

**WAP Implementation Documentation**

Ryan Chen

5/20/2024

**Purpose:**

The primary purpose of this lab is to familiarize ourselves with Cisco Access Point configuration. To do so, our task was to configure an access point that broadcasted three unique SSID’s. Two of these SSID’s were set to be using WPA2 PSK, whereas the other one uses a RADIUS server to provide authentication. All three SSID’s were required to be able to connect to the Internet. Throughout this lab, we both learned AP configuration and also reviewed previous concepts such as DHCP, NAT/PAT, and VLAN setup/trunking.

**Background Information:**

Access Point Technology

With the rise of wireless technologies in recent decades, more network topologies have shifted from traditional, physical ethernet connections to more convenient wireless connections. To do so, most networks utilize a Wireless Access Point (WAP), which enables various clients ranging from laptops to IP phones to connect wirelessly to the network. These access points function on a network topology by continuously broadcasting a Service Set Identifier (SSID) on a certain radio frequency or channel, from which endpoint devices can pick up on and connect to. Devices will send authentication requests to the access point based on the type of wireless connection. After the device is authenticated, the access point will act as an intermediary device between the client and the network topology. This enables wireless communication with minimized hassle, as the access point functions similarly to a router/switch in handling packets.

This lab required us to configure a Cisco Aironet AP, which we would use to create three different SSID’s. Before configuring, we needed to ensure that the AP was running latest software. Due to the relatively outdated nature of the AP’s, we needed to download new Cisco firmware and flash it onto the device. To do so, we learned how to enter boot recovery on an AP, and from there we transferred a firmware image through TFTP.

Typically, clients on an access point will be assigned addresses from a private network through NAT. In this lab, we assigned each client on a private network, which ran through two layers of NAT in order to get to the internet. First, we needed to run PAT on the one address assigned to us by our upstream DHCP, and then we needed to run NAT on each SSID.

Wireless Security

After successfully loading firmware and configuring basic settings on the AP, we needed to create unique SSID’s using WPA2 and WPA2 Enterprise for authentication. Since WAP’s provide wireless capabilities, packets are sent through radio frequencies. However, given that this is the case, attackers have leveraged this to perform packet sniffing. With the right equipment and setup, an attacker can intercept packets between an AP and a client, thus presenting a significant security issue. By sniffing packets, an attacker can intercept potentially critical information, such as login information to websites.

Initially, a protocol called Wired Equivalent Privacy (WEP) was created and used in the early 2000’s to provide wireless security. This protocol was aimed at providing security that matched that of wired networks, meaning that attackers would not be able to decrypt packets that were sniffed. However, this protocol has been deprecated for quite a long time now, as numerous different issues were found within its implementation. Due to limited cryptographic advances at the time, the WEP protocol is easy to brute-force and crack with modern computing power, and as such should not be used.

WPA was introduced in order to provide a fix for the faults that WEP had, and presented stronger cryptographic encryption, as well as separate protocols for industry and home use. WPA-Personal is used for home networks and provides a simple solution to network security. On the other hand, WPA-Enterprise is tailored towards corporate use, and it utilizes an external Remote Authentication Dial-In-User Service (RADIUS) to provide login. RADIUS is a protocol which provides an AAA service, consisting of:

1. Authentication: Logging in as various different users, each with unique passwords
2. Authorization: Determining what permissions are granted to logged in users
3. Accounting: Tracking user resource consumption and session history

RADIUS authenticates users using a user/pass scheme. In conjunction with WPA, it enables enterprise use. WPA2 builds off of WPA, however uses AES instead of TKIP for encryption, whereas WPA3 further increases cryptographic strength and adds various other improvements. In this lab, we utilized an external drive to setup RADIUS, mimicking a real server. However, one could also as easily setup RADIUS on a Docker image, which allows for scalability and flexibility.

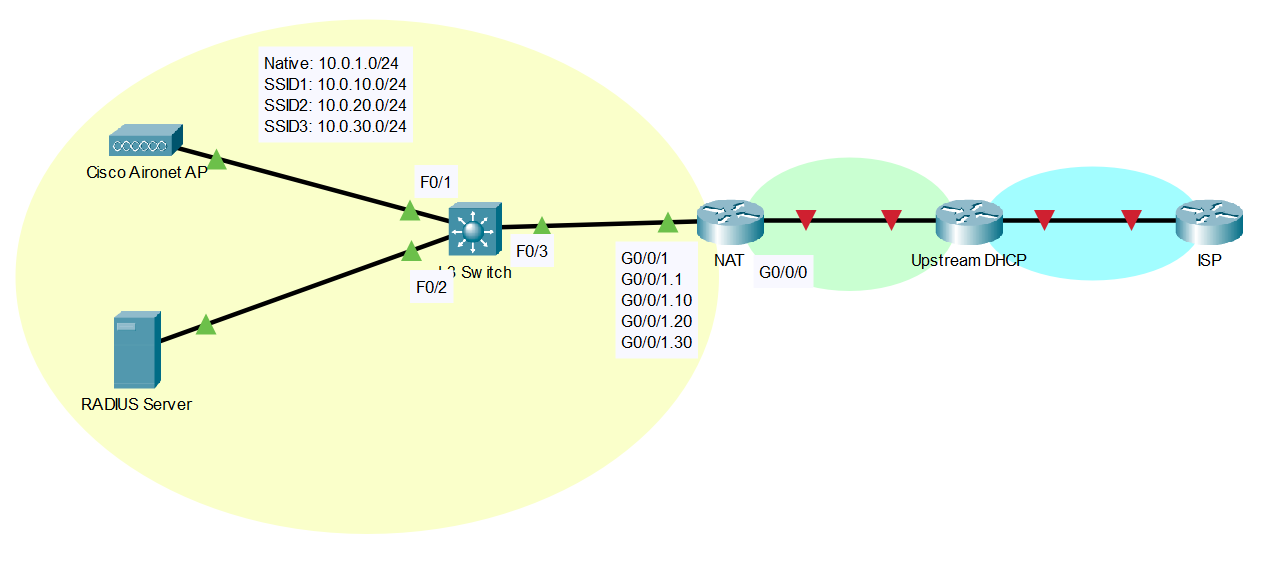
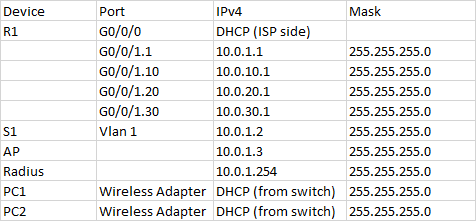
**Lab Summary:**

First off, we needed to flash our AP using the appropriate firmware. The AP’s that we received were running older firmware, and our task was to put K9W7-JD9 on it. To do so, we used TFTP in order to transfer the file over. After flashing our firmware, we continued with basic AP configuration.

In order to get this lab working, we needed to use both DHCP and PAT in order to retrieve an internet-accessible address, as well as to give addresses to clients connecting to our SSID. To do so, we had to use VLANs to separate our three different SSID networks. Each had its own subnet, and was trunked to the switch and router. This would allow each VLAN to reach the internet, but avoid security implications while doing so. Each VLAN is separate from the others and cannot ping them.

Finally, we needed to configure some form of a RADIUS server. We settled on using an external Ubuntu drive to do so, mimicking a real server scenario. However, with the rise of virtualization, this could easily have been put in a Docker container to simplify deployment and increase flexibility.

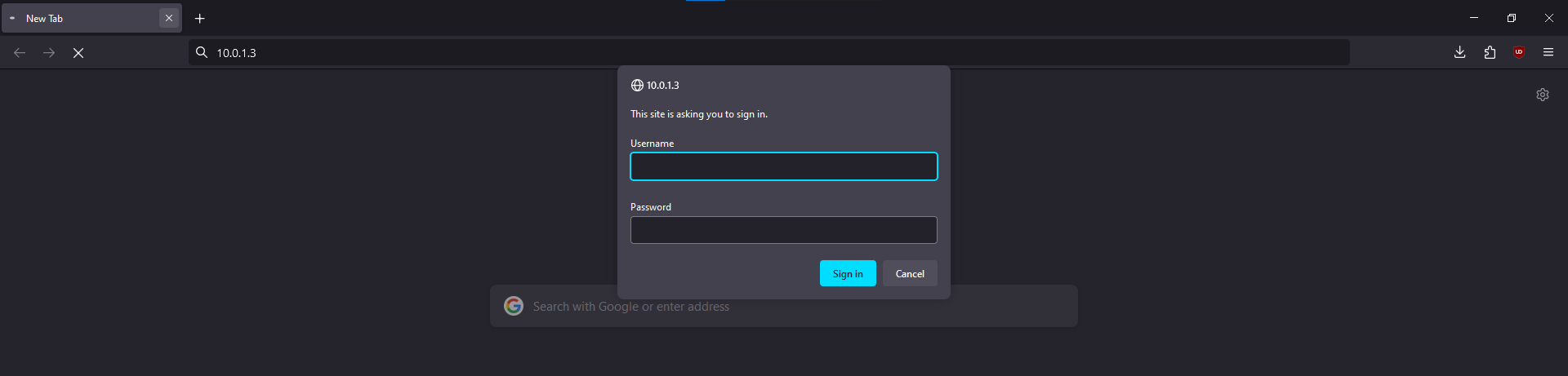
**Network Diagram and IP Table:**



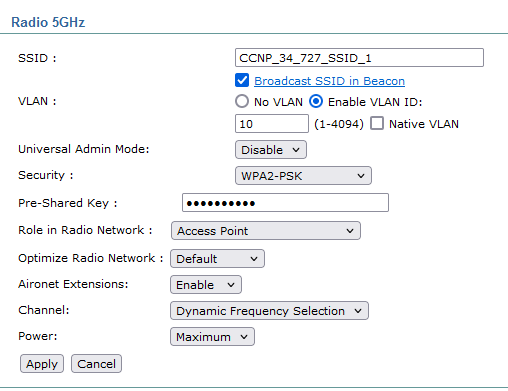
**Configurations:**

**AP WPA2 PSK SSID:**

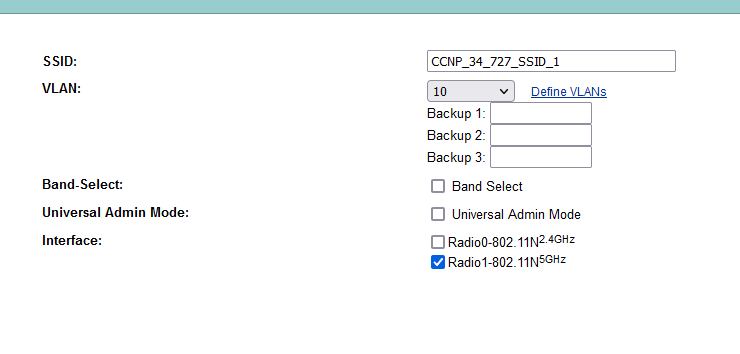
1. Enter the Cisco WAP by typing in the assigned IP address in a computer connected to the same LAN. The default username and password is “Cisco”.



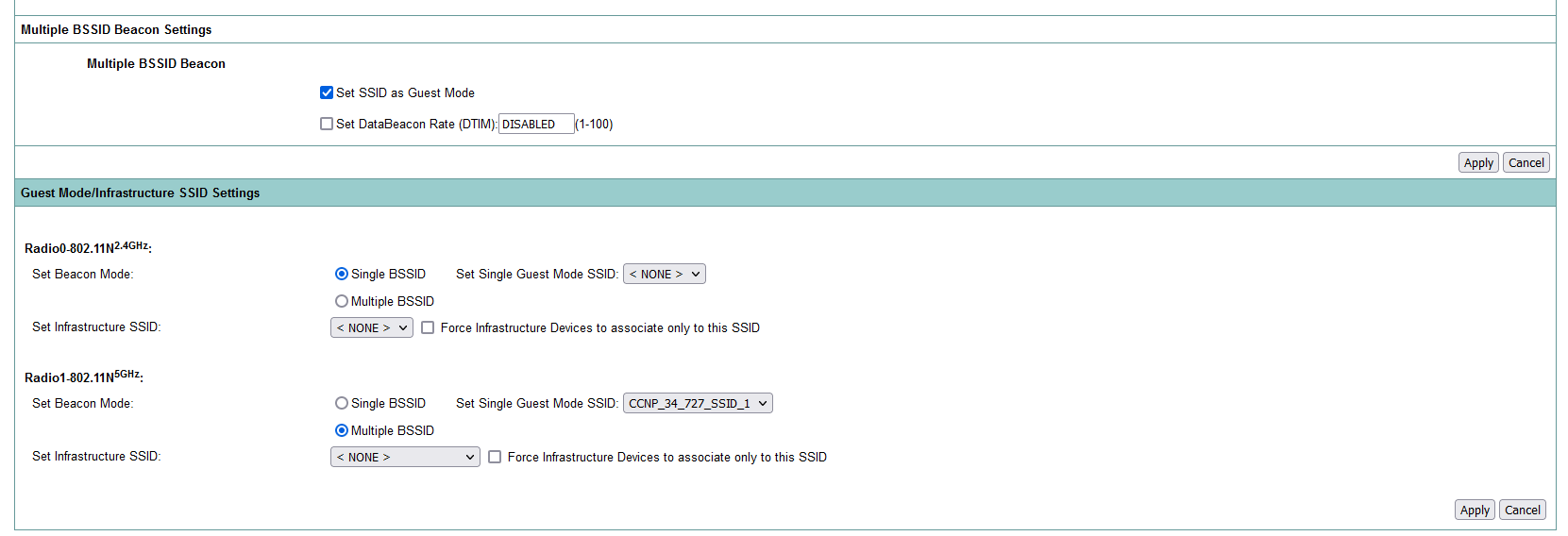
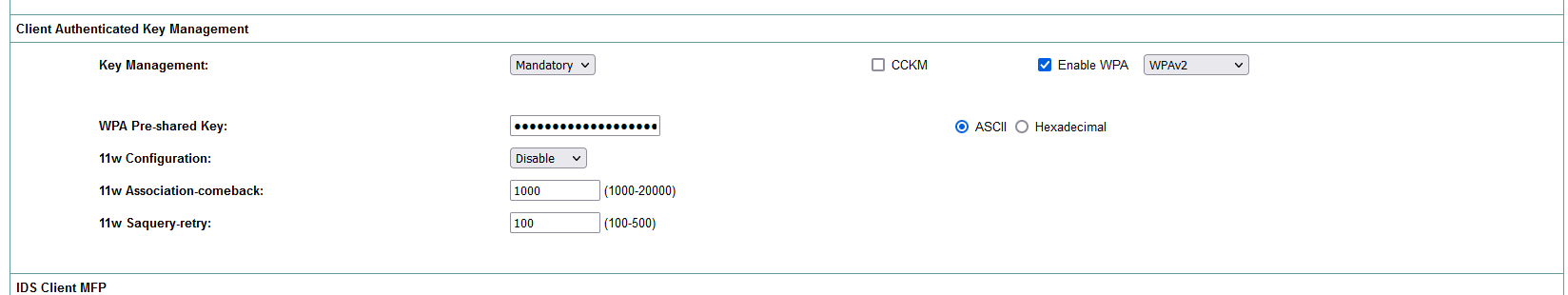
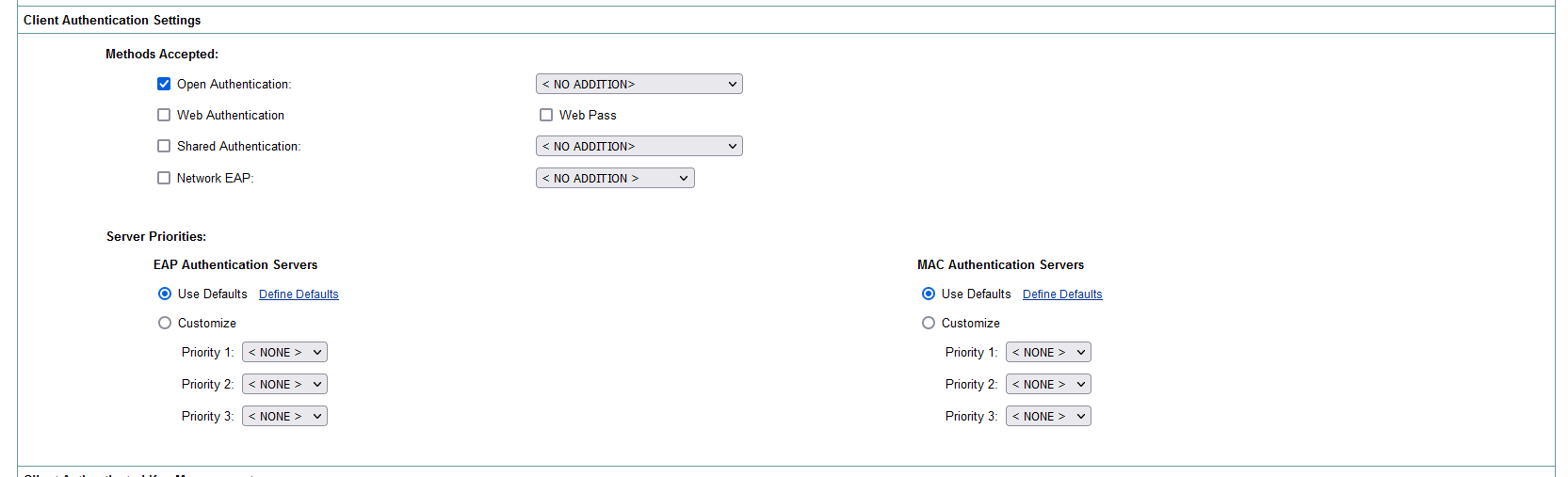
1. Using the SSID setup under “Network Configuration”, configure a new SSID with WPA2-PSK. The VLAN ID should in best practice correspond to the SSID value. For our purposes, SSID 1 was on VLAN 10, SSID 2 on VLAN 20, etc.



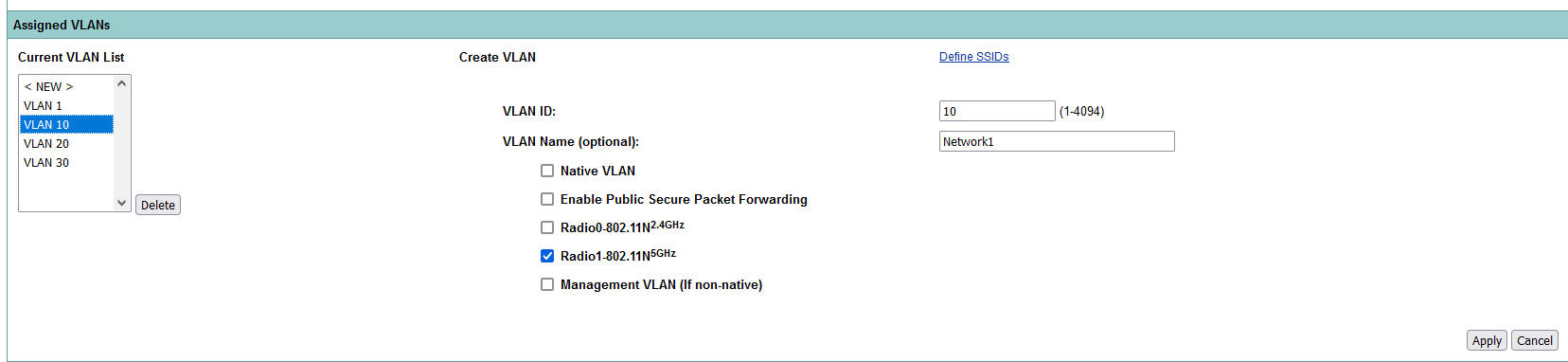
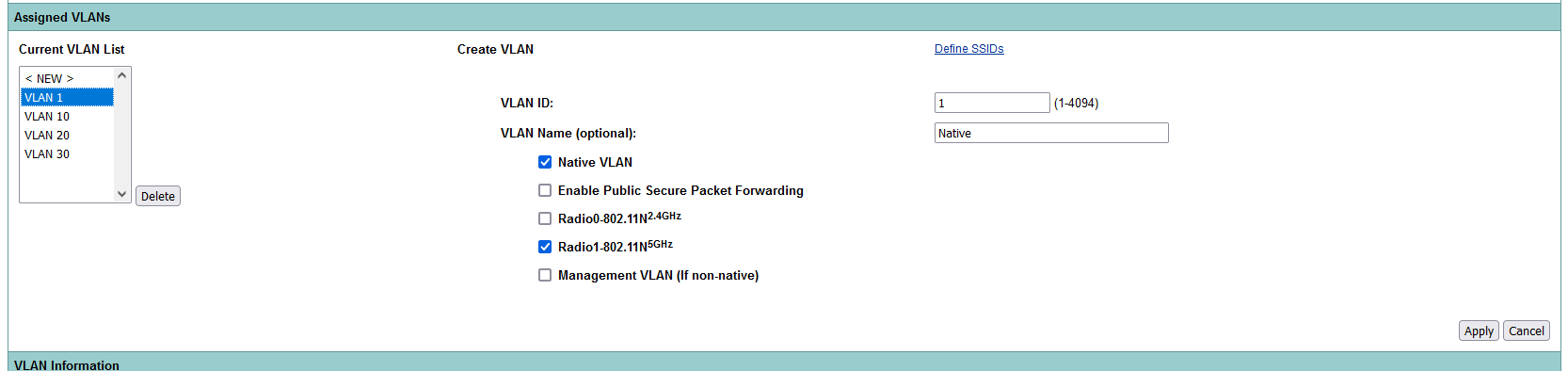
1. Navigate to “Security>SSID Manager” and select the newly created SSID. Ensure the SSID is on Radio1-5Ghz.



1. Ensure the Authentication settings is set to none. This is only used for RADIUS servers. Verify WPA2 is configured and that Guest Mode is on. This allows clients to connect to the SSID by broadcast. Furthermore, set Radio1 interface to be Multiple BSSID, allowing more than one SSID to be broadcasted.



1. Create a corresponding VLAN for your new SSID. Ensure that there is already a Native VLAN in order for the access point and switch to communicate.



1. Setup the VLAN on the switch side. Console into the switch through PuTTY and enter the interface of the port connected to the access point. Then, set the port as a trunk using:

*switchport trunk encapsulation dot1q*

*switchport mode trunk*

1. Now enter the router that the switch connects to. Enter the sub-interface corresponding to the VLAN on the interface connecting to the switch. Set the VLAN on this sub-interface.

*interface GigabitEthernet0/0/1.10*

*encapsulation dot1Q 10*

*ip address 10.0.10.1 255.255.255.0*

1. Finally, ensure that the native VLAN is up by configuring the same process on the native VLAN interface, usually G0/0/1.1:

*interface GigabitEthernet0/0/1.1*

*encapsulation dot1Q 1 native*

*ip address 10.0.1.1 255.255.255.0*

1. Now that internet connectivity is open, we must assign addresses and DNS to each of the clients connecting on an SSID. To do so, we must use DHCP. Configure DHCP on the Switch as follows:

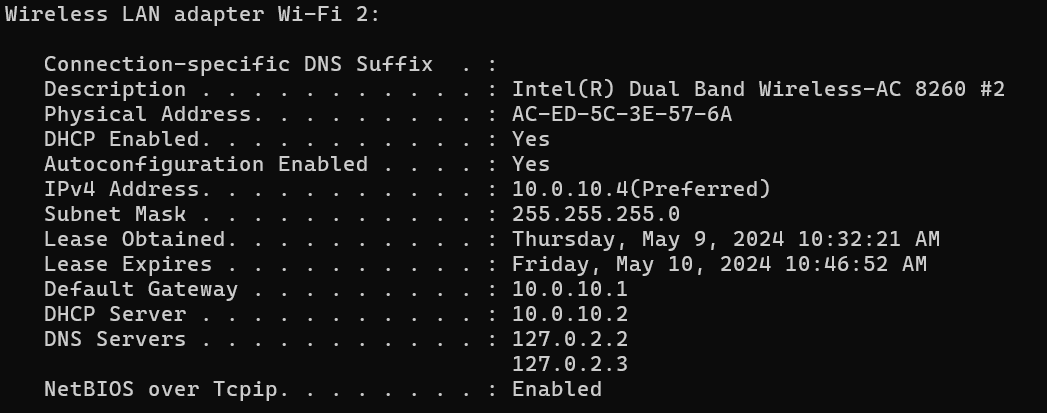
*ip dhcp pool vPool1*

*network 10.0.1.0 255.255.255.0*

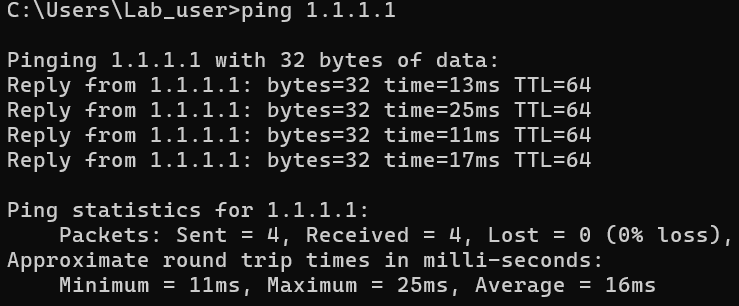
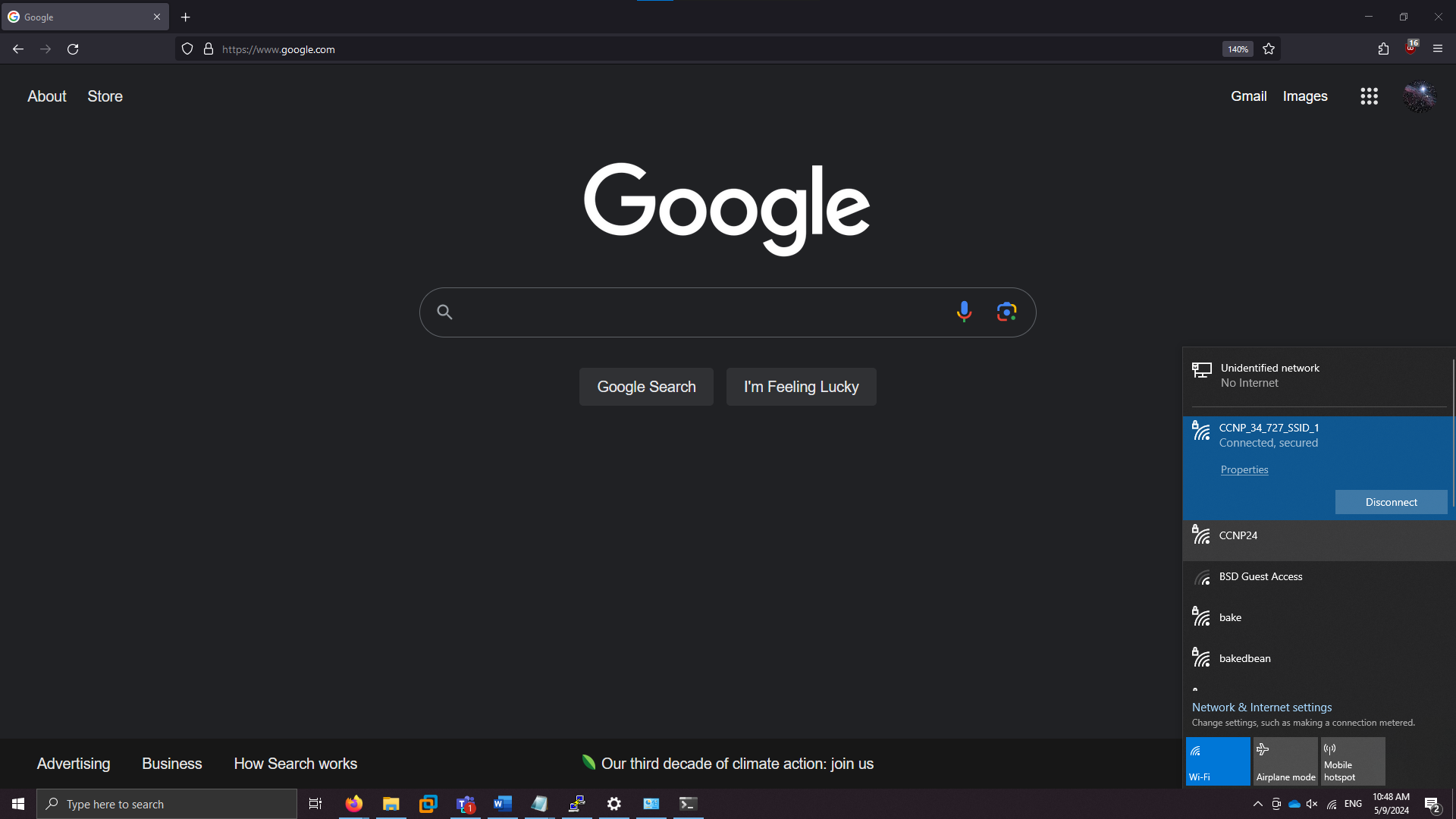
*default-router 10.0.1.1*

*dns-server 8.8.8.8*

1. Check for network connectivity by connecting to the configured SSID. It should show up as an available network on any device with WiFi. After connecting, verify that DHCP is correctly assigning addresses and the DNS server is working.

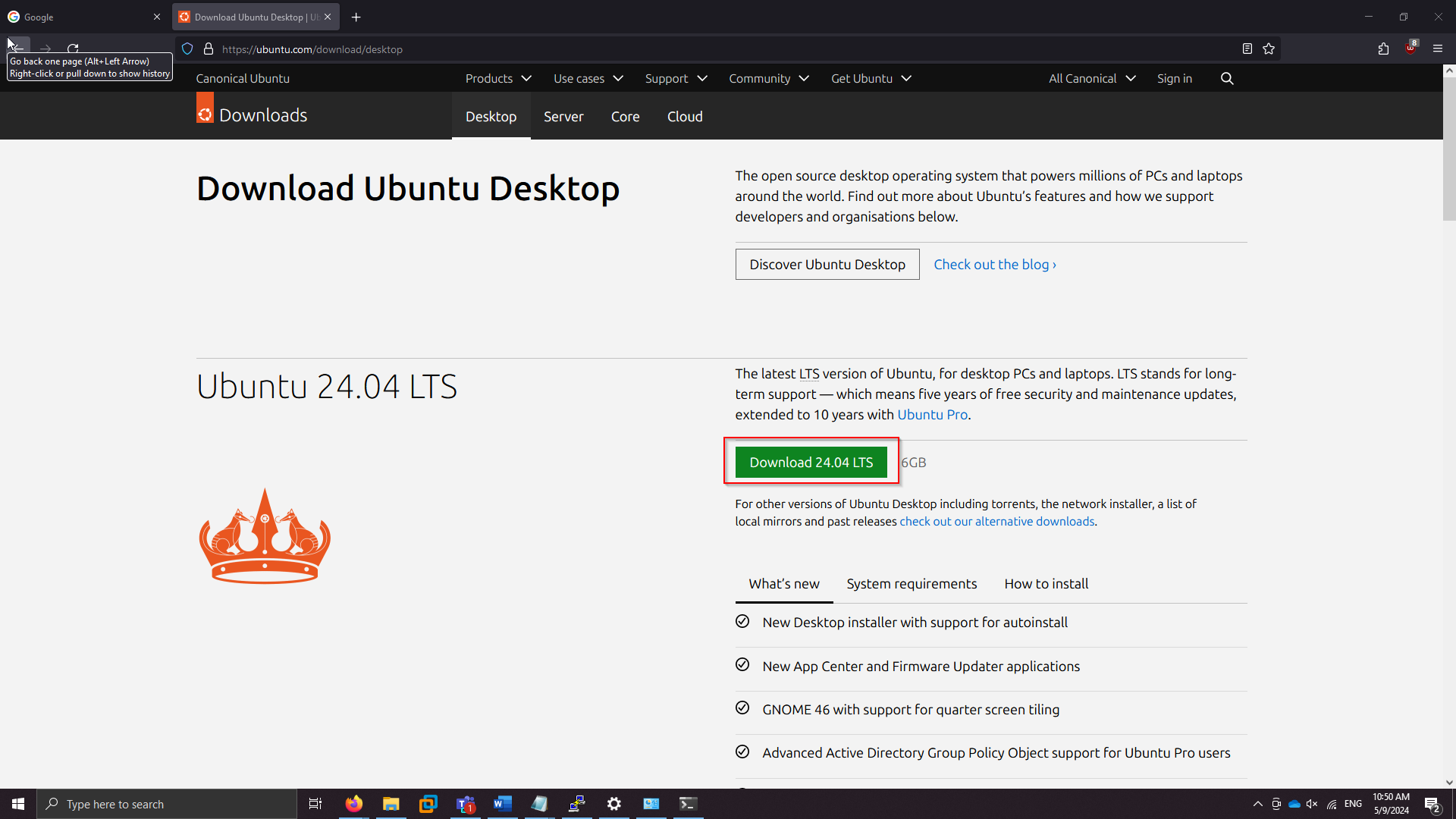


1. Finally, verify full internet connectivity by pinging any given address.

**AP WPA2 Enterprise SSID (RADIUS Configuration):**

1. Download Ubuntu desktop, or alternatively create an Ubuntu Docker container or operating system of your choice. The RADIUS server should function on virtualized software, but for our purposes we used a physical drive.



1. Complete basic setup and boot into Ubuntu, then download FreeRADIUS following the instructions on the website. For convenience and updates later, we used a package manager (apt) in order to install FreeRADIUS.

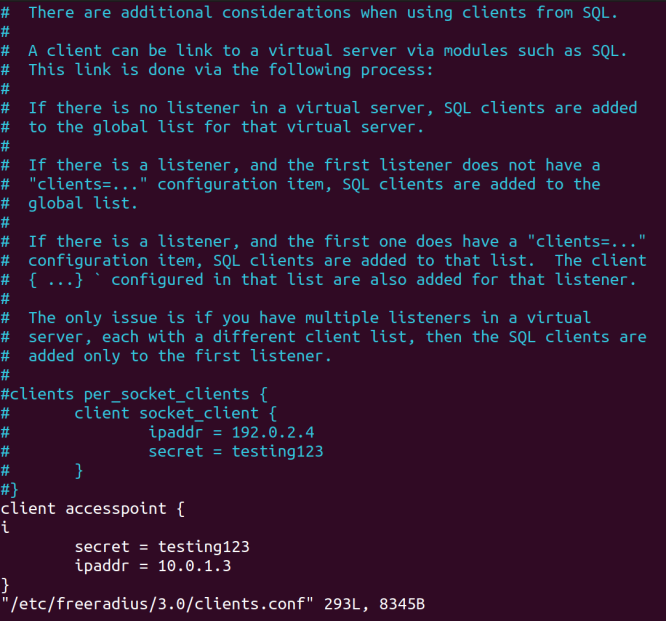
*sudo apt update*

*sudo apt install freeradius freeradius-utils*

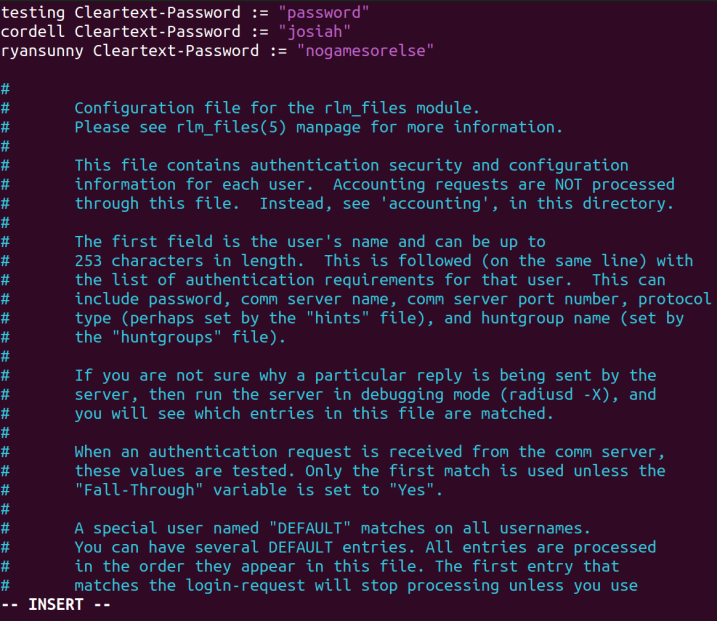
1. Temporarily switch into the root user to configure FreeRADIUS. From here, you can edit FreeRADIUS configuration files.

*sudo su*

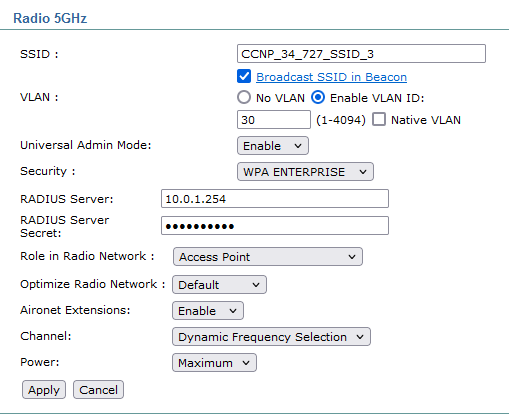
1. First, edit the *clients.conf*  file, which is responsible for setting the IP and shared secret of a client server utilizing the RADIUS service. In this case, we set the IP to be the access point, which will communicate with the RADIUS server in order to login users for SSID3.

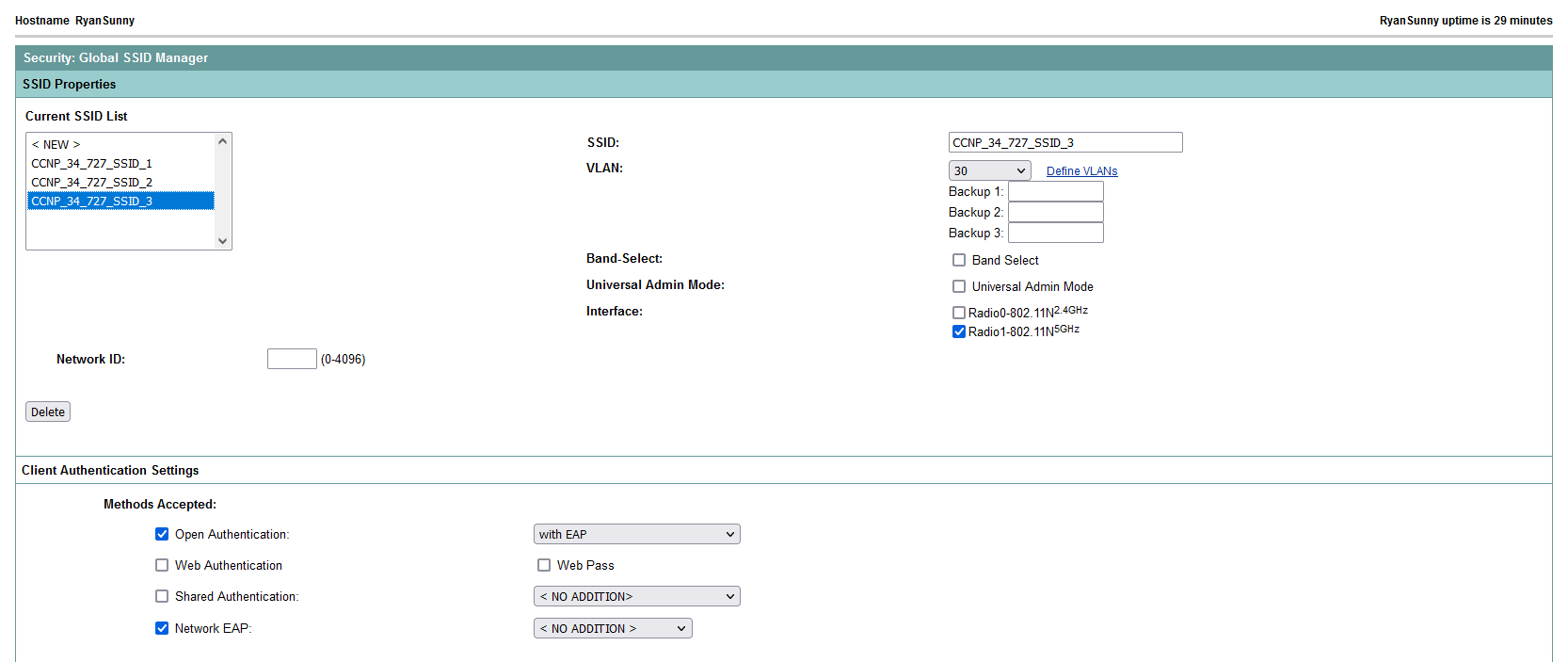
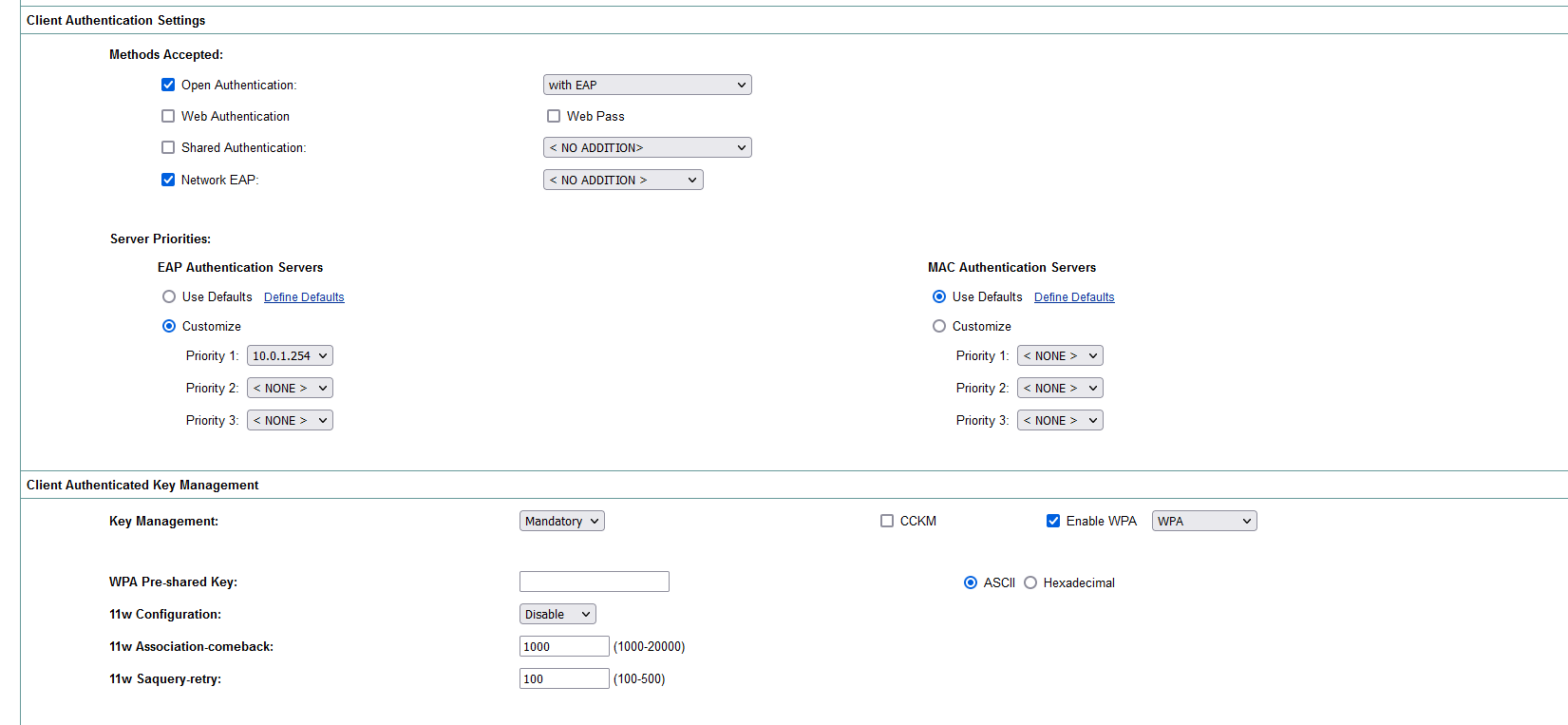
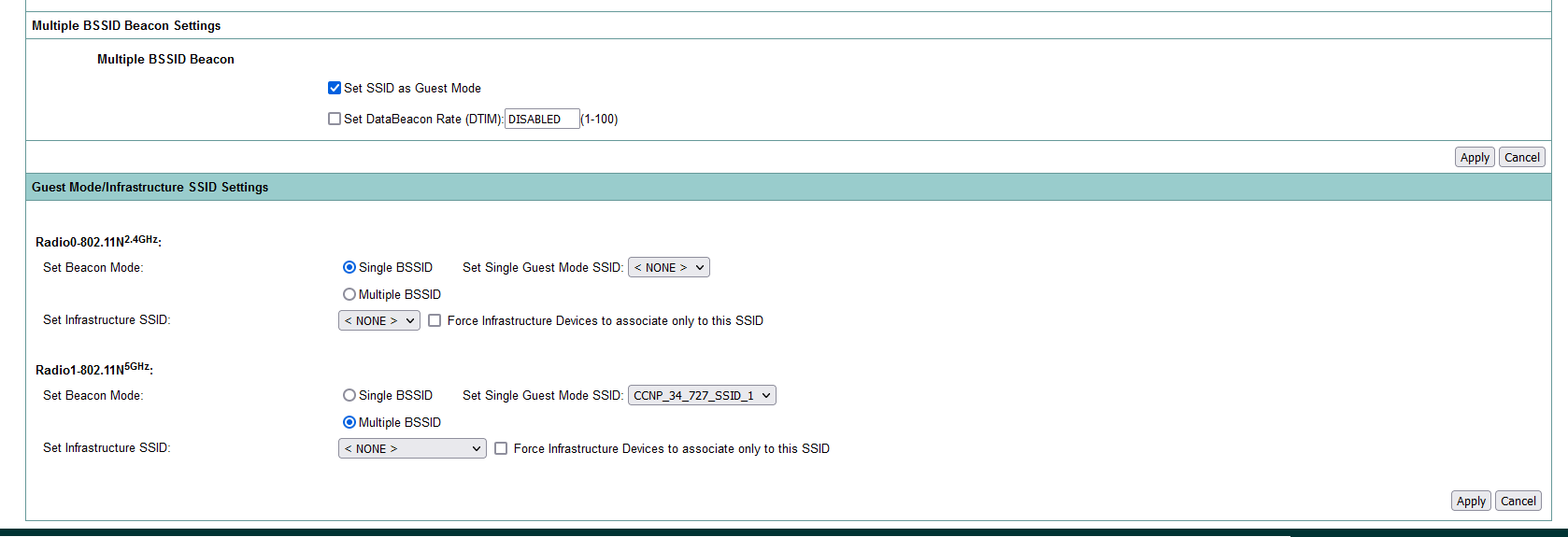
 

1. Next, edit the users file and follow the default template to add a new user. In this case, our user is “ryansunny” and the password is “nogamesorelse”. We will be using these credentials to sign in when accessing our SSID.

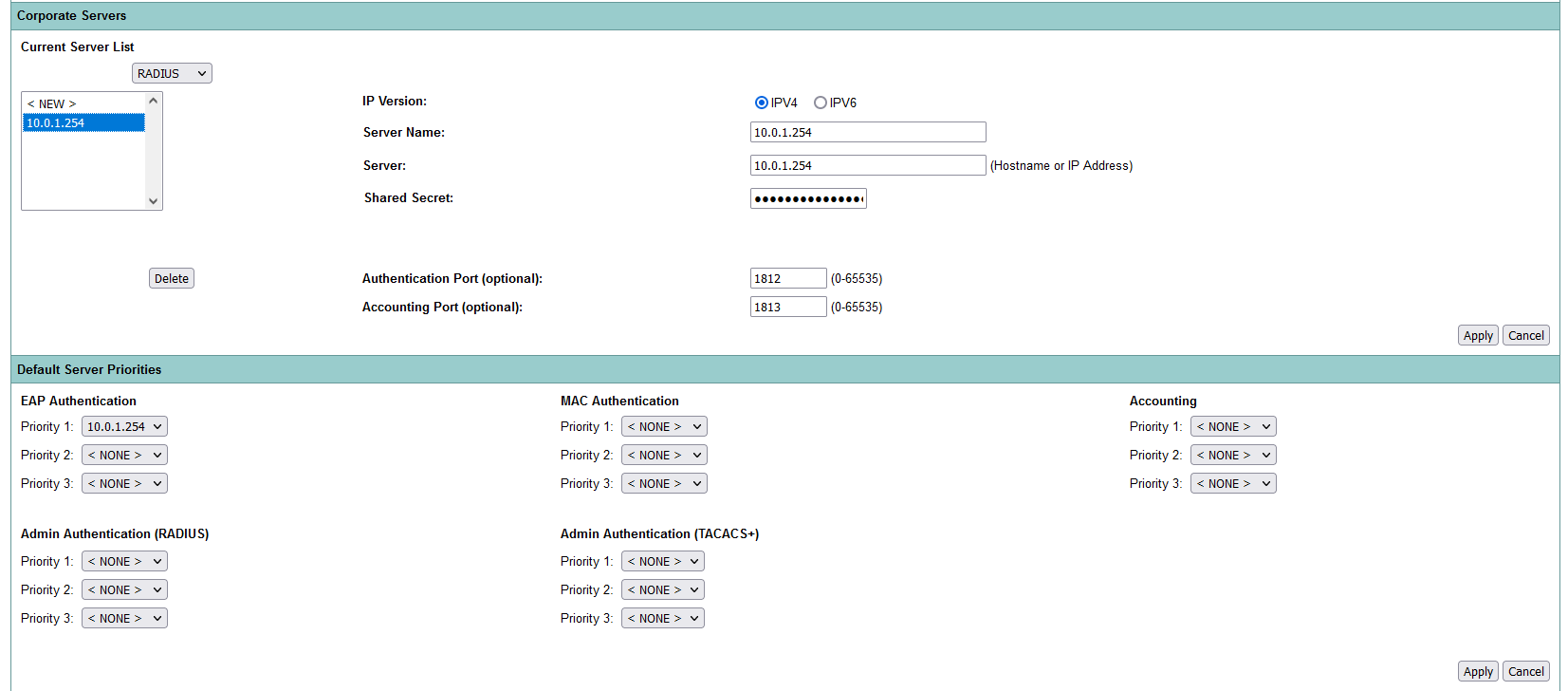
 

1. Configure the IP on Ubuntu to be an address within the subnet range. For our purposes, we configured the RADIUS server to be 10.0.1.254, in order to be consistent with our scheme.
2. Next, configure the AP to use the RADIUS server. On the interface, add a new SSID and configure it to use WPA Enterprise. Fill in the fields with the preconfigured values.



1. Next, verify the SSID is created correctly and enable WPA security. Also, enable Guest SSID so the network will be broadcast to nearby devices.   
2. Finally, go into Server Manager and add the RADIUS server as a new server on the AP. This allows the SSID to forward all AAA traffic to the RADIUS server, which in turn handles the requests. Ensure that the RADIUS server is set as the default EAP authentication server, and that Authentication is 1812 and Accounting is 1813. The shared secret continues to be the default one configured in FreeRADIUS, “testing123”.

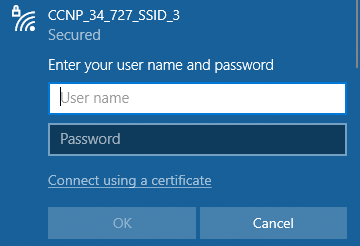
NOTE: The shared secret should be changed if the RADIUS server is going to be used.

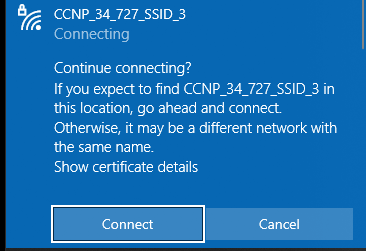


1. Confirm that the SSID is using the configured server, and then boot the FreeRADIUS process on the Ubuntu machine using:

*sudo freeradius -X*

1. Confirm that RADIUS is working as expected!





**AP Config:**

hostname RyanSunny

logging rate-limit console 9

enable secret 5 $1$Uca.$wqB9Ug2Lu/O2SPHQD/G34/

aaa new-model

aaa group server tacacs+ tac\_admin

aaa group server radius rad\_eap

server name 10.0.1.254

aaa group server radius rad\_mac

aaa group server radius rad\_acct

aaa group server radius rad\_admin

aaa group server radius rad\_pmip

aaa group server radius dummy

aaa group server radius rad\_eap3

server name 10.0.1.254

aaa authentication login eap\_methods group rad\_eap

aaa authentication login mac\_methods local

aaa authentication login eap\_methods3 group rad\_eap3

aaa authorization exec default local

aaa accounting network acct\_methods start-stop group rad\_acct

aaa session-id common

no ip source-route

no ip cef

dot11 pause-time 100

dot11 syslog

dot11 vlan-name Native vlan 1

dot11 vlan-name Network1 vlan 10

dot11 vlan-name Network2 vlan 20

dot11 vlan-name Network3 vlan 30

dot11 ssid CCNP\_34\_727\_SSID\_1

vlan 10

authentication open

authentication key-management wpa version 2

guest-mode

mbssid guest-mode

wpa-psk ascii 7 123A0C0411042F0D39282B

dot11 ssid CCNP\_34\_727\_SSID\_2

vlan 20

authentication open

authentication key-management wpa version 2

mbssid guest-mode

wpa-psk ascii 7 13261E0108032723382727

dot11 ssid CCNP\_34\_727\_SSID\_3

vlan 30

authentication open eap eap\_methods3

authentication network-eap eap\_methods3

authentication key-management wpa

mbssid guest-mode

no ipv6 cef

username Cisco password 7 01300F175804

bridge irb

interface Dot11Radio0

no ip address

shutdown

antenna gain 0

station-role root

bridge-group 1

bridge-group 1 subscriber-loop-control

bridge-group 1 spanning-disabled

bridge-group 1 block-unknown-source

no bridge-group 1 source-learning

no bridge-group 1 unicast-flooding

interface Dot11Radio1

no ip address

encryption vlan 1 mode ciphers aes-ccm

encryption vlan 2 mode ciphers aes-ccm

encryption vlan 10 mode ciphers aes-ccm

encryption vlan 20 mode ciphers aes-ccm

encryption key 2 size 128bit 7 FD719385EC0CEE0A351C749FB410 transmit-key

encryption vlan 30 mode ciphers aes-ccm tkip

ssid CCNP\_34\_727\_SSID\_1

ssid CCNP\_34\_727\_SSID\_2

ssid CCNP\_34\_727\_SSID\_3

antenna gain 0

peakdetect

dfs band 3 block

mbssid

channel dfs

station-role root

interface Dot11Radio1.1

encapsulation dot1Q 1 native

bridge-group 1

bridge-group 1 subscriber-loop-control

bridge-group 1 spanning-disabled

bridge-group 1 block-unknown-source

no bridge-group 1 source-learning

no bridge-group 1 unicast-flooding

interface Dot11Radio1.10

encapsulation dot1Q 10

ip address 10.0.10.3 255.255.255.0

bridge-group 10

bridge-group 10 subscriber-loop-control

bridge-group 10 spanning-disabled

bridge-group 10 block-unknown-source

no bridge-group 10 source-learning

no bridge-group 10 unicast-flooding

interface Dot11Radio1.11

interface Dot11Radio1.20

encapsulation dot1Q 20

bridge-group 20

bridge-group 20 subscriber-loop-control

bridge-group 20 spanning-disabled

bridge-group 20 block-unknown-source

no bridge-group 20 source-learning

no bridge-group 20 unicast-flooding

interface Dot11Radio1.30

encapsulation dot1Q 30

bridge-group 30

bridge-group 30 subscriber-loop-control

bridge-group 30 spanning-disabled

bridge-group 30 block-unknown-source

no bridge-group 30 source-learning

no bridge-group 30 unicast-flooding

interface Dot11Radio1.301

interface GigabitEthernet0

no ip address

duplex auto

speed auto

interface GigabitEthernet0.1

encapsulation dot1Q 1 native

bridge-group 1

bridge-group 1 spanning-disabled

no bridge-group 1 source-learning

interface GigabitEthernet0.10

encapsulation dot1Q 10

bridge-group 10

bridge-group 10 spanning-disabled

no bridge-group 10 source-learning

interface GigabitEthernet0.11

interface GigabitEthernet0.20

encapsulation dot1Q 20

bridge-group 20

bridge-group 20 spanning-disabled

no bridge-group 20 source-learning

interface GigabitEthernet0.30

encapsulation dot1Q 30

bridge-group 30

bridge-group 30 spanning-disabled

no bridge-group 30 source-learning

interface GigabitEthernet0.301

interface BVI1

mac-address 0007.7df5.4710

ip address 10.0.1.3 255.255.255.0

ipv6 address dhcp

ipv6 address autoconfig

ipv6 enable

ip forward-protocol nd

ip http server

no ip http secure-server

ip http help-path http://www.cisco.com/warp/public/779/smbiz/prodconfig/help/eag

ip radius source-interface BVI1

radius-server attribute 32 include-in-access-req format %h

radius server 10.0.1.254

address ipv4 10.0.1.254 auth-port 1812 acct-port 1813

key 7 111D1C16031B050B557878

bridge 1 route ip

line con 0

line vty 0 4

transport input all

end

**R1 Config:**

hostname R1

boot-start-marker

boot-end-marker

vrf definition Mgmt-intf

address-family ipv4

exit-address-family

address-family ipv6

exit-address-family

no aaa new-model

subscriber templating

multilink bundle-name authenticated

license udi pid ISR4321/K9 sn FDO214420QQ

license accept end user agreement

license boot level securityk9

spanning-tree extend system-id

redundancy

mode none

vlan internal allocation policy ascending

interface GigabitEthernet0/0/0

ip address dhcp

ip nat outside

negotiation auto

interface GigabitEthernet0/0/1

no ip address

ip nat inside

negotiation auto

interface GigabitEthernet0/0/1.1

encapsulation dot1Q 1 native

ip address 10.0.1.1 255.255.255.0

ip nat inside

interface GigabitEthernet0/0/1.10

encapsulation dot1Q 10

ip address 10.0.10.1 255.255.255.0

ip nat inside

interface GigabitEthernet0/0/1.20

encapsulation dot1Q 20

ip address 10.0.20.1 255.255.255.0

ip nat inside

interface GigabitEthernet0/0/1.30

encapsulation dot1Q 30

ip address 10.0.30.1 255.255.255.0

ip nat inside

interface Serial0/1/0

no ip address

shutdown

interface Serial0/1/1

no ip address

shutdown

interface GigabitEthernet0

vrf forwarding Mgmt-intf

no ip address

shutdown

negotiation auto

interface Vlan1

no ip address

shutdown

ip nat inside source list 10 interface GigabitEthernet0/0/0 overload

ip forward-protocol nd

no ip http server

no ip http secure-server

ip tftp source-interface GigabitEthernet0/0/0

access-list 10 permit 10.0.0.0 0.255.255.255

control-plane

line con 0

exec-timeout 0 0

stopbits 1

line aux 0

stopbits 1

line vty 0 4

login

ntp server pool.ntp.org

end

**S1 Config:**

hostname S1

boot-start-marker

boot-end-marker

no aaa new-model

system mtu routing 1500

authentication mac-move permit

ip subnet-zero

ip dhcp excluded-address 10.0.1.1

ip dhcp excluded-address 10.0.10.1

ip dhcp excluded-address 10.0.20.1

ip dhcp excluded-address 10.0.30.1

ip dhcp excluded-address 10.0.1.2

ip dhcp excluded-address 10.0.1.3

ip dhcp excluded-address 10.0.10.2

ip dhcp excluded-address 10.0.10.3

ip dhcp excluded-address 10.0.20.2

ip dhcp excluded-address 10.0.20.3

ip dhcp excluded-address 10.0.30.2

ip dhcp excluded-address 10.0.30.3

ip dhcp pool vPool1

network 10.0.1.0 255.255.255.0

default-router 10.0.1.1

dns-server 8.8.8.8

ip dhcp pool vPool10

network 10.0.10.0 255.255.255.0

default-router 10.0.10.1

dns-server 8.8.8.8

ip dhcp pool vPool20

network 10.0.20.0 255.255.255.0

default-router 10.0.20.1

dns-server 8.8.8.8

ip dhcp pool vPool30

network 10.0.30.0 255.255.255.0

default-router 10.0.30.1

dns-server 8.8.8.8

spanning-tree mode pvst

spanning-tree etherchannel guard misconfig

spanning-tree extend system-id

vlan internal allocation policy ascending

interface FastEthernet0/1

switchport trunk encapsulation dot1q

switchport mode trunk

interface FastEthernet0/2

switchport trunk encapsulation dot1q

switchport mode trunk

interface FastEthernet0/3

interface FastEthernet0/4

interface FastEthernet0/5

interface FastEthernet0/6

interface FastEthernet0/7

interface FastEthernet0/8

interface FastEthernet0/9

interface FastEthernet0/10

interface FastEthernet0/11

interface FastEthernet0/12

interface FastEthernet0/13

interface FastEthernet0/14

interface FastEthernet0/15

interface FastEthernet0/16

interface FastEthernet0/17

interface FastEthernet0/18

interface FastEthernet0/19

interface FastEthernet0/20

interface FastEthernet0/21

interface FastEthernet0/22

interface FastEthernet0/23

interface FastEthernet0/24

interface GigabitEthernet0/1

interface GigabitEthernet0/2

interface Vlan1

ip address 10.0.1.2 255.255.255.0

interface Vlan10

ip address 10.0.10.2 255.255.255.0

interface Vlan20

ip address 10.0.20.2 255.255.255.0

interface Vlan30

ip address 10.0.30.2 255.255.255.0

ip default-gateway 10.0.1.1

ip classless

ip http server

ip sla enable reaction-alerts

line con 0

login

line vty 0 4

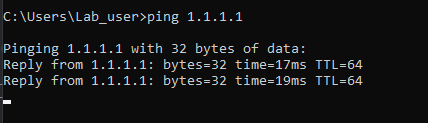
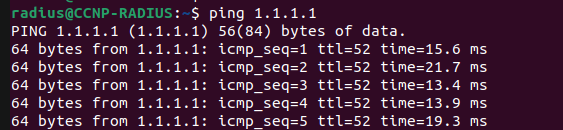
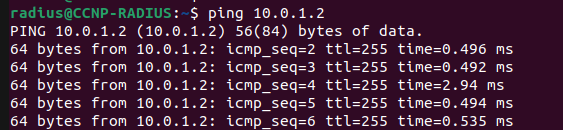
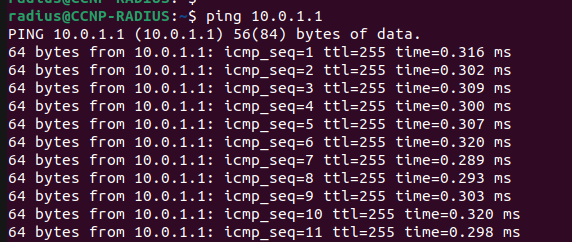
login

line vty 5 15

login

end

**Pings and Traceroute:**



**Radius Handshake:**

Ready to process requests

(10) Received Access-Request Id 17 from 10.0.1.3:1645 to 10.0.1.254:1812 length 227

(10) User-Name = "testing"

(10) Framed-MTU = 1400

(10) Called-Station-Id = "d0c2.822e.4392:CCNP\_34\_727\_SSID\_3"

(10) Calling-Station-Id = "14ab.c53c.0807"

(10) Cisco-AVPair = "ssid=CCNP\_34\_727\_SSID\_3"

(10) Service-Type = Login-User

(10) Cisco-AVPair = "service-type=Login"

(10) Message-Authenticator = 0x6cf676848da5e109f7cb8e56615e7760

(10) EAP-Message = 0x020700061900

(10) NAS-Port-Type = Wireless-802.11

(10) NAS-Port = 277

(10) NAS-Port-Id = "277"

(10) State = 0x1dfcd08519fbc9db5e3d878bc6e70d5e

(10) NAS-IP-Address = 10.0.1.3

(10) NAS-Identifier = "RyanSunny"

(10) Restoring &session-state

(10) &session-state:Framed-MTU = 994

(10) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(10) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(10) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(10) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(10) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(10) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(10) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(10) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(10) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(10) &session-state:TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(10) &session-state:TLS-Session-Version = "TLS 1.2"

(10) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default

(10) authorize {

(10) policy filter\_username {

(10) if (&User-Name) {

(10) if (&User-Name) -> TRUE

(10) if (&User-Name) {

(10) if (&User-Name =~ / /) {

(10) if (&User-Name =~ / /) -> FALSE

(10) if (&User-Name =~ /@[^@]\*@/ ) {

(10) if (&User-Name =~ /@[^@]\*@/ ) -> FALSE

(10) if (&User-Name =~ /\.\./ ) {

(10) if (&User-Name =~ /\.\./ ) -> FALSE

(10) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) {

(10) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) -> FALSE

(10) if (&User-Name =~ /\.$/) {

(10) if (&User-Name =~ /\.$/) -> FALSE

(10) if (&User-Name =~ /@\./) {

(10) if (&User-Name =~ /@\./) -> FALSE

(10) } # if (&User-Name) = notfound

(10) } # policy filter\_username = notfound

(10) [preprocess] = ok

(10) [chap] = noop

(10) [mschap] = noop

(10) [digest] = noop

(10) suffix: Checking for suffix after "@"

(10) suffix: No '@' in User-Name = "testing", looking up realm NULL

(10) suffix: No such realm "NULL"

(10) [suffix] = noop

(10) eap: Peer sent EAP Response (code 2) ID 7 length 6

(10) eap: Continuing tunnel setup

(10) [eap] = ok

(10) } # authorize = ok

(10) Found Auth-Type = eap

(10) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(10) authenticate {

(10) eap: Expiring EAP session with state 0x0275a0c70672b940

(10) eap: Finished EAP session with state 0x1dfcd08519fbc9db

(10) eap: Previous EAP request found for state 0x1dfcd08519fbc9db, released from the list

(10) eap: Peer sent packet with method EAP PEAP (25)

(10) eap: Calling submodule eap\_peap to process data

(10) eap\_peap: (TLS) Peer ACKed our handshake fragment. handshake is finished

(10) eap\_peap: Session established. Decoding tunneled attributes

(10) eap\_peap: PEAP state TUNNEL ESTABLISHED

(10) eap: Sending EAP Request (code 1) ID 8 length 40

(10) eap: EAP session adding &reply:State = 0x1dfcd08518f4c9db

(10) [eap] = handled

(10) } # authenticate = handled

(10) Using Post-Auth-Type Challenge

(10) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(10) Challenge { ... } # empty sub-section is ignored

(10) session-state: Saving cached attributes

(10) Framed-MTU = 994

(10) TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(10) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(10) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(10) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(10) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(10) TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(10) TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(10) TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(10) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(10) TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(10) TLS-Session-Version = "TLS 1.2"

(10) Sent Access-Challenge Id 17 from 10.0.1.254:1812 to 10.0.1.3:1645 length 98

(10) EAP-Message = 0x010800281900170303001de0c487f4aa2d57ae38b8a190800558dd4d2ecf52e8ed6f30a489c5be57

(10) Message-Authenticator = 0x00000000000000000000000000000000

(10) State = 0x1dfcd08518f4c9db5e3d878bc6e70d5e

(10) Finished request

Waking up in 4.9 seconds.

(11) Received Access-Request Id 18 from 10.0.1.3:1645 to 10.0.1.254:1812 length 264

(11) User-Name = "testing"

(11) Framed-MTU = 1400

(11) Called-Station-Id = "d0c2.822e.4392:CCNP\_34\_727\_SSID\_3"

(11) Calling-Station-Id = "14ab.c53c.0807"

(11) Cisco-AVPair = "ssid=CCNP\_34\_727\_SSID\_3"

(11) Service-Type = Login-User

(11) Cisco-AVPair = "service-type=Login"

(11) Message-Authenticator = 0xc164bbae234660c9bf262f5f073aedac

(11) EAP-Message = 0x0208002b190017030300200000000000000001660e925184947aac2e3e23b831ac450a37354ab1eeda6670

(11) NAS-Port-Type = Wireless-802.11

(11) NAS-Port = 277

(11) NAS-Port-Id = "277"

(11) State = 0x1dfcd08518f4c9db5e3d878bc6e70d5e

(11) NAS-IP-Address = 10.0.1.3

(11) NAS-Identifier = "RyanSunny"

(11) Restoring &session-state

(11) &session-state:Framed-MTU = 994

(11) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(11) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(11) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(11) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(11) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(11) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(11) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(11) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(11) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(11) &session-state:TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(11) &session-state:TLS-Session-Version = "TLS 1.2"

(11) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default

(11) authorize {

(11) policy filter\_username {

(11) if (&User-Name) {

(11) if (&User-Name) -> TRUE

(11) if (&User-Name) {

(11) if (&User-Name =~ / /) {

(11) if (&User-Name =~ / /) -> FALSE

(11) if (&User-Name =~ /@[^@]\*@/ ) {

(11) if (&User-Name =~ /@[^@]\*@/ ) -> FALSE

(11) if (&User-Name =~ /\.\./ ) {

(11) if (&User-Name =~ /\.\./ ) -> FALSE

(11) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) {

(11) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) -> FALSE

(11) if (&User-Name =~ /\.$/) {

(11) if (&User-Name =~ /\.$/) -> FALSE

(11) if (&User-Name =~ /@\./) {

(11) if (&User-Name =~ /@\./) -> FALSE

(11) } # if (&User-Name) = notfound

(11) } # policy filter\_username = notfound

(11) [preprocess] = ok

(11) [chap] = noop

(11) [mschap] = noop

(11) [digest] = noop

(11) suffix: Checking for suffix after "@"

(11) suffix: No '@' in User-Name = "testing", looking up realm NULL

(11) suffix: No such realm "NULL"

(11) [suffix] = noop

(11) eap: Peer sent EAP Response (code 2) ID 8 length 43

(11) eap: Continuing tunnel setup

(11) [eap] = ok

(11) } # authorize = ok

(11) Found Auth-Type = eap

(11) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(11) authenticate {

(11) eap: Expiring EAP session with state 0x0275a0c70672b940

(11) eap: Finished EAP session with state 0x1dfcd08518f4c9db

(11) eap: Previous EAP request found for state 0x1dfcd08518f4c9db, released from the list

(11) eap: Peer sent packet with method EAP PEAP (25)

(11) eap: Calling submodule eap\_peap to process data

(11) eap\_peap: (TLS) EAP Done initial handshake

(11) eap\_peap: Session established. Decoding tunneled attributes

(11) eap\_peap: PEAP state WAITING FOR INNER IDENTITY

(11) eap\_peap: Identity - testing

(11) eap\_peap: Got inner identity 'testing'

(11) eap\_peap: Setting default EAP type for tunneled EAP session

(11) eap\_peap: Got tunneled request

(11) eap\_peap: EAP-Message = 0x0208000c0174657374696e67

(11) eap\_peap: Setting User-Name to testing

(11) eap\_peap: Sending tunneled request to inner-tunnel

(11) eap\_peap: EAP-Message = 0x0208000c0174657374696e67

(11) eap\_peap: FreeRADIUS-Proxied-To = 127.0.0.1

(11) eap\_peap: User-Name = "testing"

(11) Virtual server inner-tunnel received request

(11) EAP-Message = 0x0208000c0174657374696e67

(11) FreeRADIUS-Proxied-To = 127.0.0.1

(11) User-Name = "testing"

(11) WARNING: Outer and inner identities are the same. User privacy is compromised.

(11) server inner-tunnel {

(11) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/inner-tunnel

(11) authorize {

(11) policy filter\_username {

(11) if (&User-Name) {

(11) if (&User-Name) -> TRUE

(11) if (&User-Name) {

(11) if (&User-Name =~ / /) {

(11) if (&User-Name =~ / /) -> FALSE

(11) if (&User-Name =~ /@[^@]\*@/ ) {

(11) if (&User-Name =~ /@[^@]\*@/ ) -> FALSE

(11) if (&User-Name =~ /\.\./ ) {

(11) if (&User-Name =~ /\.\./ ) -> FALSE

(11) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) {

(11) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) -> FALSE

(11) if (&User-Name =~ /\.$/) {

(11) if (&User-Name =~ /\.$/) -> FALSE

(11) if (&User-Name =~ /@\./) {

(11) if (&User-Name =~ /@\./) -> FALSE

(11) } # if (&User-Name) = notfound

(11) } # policy filter\_username = notfound

(11) [chap] = noop

(11) [mschap] = noop

(11) suffix: Checking for suffix after "@"

(11) suffix: No '@' in User-Name = "testing", looking up realm NULL

(11) suffix: No such realm "NULL"

(11) [suffix] = noop

(11) update control {

(11) &Proxy-To-Realm := LOCAL

(11) } # update control = noop

(11) eap: Peer sent EAP Response (code 2) ID 8 length 12

(11) eap: EAP-Identity reply, returning 'ok' so we can short-circuit the rest of authorize

(11) [eap] = ok

(11) } # authorize = ok

(11) Found Auth-Type = eap

(11) # Executing group from file /etc/freeradius/3.0/sites-enabled/inner-tunnel

(11) authenticate {

(11) eap: Peer sent packet with method EAP Identity (1)

(11) eap: Calling submodule eap\_mschapv2 to process data

(11) eap\_mschapv2: Issuing Challenge

(11) eap: Sending EAP Request (code 1) ID 9 length 43

(11) eap: EAP session adding &reply:State = 0x733052da733948b5

(11) [eap] = handled

(11) } # authenticate = handled

(11) } # server inner-tunnel

(11) Virtual server sending reply

(11) EAP-Message = 0x0109002b1a0109002610ca1234749e59c0acac8f17fa5fc80861667265657261646975732d332e302e3236

(11) Message-Authenticator = 0x00000000000000000000000000000000

(11) State = 0x733052da733948b52b4f2886441d1cfe

(11) eap\_peap: Got tunneled reply code 11

(11) eap\_peap: EAP-Message = 0x0109002b1a0109002610ca1234749e59c0acac8f17fa5fc80861667265657261646975732d332e302e3236

(11) eap\_peap: Message-Authenticator = 0x00000000000000000000000000000000

(11) eap\_peap: State = 0x733052da733948b52b4f2886441d1cfe

(11) eap\_peap: Got tunneled reply RADIUS code 11

(11) eap\_peap: EAP-Message = 0x0109002b1a0109002610ca1234749e59c0acac8f17fa5fc80861667265657261646975732d332e302e3236

(11) eap\_peap: Message-Authenticator = 0x00000000000000000000000000000000

(11) eap\_peap: State = 0x733052da733948b52b4f2886441d1cfe

(11) eap\_peap: Got tunneled Access-Challenge

(11) eap: Sending EAP Request (code 1) ID 9 length 74

(11) eap: EAP session adding &reply:State = 0x1dfcd0851bf5c9db

(11) [eap] = handled

(11) } # authenticate = handled

(11) Using Post-Auth-Type Challenge

(11) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(11) Challenge { ... } # empty sub-section is ignored

(11) session-state: Saving cached attributes

(11) Framed-MTU = 994

(11) TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(11) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(11) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(11) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(11) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(11) TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(11) TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(11) TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(11) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(11) TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(11) TLS-Session-Version = "TLS 1.2"

(11) Sent Access-Challenge Id 18 from 10.0.1.254:1812 to 10.0.1.3:1645 length 132

(11) EAP-Message = 0x0109004a1900170303003fe0c487f4aa2d57af8d86fbd3b602c6d146381a04252eeb1f7bc7a8a247a268e90dd4fa6495bc216a6334a709a1ca504b426edcbae34bd2c985e673d876b143

(11) Message-Authenticator = 0x00000000000000000000000000000000

(11) State = 0x1dfcd0851bf5c9db5e3d878bc6e70d5e

(11) Finished request

Waking up in 4.9 seconds.

(12) Received Access-Request Id 19 from 10.0.1.3:1645 to 10.0.1.254:1812 length 318

(12) User-Name = "testing"

(12) Framed-MTU = 1400

(12) Called-Station-Id = "d0c2.822e.4392:CCNP\_34\_727\_SSID\_3"

(12) Calling-Station-Id = "14ab.c53c.0807"

(12) Cisco-AVPair = "ssid=CCNP\_34\_727\_SSID\_3"

(12) Service-Type = Login-User

(12) Cisco-AVPair = "service-type=Login"

(12) Message-Authenticator = 0x3dc2f227b77299b6c1ac95aeb9e7553d

(12) EAP-Message = 0x020900611900170303005600000000000000029e9c2e15379526228c4960fcc0f9551ef02d5fb1d46487d335bbb4084c9a9166052b70e54d4971b3cbc5e14f4de3b973452b509c53fa8abd2d10b423a49657ef2fc3ea23854dfa06ae36a5f04373

(12) NAS-Port-Type = Wireless-802.11

(12) NAS-Port = 277

(12) NAS-Port-Id = "277"

(12) State = 0x1dfcd0851bf5c9db5e3d878bc6e70d5e

(12) NAS-IP-Address = 10.0.1.3

(12) NAS-Identifier = "RyanSunny"

(12) Restoring &session-state

(12) &session-state:Framed-MTU = 994

(12) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(12) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(12) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(12) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(12) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(12) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(12) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(12) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(12) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(12) &session-state:TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(12) &session-state:TLS-Session-Version = "TLS 1.2"

(12) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default

(12) authorize {

(12) policy filter\_username {

(12) if (&User-Name) {

(12) if (&User-Name) -> TRUE

(12) if (&User-Name) {

(12) if (&User-Name =~ / /) {

(12) if (&User-Name =~ / /) -> FALSE

(12) if (&User-Name =~ /@[^@]\*@/ ) {

(12) if (&User-Name =~ /@[^@]\*@/ ) -> FALSE

(12) if (&User-Name =~ /\.\./ ) {

(12) if (&User-Name =~ /\.\./ ) -> FALSE

(12) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) {

(12) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) -> FALSE

(12) if (&User-Name =~ /\.$/) {

(12) if (&User-Name =~ /\.$/) -> FALSE

(12) if (&User-Name =~ /@\./) {

(12) if (&User-Name =~ /@\./) -> FALSE

(12) } # if (&User-Name) = notfound

(12) } # policy filter\_username = notfound

(12) [preprocess] = ok

(12) [chap] = noop

(12) [mschap] = noop

(12) [digest] = noop

(12) suffix: Checking for suffix after "@"

(12) suffix: No '@' in User-Name = "testing", looking up realm NULL

(12) suffix: No such realm "NULL"

(12) [suffix] = noop

(12) eap: Peer sent EAP Response (code 2) ID 9 length 97

(12) eap: Continuing tunnel setup

(12) [eap] = ok

(12) } # authorize = ok

(12) Found Auth-Type = eap

(12) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(12) authenticate {

(12) eap: Expiring EAP session with state 0x0275a0c70672b940

(12) eap: Finished EAP session with state 0x1dfcd0851bf5c9db

(12) eap: Previous EAP request found for state 0x1dfcd0851bf5c9db, released from the list

(12) eap: Peer sent packet with method EAP PEAP (25)

(12) eap: Calling submodule eap\_peap to process data

(12) eap\_peap: (TLS) EAP Done initial handshake

(12) eap\_peap: Session established. Decoding tunneled attributes

(12) eap\_peap: PEAP state phase2

(12) eap\_peap: EAP method MSCHAPv2 (26)

(12) eap\_peap: Got tunneled request

(12) eap\_peap: EAP-Message = 0x020900421a0209003d311fcc349d2063c23ba8c84813ebcd4a6e0000000000000000fe131f55fa15f12f9ef7df1e17cb976347f94361dc9cfef90074657374696e67

(12) eap\_peap: Setting User-Name to testing

(12) eap\_peap: Sending tunneled request to inner-tunnel

(12) eap\_peap: EAP-Message = 0x020900421a0209003d311fcc349d2063c23ba8c84813ebcd4a6e0000000000000000fe131f55fa15f12f9ef7df1e17cb976347f94361dc9cfef90074657374696e67

(12) eap\_peap: FreeRADIUS-Proxied-To = 127.0.0.1

(12) eap\_peap: User-Name = "testing"

(12) eap\_peap: State = 0x733052da733948b52b4f2886441d1cfe

(12) Virtual server inner-tunnel received request

(12) EAP-Message = 0x020900421a0209003d311fcc349d2063c23ba8c84813ebcd4a6e0000000000000000fe131f55fa15f12f9ef7df1e17cb976347f94361dc9cfef90074657374696e67

(12) FreeRADIUS-Proxied-To = 127.0.0.1

(12) User-Name = "testing"

(12) State = 0x733052da733948b52b4f2886441d1cfe

(12) WARNING: Outer and inner identities are the same. User privacy is compromised.

(12) server inner-tunnel {

(12) session-state: No cached attributes

(12) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/inner-tunnel

(12) authorize {

(12) policy filter\_username {

(12) if (&User-Name) {

(12) if (&User-Name) -> TRUE

(12) if (&User-Name) {

(12) if (&User-Name =~ / /) {

(12) if (&User-Name =~ / /) -> FALSE

(12) if (&User-Name =~ /@[^@]\*@/ ) {

(12) if (&User-Name =~ /@[^@]\*@/ ) -> FALSE

(12) if (&User-Name =~ /\.\./ ) {

(12) if (&User-Name =~ /\.\./ ) -> FALSE

(12) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) {

(12) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) -> FALSE

(12) if (&User-Name =~ /\.$/) {

(12) if (&User-Name =~ /\.$/) -> FALSE

(12) if (&User-Name =~ /@\./) {

(12) if (&User-Name =~ /@\./) -> FALSE

(12) } # if (&User-Name) = notfound

(12) } # policy filter\_username = notfound

(12) [chap] = noop

(12) [mschap] = noop

(12) suffix: Checking for suffix after "@"

(12) suffix: No '@' in User-Name = "testing", looking up realm NULL

(12) suffix: No such realm "NULL"

(12) [suffix] = noop

(12) update control {

(12) &Proxy-To-Realm := LOCAL

(12) } # update control = noop

(12) eap: Peer sent EAP Response (code 2) ID 9 length 66

(12) eap: No EAP Start, assuming it's an on-going EAP conversation

(12) [eap] = updated

(12) files: users: Matched entry testing at line 1

(12) [files] = ok

(12) [expiration] = noop

(12) [logintime] = noop

(12) pap: WARNING: Auth-Type already set. Not setting to PAP

(12) [pap] = noop

(12) } # authorize = updated

(12) Found Auth-Type = eap

(12) # Executing group from file /etc/freeradius/3.0/sites-enabled/inner-tunnel

(12) authenticate {

(12) eap: Expiring EAP session with state 0x0275a0c70672b940

(12) eap: Finished EAP session with state 0x733052da733948b5

(12) eap: Previous EAP request found for state 0x733052da733948b5, released from the list

(12) eap: Peer sent packet with method EAP MSCHAPv2 (26)

(12) eap: Calling submodule eap\_mschapv2 to process data

(12) eap\_mschapv2: # Executing group from file /etc/freeradius/3.0/sites-enabled/inner-tunnel

(12) eap\_mschapv2: authenticate {

(12) mschap: Found Cleartext-Password, hashing to create NT-Password

(12) mschap: Creating challenge hash with username: testing

(12) mschap: Client is using MS-CHAPv2

(12) mschap: Adding MS-CHAPv2 MPPE keys

(12) eap\_mschapv2: [mschap] = ok

(12) eap\_mschapv2: } # authenticate = ok

(12) eap\_mschapv2: MSCHAP Success

(12) eap: Sending EAP Request (code 1) ID 10 length 51

(12) eap: EAP session adding &reply:State = 0x733052da723a48b5

(12) [eap] = handled

(12) } # authenticate = handled

(12) } # server inner-tunnel

(12) Virtual server sending reply

(12) EAP-Message = 0x010a00331a0309002e533d38433933343542353833373344334434423337314141424345334434373442313132303033373933

(12) Message-Authenticator = 0x00000000000000000000000000000000

(12) State = 0x733052da723a48b52b4f2886441d1cfe

(12) eap\_peap: Got tunneled reply code 11

(12) eap\_peap: EAP-Message = 0x010a00331a0309002e533d38433933343542353833373344334434423337314141424345334434373442313132303033373933

(12) eap\_peap: Message-Authenticator = 0x00000000000000000000000000000000

(12) eap\_peap: State = 0x733052da723a48b52b4f2886441d1cfe

(12) eap\_peap: Got tunneled reply RADIUS code 11

(12) eap\_peap: EAP-Message = 0x010a00331a0309002e533d38433933343542353833373344334434423337314141424345334434373442313132303033373933

(12) eap\_peap: Message-Authenticator = 0x00000000000000000000000000000000

(12) eap\_peap: State = 0x733052da723a48b52b4f2886441d1cfe

(12) eap\_peap: Got tunneled Access-Challenge

(12) eap: Sending EAP Request (code 1) ID 10 length 82

(12) eap: EAP session adding &reply:State = 0x1dfcd0851af6c9db

(12) [eap] = handled

(12) } # authenticate = handled

(12) Using Post-Auth-Type Challenge

(12) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(12) Challenge { ... } # empty sub-section is ignored

(12) session-state: Saving cached attributes

(12) Framed-MTU = 994

(12) TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(12) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(12) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(12) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(12) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(12) TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(12) TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(12) TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(12) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(12) TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(12) TLS-Session-Version = "TLS 1.2"

(12) Sent Access-Challenge Id 19 from 10.0.1.254:1812 to 10.0.1.3:1645 length 140

(12) EAP-Message = 0x010a005219001703030047e0c487f4aa2d57b05efd44ddaf69b13cf6dfd07add935d8649931f3bdeff4a7013a7a581348e5361d8acf09a236d7edbc016172e88c7e3be4dad79fb299a09f804f2b29f9e9845

(12) Message-Authenticator = 0x00000000000000000000000000000000

(12) State = 0x1dfcd0851af6c9db5e3d878bc6e70d5e

(12) Finished request

Waking up in 4.9 seconds.

(13) Received Access-Request Id 20 from 10.0.1.3:1645 to 10.0.1.254:1812 length 258

(13) User-Name = "testing"

(13) Framed-MTU = 1400

(13) Called-Station-Id = "d0c2.822e.4392:CCNP\_34\_727\_SSID\_3"

(13) Calling-Station-Id = "14ab.c53c.0807"

(13) Cisco-AVPair = "ssid=CCNP\_34\_727\_SSID\_3"

(13) Service-Type = Login-User

(13) Cisco-AVPair = "service-type=Login"

(13) Message-Authenticator = 0x0af8a0a20654c00fedda5a8e2ee2a165

(13) EAP-Message = 0x020a00251900170303001a0000000000000003f5a01b4555235082f99ca25bc91201efe22c

(13) NAS-Port-Type = Wireless-802.11

(13) NAS-Port = 277

(13) NAS-Port-Id = "277"

(13) State = 0x1dfcd0851af6c9db5e3d878bc6e70d5e

(13) NAS-IP-Address = 10.0.1.3

(13) NAS-Identifier = "RyanSunny"

(13) Restoring &session-state

(13) &session-state:Framed-MTU = 994

(13) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(13) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(13) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(13) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(13) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(13) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(13) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(13) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(13) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(13) &session-state:TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(13) &session-state:TLS-Session-Version = "TLS 1.2"

(13) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default

(13) authorize {

(13) policy filter\_username {

(13) if (&User-Name) {

(13) if (&User-Name) -> TRUE

(13) if (&User-Name) {

(13) if (&User-Name =~ / /) {

(13) if (&User-Name =~ / /) -> FALSE

(13) if (&User-Name =~ /@[^@]\*@/ ) {

(13) if (&User-Name =~ /@[^@]\*@/ ) -> FALSE

(13) if (&User-Name =~ /\.\./ ) {

(13) if (&User-Name =~ /\.\./ ) -> FALSE

(13) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) {

(13) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) -> FALSE

(13) if (&User-Name =~ /\.$/) {

(13) if (&User-Name =~ /\.$/) -> FALSE

(13) if (&User-Name =~ /@\./) {

(13) if (&User-Name =~ /@\./) -> FALSE

(13) } # if (&User-Name) = notfound

(13) } # policy filter\_username = notfound

(13) [preprocess] = ok

(13) [chap] = noop

(13) [mschap] = noop

(13) [digest] = noop

(13) suffix: Checking for suffix after "@"

(13) suffix: No '@' in User-Name = "testing", looking up realm NULL

(13) suffix: No such realm "NULL"

(13) [suffix] = noop

(13) eap: Peer sent EAP Response (code 2) ID 10 length 37

(13) eap: Continuing tunnel setup

(13) [eap] = ok

(13) } # authorize = ok

(13) Found Auth-Type = eap

(13) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(13) authenticate {

(13) eap: Expiring EAP session with state 0x0275a0c70672b940

(13) eap: Finished EAP session with state 0x1dfcd0851af6c9db

(13) eap: Previous EAP request found for state 0x1dfcd0851af6c9db, released from the list

(13) eap: Peer sent packet with method EAP PEAP (25)

(13) eap: Calling submodule eap\_peap to process data

(13) eap\_peap: (TLS) EAP Done initial handshake

(13) eap\_peap: Session established. Decoding tunneled attributes

(13) eap\_peap: PEAP state phase2

(13) eap\_peap: EAP method MSCHAPv2 (26)

(13) eap\_peap: Got tunneled request

(13) eap\_peap: EAP-Message = 0x020a00061a03

(13) eap\_peap: Setting User-Name to testing

(13) eap\_peap: Sending tunneled request to inner-tunnel

(13) eap\_peap: EAP-Message = 0x020a00061a03

(13) eap\_peap: FreeRADIUS-Proxied-To = 127.0.0.1

(13) eap\_peap: User-Name = "testing"

(13) eap\_peap: State = 0x733052da723a48b52b4f2886441d1cfe

(13) Virtual server inner-tunnel received request

(13) EAP-Message = 0x020a00061a03

(13) FreeRADIUS-Proxied-To = 127.0.0.1

(13) User-Name = "testing"

(13) State = 0x733052da723a48b52b4f2886441d1cfe

(13) WARNING: Outer and inner identities are the same. User privacy is compromised.

(13) server inner-tunnel {

(13) session-state: No cached attributes

(13) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/inner-tunnel

(13) authorize {

(13) policy filter\_username {

(13) if (&User-Name) {

(13) if (&User-Name) -> TRUE

(13) if (&User-Name) {

(13) if (&User-Name =~ / /) {

(13) if (&User-Name =~ / /) -> FALSE

(13) if (&User-Name =~ /@[^@]\*@/ ) {

(13) if (&User-Name =~ /@[^@]\*@/ ) -> FALSE

(13) if (&User-Name =~ /\.\./ ) {

(13) if (&User-Name =~ /\.\./ ) -> FALSE

(13) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) {

(13) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) -> FALSE

(13) if (&User-Name =~ /\.$/) {

(13) if (&User-Name =~ /\.$/) -> FALSE

(13) if (&User-Name =~ /@\./) {

(13) if (&User-Name =~ /@\./) -> FALSE

(13) } # if (&User-Name) = notfound

(13) } # policy filter\_username = notfound

(13) [chap] = noop

(13) [mschap] = noop

(13) suffix: Checking for suffix after "@"

(13) suffix: No '@' in User-Name = "testing", looking up realm NULL

(13) suffix: No such realm "NULL"

(13) [suffix] = noop

(13) update control {

(13) &Proxy-To-Realm := LOCAL

(13) } # update control = noop

(13) eap: Peer sent EAP Response (code 2) ID 10 length 6

(13) eap: No EAP Start, assuming it's an on-going EAP conversation

(13) [eap] = updated

(13) files: users: Matched entry testing at line 1

(13) [files] = ok

(13) [expiration] = noop

(13) [logintime] = noop

(13) pap: WARNING: Auth-Type already set. Not setting to PAP

(13) [pap] = noop

(13) } # authorize = updated

(13) Found Auth-Type = eap

(13) # Executing group from file /etc/freeradius/3.0/sites-enabled/inner-tunnel

(13) authenticate {

(13) eap: Expiring EAP session with state 0x0275a0c70672b940

(13) eap: Finished EAP session with state 0x733052da723a48b5

(13) eap: Previous EAP request found for state 0x733052da723a48b5, released from the list

(13) eap: Peer sent packet with method EAP MSCHAPv2 (26)

(13) eap: Calling submodule eap\_mschapv2 to process data

(13) eap: Sending EAP Success (code 3) ID 10 length 4

(13) eap: Freeing handler

(13) [eap] = ok

(13) } # authenticate = ok

(13) # Executing section post-auth from file /etc/freeradius/3.0/sites-enabled/inner-tunnel

(13) post-auth {

(13) if (0) {

(13) if (0) -> FALSE

(13) } # post-auth = noop

(13) } # server inner-tunnel

(13) Virtual server sending reply

(13) MS-MPPE-Encryption-Policy = Encryption-Allowed

(13) MS-MPPE-Encryption-Types = RC4-40or128-bit-Allowed

(13) MS-MPPE-Send-Key = 0x131754fa87cdb097689864f121ea9f7d

(13) MS-MPPE-Recv-Key = 0xbb7892aa92e0522b086574bddaf60e68

(13) EAP-Message = 0x030a0004

(13) Message-Authenticator = 0x00000000000000000000000000000000

(13) User-Name = "testing"

(13) eap\_peap: Got tunneled reply code 2

(13) eap\_peap: MS-MPPE-Encryption-Policy = Encryption-Allowed

(13) eap\_peap: MS-MPPE-Encryption-Types = RC4-40or128-bit-Allowed

(13) eap\_peap: MS-MPPE-Send-Key = 0x131754fa87cdb097689864f121ea9f7d

(13) eap\_peap: MS-MPPE-Recv-Key = 0xbb7892aa92e0522b086574bddaf60e68

(13) eap\_peap: EAP-Message = 0x030a0004

(13) eap\_peap: Message-Authenticator = 0x00000000000000000000000000000000

(13) eap\_peap: User-Name = "testing"

(13) eap\_peap: Got tunneled reply RADIUS code 2

(13) eap\_peap: MS-MPPE-Encryption-Policy = Encryption-Allowed

(13) eap\_peap: MS-MPPE-Encryption-Types = RC4-40or128-bit-Allowed

(13) eap\_peap: MS-MPPE-Send-Key = 0x131754fa87cdb097689864f121ea9f7d

(13) eap\_peap: MS-MPPE-Recv-Key = 0xbb7892aa92e0522b086574bddaf60e68

(13) eap\_peap: EAP-Message = 0x030a0004

(13) eap\_peap: Message-Authenticator = 0x00000000000000000000000000000000

(13) eap\_peap: User-Name = "testing"

(13) eap\_peap: Tunneled authentication was successful

(13) eap\_peap: SUCCESS

(13) eap: Sending EAP Request (code 1) ID 11 length 46

(13) eap: EAP session adding &reply:State = 0x1dfcd08515f7c9db

(13) [eap] = handled

(13) } # authenticate = handled

(13) Using Post-Auth-Type Challenge

(13) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(13) Challenge { ... } # empty sub-section is ignored

(13) session-state: Saving cached attributes

(13) Framed-MTU = 994

(13) TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(13) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(13) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(13) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(13) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(13) TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(13) TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(13) TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(13) TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(13) TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(13) TLS-Session-Version = "TLS 1.2"

(13) Sent Access-Challenge Id 20 from 10.0.1.254:1812 to 10.0.1.3:1645 length 104

(13) EAP-Message = 0x010b002e19001703030023e0c487f4aa2d57b13408ce713d0693c8a1aa926d50b780e92ff6fef36181a99bf260dc

(13) Message-Authenticator = 0x00000000000000000000000000000000

(13) State = 0x1dfcd08515f7c9db5e3d878bc6e70d5e

(13) Finished request

Waking up in 4.9 seconds.

(14) Received Access-Request Id 21 from 10.0.1.3:1645 to 10.0.1.254:1812 length 267

(14) User-Name = "testing"

(14) Framed-MTU = 1400

(14) Called-Station-Id = "d0c2.822e.4392:CCNP\_34\_727\_SSID\_3"

(14) Calling-Station-Id = "14ab.c53c.0807"

(14) Cisco-AVPair = "ssid=CCNP\_34\_727\_SSID\_3"

(14) Service-Type = Login-User

(14) Cisco-AVPair = "service-type=Login"

(14) Message-Authenticator = 0x4f32d493bd6893ce08aa9b8f2254b8b6

(14) EAP-Message = 0x020b002e1900170303002300000000000000042ed1a9d8b5f207ef498bb4c0850a47564b00b009dfc346afa31a7f

(14) NAS-Port-Type = Wireless-802.11

(14) NAS-Port = 277

(14) NAS-Port-Id = "277"

(14) State = 0x1dfcd08515f7c9db5e3d878bc6e70d5e

(14) NAS-IP-Address = 10.0.1.3

(14) NAS-Identifier = "RyanSunny"

(14) Restoring &session-state

(14) &session-state:Framed-MTU = 994

(14) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"

(14) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"

(14) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"

(14) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"

(14) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"

(14) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"

(14) &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"

(14) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"

(14) &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"

(14) &session-state:TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"

(14) &session-state:TLS-Session-Version = "TLS 1.2"

(14) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default

(14) authorize {

(14) policy filter\_username {

(14) if (&User-Name) {

(14) if (&User-Name) -> TRUE

(14) if (&User-Name) {

(14) if (&User-Name =~ / /) {

(14) if (&User-Name =~ / /) -> FALSE

(14) if (&User-Name =~ /@[^@]\*@/ ) {

(14) if (&User-Name =~ /@[^@]\*@/ ) -> FALSE

(14) if (&User-Name =~ /\.\./ ) {

(14) if (&User-Name =~ /\.\./ ) -> FALSE

(14) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) {

(14) if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/)) -> FALSE

(14) if (&User-Name =~ /\.$/) {

(14) if (&User-Name =~ /\.$/) -> FALSE

(14) if (&User-Name =~ /@\./) {

(14) if (&User-Name =~ /@\./) -> FALSE

(14) } # if (&User-Name) = notfound

(14) } # policy filter\_username = notfound

(14) [preprocess] = ok

(14) [chap] = noop

(14) [mschap] = noop

(14) [digest] = noop

(14) suffix: Checking for suffix after "@"

(14) suffix: No '@' in User-Name = "testing", looking up realm NULL

(14) suffix: No such realm "NULL"

(14) [suffix] = noop

(14) eap: Peer sent EAP Response (code 2) ID 11 length 46

(14) eap: Continuing tunnel setup

(14) [eap] = ok

(14) } # authorize = ok

(14) Found Auth-Type = eap

(14) # Executing group from file /etc/freeradius/3.0/sites-enabled/default

(14) authenticate {

(14) eap: Expiring EAP session with state 0x0275a0c70672b940

(14) eap: Finished EAP session with state 0x1dfcd08515f7c9db

(14) eap: Previous EAP request found for state 0x1dfcd08515f7c9db, released from the list

(14) eap: Peer sent packet with method EAP PEAP (25)

(14) eap: Calling submodule eap\_peap to process data

(14) eap\_peap: (TLS) EAP Done initial handshake

(14) eap\_peap: Session established. Decoding tunneled attributes

(14) eap\_peap: PEAP state send tlv success

(14) eap\_peap: Received EAP-TLV response

(14) eap\_peap: Success

(14) eap: Sending EAP Success (code 3) ID 11 length 4

(14) eap: Freeing handler

(14) [eap] = ok

(14) } # authenticate = ok

(14) # Executing section post-auth from file /etc/freeradius/3.0/sites-enabled/default

(14) post-auth {

(14) if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name == request:User-Name)) {

(14) if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name == request:User-Name)) -> FALSE

(14) update {

(14) &reply::Framed-MTU += &session-state:Framed-MTU[\*] -> 994

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) recv TLS 1.3 Handshake, ClientHello'

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) send TLS 1.2 Handshake, ServerHello'

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) send TLS 1.2 Handshake, Certificate'

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) send TLS 1.2 Handshake, ServerKeyExchange'

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) send TLS 1.2 Handshake, ServerHelloDone'

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) recv TLS 1.2 Handshake, ClientKeyExchange'

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) recv TLS 1.2 Handshake, Finished'

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) send TLS 1.2 ChangeCipherSpec'

(14) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[\*] -> '(TLS) send TLS 1.2 Handshake, Finished'

(14) &reply::TLS-Session-Cipher-Suite += &session-state:TLS-Session-Cipher-Suite[\*] -> 'ECDHE-RSA-AES256-GCM-SHA384'

(14) &reply::TLS-Session-Version += &session-state:TLS-Session-Version[\*] -> 'TLS 1.2'

(14) } # update = noop

(14) [exec] = noop

(14) policy remove\_reply\_message\_if\_eap {

(14) if (&reply:EAP-Message && &reply:Reply-Message) {

(14) if (&reply:EAP-Message && &reply:Reply-Message) -> FALSE

(14) else {

(14) [noop] = noop

(14) } # else = noop

(14) } # policy remove\_reply\_message\_if\_eap = noop

(14) if (EAP-Key-Name && &reply:EAP-Session-Id) {

(14) if (EAP-Key-Name && &reply:EAP-Session-Id) -> FALSE

(14) } # post-auth = noop

(14) Sent Access-Accept Id 21 from 10.0.1.254:1812 to 10.0.1.3:1645 length 175

(14) MS-MPPE-Recv-Key = 0x171c4112125f9828825826210a235d897df09241b7a5ea85135ff5a7c53c48d3

(14) MS-MPPE-Send-Key = 0xc28dc6f60be98571c8a439836c2ca480ab700efdd78ef2952ed6276ba53a9b14

(14) EAP-Message = 0x030b0004

(14) Message-Authenticator = 0x00000000000000000000000000000000

(14) User-Name = "testing"

(14) Framed-MTU += 994

(14) Finished request

Waking up in 4.9 seconds.

(10) Cleaning up request packet ID 17 with timestamp +73 due to cleanup\_delay was reached

(11) Cleaning up request packet ID 18 with timestamp +73 due to cleanup\_delay was reached

(12) Cleaning up request packet ID 19 with timestamp +74 due to cleanup\_delay was reached

(13) Cleaning up request packet ID 20 with timestamp +74 due to cleanup\_delay was reached

(14) Cleaning up request packet ID 21 with timestamp +74 due to cleanup\_delay was reached

Ready to process requests

**IPCONFIG:**

C:\Users\Lab\_user>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Ethernet adapter vEthernet (Default Switch):

Connection-specific DNS Suffix . :

Link-local IPv6 Address . . . . . : fe80::9d3c:f356:9171:90c8%45

IPv4 Address. . . . . . . . . . . : 172.27.144.1

Subnet Mask . . . . . . . . . . . : 255.255.240.0

Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 2:

Connection-specific DNS Suffix . :

Link-local IPv6 Address . . . . . : fe80::e526:ac31:b011:c4af%8

IPv4 Address. . . . . . . . . . . : 192.168.56.1

Subnet Mask . . . . . . . . . . . : 255.255.255.0

Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection\* 1:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection\* 2:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :

Link-local IPv6 Address . . . . . : fe80::f62b:f747:72db:ca85%12

IPv4 Address. . . . . . . . . . . : 10.0.20.4

Subnet Mask . . . . . . . . . . . : 255.255.255.0

Default Gateway . . . . . . . . . : 10.0.20.1

**Tracert to AP:**

C:\Users\Lab\_user>tracert 10.0.1.3

Tracing route to 10.0.1.3 over a maximum of 30 hops

1 1 ms 1 ms 1 ms 10.0.20.1

2 2 ms 1 ms 1 ms 10.0.1.3

Trace complete.

**Tracert to Switch:**

C:\Users\Lab\_user>tracert 10.0.1.2

Tracing route to 10.0.1.2 over a maximum of 30 hops

1 1 ms 1 ms 1 ms 10.0.20.1

2 3 ms 3 ms 2 ms 10.0.1.2

Trace complete.

**Tracert to Router (InterVLAN):**

C:\Users\Lab\_user>tracert 10.0.1.1

Tracing route to 10.0.1.1 over a maximum of 30 hops

1 1 ms 1 ms 1 ms 10.0.1.1

Trace complete.

**Tracert to Internet:**

C:\Users\Lab\_user>tracert 1.1.1.1

Tracing route to one.one.one.one [1.1.1.1]

over a maximum of 30 hops:

1 1 ms 1 ms 1 ms 10.0.20.1

2 2 ms 1 ms 1 ms 192.168.40.1

3 2 ms 1 ms 1 ms 172.28.128.1

4 2 ms 2 ms 2 ms 192.168.15.1

5 13 ms 9 ms 14 ms 10.61.242.194

6 14 ms 17 ms 14 ms po-316-315-rur401.bellevue.wa.seattle.comcast.net [24.153.84.161]

7 13 ms 13 ms 14 ms po-2-rur402.bellevue.wa.seattle.comcast.net [96.216.61.78]

8 13 ms 18 ms 15 ms po-400-xar02.bellevue.wa.seattle.comcast.net [96.216.61.61]

9 22 ms 22 ms 18 ms be-300-arsc1.seattle.wa.seattle.comcast.net [24.124.128.89]

10 \* 98 ms 17 ms be-36131-cs03.seattle.wa.ibone.comcast.net [68.86.93.9]

11 14 ms 20 ms 17 ms be-2313-pe13.seattle.wa.ibone.comcast.net [96.110.44.90]

12 12 ms \* 17 ms 66.208.232.210

13 42 ms 89 ms 14 ms 172.71.148.5

14 19 ms 17 ms 16 ms one.one.one.one [1.1.1.1]

Trace complete.

**Routing Table:**

**DHCP:**

S1#show ip dhcp binding

Bindings from all pools not associated with VRF:

IP address Client-ID/ Lease expiration Type

Hardware address/

User name

10.0.10.4 01ac.ed5c.3e57.6a Mar 02 1993 02:29 AM Automatic

10.0.20.4 0114.abc5.3c08.07 Mar 02 1993 02:20 AM Automatic

**NAT:**

R1#show ip nat translations

Pro Inside global Inside local Outside local Outside global

tcp 192.168.40.115:4520 10.0.20.4:51161 13.69.116.107:443 13.69.116.107:443

tcp 192.168.40.115:4515 10.0.20.4:51158 52.112.127.50:443 52.112.127.50:443

udp 192.168.40.115:512 10.0.20.4:137 192.168.15.1:137 192.168.15.1:137

udp 192.168.40.115:512 10.0.20.4:137 10.61.242.194:137 10.61.242.194:137

tcp 192.168.40.115:4531 10.0.20.4:51151 13.89.179.8:443 13.89.179.8:443

udp 192.168.40.115:512 10.0.20.4:137 172.28.128.1:137 172.28.128.1:137

tcp 192.168.40.115:4517 10.0.20.4:51058 40.83.240.146:443 40.83.240.146:443

tcp 192.168.40.115:4504 10.0.20.4:51154 13.107.136.10:443 13.107.136.10:443

tcp 192.168.40.115:4525 10.0.20.4:51150 192.168.40.76:7680 192.168.40.76:7680

tcp 192.168.40.115:4505 10.0.20.4:51155 52.104.79.25:443 52.104.79.25:443

tcp 192.168.40.115:4508 10.0.20.4:51049 40.83.240.146:443 40.83.240.146:443

tcp 192.168.40.115:4522 10.0.20.4:51163 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4506 10.0.20.4:51156 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4512 10.0.20.4:51145 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4516 10.0.20.4:51159 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4503 10.0.20.4:56084 52.112.87.139:443 52.112.87.139:443

tcp 192.168.40.115:4519 10.0.20.4:51160 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4501 10.0.20.4:51153 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4502 10.0.20.4:51148 104.18.12.46:80 104.18.12.46:80

udp 192.168.40.115:512 10.0.20.4:137 66.208.232.210:137 66.208.232.210:137

udp 192.168.40.115:4518 10.0.20.4:58947 8.8.4.4:443 8.8.4.4:443

udp 192.168.40.115:4514 10.0.20.4:53615 8.8.4.4:443 8.8.4.4:443

tcp 192.168.40.115:4518 10.0.20.4:51059 52.112.107.138:443 52.112.107.138:443

tcp 192.168.40.115:4527 10.0.20.4:51146 52.184.216.174:443 52.184.216.174:443

tcp 192.168.40.115:4513 10.0.20.4:51149 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4532 10.0.20.4:51152 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4529 10.0.20.4:51147 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4511 10.0.20.4:51052 162.159.134.234:443 162.159.134.234:443

udp 192.168.40.115:4512 10.0.20.4:58794 8.8.8.8:53 8.8.8.8:53

tcp 192.168.40.115:4521 10.0.20.4:51162 104.18.12.46:80 104.18.12.46:80

tcp 192.168.40.115:4514 10.0.20.4:51157 104.18.12.46:80 104.18.12.46:80

udp 192.168.40.115:4516 10.0.20.4:62045 8.8.8.8:53 8.8.8.8:53

udp 192.168.40.115:512 10.0.20.4:137 192.168.40.1:137 192.168.40.1:137

udp 192.168.40.115:512 10.0.20.4:137 172.71.148.5:137 172.71.148.5:137

udp 192.168.40.115:4517 10.0.20.4:57236 8.8.8.8:53 8.8.8.8:53

icmp 192.168.40.115:1 10.0.20.4:1 1.1.1.1:1 1.1.1.1:1

tcp 192.168.40.115:4530 10.0.20.4:51092 40.83.247.108:443 40.83.247.108:443

Total number of translations: 37

**Problems:**

1. **Access Point Nonfunctional**

The bootloader on the first access point we received was corrupted and after several attempts to enter recovery mode, continued to be unresponsive. The access point continued to boot loop, not even reaching OS load. We then exchanged this access point for another one, which worked.

1. **Access Point Corrupted**

After loading the appropriate OS and setting up basic interfaces, our access point suddenly corrupted the next day, rendering our previous work unusable. We flashed the OS again and continued working.

1. **Rogue DHCP server**

Due to sharing conflicts on a single rack with other groups running the same lab, we realized that address conflicts arose. A DHCP server from another group was configured incorrectly, causing it to use our same subnet.

1. **DHCP -> PAT -> DHCP**

Initially we did not realize that the upstream DHCP server provided by our class ran on the 192.168.x.x subnets. Since our topology was based on the 10.0.1.x subnet, we needed to use PAT in order to convert the outside 192.168.x.x addresses into our network. After doing that, we were able to confirm that it was functional by connecting our PC to the switch which the interface was connected to. However, we also needed to run PAT back to our own internal DHCP server running on a router. This was necessary in assigning AP clients addresses to get them on the internet.

At first, we tried to run all of these configurations on a Layer 3 switch to reduce topology size and complexity but later realized that the switch was incapable of this. Because of this, we needed to run an interface down to another router, which handled our DHCP and PAT. Then, this interface would run back up to our Layer 3 switch. One interface on the router served as a PAT outside port, and one served as the internal DHCP port.

1. **Assigning Clients DNS**

After we initially got WPA2 PSK working, we realized we could ping to an address like 8.8.8.8 (Google DNS) but could not access <https://google.com> on a web browser. After doing an ipconfig, we quickly realized that we had forgotten to assign a default DNS server to the clients on the switch. After assigning a common DNS server such as 1.1.1.1 or 8.8.8.8, we were able to access websites.

1. **Port Forwarding in VirtualBox/VMware/WSL**

Port forwarding in VM applications did not work as intended, neither did WSL. Initially, we intended to run FreeRadius in a hypervisor setup, but due to conflicts decided it would be better to run it on bare metal with another Ubuntu drive acting as the server.

1. **AP Radius Server Issues**

Initially we intended to use the Cisco AP’s built-in local RADIUS server, but after much configuration and confusion due to absolute lack of documentation, we ditched this idea, opting to use something more standard such as FreeRadius.

1. **Switch VLAN Issues**

Our configs were not persistent since our racks were shared amongst class members. When loading our switch config, we realized that VLAN interfaces by default were not included in the config. This led us to believe that our switch config was incorrect, but after troubleshooting realized that the VLAN interfaces needed to be manually activated.

1. **FreeRadius Configuration Troubles**

FreeRadius documentation labelled out of date configuration file locations, leading us to believe that the installation had not fully worked.

FreeRadius refused to accept a shared secret between the AP and the server, however upon changing this key worked as intended.

FreeRadius documentation was scarce and labelled poorly, leading us to configure wrong IP addresses, mixing up the server and the host. However, this did not yield in a useful error prompt, instead telling us that the shared secret was once again incorrect. Upon changing the IP address, the FreeRadius server worked as intended.

**Conclusion:**

Overall, this lab was a good introduction to AP configuration, as well as a refresher on configuring DHCP and PAT on routers/switches. We needed to design our topology and consider certain restrictions and limitations from the ground up, as we were only given a task. This allowed us to put prior networking knowledge into use, giving us a greater understanding of how these things function in a real topology.

WAP Signoff Sheet

Ryan Chen, P3-4 Cisco CCNP, Mr. Mason

