Debug Secure Shell (SSH) on NCS1K

Contents

Introduction

This document describes basic troubleshooting practices for Secure Shell (SSH) on the NCS1K platform.

Prerequisites

This document assumes proficiency with XR-based operating systems on devices such as the Network Convergence System (NCS) 1002.

Requirements

Cisco recommends that you have knowledge of these topics for SSH connection requirements:

- The relevant k9sec package for the XR image
- SSH configuration present on the Cisco device
- A successful key generation, key exchange, and cipher negotiation between the host and server

Components Used

The information in this document is based on these software and hardware versions:

- NCS1002 with XR 7.3.1
- NCS1004 with XR 7.9.1

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Verify Installed Packages

The commands show install active and show install committed identify the presence of the k9sec package. Without this package installed, you cannot generate crypto keys to initiate an SSH session.

<#root>

```
RP/0/RP0/CPU0:NCS1002_1#
```

show install active

```
Wed Jul 19 09:31:18.977 UTC
Label : 7.3.1

Node 0/RP0/CPU0 [RP]
Boot Partition: xr_lv58
Active Packages: 4
ncs1k-xr-7.3.1 version=7.3.1 [Boot image]
ncs1k-mpls-te-rsvp-3.1.0.0-r731
ncs1k-mpls-2.1.0.0-r731
ncs1k-k9sec-3.1.0.0-r731
```

```
RP/0/RP0/CPU0:NCS1002_1#
```

show install committed

```
Wed Jul 19 09:31:37.359 UTC
Label : 7.3.1

Node 0/RP0/CPU0 [RP]
Boot Partition: xr_lv58
Committed Packages: 4
ncs1k-xr-7.3.1 version=7.3.1 [Boot image]
ncs1k-mpls-te-rsvp-3.1.0.0-r731
ncs1k-mpls-2.1.0.0-r731
ncs1k-k9sec-3.1.0.0-r731
```

Configuration

At the very least, the NCS1K requires the configuration ssh server v2 in order to allow SSH connections. Enter show run ssh in order to ensure this configuration is present:

<#root>

```
RP/0/RP0/CPU0:NCS1004_1#
```

show run ssh

```
Wed Jul 19 13:06:57.207 CDT
ssh server rate-limit 600
ssh server v2
ssh server netconf vrf default
```

Identify Generated Keys

In order to establish an SSH session, the NCS1K must have a public cryptographic key present. Identify the presence of generated keys with show crypto key mypubkey { dsa | ecdsa | ed25519 | rsa }. The default key type is rsa. The key appears as a hexadecimal string, omitted here for security purposes.

```
<#root>

RP/0/RP0/CPU0:NCS1002_1#

show crypto key mypubkey rsa


Wed Jul 19 10:30:09.333 UTC
Key label: the_default
Type : RSA General purpose
Size : 2048
Created : 11:59:56 UTC Tue Aug 23 2022
Data : <key>
```

In order to generate a key of a particular type, enter the command crypto key generate { dsa | ecdsa | ed25519 | rsa } and choose a key modulus. The modulus size varies by algorithm.

Key Type	Allowed Modulus/Curve Types	Default Modulus Length (bits)
dsa	512, 768, 1024	1024
ecdsa	nistp256, nistp384, nistp521	none
ed25519	256	256
rsa	512 to 4096	2048

Verify the key generated successfully with show crypto key mypubkey.

In order to remove an existing key, enter the command crypto key zeroize { authentication | dsa | ecdsa | ed25519 | rsa } [label]. Ensure you have access to the device through other means as disconnection from a device with no crypto keys blocks access with SSH.

Identify SSH Server Capabilities

The server and host must agree on a key exchange, host key, and cipher before establishing an SSH session. In order to identify the capabilities of the NCS1K platform, enter the command show ssh server.

```
<#root>
```

```
RP/0/RP0/CPU0:NCS1004_1#
```

```
show ssh server
```

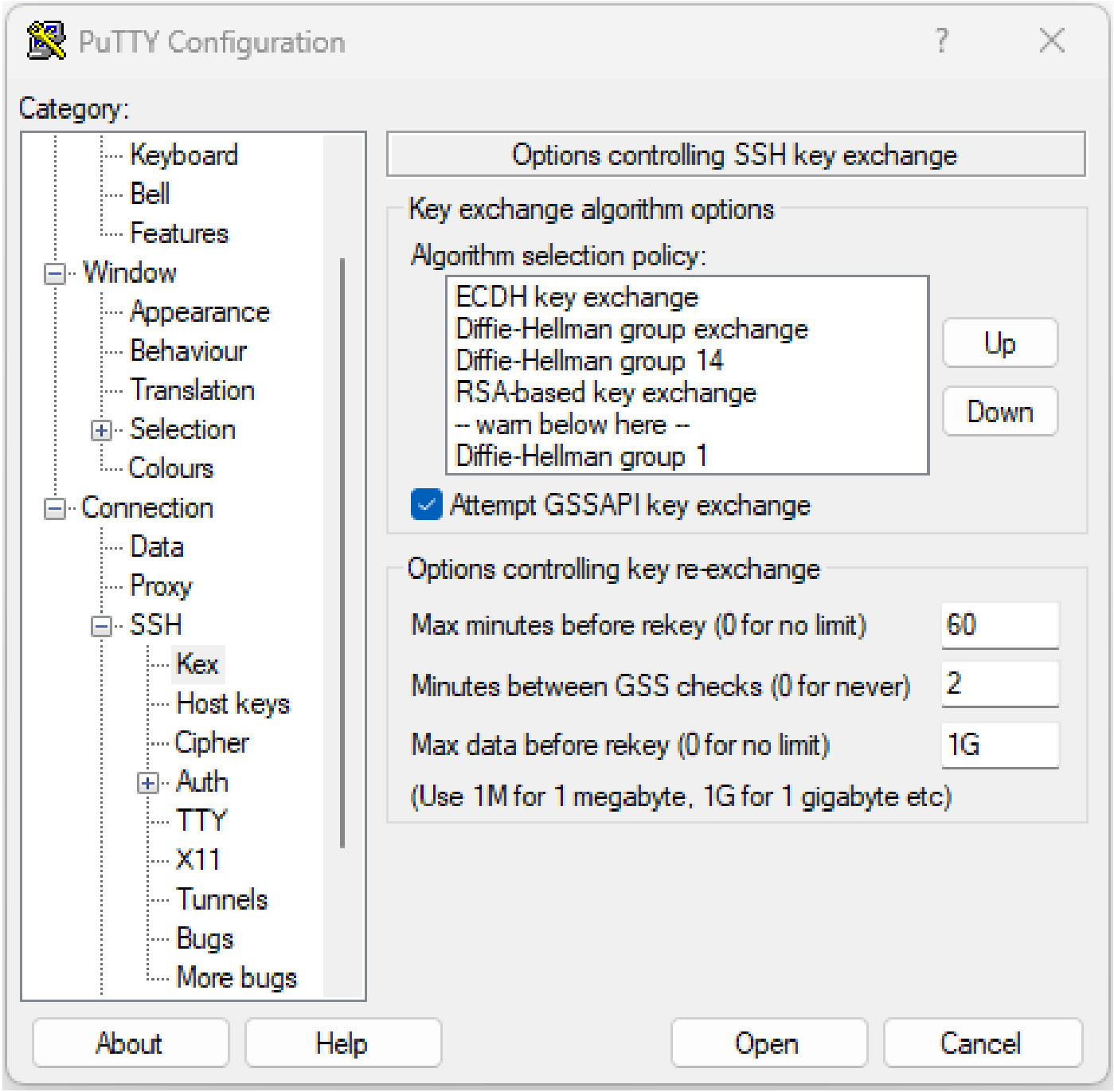
```
Wed Jul 19 13:28:04.820 CDT
---------------------
SSH Server Parameters
---------------------

Current supported versions := v2
SSH port := 22
SSH vrfs := vrfname:=default(v4-acl:=, v6-acl:=)
Netconf Port := 830
Netconf Vrfs := vrfname:=default(v4-acl:=, v6-acl:=)

Algorithms
---------------
Hostkey Algorithms := x509v3-ssh-rsa,ecdsa-sha2-nistp521,ecdsa-sha2-nistp384,ecdsa-sha2-nistp256,rsa-sh
Key-Exchange Algorithms := ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-grou
Encryption Algorithms := aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com
Mac Algorithms := hmac-sha2-512,hmac-sha2-256,hmac-sha1

Authentication Method Supported
------------------------------------
PublicKey := Yes
Password := Yes
Keyboard-Interactive := Yes
Certificate Based := Yes

Others
------------
DSCP := 16
Ratelimit := 600
Sessionlimit := 64
Rekeytime := 60
Server rekeyvolume := 1024
TCP window scale factor := 1
Backup Server := Disabled
Host Trustpoint :=
User Trustpoint :=
Port Forwarding := Disabled
Max Authentication Limit := 20
Certificate username := Common name(CN)
```

Identify Host SSH Capabilities

The host attempting to connect must match at least one hostkey, key exchange, and encryption algorithm from the server in order to establish an SSH session.

PuTTY

PuTTY lists the supported key exchange, host key, and cipher algorithms under Connections > SSH. The host automatically negotiates the algorithms based on its capabilities, preferring the key exchange algorithm in order of user preference. The option Attempt GSSAPI key exchange is not required in order to connect to an NCS1K device.



Screenshot of PuTTY SSH options

Linux

Linux servers typically keep the supported algorithms in the /etc/ssh/ssh_config file. This example originates from Ubuntu Server 18.04.3.

```
Host *
# ForwardAgent no
# ForwardX11 no
# ForwardX11Trusted yes
# PasswordAuthentication yes
# HostbasedAuthentication no
# GSSAPIAuthentication no
# GSSAPIDelegateCredentials no
# GSSAPIKeyExchange no
```

```
# GSSAPITrustDNS no
# BatchMode no
# CheckHostIP yes
# AddressFamily any
# ConnectTimeout 0
# StrictHostKeyChecking ask
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
# IdentityFile ~/.ssh/id_ecdsa
# IdentityFile ~/.ssh/id_ed25519
# Port 22
# Protocol 2
# Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
# MACs hmac-md5,hmac-sha1,umac-64@openssh.com
# EscapeChar ~
# Tunnel no
# TunnelDevice any:any
# PermitLocalCommand no
# VisualHostKey no
# ProxyCommand ssh -q -W %h:%p gateway.example.com
# RekeyLimit 1G 1h
SendEnv LANG LC_*
HashKnownHosts yes
GSSAPIAuthentication yes
```

Troubleshoot SSH Connections

These commands can help isolate failures with SSH connections.

See current incoming and outgoing SSH sessions with show ssh session details.

<#root>

```
RP/0/RP0/CPU0:NCS1002_1#
```

show ssh session details

```
Wed Jul 19 13:08:46.147 UTC
SSH version : Cisco-2.0

id key-exchange pubkey incipher outcipher inmac outmac
------------------------------------------------------------------------------------------------
Incoming Sessions
128733 ecdh-sha2-nistp256 ssh-rsa aes256-ctr aes256-ctr hmac-sha2-256 hmac-sha2-256
128986 diffie-hellman-group14 ssh-rsa aes128-ctr aes128-ctr hmac-sha1 hmac-sha1
128988 diffie-hellman-group14 ssh-rsa aes128-ctr aes128-ctr hmac-sha1 hmac-sha1

Outgoing sessions
```

Historical SSH sessions include failed connection attempts with the command show ssh history detail.

<#root>

```
RP/0/RP0/CPU0:NCS1002_1#
```

```
show ssh history details


Wed Jul 19 13:13:26.821 UTC
SSH version : Cisco-2.0

id key-exchange pubkey incipher outcipher inmac outmac start_time end_time
-------------------------------------------------------------------------------------------------------
Incoming Session
128869diffie-hellman-group14-sha1ssh-rsa aes128-ctr aes128-ctr hmac-sha1 hmac-sha1 19-07-23 11:28:55 19
```

SSH traces give a fine level of detail on the connection process with show ssh trace all.

```
<#root>

RP/0/RP0/CPU0:NCS1002_1#

show ssh trace all


Wed Jul 19 13:15:53.701 UTC
3986 wrapping entries (57920 possible, 40896 allocated, 0 filtered, 392083 total)
Apr 29 19:13:19.438 ssh/backup-server/event 0/RP0/CPU0 t6478 [SId:=0] Respawn-count:=1, Starting SSH Se
Apr 29 19:13:19.438 ssh/backup-server/shmem 0/RP0/CPU0 t6478 [SId:=0] Shared memory does not exist duri
```

Configure SSH Re-Key Values

SSH re-key configuration determines the time and number of bytes before a new key exchange occurs. See the current values using show ssh rekey.

```
<#root>

RP/0/RP0/CPU0:NCS1004_1#

 show ssh rekey


Wed Jul 19 15:23:06.379 CDT
SSH version : Cisco-2.0

id RekeyCount TimeToRekey(min) VolumeToRekey(MB)
-----------------------------------------------------------
Incoming Session
1015      6      6.4              1024.0
1016      0      58.8             1024.0

Outgoing sessions
```

In order to set the re-key volume, use the command ssh server rekey-volume [size]. The default re-key size is 1024 MB.

```
<#root>
```

```
RP/0/RP0/CPU0:NCS1004_1(config)#
ssh server rekey-volume 4095

RP/0/RP0/CPU0:NCS1004_1(config)#
commit
```

Likewise, set the re-key timer value with ssh server rekey-time [time]. The default value is 60 minutes.

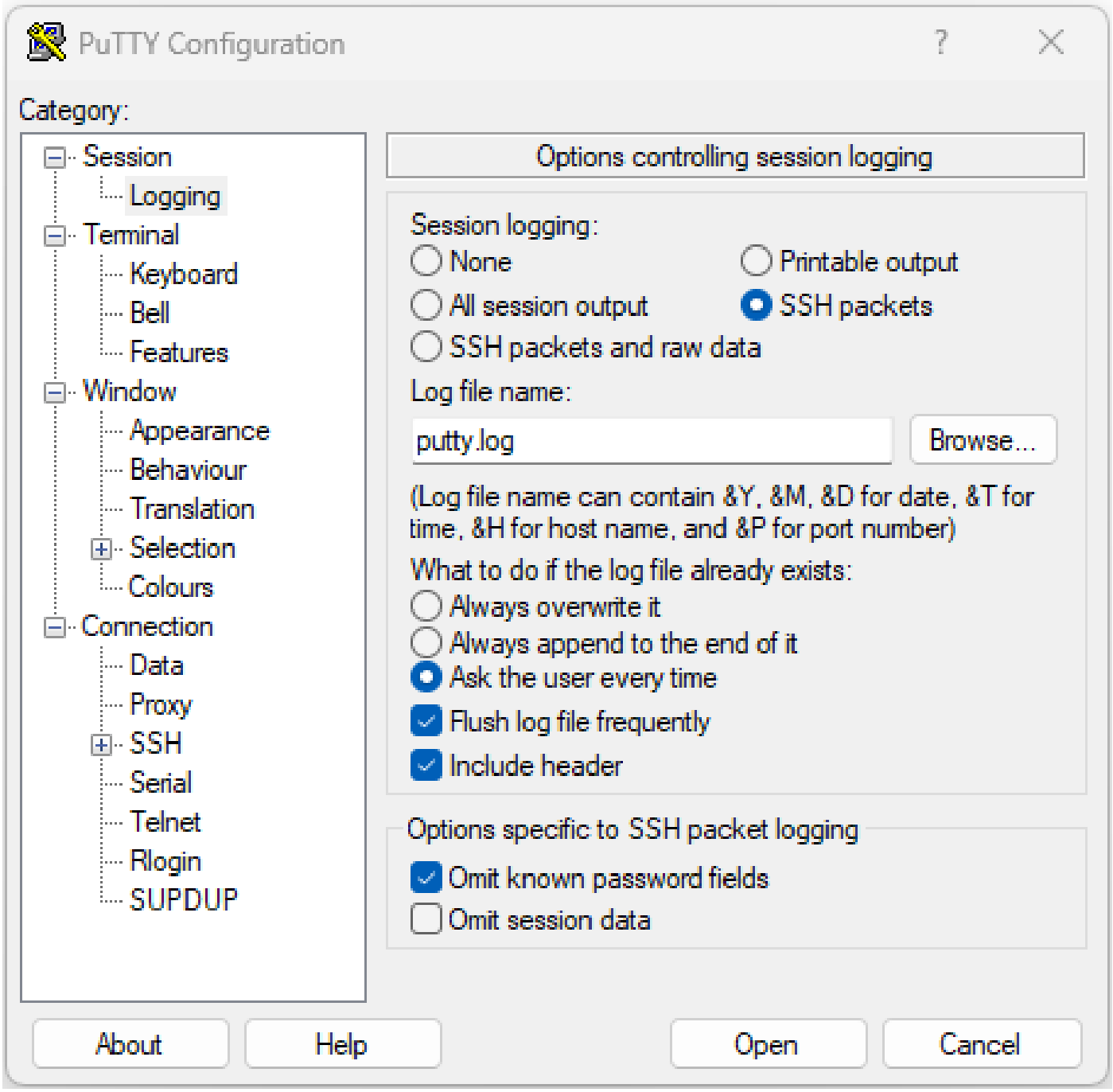
```
RP/0/RP0/CPU0:NCS1004_1(config)# ssh server rekey-time 120
RP/0/RP0/CPU0:NCS1004_1(config)# commit
```

SSH Debug

The debug ssh server command displays real-time outputs for active SSH sessions and connection attempts. In order to troubleshoot a failing connection, enable the debug, attempt the connection, and then stop the debug with undebug all. Log the session using PuTTY or another terminal application for analysis.

```
<#root>
RP/0/RP0/CPU0:NCS1002_1#
debug ssh server
```

PuTTY includes a feature for logging SSH packets under Session > Logging.



Screenshot of PuTTY SSH logging

In Linux, ssh -vv (very verbose) gives detailed information on the SSH connection process.

<#root>

```
ubuntu-18@admin:/$
```

```
ssh -vv admin@192.168.190.2
```

Additional Logs

Several show techs capture useful information on SSH.

- **show tech { ncs1k | ncs1001 | ncs1004 } detail**
- **show tech crypto session**
- **show tech ssh**
- **admin show tech { ncs1k | ncs1001 | ncs1004 }-admin**