **iked/certs/** will be checked for a full chain of trust against the CA certs in **iked/ca/** on startup. Additionally, non-self-signed CA's in **iked/ca** will be checked for a valid chain terminating in a self-signed root.

```
/tmp/iked/ca/       # CA files such as the Root CA and any Intermediate CA's to
be trusted

/tmp/iked/certs/    # Identity certs (leaf) to be used by the device
itself for authentication

/tmp/iked/crls/     # Certificate Revocation List files for each CA
```

15.5 Using X.509 Certificates for Authentication

The NIAP-validated configuration of this device relies on a self-signed root CA, at least 1 intermediate CA signed by the root, and leaf identity certs that are signed by the intermediate.

Both the root and the intermediate certs that are considered "trusted" and should be loaded into the /tmp/iked/ca directory, and the identity cert to be used by this device is to be loaded into the /tmp/iked/certs directory, as shown above. The TOE supports only one certificate for itself.

If a CA certificate is self-signed and loaded into /tmp/iked/ca/ it will be designated as a trust anchor.

Only administrators are able to modify these directories and files

The “reference identifier” is the Common Name. In addition to the Common Name (CN) part of the Distinguished Name (DN) field, the TOE supports the SAN extension for x509 certificates, allowing the TOE to identify a peer by Fully Qualified Domain Name (FQDN). These FQDNs should be unique across

any peer available to authenticate with the TOE. If the SAN field is present, it takes priority over the CN field.

15.6 Reading the public key

The private key is stored only on the secure element and is not accessible, but is made available for signing. However, the public key can be read from the bin device in the key pair directory:

```
$ hexdump -Cv secure_objects/ec_key_pairs/TEST/bin
```

The file always starts with 0x04 , meaning it is an uncompressed public key pair (X,Y).

16 SSH Configuration options

SSH server configuration is controlled via the **/etc/sshd_include** file on the device. A sample file configuration is shown below, and can be viewed on the device by running **cat /etc/sshd_include** as shown:

```
[root@node1 ~]# cat /etc/ssh/sshd_include


# Set log level for reporting login/logout

LogLevel VERBOSE


# Cryptographic Algorithms

Ciphers aes256-gcm@openssh.com

KexAlgorithms ecdh-sha2-nistp384   ecdh-sha2-nistp256




# Session timeouts

ClientAliveInterval 120

ClientAliveCountMax 5


# Rekeying parameters, units map to size,time->M,s

RekeyLimit 512M 2700


# Max retries

MaxAuthTries 5
```

```
CASignatureAlgorithms ecdsa-sha2-nistp384

HostbasedAcceptedAlgorithms ecdsa-sha2-nistp384


#  for SSH agent using Secure Element chip

HostKeyAlgorithms ecdsa-sha2-nistp384

PubkeyAcceptedAlgorithms ecdsa-sha2-nistp384


PasswordAuthentication no

PermitRootLogin without-password


# DO NOT MODIFY: Enable to use the SE050's random file for generation of private keys

# in the ECDH key exchange.

UseSE050RandomForECDHKeyExchange yes


# DO NOT MODIFY: Required for performing signatures with the device's secure element
chip

SE050DeviceFolder /sys/bus/i2c/drivers/se050/0-0048
```

For configuring allowed ciphers, key exchange (kex) algorithms, integrity algorithms (MACs), and authentication methods in a NIAP-compliant manner, the following options can be modified in this file by changing the value following the keyword. For example, to set the Cipher algorithms to an option supported by this device, set **Ciphers <name_of_algorithm>** as seen in the above files. Only the following NIAP-compliant options should be selected in this file for configurable fields:

Option Field	Supported Values	Notes
Ciphers	aes256gcm@openssh.com	
KexAlgorithms	ecdh-sha2-nistp384 , ecdh-sha2-nistp256	
MACs	hmac-sha2-512	Since **aes256-gcm@openssh.com** is the only supported cipher, the negotiated MAC will appear as **<IMPLICIT>** in the negotiation, even when **hmac-sha2-512** is selected. See NIAP TD0446 for more details on this distinction
HostKeyAlgorithms	ecdsa-sha2-nistp384	

Option Field	Supported Values	Notes
PubkeyAcceptedAlgorithms	ecdsa-sha2-nistp384	
RekeyLimit	1000M 3600	Not to exceed **1000M** for size and **3600** for time (seconds), lesser values are supported (e.g. 512M 2700 in the above example). See requirement FCS_SSHS_EXT.1.8 for more information on session keying compliance. 1 gigabyte and 1 hour are the maximum permissible values. Whichever threshold is reached first will trigger the rekeying operation.

16.1 Loading and Re-Loading SSH Configurations

As the root administrative user, configurations can be applied and queried by running the following commands

```
# Restart and apply SSHS configurations

[root@node1 ~]# killall -HUP sshd


# Query running SSHS configuration options

[root@node1 ~]# sshd -T
```