NSaaS Pi

>> First up - enable the SSH server as on Raspbian it's not enabled by default/hostapd

sudo raspi-config - enable SSH server

>> Connect wired ethernet, make sure you receive an IP address on eth0 and have internet access. Then update APT

sudo apt update

>> Install packages required. DNSmasq is a DNS forwarder and DHCP server. HostAP is a driver that allows the wireless card to run in HostAP mode. StrongSwan is an open-source IPsec solution.

sudo apt install strongswan hostapd dnsmasq

>> Stop the newly installed DNS and DHCP services as they're not configured yet.

sudo systemctl stop dnsmasq
sudo systemctl stop hostapd

>> Configure the DHCP file and make a note of the range. This will be used later in the VPN configuration.

sudo nano /etc/dhcpcd.conf

>> Add the following to the end of the file to configure your wireless card

interface wlan0
static ip_Address=192.168.200.1/24
nohook wpa_supplicant

>> Restart the DHCP service

sudo service dhcpcd restart

>> Take a backup of the DNSmasq config and create a new one

sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf

>> Add the following to the new /etc/dnsmasq.conf file

interface=wlan0
dhcp-range=192.168.200.2,192.168.200.100,24h

>> Restart dnsmasq

sudo systemctl reload dnsmasq

>> Configure the wireless AP settings

sudo nano /etc/hostapd/hostapd.conf

>> Add the following (you should be able to pick out the relevant parts to change for PSK and SSID values if you want to change yet)

```
interface=wlan0
hw_mode=g
driver=nl80211
ssid=CheckPoint_NSaaS
channel=11
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Cpwins1!
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

>> Next, configure the OS to know that this is the config file to use for hostapd. Edit the file /etc/default/hostapd

```
sudo nano /etc/default/hostapd
```

>> Find the section starting #DAEMON_CONF. Add the line below:

```
DAMON_CONF="/etc/hostapd/hostapd.conf"
```

>> Enable and start our new services

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd
sudo systemctl enable
```

>> At this point, we should have an SSID being broadcast and DNS / DHCP services ready to go. You'll be able to connect to the SSID at this point but you won't have internet access.
>> Enable IP forwarding.

```
sudo nano /etc/sysctl.conf
```

>> Uncomment the line

```
net.ipv4.ip_forward=1
```

>> Add a masquerade rule for outbound traffic on eth0. This is the IPtables equivalent of 'hide NAT'.

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

>> As we're using a VPN, we'll need to add a couple of rules to 'mangle' the MTU and MSS values.

```
sudo iptables -t mangle -A FORWARD -m policy --pol ipsec --dir in -p tcp -m tcp --tcp-flags SYN,RST SYN -m tcpmss --mss 1361:1536 -j TCPMSS --set-mss 1360
sudo iptables -t mangle -A FORWARD -m policy --pol ipsec --dir out -p tcp -m tcp --tcp-flags SYN,RST SYN -m tcpmss --mss 1361:1536 -j TCPMSS --set-mss 1360
```

>> Save the IPtables policy

```
sudo iptables-save > /etc/iptables.ipsec_rules
```

>> Make sure these settings are loaded every time the Pi reboots. Also make sure PMTU discovery is disabled. Edit the file /etc/rc.local and these lines above 'exit 0':

```
iptables-restore < /etc/iptables.ipsec_rules
echo 1 > /proc/sys/net/ipv4/no_ip_pmtu_disc
```

ifconfig eth0 mtu 1400 up
ipsec stop
ipsec start

>> That's the OS configured - now we can configure our IPsec settings. From the zip / GIT repo copy the files to the following locations:

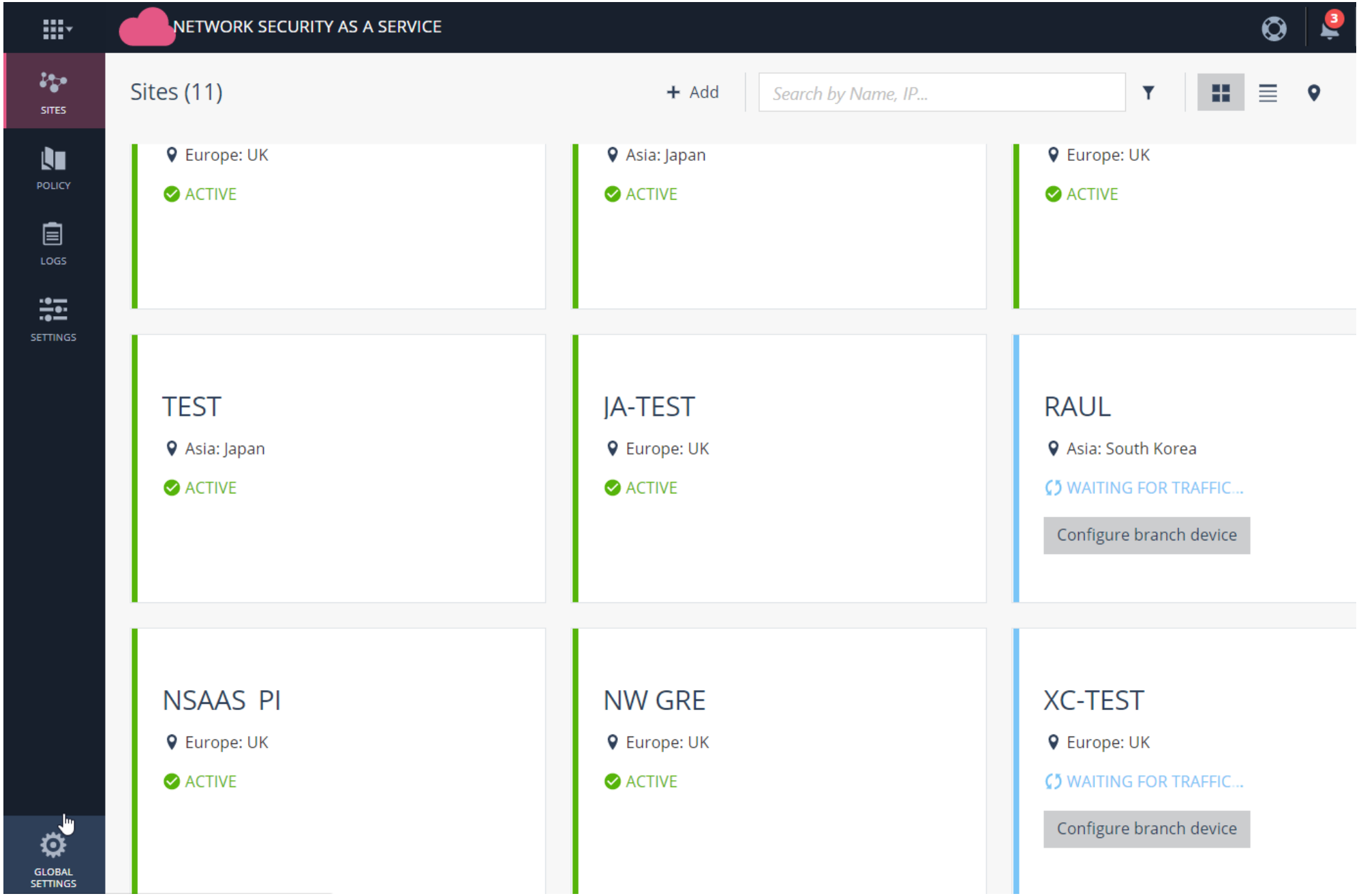
60-trigger_api.sh -> /home/
ipsec.conf -> /etc/
ipsec.secrets -> /etc/
ip_update.py -> /home/

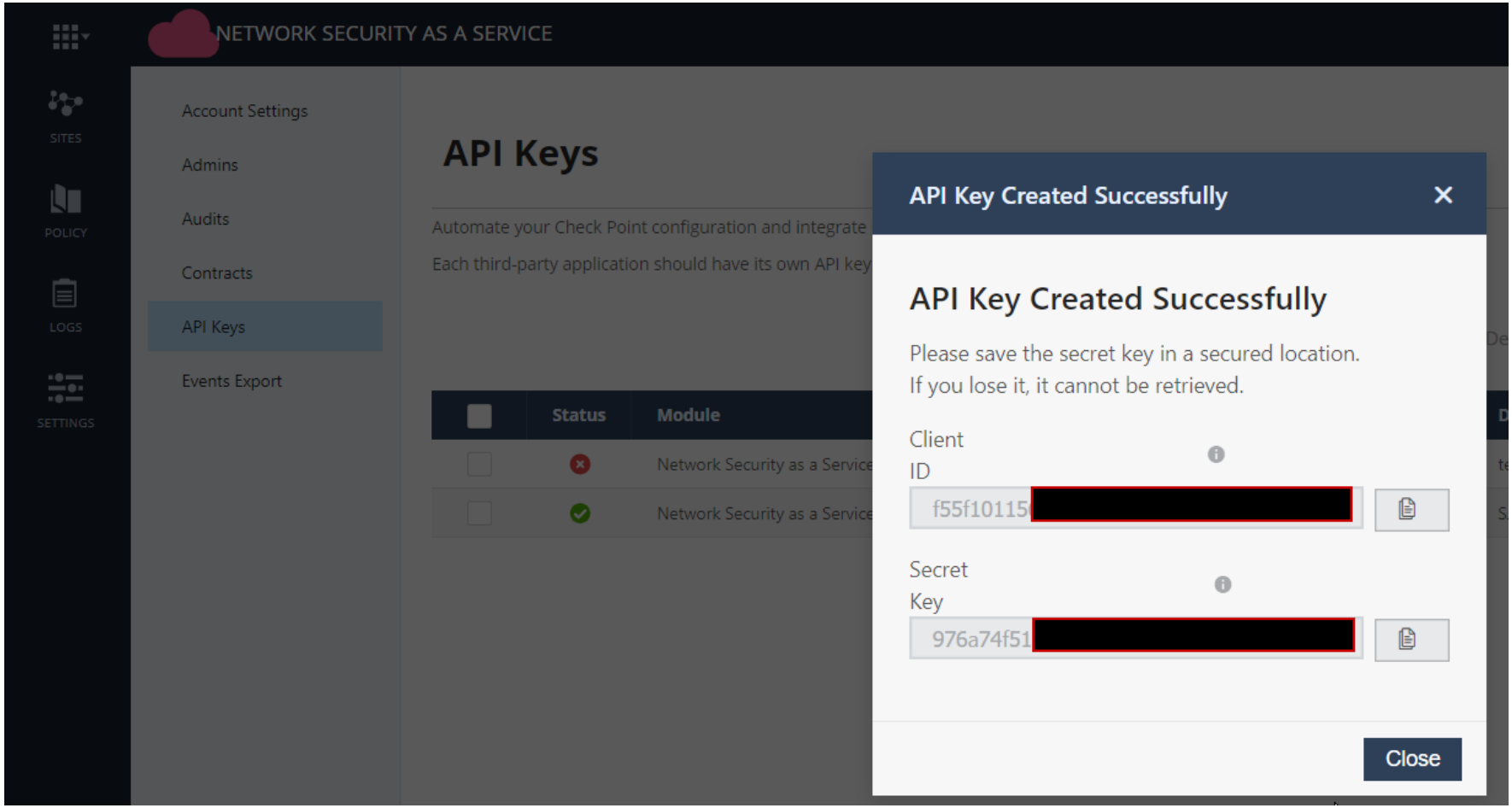
>> Create a DHCPCD exit hook to trigger the API update script anytime the ethernet port is replugged or ip is updated.
sudo cp /home/60-trigger_api.sh /etc/dhcpcd.exit-hook
sudo chmod +x /etc/dhcpcd.exit-hook

>> Create a cronjob to restart the VPN every 30 mins (seems to be a stability issue with Strongswan - looking into this).

sudo crontab -l > current_cron
sudo echo “*/30 * * * * ipsec stop; ipsec start" >> current_cron
sudo crontab current_cron

>> At this point, you'll need to have access to the NSaaS portal and have a site setup.
>> Create your site and make a note of the name (IMPORTANT - make sure it's a unique name - the portal allows duplicates!), PSK and cloud gw address. Make sure the internal network matches the wifi DHCP range you've setup on the Pi.
>> In the portal - go to global settings and create an API key for NSaaS. Copy out the client ID and secret key, you'll need to add these to the Python script ip_update.
>> While you're here - go to the site you want to connect to and copy out the exact site name - you'll need this later.


NETWORK SECURITY AS A SERVICE
3
SITES
POLICY
LOGS
SETTINGS
GLOBAL SETTINGS
Sites (11)
+ Add
Search by Name, IP...
Europe: UK
ACTIVE
Asia: Japan
ACTIVE
Europe: UK
ACTIVE
TEST
Asia: Japan
ACTIVE
JA-TEST
Europe: UK
ACTIVE
RAUL
Asia: South Korea
WAITING FOR TRAFFIC...
Configure branch device
NSAAS PI
Europe: UK
ACTIVE
NW GRE
Europe: UK
ACTIVE
XC-TEST
Europe: UK
WAITING FOR TRAFFIC...
Configure branch device



>> Now you've got the keys, add them to the ip_update.py file and make sure everything in the #connection info section is completed.

>> Next edit /etc/ipsec.conf. There are two sections to focus on, conn local-connections and conn local-to-cgnsaas.
>> In local-connections, make sure leftsubnet and rightsubnet are set to be the network for the WIFI. This section prevents local traffic being forced over the VPN
>> In local-to-cgnsaas configure the following properties:

right= This should be the FQDN you get from the NSaaS portal that you connect TO. You're given two by NSaaS, pick the first one only. For this demo, we don't need two tunnels.
leftsubnet= This should be set to your local WIFI network (which will match what you configured for the network on the NSaaS side).

>> Next, edit /etc/ipsec.secrets. This is the file that maps a site address / FQDN to a PSK. You'll want a line that looks like the below (but use the details from the portal for your site)...

g-1183-f26476d972d0fb1d6552a6f4b0bb9c8b.checkpoint.cloud : PSK "6NCXCVXCVogD7Ky4vXwc8bhBTUJoODFczyA"

>> You're done! Reboot the Pi and then when the wired interface comes up the following will happen:

1. The interface up / down script fires
2. ip_update.py file is executed - checking your external IP and then sending that to NSaaS via API.

3. When site is updated - the IPSEC services are restarted and the VPN tunnel comes up.
4. You can connect clients to the CheckPoint_NSaaS SSID and view the logs in the portal.

>> To verify IPSEC connectivity you can run

sudo ipsec statusall

>> The output should look like this:

```
pi@raspberrypi:~ $ sudo ipsec status
Shunted Connections:
local-connections:  192.168.200.0/24 === 192.168.200.0/24 PASS
Security Associations (1 up, 0 connecting):
local-to-cgnsaas[1]: ESTABLISHED 1 second ago, 192.168.124.154[192.168.124.154]...35.176.146.122[g-607-49d87846a4aae70778fdc6504b1eb463.checkpoin
t.cloud]
local-to-cgnsaas{1}:  INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: c9d570dc_i 392b5242_o
local-to-cgnsaas{1}:   192.168.200.0/24 === 0.0.0.0/0
pi@raspberrypi:~ $
```