\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Securing a Cisco Router with AAA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Author*:

Gabriel Rosas



Purpose

An administrator can access a router or device through a console cable, but that would require them to be within physical range. What if they monitor and control devices across the world? An admin can’t be everywhere at once, so thankfully they can use remote connection to access a faraway device. Unfortunately, this presents many security risks. If an admin can remotely connect to a device, why wouldn’t a hacker be able to do the same? These are the questions that AAA attempts to solve, providing security for remote devices via authentication, authorization, and accounting.

Background Information

This lab will be split up into two sections: one part on *RADIUS* and another for *TACACS+*. I set up RADIUS on a Linux server and TACACS+ on Windows Server 2019. The goal was to secure login for administrative access. *Secure Shell* (SSH) is a common remote connection protocol used to access network devices that are not local. It is a secure alternative to other remote login protocols, such as telnet or rlogin. To test login access, I will *SSH* into the router and be prompted to enter login details verified by a AAA server. Guides I made for configuring both RADIUS and TACACS can be found on my [GitHub](https://github.com/TherieI/Cisco-Projects).

Authentication, Authorization, Accounting

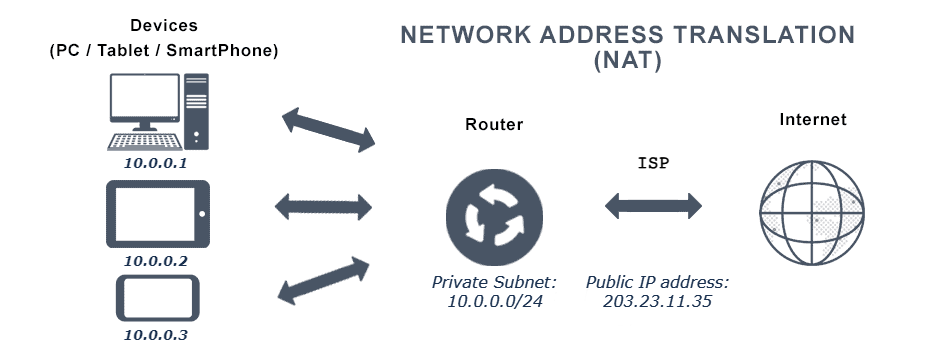
*Authentication, Authorization, Accounting* (AAA, pronounced “triple A”) is a standard used to control who has access to network devices, what level of permission they have, and the logging of a user’s activity. If AAA is not used, then authentication would be handled locally on each individual device, likely using shared usernames and passwords. Having to manage everything locally and individually is a huge strain on human resources and a security risk, prompting admins to use AAA to secure their services. Popular AAA services include RADIUS, TACACS+, and Diameter.

* *Authentication* provides a way to identify a user through a valid username and password before access is granted. When a user wants to remotely access a device, the device will compare the user’s credentials with those stored on the AAA server. If the credentials match on the AAA server, the user is granted access to the device. Otherwise, the device is denied.
* *Authorization* is the process of determining the level of permission a user has, namely, what they are permitted to access. After a user is authenticated, they may be authorized to view or edit certain files.
* *Accounting* logs the activity of a user during access, which may include the length of time spent, what they accessed, or changes they made. These statistics can be used by higher officials to determine billing, trend analysis, time management, and such.

In this lab, I configured both RADIUS and TACACS to test different AAA services.

Network Address Translation

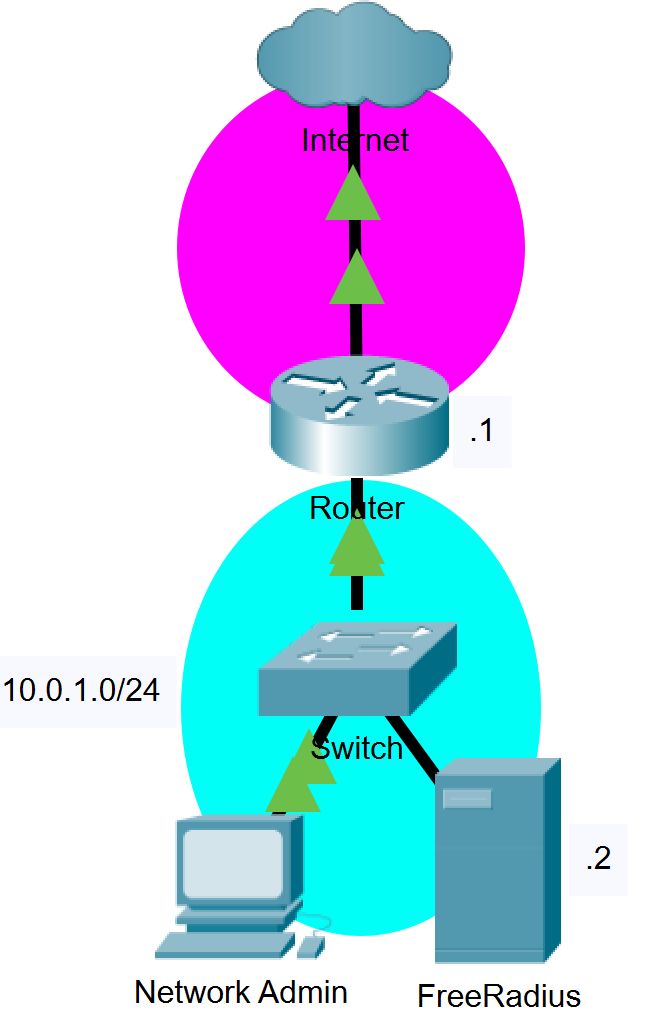
Network Address Translation (NAT) is a process that enables a unique IP address to represent an entire group of devices. Typically, this IP address is “Public”, administered by one’s ISP.



In this example, the public IP, *203.23.11.35*, is representing the *10.0.0.0/24* subnet. There can be multiple networks that use the same private subnet – what matters is that the public IP is unique. With so many devices nowadays, the total IPv4 address pool is running out. NAT helps conserve these addresses by translating multiple private IPs through one public IP. This is known as dynamic NAT. In this lab, I used NAT to provide internet access for my Linux server so I could download *FreeRADIUS* dependencies.

RADIUS

Topology



Background Information

Like Windows and MacOS, Linux is an operating system which can run applications, but particularly excels at server-based services, such as hosting a RADIUS server, since it is much more lightweight than other software. In computing, something “lightweight” refers to software designed to have a small memory footprint (RAM), low CPU, and low overall usage of system resources. Perfect for something like a server, but less user friendly. Users will typically navigate Linux through the command-line with less focus on graphical applications.

Since Linux is an open-source OS, where anyone can take the base code and manufacture it to their liking, there are many *distributions* designed for specific purposes. For example, in a previous project I worked with *Kali* Linux, a distribution focused on network penetration testing. In this project I used *Raspbian* – a modification of *Debian* – also a distribution of Linux. Raspbian is the official OS of the Raspberry Pi, which is just a tiny computer. I used the Raspberry Pi as my server, though this project could be replicated simply with any distribution of Linux.

Process

*RADIUS* was the first AAA protocol I stumbled upon when investigating AAA services. After looking into possible RADIUS server options, *FreeRADIUS* on Linux appeared to be popular among developers and I wanted to get more familiar with Linux as an OS. It seemed like a win-win situation. In an older project I installed Kali Linux on a laptop, so my initial thought was to use that as my server. I ran the command *sudo apt-get install freeradius freeradius-utils* on the Kali laptop to install the FreeRADIUS packages. Everything went smoothly until I tried running the server, which ended up crashing every time with an incomprehensible failure error. I tried reinstalling the packages, but the error persisted.

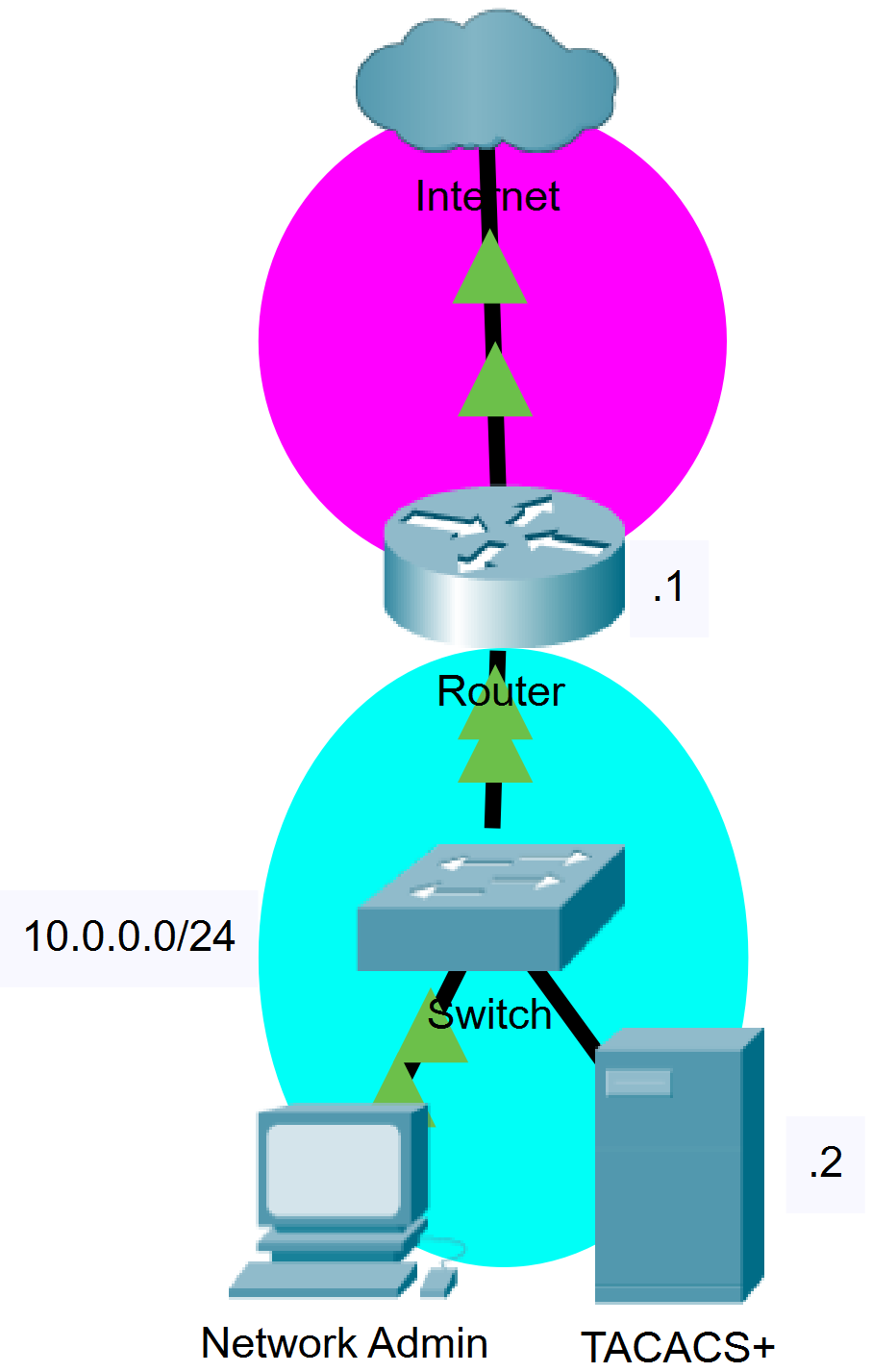
Perhaps Kali Linux wasn’t the best Linux distribution to choose as my server. Why not use the spare Raspberry Pi laying around instead? Raspbian is a distribution of Linux, so most (if not all) of the commands I used on the Kali should work on the Pi. I began by installing the FreeRADIUS packages on the Pi the same way I did for Kali. When I ran the server with no configuration, there was no error like Kali, so I felt safe to continue.

I added the router as a client in my FreeRADIUS configuration by editing the *clients.conf* file, then edited the user file to create an account. On the router, I configured various *aaa authentication* commands. These commands tell the router to verify credentials with a specified AAA server, which you can define using the *radius server* command. I ran the AAA server and attempted to login to the router. The authentication failed.

To troubleshoot, my first idea was checking the debug log of the FreeRADIUS server. The debug log contained the username and password I entered on the router that the server intended to verify. The username I entered was correct, but the password was a mess of random Unicode characters that I didn’t enter, like so: *\304<\306D\202\346jh\371\n\233*… I presumed the router might be encrypting the password, sending it to the AAA server, and the server not doing the necessary decryption. My solution was to rework the *aaa authentication* commands I had on the router and restart the FreeRADIUS server. In my opinion, the best feature of FreeRADIUS was the *debug* mode (*sudo freeradius -X*), which would run the server and output helpful information in real time like login attempts or login failures.

TACACS+

Topology



Background Information

Windows Server is an OS developed by Microsoft that supports enterprise-level management, data storage, applications, and communications. Many argue that Windows Server is more beginner friendly due to its intuitive graphical interface and requires less maintenance thanks to automated updates.

TACACS is a Cisco proprietary protocol. In this regard, it is better suited to handle AAA between Cisco devices, which is ideal for this project. I found that TACACS provided many more *authorization* features than RADIUS. For example, if I wanted a user whose sole purpose is to debug, I would only let them type “show” commands. TACACS turns this seemingly complicated task into a simple process with access-list like syntax. I would recommend TACACS over RADIUS for securing Cisco devices just because of this feature.

Process

After configuring RADIUS using a Linux server, I was intrigued in doing the same for *TACACS+*, but on Windows Server. The concepts of AAA should translate the same across the services, so there was not much extra to learn. I intended to host the Windows Server on VirtualBox, a virtualization program.

I ran through the process of installing a Windows Server 2019 VM without any issues. We decided to use the TACACS service from *TACACS.net*, which was kindly provided to me by Harsha Bhat on a USB drive. The program is an executable that can be run as a Windows service. The main problem was transferring the file from the USB to my virtual machine. It sounds like a trivial issue, *the VM is running on your PC, so just copy the file*, but frustratingly was not so easy. There are two options in VirtualBox to transfer files: through USB or through “file sharing”. I got an error when I tried to access my USB and file sharing was not sharing my files. Eventually, I found out that VirtualBox does not support USB 3.0 by default. You need to install an extension pack for it to work.

With the extension pack installed, I transferred the TACACS file from the USB to my virtual machine. I ran the program and went through the initial TACACS install. Much like RADIUS on Linux, I had to edit files to configure TACACS on Windows Server. Once I had the client (the router) and a user account set up, I checked my configuration with TACAC’s testing commands. *Tacverify* can be used to check the files for any syntax errors and *tactest* can be used for troubleshooting authentication requests. For example, by running the command *tactest -s 127.0.0.1 -k key -u giga -p chad*, I’m asking the server to verify an account with a username of “giga” and password of “chad”. The “key” is a credential that the AAA server and router share since the router is a client of the AAA server. The AAA server must first verify the router with the key. I had a minor lapse when *tactest* failed for the IP *10.0.0.2* (the AAA server), though restarting the TACACS service with *sc start/stop TACACS.net* solved that problem.

Lab Commands

|  |  |
| --- | --- |
| **Command** | A statement necessary for a configuration to work, denoted in bold |
| **[*Argument*]** | An argument necessary for a command to function, denoted in bold italics. |
| *Optional-Statement*  *<Optional Argument>* | An optional argument or statement, not necessary for a command to function, denoted in italics |

Router(config)# **aaa new-model**

* Specifies AAA as the authentication method for VTY lines on the router

*To configure any AAA services, you must first define aaa new-model.*

// RADIUS Authentication

Router(config)# **aaa authentication attempts login [#]**

* Specifics the number of login attempts a user gets before the connection terminates

Router(config)# **aaa authentication banner `***message***`**

* Set a message for a user when they connect to the device

Router(config)# **aaa authentication fail-message `***message***`**

* Set a message if the user fails their credentials

Router(config)# **aaa authentication *login default group radius***

* Make the router verify login credentials with a radius server

Router(config)# **aaa authentication *enable default group radius***

* Make the router verify privilege exec mode credentials with a radius server

// Defining a RADIUS Server

Router(config)# **radius server [*name*]**

* Define a radius server

*The router will use the ip of the radius server subsequently provided to verify credentials. This command can only be typed after* aaa new-model *has been declared.*

Router(config-radius-server)# **address ipv4 [*ip*] auth-port *1812* acct-port *1813***

* Define the *ip* of the radius server

Router(config-radius-server)# **key [*key*]**

* Define the *key* of the radius server

*The key on the router should match the key in the radius server’s configuration files.*

// TACACS+ Authentication

Router(config)# **aaa authentication *login default group tacacs+***

* Make the router verify login credentials with a tacacs server

Router(config)# **aaa authentication *enable default group tacacs+***

* Make the router verify privilege exec mode credentials with a tacacs server

// Defining a TACACS Server

Router(config)# **tacacs server [*name*]**

* Define a tacacs server

*The router will use the ip of the tacacs server subsequently provided to verify credentials. This command can only be typed after* aaa new-model *has been declared.*

Router(config-server-tacacs)# **address ipv4 [*ip*]**

* Define the *ip* of the tacacs server

Router(config-server-tacacs)# **key [*key*]**

* Define the *key* of the tacacs server

*The key on the router should match the key in the tacacs server’s configuration files.*

// Dynamic NAT Configuration

Router(config-if)# **ip nat [*inside/outside*]**

* Configure an interface to be internal or external

Inside interfaces are translated through the outside interface.

Router(config)# **access-list [*#*] permit [*network address*] [*wildcard mask*]**

* Create an access list that permits a subnet

The subnet specified here should be on an *inside* interface. NAT will translate the subnet out the *outside* interface. Keep note of the [*#*] defined, for that will be used in a later command.

Router(config)# **ip nat inside source list [*#*] interface [*id*] overload**

* Enable the translation of an access list through an outside interface

The source list [*#*] should be an access list created earlier; the *interface id* should be the outside interface.

Configurations

Router (RADIUS)

#show running-config

service timestamps debug datetime msec

service timestamps log datetime msec

platform qfp utilization monitor load 80

platform punt-keepalive disable-kernel-core

hostname Router

boot-start-marker

boot-end-marker

vrf definition Mgmt-intf

address-family ipv4

exit-address-family

address-family ipv6

exit-address-family

aaa new-model

aaa authentication attempts login 5

aaa authentication banner `Stop mortal. Speak thine Words of Entry.`

aaa authentication fail-message `You have failed. Begone.`

aaa authentication login default group radius

aaa authentication enable default group radius

aaa session-id common

login on-success log

subscriber templating

multilink bundle-name authenticated

no license smart enable

diagnostic bootup level minimal

spanning-tree extend system-id

redundancy

mode none

interface GigabitEthernet0/0/0

ip address dhcp

ip nat outside

negotiation auto

no shutdown

interface GigabitEthernet0/0/1

ip address 10.0.1.1 255.255.255.0

ip nat inside

negotiation auto

no shutdown

interface GigabitEthernet0

vrf forwarding Mgmt-intf

no ip address

shutdown

negotiation auto

ip forward-protocol nd

ip http server

ip http authentication local

ip http secure-server

ip http client source-interface GigabitEthernet0/0/0

ip nat inside source list 1 interface GigabitEthernet0/0/0 overload

access-list 1 permit 10.0.1.0 0.0.0.255

radius server PI\_RADIUS

address ipv4 10.0.1.2 auth-port 1812 acct-port 1813

timeout 30

retransmit 3

key chad

control-plane

line con 0

exec-timeout 0 0

transport input none

stopbits 1

line aux 0

stopbits 1

line vty 0 4

end

Router (TACACS)

Router#show running-config

service timestamps debug datetime msec

service timestamps log datetime msec

platform qfp utilization monitor load 80

platform punt-keepalive disable-kernel-core

hostname Router

boot-start-marker

boot-end-marker

vrf definition Mgmt-intf

address-family ipv4

exit-address-family

address-family ipv6

exit-address-family

aaa new-model

aaa authentication login default group tacacs+

aaa authentication enable default group tacacs+

aaa session-id common

login on-success log

subscriber templating

multilink bundle-name authenticated

no license smart enable

diagnostic bootup level minimal

spanning-tree extend system-id

redundancy

mode none

interface GigabitEthernet0/0/0

ip address dhcp

ip nat outside

negotiation auto

no shutdown

interface GigabitEthernet0/0/1

ip address 10.0.0.1 255.255.255.0

ip nat inside

negotiation auto

no shutdown

interface GigabitEthernet0

vrf forwarding Mgmt-intf

no ip address

shutdown

negotiation auto

ip forward-protocol nd

ip http server

ip http authentication local

ip http secure-server

ip http client source-interface GigabitEthernet0/0/0

ip nat inside source list 1 interface GigabitEthernet0/0/0 overload

access-list 1 permit 10.0.0.0 0.0.0.255

tacacs server WINSER2019

address ipv4 10.0.0.2

key chad

control-plane

line con 0

exec-timeout 0 0

transport input none

stopbits 1

line aux 0

stopbits 1

line vty 0 4

end

Conclusion

In this lab, I secured a Cisco router using *RADIUS* then *TACACS+*, implementing authentication and authorization. Considering how widely used AAA is in industry, it is important to learn for network security. Links to AAA guides can be found on my [GitHub](https://github.com/TherieI/Cisco-Projects), as can many of my other projects.