

FACULDADE DE TECNOLOGIA SENAC GOIÁS

PROJETO DE REDES DE COMPUTADORES



Jefferson Medrado

Desiron Gonçalves

FreeRADIUS

GOIÂNIA,

2018

Jefferson Medrado

Desiron Gonçalves

FreeRADIUS

Componente do Projeto Integrador 2018-1 apresentado como requisito parcial de avaliação na disciplina de Projeto de Redes de Computadores, no Curso de Gestão da Tecnologia da Informação, na Faculdade de Tecnologia Senac Goiás, ministrado pela professora Kelly Alves.

GOIÂNIA,

2018

RESUMO

FreeRADIUS é o servidor open source que suporta o maior número de tipos de autenticação e, atualmente, é o único servidor RADIUS de código livre que suporta o protocolo EAP - *Extensible Authentication Protocol*. Além disto, FreeRADIUS é o único que suporta virtualização, mantendo os custos de implantação e manutenção baixos. Seu desenho modular é fácil de entender, permitindo facilmente a inclusão ou remoção de módulos sem afetar o desempenho, os requisitos de hardware, de memória ou a segurança do sistema. Tem-se o com principal função o controle de acesso através de autenticações utilizando a rede WIFI.

Palavras-chave: FreeRADIUS, servidor, autenticação e controle de acesso.

SUMÁRIO

INTRODUÇÃO	5
CONHECENDO O FREERADIUS	6
INSTALANDO O FREERADIUS	7
CONFIGURANDO O FREERADIUS	9
ADICIONANDO CLIENTES.....	11
DEFINIR MÉTODOS EAP.....	12
ADICIONAR USUÁRIOS.....	13
AUTENTICAÇÃO DE TESTE	14
INICIE O FREERADIUS	14
CONCLUSÃO	15
REFERENCIAS BIBLIOGRÁFICAS.....	16

INTRODUÇÃO

Este trabalho tem o objetivo de apresentar a ferramenta FreeRADIUS ao leitor e a efetuar a capacitação de sua respectiva instalação. Esse software tem como função ser um servidor RADIUS de gerenciamento e controle de acesso através de autenticações utilizando a rede WIFI.

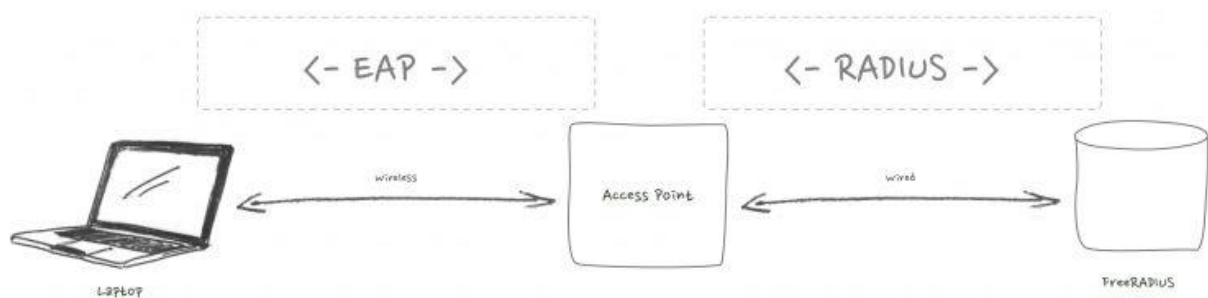
CONHECENDO O FREERADIUS

O FreeRADIUS é um servidor RADIUS de código aberto utilizado por muitas organizações. Possui suporte a muitos protocolos de autenticação e é muito popular porque é modular e escalável tendo o papel de proteger credenciais de usuário via Wi-Fi através de autenticações direcionadas ao servidor do qual será realizado o controle e gerenciamento dessas autenticações. Em nosso laboratório de Wi-Fi, usaremos o FreeRADIUS para autenticar usuários de Wi-Fi com o 802.1X.

IEEE 802.1X é um padrão IEEE pertencente ao grupo do protocolo IEEE 802.1 para controle de acesso sendo um mecanismo de autenticação para dispositivos que desejam juntar-se a uma porta na WLAN, seja estabelecendo uma conexão ponto-a-ponto ou prevenindo acesso para esta porta se a autenticação falhar. No 802.1X e em nosso laboratório, o FreeRADIUS desempenhará o papel de servidor de autenticação.

O mecanismo de autenticação 802.1X possui três componentes:

- Suplicante (dispositivo móvel)
- Autenticador (AP)
- Servidor de Autenticação (FreeRADIUS)



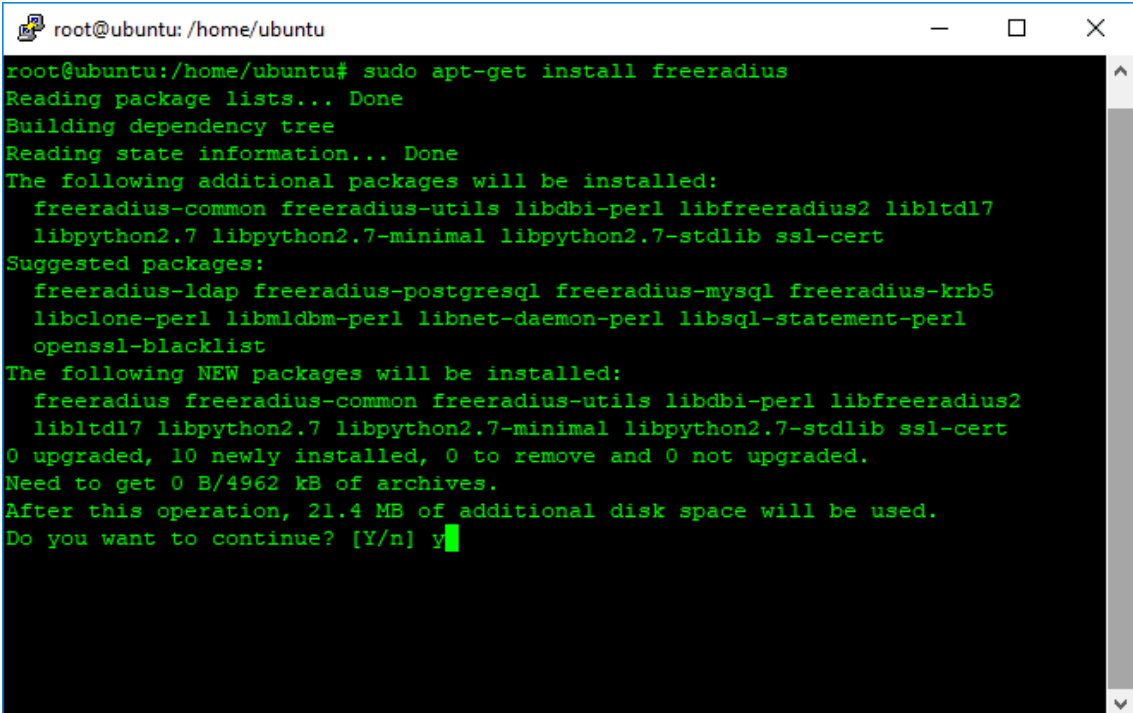
O suplicante solicitará a associação a um SSID, o autenticador solicitará uma identidade do dispositivo móvel, o autenticador encaminhará a identidade para o servidor de autenticação que responderá com êxito ou negação.

Este trabalho irá colocá-lo em funcionamento para isso implementaremos o FreeRADIUS no Ubuntu 16.04.1 LTS que eu temos rodando como uma VM.

INSTALANDO O FREERADIUS

Para realizarmos a instalação, pressupomos que o leitor possua uma máquina VM com Ubuntu instalado e com todas as suas dependências em dia, sendo assim, seguiremos com a instalação. Para iniciarmos podemos começar com o apt-get. A instalação do FreeRADIUS também instalará dependências e pacotes adicionais necessários para a operação.

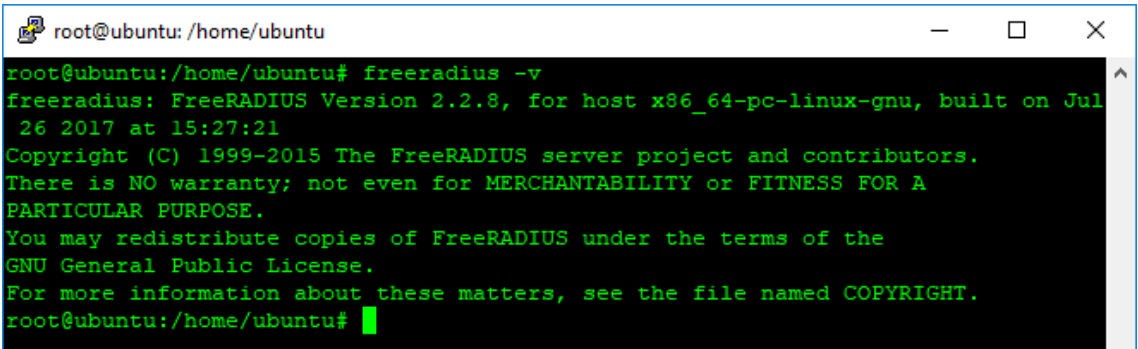
\$ sudo apt-get install freeradius

A terminal window titled 'root@ubuntu: /home/ubuntu' showing the output of the command 'sudo apt-get install freeradius'. The output indicates that 10 new packages will be installed, including freeradius, freeradius-common, freeradius-utils, libdbi-perl, libfreeradius2, libltdl7, libpython2.7, libpython2.7-minimal, libpython2.7-stdlib, and ssl-cert. It also lists suggested packages like freeradius-ldap, freeradius-postgresql, freeradius-mysql, freeradius-krb5, libclone-perl, libmldbm-perl, libnet-daemon-perl, libsql-statement-perl, and openssl-blacklist. The total disk space required is 21.4 MB.

```
root@ubuntu:/home/ubuntu# sudo apt-get install freeradius
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  freeradius-common freeradius-utils libdbi-perl libfreeradius2 libltdl7
  libpython2.7 libpython2.7-minimal libpython2.7-stdlib ssl-cert
Suggested packages:
  freeradius-ldap freeradius-postgresql freeradius-mysql freeradius-krb5
  libclone-perl libmldbm-perl libnet-daemon-perl libsql-statement-perl
  openssl-blacklist
The following NEW packages will be installed:
  freeradius freeradius-common freeradius-utils libdbi-perl libfreeradius2
  libltdl7 libpython2.7 libpython2.7-minimal libpython2.7-stdlib ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/4962 kB of archives.
After this operation, 21.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

➤ Verifique a versão do FreeRADIUS e se ele foi instalado, verificando a versão.

\$ freeradius -v

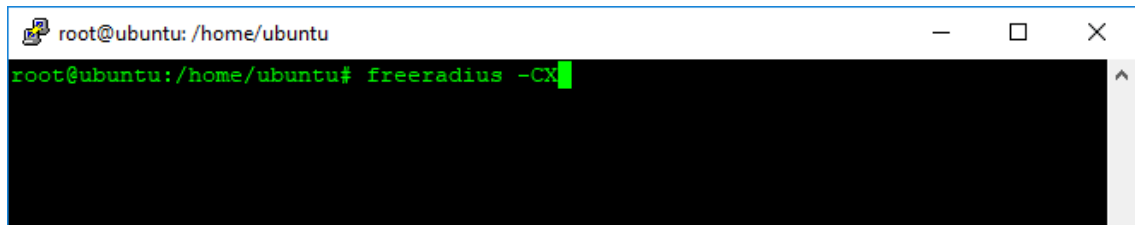
A terminal window titled 'root@ubuntu: /home/ubuntu' showing the output of the command 'freeradius -v'. The output displays the FreeRADIUS version as 2.2.8, built on July 26, 2017, at 15:27:21. It also includes copyright information and a disclaimer.

```
root@ubuntu:/home/ubuntu# freeradius -v
freeradius: FreeRADIUS Version 2.2.8, for host x86_64-pc-linux-gnu, built on Jul
 26 2017 at 15:27:21
Copyright (C) 1999-2015 The FreeRADIUS server project and contributors.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
You may redistribute copies of FreeRADIUS under the terms of the
GNU General Public License.
For more information about these matters, see the file named COPYRIGHT.
root@ubuntu:/home/ubuntu#
```

- Execute uma verificação rápida de configuração.

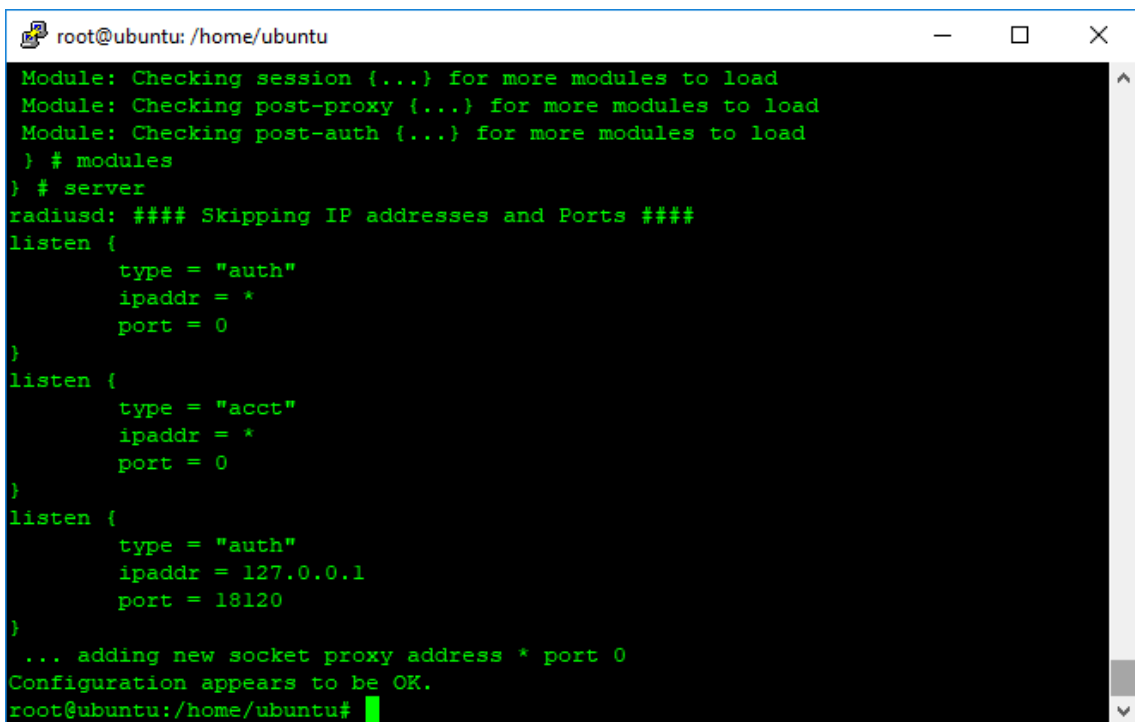
FreeRADIUS deve ser capaz de rodar com sucesso com todos os padrões.

\$ sudo freeradius -CX



```
root@ubuntu: /home/ubuntu
root@ubuntu:/home/ubuntu# freeradius -CX
```

A saída na parte inferior exibirá:



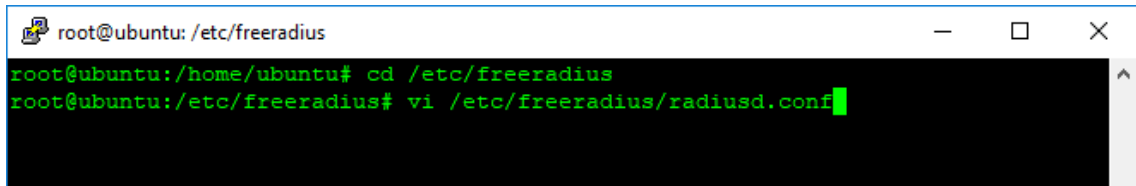
```
root@ubuntu: /home/ubuntu
Module: Checking session {...} for more modules to load
Module: Checking post-proxy {...} for more modules to load
Module: Checking post-auth {...} for more modules to load
} # modules
} # server
radiusd: #### Skipping IP addresses and Ports ####
listen {
    type = "auth"
    ipaddr = *
    port = 0
}
listen {
    type = "acct"
    ipaddr = *
    port = 0
}
listen {
    type = "auth"
    ipaddr = 127.0.0.1
    port = 18120
}
... adding new socket proxy address * port 0
Configuration appears to be OK.
root@ubuntu:/home/ubuntu#
```


CONFIGURANDO O FREERADIUS

- Os arquivos de configurações do FreeRADIUS estão localizados em / etc:

cd /etc/freeradius

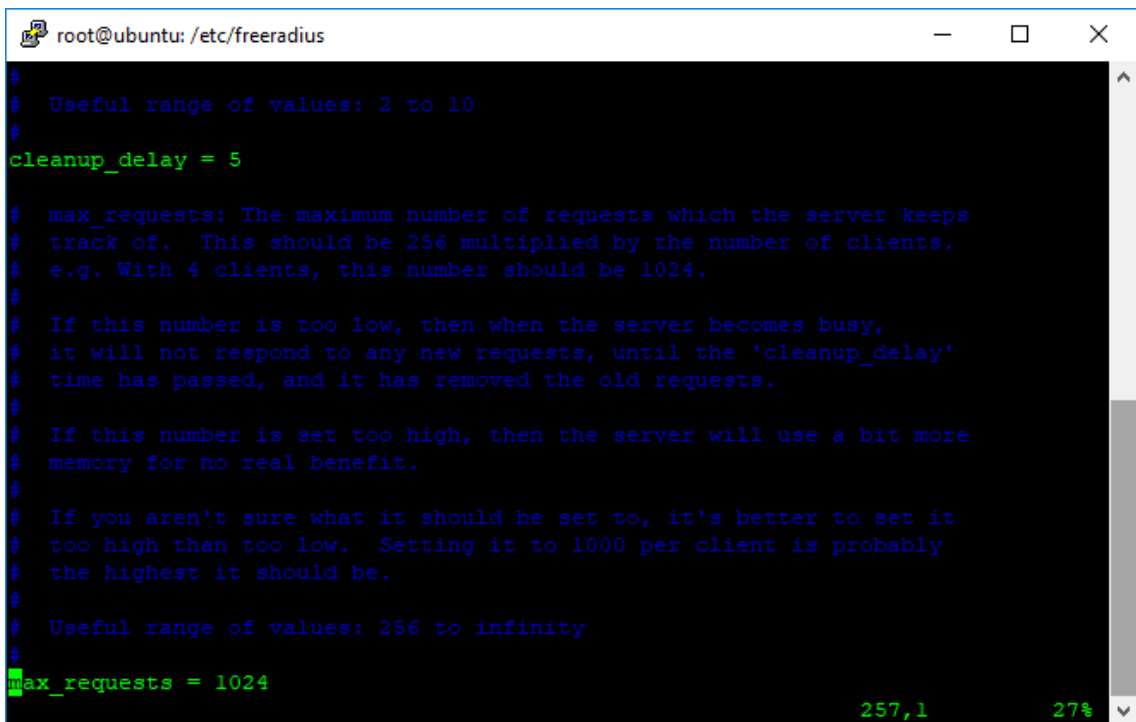
\$ sudo vi /etc/freeradius/radiusd.conf



```
root@ubuntu: /etc/freeradius
root@ubuntu:/home/ubuntu# cd /etc/freeradius
root@ubuntu:/etc/freeradius# vi /etc/freeradius/radiusd.conf
```

Número máximo de pedidos:

Aumente o valor padrão de 1024 se você planeja ter mais de 4 clientes autenticando por vez.

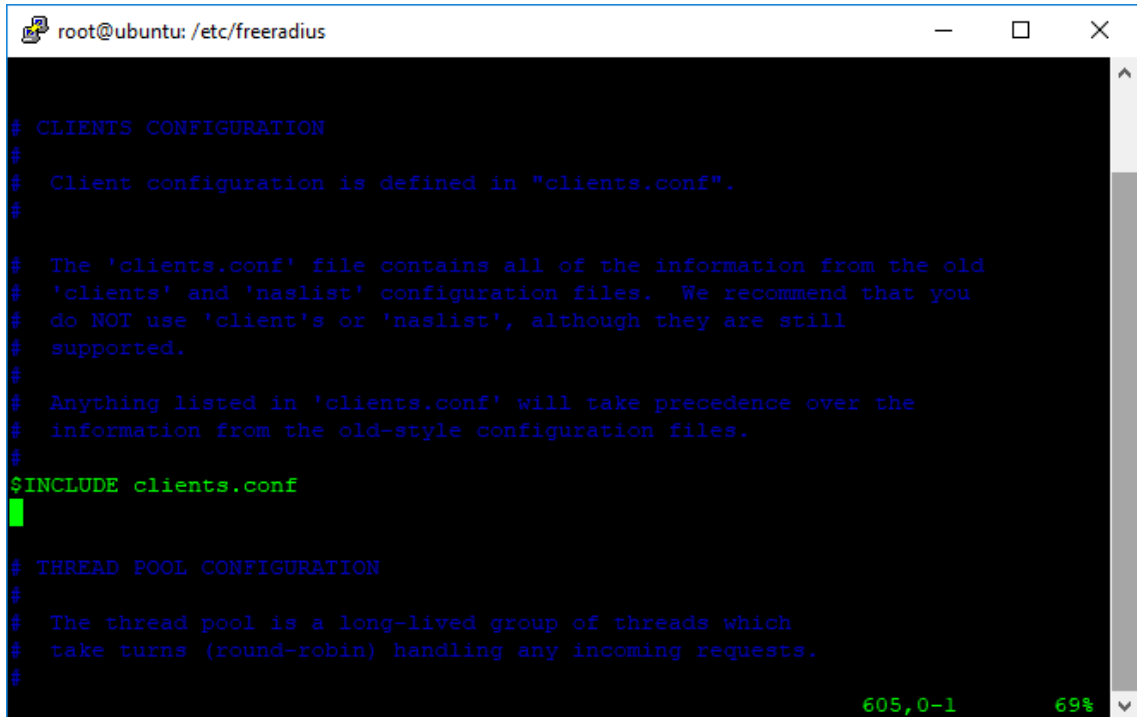


```
root@ubuntu: /etc/freeradius
#
# Useful range of values: 2 to 10
#
cleanup_delay = 5

# max_requests: The maximum number of requests which the server keeps
# track of. This should be 256 multiplied by the number of clients.
# e.g. With 4 clients, this number should be 1024.
#
# If this number is too low, then when the server becomes busy,
# it will not respond to any new requests, until the 'cleanup_delay'
# time has passed, and it has removed the old requests.
#
# If this number is set too high, then the server will use a bit more
# memory for no real benefit.
#
# If you aren't sure what it should be set to, it's better to set it
# too high than too low. Setting it to 1000 per client is probably
# the highest it should be.
#
# Useful range of values: 256 to infinity
#
max_requests = 1024
```

- Aqui é onde informamos ao FreeRADIUS para procurar por clientes autorizados (autenticadores).

Arquivo de configuração para seus **clientes** (pontos de acesso ou controladores)

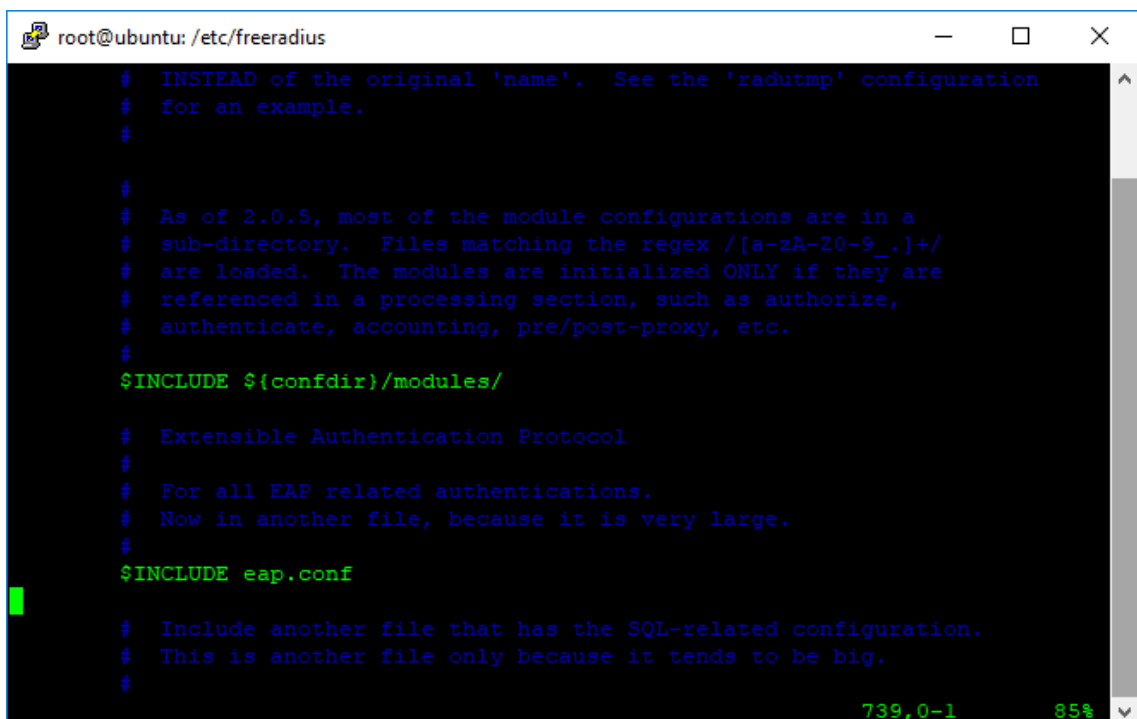


```
root@ubuntu: /etc/freeradius

# CLIENTS CONFIGURATION
#
# Client configuration is defined in "clients.conf".
#
# The 'clients.conf' file contains all of the information from the old
# 'clients' and 'naslist' configuration files. We recommend that you
# do NOT use 'client's or 'naslist', although they are still
# supported.
#
# Anything listed in 'clients.conf' will take precedence over the
# information from the old-style configuration files.
#
$INCLUDE clients.conf

# THREAD POOL CONFIGURATION
#
# The thread pool is a long-lived group of threads which
# take turns (round-robin) handling any incoming requests.
#
605,0-1 69%
```

Mais abaixo, no arquivo radiusd.conf, estão localizados os métodos EAP definidos, que é o arquivo eap.conf. Este arquivo de configuração é utilizado para definir métodos EAP usados:



```
root@ubuntu: /etc/freeradius

# INSTEAD of the original 'name'. See the 'radutmp' configuration
# for an example.
#
#
# As of 2.0.5, most of the module configurations are in a
# sub-directory. Files matching the regex /[a-zA-Z0-9_.]+/
# are loaded. The modules are initialized ONLY if they are
# referenced in a processing section, such as authorize,
# authenticate, accounting, pre/post-proxy, etc.
#
$INCLUDE ${confdir}/modules/

# Extensible Authentication Protocol
#
# For all EAP related authentications.
# Now in another file, because it is very large.
#
$INCLUDE eap.conf

# Include another file that has the SQL-related configuration.
# This is another file only because it tends to be big.
#
739,0-1 85%
```

ADICIONANDO CLIENTES

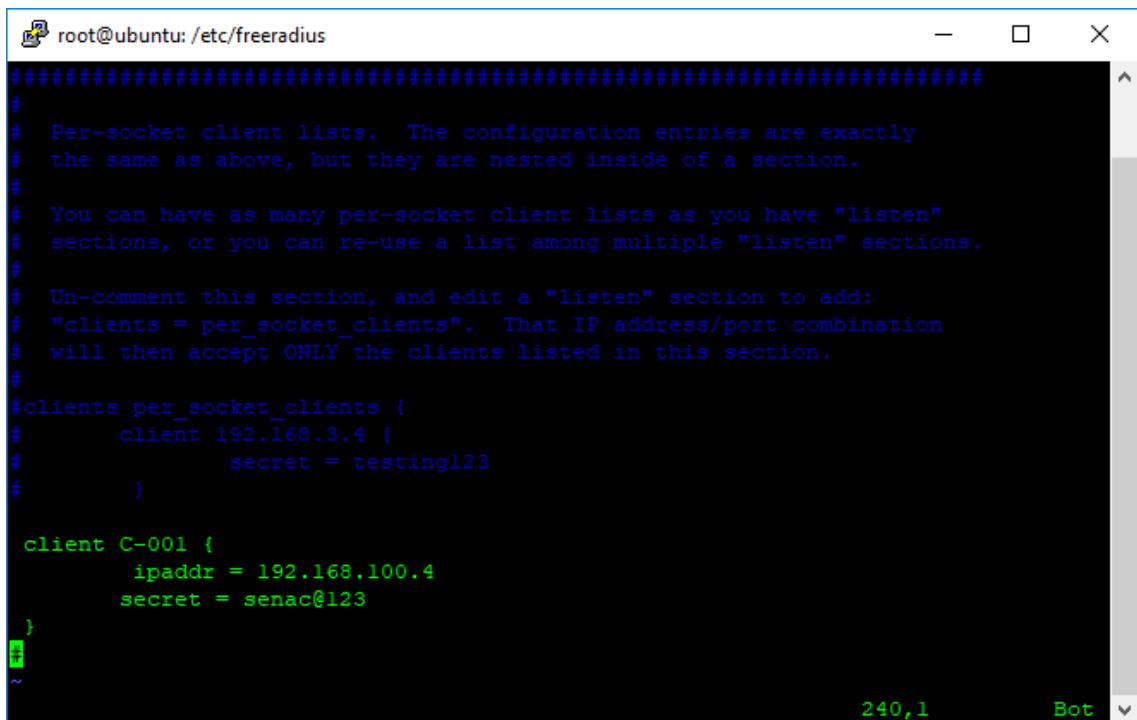
A palavra “clientes” pode ser enganosa, mas em termos de FreeRADIUS isso significa que os autenticadores, como os APs ou controladores de WLAN. Queremos verificar se apenas autenticadores autorizados são usados em nossa rede. Os autenticadores e o servidor de autenticação usarão segredos compartilhados para verificar um ao outro.

- Abra o arquivo `clients.conf` para adicionar seu (s) autenticador (es).

\$ sudo vi clients.conf

Se seus APs ou controladores estiverem em uma rede específica, você pode definir toda a rede ou especificar um segredo para clientes individuais.

Aqui é onde eu adiciono nosso ponto de acesso, que é um Cliente Networks C-001.



```
root@ubuntu: /etc/freeradius
#####
#
# Per-socket client lists. The configuration entries are exactly
# the same as above, but they are nested inside of a section.
#
# You can have as many per-socket client lists as you have "listen"
# sections, or you can re-use a list among multiple "listen" sections.
#
# Un-comment this section, and edit a "listen" section to add:
# "clients = per_socket_clients". That IP address/port combination
# will then accept ONLY the clients listed in this section.
#
#clients per_socket_clients {
#    client 192.168.3.4 {
#        secret = testing123
#    }
#
#    client C-001 {
#        ipaddr = 192.168.100.4
#        secret = senac@123
#    }
#
#}
~
240,1 Bot
```

Em seguida, salve o arquivo e saia.

DEFINIR MÉTODOS EAP

Agora, a parte divertida é saber quais métodos EAP você deseja usar. Eu não vou fazer um tutorial EAP aprofundado sobre este segmento. Em nosso laboratório estamos usando o EAP-TTLS porque requer o uso de um certificado do lado do servidor, mas os certificados do cliente são opcionais.

Por padrão, o FreeRadius usará o MD5, que não é muito forte.

```
# EAP types not listed here may be supported via the eap module.
# See experimental.conf for documentation.
#
eap {
    # Invoke the default supported EAP type when
    # EAP-Identity response is received.
    #
    # The incoming EAP messages DO NOT specify which EAP
    # type they will be using, so it MUST be set here.
    #
    # For now, only one default EAP type may be used at a time.
    #
    # If the EAP-Type attribute is set by another module,
    # then that EAP type takes precedence over the
    # default type configured here.
    #
    default_eap_type = ttls
}
```

Comente a de `default_eap_type = md5` e altere para: **`default_eap_type = ttls`**

O FreeRadius vem com um certificado de servidor por padrão que usaremos para testes iniciais. Comente os tipos de EAP suportados de MD5, LEAP, GTC

➤ Em ttls, altere **`default_eap_type`** para **`mschapv2`**

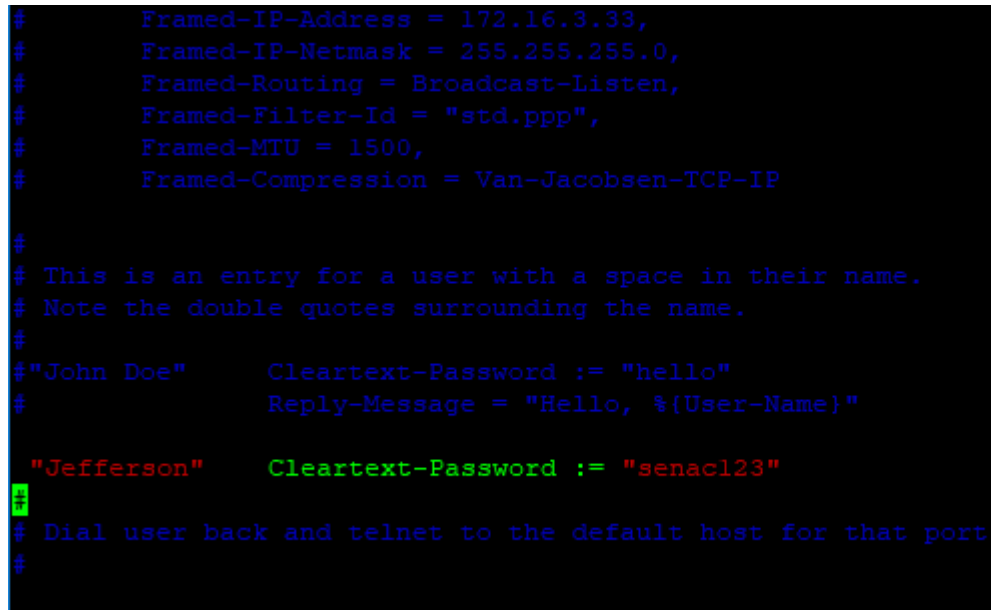
```
#
ttls {
    # The tunneled EAP session needs a default
    # EAP type which is separate from the one for
    # the non-tunneled EAP module. Inside of the
    # TTLS tunnel, we recommend using EAP-MD5.
    # If the request does not contain an EAP
    # conversation, then this configuration entry
    # is ignored.
    default_eap_type = mschapv2
    # The tunneled authentication request does
```

ADICIONAR USUÁRIOS

Vamos adicionar usuários que serão autenticados neste servidor RADIUS. Edite o arquivo de usuários com este comando:

\$ sudo vi users

Adicione uma conta ao arquivo:



```
# Framed-IP-Address = 172.16.3.33,
# Framed-IP-Netmask = 255.255.255.0,
# Framed-Routing = Broadcast-Listen,
# Framed-Filter-Id = "std.ppp",
# Framed-MTU = 1500,
# Framed-Compression = Van-Jacobson-TCP-IP
#
# This is an entry for a user with a space in their name.
# Note the double quotes surrounding the name.
#
#"John Doe"      Cleartext-Password := "hello"
#                Reply-Message = "Hello, %{User-Name}"
#
#"Jefferson"     Cleartext-Password := "senac123"
#
# Dial user back and telnet to the default host for that port
#
```

➤ Jefferson = meu nome de usuário.

Cleartext-Password = declaração indicando que vamos atribuir uma senha de texto claro a esse nome de usuário. Observe que “:=” é usado para atribuição.

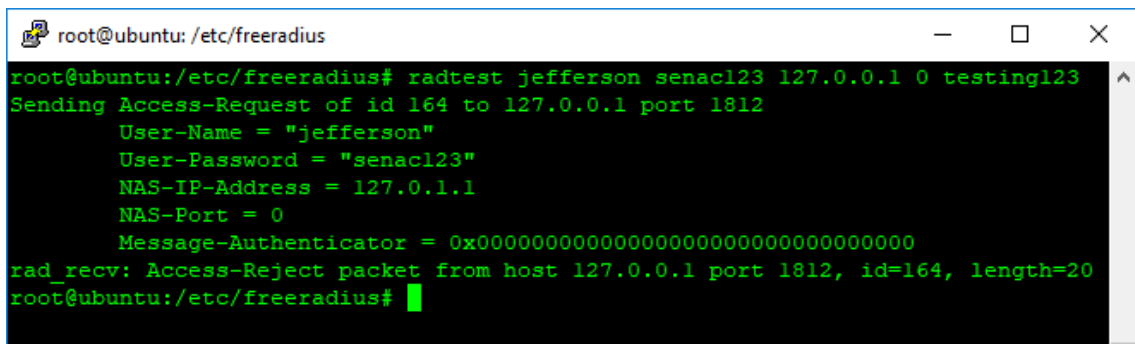
➤ senac123 = Minha senha super forte entre aspas.

Nesse cenário, os usuários são definidos em um arquivo usando senhas de texto não criptografado. Não é a coisa mais inteligente a se fazer na produção. Se você seguir esse caminho, você deve proteger este servidor muito bem.

AUTENTICAÇÃO DE TESTE

Execute um teste rápido para ver se o FreeRADIUS aceitará o nome de usuário e a senha recém-criados. Executar isto a partir do servidor significa que você terá que usar o segredo configurado para o host local que é definido no arquivo clients.conf:

\$ radtest jefferson senac123 127.0.0.1 0 testing123

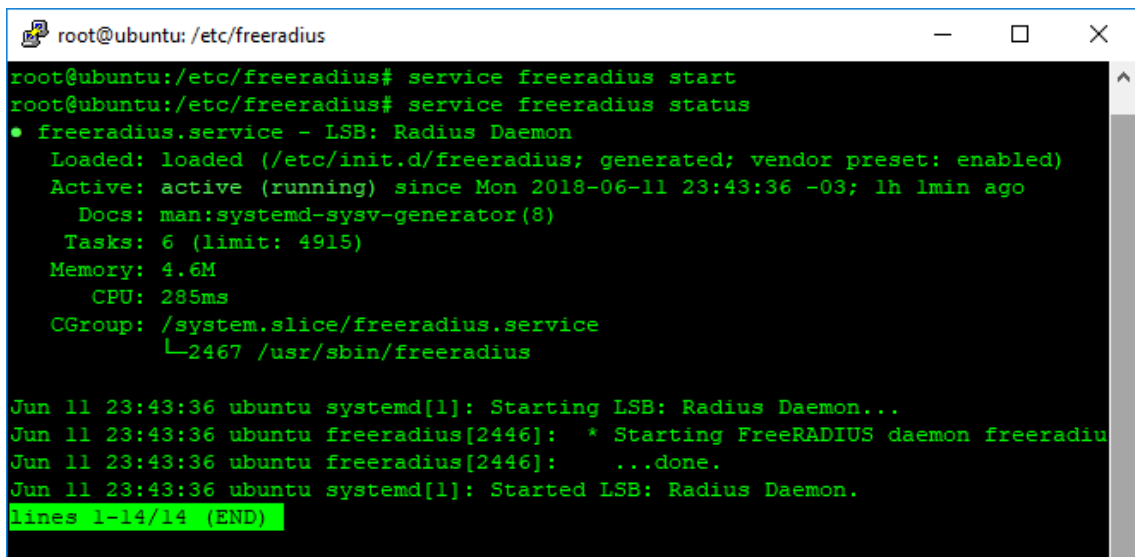


```
root@ubuntu: /etc/freeradius
root@ubuntu:/etc/freeradius# radtest jefferson senac123 127.0.0.1 0 testing123
Sending Access-Request of id 164 to 127.0.0.1 port 1812
    User-Name = "jefferson"
    User-Password = "senac123"
    NAS-IP-Address = 127.0.1.1
    NAS-Port = 0
    Message-Authenticator = 0x00000000000000000000000000000000
rad_recv: Access-Reject packet from host 127.0.0.1 port 1812, id=164, length=20
root@ubuntu:/etc/freeradius#
```

INICIE O FREERADIUS

\$ serviço freeradius start

\$ status de serviço freeradius



```
root@ubuntu: /etc/freeradius
root@ubuntu:/etc/freeradius# service freeradius start
root@ubuntu:/etc/freeradius# service freeradius status
• freeradius.service - LSB: Radius Daemon
   Loaded: loaded (/etc/init.d/freeradius; generated; vendor preset: enabled)
   Active: active (running) since Mon 2018-06-11 23:43:36 -03; 1h 1min ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 6 (limit: 4915)
   Memory: 4.6M
      CPU: 285ms
   CGroup: /system.slice/freeradius.service
           └─2467 /usr/sbin/freeradius

Jun 11 23:43:36 ubuntu systemd[1]: Starting LSB: Radius Daemon...
Jun 11 23:43:36 ubuntu freeradius[2446]: * Starting FreeRADIUS daemon freeradiu
Jun 11 23:43:36 ubuntu freeradius[2446]: ...done.
Jun 11 23:43:36 ubuntu systemd[1]: Started LSB: Radius Daemon.
lines 1-14/14 (END)
```

Agora você tem um servidor FreeRADIUS pronto para autenticar usuários em sua rede Wi-Fi. O próximo passo é configurar um ponto de acesso ou controlador para apontar para o seu servidor RADIUS. Tenha em mente que esses autenticadores estão listados no arquivo clients.conf com suas senhas. A senha será configurada no AP ou no controlador.

CONCLUSÃO

A utilização de um sistema de autenticação como o FreeRADIUS é em uma opção extremamente interessante para empresas que desejam constituir uma rede WIFI com segurança e divisão de categorias de autenticação.

Este trabalho apresentou o conceito de segurança do FreeRADIUS, sua importância e um passo a passo de como realizar sua instalação e configuração básica, sendo assim, concluímos que o leitor está completamente apto a realizar todos os procedimentos desse trabalho conseguindo obter sucesso em sua instalação e uso da ferramenta.

REFERENCIAS BIBLIOGRÁFICAS

INSTALAÇÃO E CONFIGURAÇÃO DO FREERADIUS. *GIT HUB*. Disponível em: <<https://github.com/RafaelFazzolino/ServicoRadiusLdap/wiki/Instala%C3%A7%C3%A3o-e-configura%C3%A7%C3%A3o-do-FreeRadius>>. Acesso em: 10 de Junho de 2018.

FREERADIUS – NOÇÕES PARTE I. Disponível em: <<https://www.vivaolinux.com.br/artigo/FreeRADIUS-Noco-es-basicas-Parte-I>>. Acesso em: 12 Junho de 2018.

ENTENDENDO O FREERADIUS. Disponível em: <<https://medium.com/@m0blabs/entendendo-o-freeradius-cfd788b47753>>. Acesso em: 11 Junho de 2018.

FREERADIUS - UM SERVIDOR DE AUTENTICAÇÃO GRATUITO. Disponível em: <<https://pplware.sapo.pt/linux/freeradius-um-servidor-de-autenticacao-gratuito/>>. Acesso em: 15 Junho de 2018.