Módulo 04 - Aulas 3 e 4

Tarefas

No final deste módulo você deve submeter em um ÚNICO arquivo PDF os seguintes prints:

- Atividade 4.6: passo 11.
- Atividade 4.7: passo 9.
- Atividade 4.8: passo 14.
- Atividade 4.9: passo 5.
- Atividade 4.10: passo 11.

Regras para a elaboração do documento:

1. Antes de cada Print, adicione obrigatoriamente uma frase explicativa que sinalize do que se trata o print. A inserção de prints sem a devida frase explicativa será considerada como tentativa de atrapalhar a correção do instrutor e será penalizada a critério do instrutor. Exemplo:

Print da atividade 4.6: [Imagem com o Print]

2. Os Prints devem ser tirados da TELA CHEIA. Quando tirados da VM da AWS, devem ser capturados **obrigatoriamente** da tela cheia clicando no botão "Screenshot" → "Take screenshot" da barra de ferramentas do Hypervisor da AWS.
3. Insira **somente a quantidade de Prints solicitados por atividade** usando exclusivamente 1 página por print. **A página do documento onde você vai inserir o Print deve estar com a orientação no modo PAISAGEM** ´para termos melhor aproveitamento do espaço. Ou seja, seu documento deverá ter a mesma quantidade de páginas que a quantidade do total de Prints! A inserção de prints desnecessários será considerada como tentativa de atrapalhar a correção do instrutor e será penalizada com nota 0.

4. Apresente Prints legíveis e com tamanho correto para fácil leitura. O envio de prints com letras minúsculas poderá ser considerado como tentativa de atrapalhar a correção do instrutor e será penalizada a critério do instrutor.

Atividade 4.6 – Implementando um servidor RADIUS no Kali Linux

Nesta atividade, vamos implementar um servidor RADIUS com a ferramenta FreeRADIUS no Kali Linux. O FreeRADIUS é um servidor de autenticação, autorização e contabilidade de código aberto, utilizado para implementar o protocolo RADIUS (Remote Authentication Dial-In User Service). Projetado para fornecer serviços de autenticação centralizada em redes, o FreeRADIUS permite que sistemas, como pontos de acesso Wi-Fi, switches e dispositivos de rede, autentiquem usuários remotamente. Suportando diversos métodos de autenticação, incluindo autenticação baseada em texto, EAP (Extensible Authentication Protocol) e integração com diretórios LDAP, o FreeRADIUS é amplamente utilizado em ambientes que demandam controle de acesso seguro e centralizado, como provedores de serviços de Internet, empresas e instituições educacionais. Mais informações do FreeRADIUS disponíveis em https://freeradius.org/. Todo material apresentado aqui deve ser usado somente para fins acadêmicos.

Vamos começar inicializando nosso Kali Linux via RDP ao IP: 192.168.98.40, com usuário “aluno” e senha “rnpesr”.

1. Abra o Terminal e execute o seguinte comando (com senha “rnpesr”) para ser super usuário:

```
┌──(aluno㉿kali)-[~]
└─$ sudo -i
[sudo] senha para aluno:
```

2. Verifique seu IP, principalmente a rede Loopback 127.0.0.1:

```
┌──(root㉿kali)-[~]
└─# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:df:26:13:80  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 192.168.98.40  netmask 255.255.255.0  broadcast
192.168.98.255
        inet6 fe80::1005:27ff:fe44:1631  prefixlen 64  scopeid
0x20<link>
        ether 12:05:27:44:16:31  txqueuelen 1000  (Ethernet)
        RX packets 1818194  bytes 2694820836 (2.5 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 61082  bytes 77348531 (73.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Loopback Local)
        RX packets 23  bytes 1937 (1.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23  bytes 1937 (1.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

O resultado do comando ifconfig mostra três interfaces de rede: docker0, eth0, e lo.

3. Veja os arquivos de configuração do FreeRADIUS:

```
┌──(root㉿kali)-[~]
└─# ls  /etc/freeradius/3.0
certs         experimental.conf  mods-available  panic.gdb
radiusd.conf     sites-enabled   users
clients.conf  hints              mods-config     policy.d    README.rst
templates.conf
dictionary    huntgroups         mods-enabled    proxy.conf  sites-
available  trigger.conf
```

4. Veja os clientes pré-configurados:

```
┌──(root㉿kali)-[~]
└─# cat /etc/freeradius/3.0/clients.conf
```

5. Perceba que há o cliente “client localhost” pré-configurado e conta com a senha testing123:

```
        #  The default secret below is only for testing, and should
        #  not be used in any real environment.
        #
        secret = testing123
```

Usaremos esse cliente para verificar se o servidor RADIUS está rodando.

6. Crie o usuário que poderá ter acesso.

```
┌──(root㉿kali)-[~]
└─# nano /etc/freeradius/3.0/users
```

7. Abaixo do usuário comentado “bob”, crie o usuário usuario1 com as seguintes características:

```
#bob    Cleartext-Password := "hello"
#       Reply-Message := "Hello, %{User-Name}"
#

usuario1 Cleartext-Password := "rnpesr"
```

Aperte “Ctrl + x”, “s” e “ENTER” para sair e salvar.

8. Coloque o FreeRADIUS para rodar (importante, é X **maiúscula**, não minúscula):

```
┌──(root㉿kali)-[~]
└─# freeradius -X
FreeRADIUS Version 3.2.3
Copyright (C) 1999-2022 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
Starting - reading configuration files ...
including dictionary file /usr/share/freeradius/dictionary
including dictionary file /usr/share/freeradius/dictionary.dhcp

...

listen {
        type = "acct"
        ipv6addr = ::
        port = 0
   limit {
        max_connections = 16
        lifetime = 0
        idle_timeout = 30
   }
}
listen {
        type = "auth"
        ipaddr = 127.0.0.1
        port = 18120
}
Listening on auth address * port 1812 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on auth address :: port 1812 bound to server default
Listening on acct address :: port 1813 bound to server default
Listening on auth address 127.0.0.1 port 18120 bound to server inner-
tunnel
Listening on proxy address * port 43055
Listening on proxy address :: port 36998
Ready to process requests
```

Veja que o servidor FreeRADIUS está em funcionamento e aguardando requisições.

9. Abra um segundo Terminal e estabeleça um teste de conexão com a rede Loopback:

```
┌──(aluno㉿kali)-[~]
└─$ sudo -i
[sudo] senha para aluno:

┌──(root㉿kali)-[~]
└─# radtest usuario1 rnpesr 127.0.0.1 0 testing123
Sent Access-Request Id 246 from 0.0.0.0:af56 to 127.0.0.1:1812 length
78
        User-Name = "usuario1"
        User-Password = "rnpesr"
        NAS-IP-Address = 127.0.1.1
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "rnpesr"
Received Access-Accept Id 246 from 127.0.0.1:714 to 127.0.0.1:44886
length 20
Message-Authenticator = 0xca5dfd87ba2b797e16edfe5c7ecb1d40
```

Vaja os parâmetros do comando:

- **radtest**: Este comando é utilizado para testar a autenticação em um servidor RADIUS (Remote Authentication Dial-In User Service).
- **usuario1**: Nome do usuário a ser autenticado.
- **rnpesr**: Senha do usuário a ser usada para autenticação.
- **127.0.0.1**: Endereço IP do servidor RADIUS.
- **0**: Número de tentativas de autenticação.
- **testing123**: Uma senha secreta compartilhada entre o cliente e o servidor RADIUS para garantir a integridade da comunicação.

A saída do comando gerou o seguinte:

- **Sent Access-Request...**: Indica que uma solicitação de acesso foi enviada ao servidor RADIUS contendo informações como o nome de usuário, senha, endereço IP do NAS (Network Access Server), porta NAS e outros detalhes da requisição.
- **Received Access-Accept...**: Indica que o servidor RADIUS respondeu com sucesso à solicitação de acesso, concedendo permissão para o usuário se autenticar. O código Access-Accept é uma confirmação de que a autenticação foi bem-sucedida.

10. Volte ao Terminal 1 e veja os logs da conexão estabelecida:

```
Ready to process requests
(0) Received Access-Request Id 246 from 127.0.0.1:40968 to
127.0.0.1:1812 length 78
(0)   Message-Authenticator = 0x80ae90efd30a3e6e0d4f9cac1c5a16b6
(0)   User-Name = "usuario1"
(0)   User-Password = "rnpesr"
(0)   NAS-IP-Address = 127.0.1.1
(0)   NAS-Port = 0
(0) # Executing section authorize from file /etc/freeradius/3.0/sites-
enabled/default
(0)   authorize {
(0)     policy filter_username {
(0)       if (&User-Name) {
(0)       if (&User-Name)  -> TRUE
(0)       if (&User-Name)  {
(0)         if (&User-Name =~ / /) {
(0)         if (&User-Name =~ / /)  -> FALSE
(0)         if (&User-Name =~ /@[^@]*@/ ) {
(0)         if (&User-Name =~ /@[^@]*@/ )  -> FALSE
(0)         if (&User-Name =~ /\.\./ ) {
(0)         if (&User-Name =~ /\.\./ )  -> FALSE
(0)         if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/))
{
(0)         if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/))
-> FALSE
(0)         if (&User-Name =~ /\.$/)  {
(0)         if (&User-Name =~ /\.$/)   -> FALSE
(0)         if (&User-Name =~ /@\./)  {
(0)         if (&User-Name =~ /@\./)   -> FALSE
(0)       } # if (&User-Name)  = notfound
(0)     } # policy filter_username = notfound
(0)     [preprocess] = ok
(0)     [chap] = noop
(0)     [mschap] = noop
(0)     [digest] = noop
(0) suffix: Checking for suffix after "@"
(0) suffix: No '@' in User-Name = "usuario1", looking up realm NULL
(0) suffix: No such realm "NULL"
(0)     [suffix] = noop
(0) eap: No EAP-Message, not doing EAP
(0)     [eap] = noop
(0) files: users: Matched entry usuario1 at line 91
(0)     [files] = ok
(0)     [expiration] = noop
(0)     [logintime] = noop
(0)     [pap] = updated
(0)   } # authorize = updated
(0) Found Auth-Type = PAP
```

```
(0) Found Auth-Type = PAP
(0) # Executing group from file /etc/freeradius/3.0/sites-enabled/
default
(0)   Auth-Type PAP {
(0) pap: Login attempt with password
(0) pap: Comparing with "known good" Cleartext-Password
(0) pap: User authenticated successfully
(0)     [pap] = ok
(0)   } # Auth-Type PAP = ok
(0) # Executing section post-auth from file /etc/freeradius/3.0/sites-
enabled/default
(0)   post-auth {
(0)     if (session-state:User-Name && reply:User-Name && request:User-
Name && (reply:User-Name == request:User-Name)) {
(0)     if (session-state:User-Name && reply:User-Name && request:User-
Name && (reply:User-Name == request:User-Name))  -> FALSE
(0)     update {
(0)       No attributes updated for RHS &session-state:
(0)     } # update = noop
(0)     [exec] = noop
(0)     policy remove_reply_message_if_eap {
(0)       if (&reply:EAP-Message && &reply:Reply-Message) {
(0)       if (&reply:EAP-Message && &reply:Reply-Message)  -> FALSE
(0)       else {
(0)         [noop] = noop
(0)       } # else = noop
(0)     } # policy remove_reply_message_if_eap = noop
(0)     if (EAP-Key-Name && &reply:EAP-Session-Id) {
(0)     if (EAP-Key-Name && &reply:EAP-Session-Id)  -> FALSE
(0)   } # post-auth = noop
(0) Sent Access-Accept Id 246 from 127.0.0.1:1812 to 127.0.0.1:44886
length 20
(0) Finished request
Waking up in 4.9 seconds.
(0) Cleaning up request packet ID 246 with timestamp +69 due to
cleanup_delay was reached
Ready to process requests
```

A saída fornecida mostra o processo de autenticação e autorização de um usuário ("usuario1") no servidor FreeRADIUS:

- **Ready to process requests**: Indica que o servidor FreeRADIUS está pronto para processar solicitações.
- **(0) Received Access-Request...**: Informa que uma solicitação de acesso foi recebida do endereço IP 127.0.0.1 na porta 44886.
- **(0) User-Name = "usuario1"**: Indica o nome de usuário incluído na solicitação.

- **(0) User-Password = "rnpesr"**: Mostra a senha incluída na solicitação, que será utilizada para autenticação.
- **(0) Found Auth-Type = PAP**: Indica que o método de autenticação PAP (Password Authentication Protocol) foi selecionado.
- **(0) Auth-Type PAP {**: Inicia o bloco de configuração para o método de autenticação PAP.
- **(0) pap: Login attempt with password**: Sinaliza uma tentativa de login utilizando o método PAP com senha.
- **(0) pap: User authenticated successfully**: Confirma que o usuário foi autenticado com sucesso utilizando o método PAP.
- **(0) Sent Access-Accept...**: Indica que uma resposta de aceitação de acesso foi enviada para o cliente.
- **(0) Finished request**: Indica que o processamento da solicitação foi concluído.
- **Waking up in 4.9 seconds.**: Informa que o servidor entrará em um estado de espera antes de processar a próxima solicitação.
- **(0) Cleaning up request packet ID 246 with timestamp +69...**: Refere-se à limpeza de recursos associados à solicitação identificada pelo ID 246.
- **Ready to process requests**: Indica novamente que o servidor FreeRADIUS está pronto para processar solicitações.

11. Volte ao Terminal 2 e tente novamente o acesso com um usuário errado, neste exemplo é o “usuarioX” (**NÃO SE ESQUEÇA DE PRINTAR ESTE PASSO!**):

```
┌──(root㉿kali)-[~]
└─# radtest usuarioX rnpesr 127.0.0.1 0 testing123
Sent Access-Request Id 211 from 0.0.0.0:bd19 to 127.0.0.1:1812 length
78
        User-Name = "usuarioX"
        User-Password = "rnpesr"
        NAS-IP-Address = 127.0.1.1
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "rnpesr"
Received Access-Reject Id 211 from 127.0.0.1:714 to 127.0.0.1:60653
length 20
        Message-Authenticator = 0xf9a00ead8e578adcede7fa8190a3bffb
(0) -: Expected Access-Accept got Access-Reject
```

Veja que o acesso foi rejeitado:

- **Received Access-Reject...**: O servidor RADIUS respondeu com uma mensagem de rejeição de acesso (Access-Reject). O ID 211 corresponde ao ID da solicitação original. O endereço IP de origem é 127.0.0.1:714, e o destino é 127.0.0.1:60653. A resposta é do tipo Access-Reject, indicando que a autenticação falhou.
- **(0) -: Expected Access-Accept got Access-Reject**: Esta linha resume o resultado da autenticação. O número entre parênteses (0) indica que a solicitação foi a primeira (ou única). O sinal de menos (-) indica que essa entrada no log não está associada a uma seção específica. A mensagem "Expected Access-Accept got Access-Reject" indica que o servidor esperava uma resposta positiva (Access-Accept) mas recebeu uma resposta negativa (Access-Reject), indicando que a autenticação falhou.

12. Volte ao Terminal 1 e veja os logs desse acesso rejeitado:

```
Ready to process requests
(1) Received Access-Request Id 211 from 127.0.0.1:60653 to
127.0.0.1:1812 length 78
(1)   Message-Authenticator = 0x82d0c1a625c7db2b6d59b94efb8b37ba
(1)   User-Name = "usuarioX"
(1)   User-Password = "rnpesr"
(1)   NAS-IP-Address = 127.0.1.1
(1)   NAS-Port = 0
(1) # Executing section authorize from file /etc/freeradius/3.0/sites-
enabled/default
(1)   authorize {
(1)     policy filter_username {
(1)       if (&User-Name) {
(1)       if (&User-Name)  -> TRUE
(1)       if (&User-Name)  {
(1)         if (&User-Name =~ / /) {
(1)         if (&User-Name =~ / /)  -> FALSE
(1)         if (&User-Name =~ /@[^@]*@/ ) {
(1)         if (&User-Name =~ /@[^@]*@/ )  -> FALSE
(1)         if (&User-Name =~ /\.\./ ) {
(1)         if (&User-Name =~ /\.\./ )  -> FALSE
(1)         if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/))
{
(1)         if ((&User-Name =~ /@/) && (&User-Name !~ /@(.+)\.(.+)$/))
-> FALSE
(1)         if (&User-Name =~ /\.$/)  {
(1)         if (&User-Name =~ /\.$/)   -> FALSE
(1)         if (&User-Name =~ /@\./)  {
(1)         if (&User-Name =~ /@\./)   -> FALSE
(1)       } # if (&User-Name)  = notfound
(1)     } # policy filter_username = notfound
(1)     [preprocess] = ok
(1)     [chap] = noop
(1)     [mschap] = noop
(1)     [digest] = noop
(1) suffix: Checking for suffix after "@"
(1) suffix: No '@' in User-Name = "usuarioX", looking up realm NULL
(1) suffix: No such realm "NULL"
(1)     [suffix] = noop
(1) eap: No EAP-Message, not doing EAP
(1)     [eap] = noop
(1)     [files] = noop
(1)     [expiration] = noop
(1)     [logintime] = noop
(1) pap: WARNING: No "known good" password found for the user.  Not
setting Auth-Type
(1) pap: WARNING: Authentication will fail unless a "known good"
password is available
```

```
password is available
(1)     [pap] = noop
(1)   } # authorize = ok
(1) ERROR: No Auth-Type found: rejecting the user via Post-Auth-Type =
Reject
(1) Failed to authenticate the user
(1) Using Post-Auth-Type Reject
(1) # Executing group from file /etc/freeradius/3.0/sites-enabled/
default
(1)   Post-Auth-Type REJECT {
(1) attr_filter.access_reject: EXPAND %{User-Name}
(1) attr_filter.access_reject:    --> usuarioX
(1) attr_filter.access_reject: Matched entry DEFAULT at line 11
(1)     [attr_filter.access_reject] = updated
(1)     [eap] = noop
(1)     policy remove_reply_message_if_eap {
(1)       if (&reply:EAP-Message && &reply:Reply-Message) {
(1)       if (&reply:EAP-Message && &reply:Reply-Message)  -> FALSE
(1)       else {
(1)         [noop] = noop
(1)       } # else = noop
(1)     } # policy remove_reply_message_if_eap = noop
(1)   } # Post-Auth-Type REJECT = updated
(1) Delaying response for 1.000000 seconds
Waking up in 0.3 seconds.
Waking up in 0.6 seconds.
(1) Sending delayed response
(1) Sent Access-Reject Id 211 from 127.0.0.1:1812 to 127.0.0.1:60653
length 20
Waking up in 3.9 seconds.
(1) Cleaning up request packet ID 211 with timestamp +412 due to
cleanup_delay was reached
Ready to process requests
```

O erro de autenticação fica aparente:

- **(1) pap: WARNING: No "known good" password found for the user. Not setting Auth-Type**: O FreeRADIUS emite um aviso indicando que não foi encontrado uma senha válida ("known good") para o usuário em questão.O Auth-Type (método de autenticação) não será definido porque a senha não está disponível.
- **(1) pap: WARNING: Authentication will fail unless a "known good" password is available**: Um segundo aviso enfatiza que a autenticação falhará, a menos que uma senha válida seja configurada para o usuário. Sem uma senha conhecida e válida, o processo de autenticação não pode ser concluído com sucesso.

- **(1) [pap] = noop**: Indica que, devido à falta de uma senha conhecida, a seção PAP (Password Authentication Protocol) não realiza nenhuma operação (noop).
- **(1) } # authorize = ok**: A seção de autorização é concluída com sucesso, mas a falta de uma senha válida significa que a autenticação não pode prosseguir.
- **(1) ERROR: No Auth-Type found: rejecting the user via Post-Auth-Type = Reject**: O FreeRADIUS relata um erro, indicando que nenhum tipo de autenticação (Auth-Type) foi encontrado devido à ausência de uma senha válida. O usuário será rejeitado usando a etapa de pós-autenticação (Post-Auth-Type) configurada como "Reject".
- **(1) Failed to authenticate the user**: Conclui que a tentativa de autenticação para o usuário falhou devido à falta de uma senha conhecida e válida.

13. Feche os Terminais.

Parabéns! Agora você sabe como subir e testar um servidor FreeRADIUS no Kali Linux!

Atividade 4.7 – Explorando o Google Authenticator no Kali Linux

Nesta atividade, vamos integrar o Google Authenticator (você precisa usar um Smart Phone) no Kali Linux por meio da utilização de TOTP (Time-based One-Time Password) via QR Code. O Google Authenticator é uma aplicação de autenticação de dois fatores desenvolvida pelo Google, projetada para reforçar a segurança dos serviços online. Funciona gerando códigos temporários de seis dígitos que mudam a cada 30 segundos, proporcionando uma camada adicional de proteção além das senhas convencionais. Os usuários vinculam suas contas online ao Google Authenticator, e, durante o processo de login, precisam fornecer não apenas suas credenciais de senha, mas também o código único gerado pela aplicação em seus dispositivos móveis. Todo material apresentado aqui deve ser usado somente para fins acadêmicos.

Vamos começar inicializando nosso Kali Linux via RDP ao IP: 192.168.98.40, com usuário “aluno” e senha “rnpesr”.

1. Instale o Google Authenticator no seu celular!
2. Abra o Terminal e execute o seguinte comando (com senha “rnpesr”) para ser super usuário:

```
┌──(aluno㉿kali)-[~]
└─$ sudo -i
[sudo] senha para aluno:
```

3. Por precaução e para evitar erros, defina o tempo e zona do Brasil e verifique a alteração

```
┌──(root㉿kali)-[~]
└─# timedatectl set-timezone America/Sao_Paulo

┌──(root㉿kali)-[~]
└─# timedatectl
               Local time: sáb 2025-09-06 20:46:18 -03
           Universal time: sáb 2025-09-06 23:46:18 UTC
                 RTC time: sáb 2025-09-06 23:46:18
                Time zone: America/Sao_Paulo (-03, -0300)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

4. Inicialize o Google Authenticator no Kali Linux:

```
┌──(root㉿kali)-[~]
└─# google-authenticator
```

5. Maximize a janela do Terminal e ative o TOTP (aperte “y”) e veja que é apresentado um QR Code no Terminal:

```
Do you want authentication tokens to be time-based (y/n) y
Warning: pasting the following URL into your browser exposes the OTP
secret to Google:
  https://www.google.com/chart?chs=200x200&chld=M|
0&cht=qr&chl=otpauth://totp/
root@kali%3Fsecret%3DRPWPLFITJZIYYKKMFUNF2ASJVQ%26issuer%3Dkali

...

Your new secret key is: RPWPLFITJZIYYKKMFUNF2ASJVQ
```

6. Veja que um QR Code é apresentado no Terminal.
7. No celular, abra o Google Authenticator, clique no botão colorido “+” e em “Faça a leitura de um QR code”. A câmera será aberta.
8. Aponte a câmera ao QR Code apresentado no Terminal (passo 6). Agora, sua sessão no Google Authenticator está gerando TOTP conforme apresentado no seu celular.
9. Volte ao terminal e escreva o TOTP apresentado no Google Authenticator do seu celular (**NÃO SE ESQUEÇA DE PRINTAR ESTE PASSO!**):

```
Your new secret key is: RPWPLFITJZIYYKKMFUNF2ASJVQ
Enter code from app (-1 to skip): 987324
Code confirmed
Your emergency scratch codes are:
  23897781
  15339847
  15649329
  87445579
  85367750
```

10. Aceite as próximas solicitações (escreva “y”):

```
Do you want me to update your "/root/.google_authenticator" file? (y/n)
y

Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it
increases
your chances to notice or even prevent man-in-the-middle attacks (y/n)
y

By default, a new token is generated every 30 seconds by the mobile
app.
In order to compensate for possible time-skew between the client and
the server,
we allow an extra token before and after the current time. This allows
for a
time skew of up to 30 seconds between authentication server and client.
If you
experience problems with poor time synchronization, you can increase
the window
from its default size of 3 permitted codes (one previous code, the
current
code, the next code) to 17 permitted codes (the 8 previous codes, the
current
code, and the 8 next codes). This will permit for a time skew of up to
4 minutes
between client and server.
Do you want to do so? (y/n) y

If the computer that you are logging into isn't hardened against brute-
force
login attempts, you can enable rate-limiting for the authentication
module.
By default, this limits attackers to no more than 3 login attempts
every 30s.
Do you want to enable rate-limiting? (y/n) y
```

Os parâmetros foram para:

- **Do you want me to update your '/root/.google_authenticator' file? (y/n) y"**: Essa solicitação pergunta se você deseja que o Google Authenticator atualize o arquivo de configuração específico do usuário localizado em /root/.google_authenticator. Responder 'y' (sim) indica que você deseja que as configurações do Google Authenticator para esse usuário sejam atualizadas.

- **"Do you want to disallow multiple uses of the same authentication token? ... (y/n) y"**: Essa pergunta refere-se à política de permitir ou não o uso múltiplo do mesmo token de autenticação. Responder 'y' (sim) significa que você deseja desabilitar o uso múltiplo do mesmo token, limitando a um login a cada 30 segundos. Isso aumenta a segurança, ajudando a detectar ou prevenir possíveis ataques do tipo man-in-the-middle.
- **"By default, a new token is generated every 30 seconds... Do you want to do so? (y/n) y"**: Essa pergunta trata da geração frequente de novos tokens a cada 30 segundos pelo aplicativo móvel. Se você enfrentar problemas de sincronização de tempo entre o cliente e o servidor, pode aumentar a janela de tempo permitida para até 17 códigos. Responder 'y' (sim) aumentará a janela de tempo para compensar eventuais discrepâncias de até 4 minutos entre o cliente e o servidor.
- **"If the computer that you are logging into isn't hardened against brute-force... Do you want to enable rate-limiting? (y/n) y"**: Esta solicitação pergunta se você deseja habilitar o limite de taxa para o módulo de autenticação, limitando os atacantes a não mais que 3 tentativas de login a cada 30 segundos. Responder 'y' (sim) significa que você deseja habilitar o controle de taxa para proteger contra tentativas de login por força bruta, especialmente se a máquina não estiver configurada para resistir a esses ataques.

11. Feche o Terminal.

Parabéns! Agora você configurou o Google Authenticator com o Kali Linux!

Atividade 4.8 – Incorporando o Google Authenticator ao Mozilla Firefox

Nesta atividade, vamos incorporar o Google Authenticator no Mozilla Firefox usando Plugin no Kali Linux. Veremos como pode ser utilizado o Google Authenticator sem a necessidade do uso de um celular ou conta Google.

Nesta atividade usaremos 2 ambientes: seu computador pessoal e a VM Kali Linux da AWS.

1. No seu computador pessoal (não use a VM do laboratório), abra o Mozilla Firefox. Caso não o tenha, realize a instalação. Insira o endereço:

```
https://addons.mozilla.org/en-US/firefox/addon/auth-helper/
```

2. Ao lado da mensagem “Authenticator by mymindstorm”, clique no botão azul “Add to Firefox”.
3. Clique em “Adicionar/Add” e clique em “OK” no aviso que mostra “Autenticador foi adicionado”.
4. Veja que uma extensão nova foi adicionada ao lado direito da barra de navegação do Mozilla Firefox (botão “Extensões”).
5. Entre na VM Landing do nosso laboratório e inicialize o Kali Linux via RDP ao IP: 192.168.98.40, com usuário “aluno” e senha “rnpesr”. Abra o Terminal e execute o seguinte comando (com senha “rnpesr”) para ser super usuário:

```
┌──(aluno㉿kali)-[~]
└─$ sudo -i
[sudo] senha para aluno:
```

6. Inicialize o Google Authenticator no Kali Linux:

```
┌──(root㉿kali)-[~]
└─# google-authenticator
```

7. Ative o TOTP e veja que é apresentada uma chave secreta no Terminal:

```
Do you want authentication tokens to be time-based (y/n) y
Warning: pasting the following URL into your browser exposes the OTP
secret to Google:
  https://www.google.com/chart?chs=200x200&chld=M|
0&cht=qr&chl=otpauth://totp/
root@kali%3Fsecret%3DN474TDDVIGUWZO3EMZP3UXUG3Y%26issuer%3Dkali

...

Your new secret key is: N474TDDVIGUWZO3EMZP3UXUG3Y
```

8. Copie a chave secreta mostrada na última linha do passo anterior para seu computador pessoal por meio da ferramenta “Clipboard” do Remote Connection.

9. Minimize a aba que contém a VM da AWS e, no seu computador, volte ao Firefox. Clique no ícone do Authenticator do passo 4.
10. Clique no símbolo do lápis e no símbolo “+”.
11. Clique em “Inserir Manualmente”.
12. No campo “Emissor”, insira um nome descritivo, como por exemplo, “Kali Linux”.
13. No campo “Segredo”, cole a chave secreta copiada no item 8 e clique em OK.
14. Clique no ícone do Authenticator do passo 4. Veja que o código do Google Authenticator gerado no Terminal do Kali Linux está agora no seu navegador sem a necessidade do uso de um celular (**NÃO SE ESQUEÇA DE PRINTAR ESTE PASSO!**).
15. Feche o Firefox, volte à VM da AWS com Kali Linux e feche o Terminal.

Parabéns! Você sabe como vincular o Google Authenticator no Mozilla Firefox sem a necessidade de um celular!

Atividade 4.9 – Verificando a presença de TPM e USB no Kali Linux

Nesta atividade, vamos verificar a presença do módulo TPM e de portas USB que possam servir como meio de entrada de Tokens no Kali Linux.

Vamos começar inicializando nosso Kali Linux via RDP ao IP: 192.168.98.40, com usuário “aluno” e senha “rnpesr”.

1. Abra o Terminal e execute o seguinte comando (com senha “rnpesr”) para ser super usuário:

```
┌──(aluno㉿kali)-[~]
└─$ sudo -i
[sudo] senha para aluno:
```

2. Verifique a presenta do módulo TPM (ausente nesta máquina virtual):

```
┌──(root㉿kali)-[~]
└─# journalctl -k --grep=tpm
```

3. Veja que a seguinte mensagem é apresentada (aperte “Ctrl + C” após apresentada a mensagem):

```
fev 09 22:01:37 kali kernel: ima: No TPM chip found, activating TPM-
bypass!
fev 09 22:01:37 kali systemd[1]: systemd 252.6-1 running in system mode
(+PAM +AUDIT +SELINUX +APPARMOR +IMA +>
```

A saída do comando significa:

- **fev 09 22:01:37 kali kernel**: Indica a data, hora e nome do sistema onde a mensagem foi registrada, seguido pelo identificador do processo que gerou a mensagem (nesse caso, o kernel).
- **ima: No TPM chip found, activating TPM-bypass!**: Esta mensagem indica que o IMA (Integrity Measurement Architecture) não encontrou um chip TPM (Trusted Platform Module) no sistema e, portanto, ativou o modo de bypass do TPM. O TPM é um componente de hardware que oferece recursos de segurança, como armazenamento de chaves criptográficas e medidas de integridade do sistema.
- **fev 09 22:01:37 kali systemd[1]**: Esta linha indica que a mensagem foi gerada pelo systemd, que é o gerenciador de sistema do Linux, responsável por inicializar o sistema e gerenciar processos.
- **systemd 252.6-1 running in system mode (+PAM +AUDIT +SELINUX +APPARMOR +IMA +...**: Esta é uma mensagem informativa do systemd, indicando a versão do systemd em execução no sistema e os recursos compilados nele. Neste caso, ele inclui suporte para PAM (Pluggable Authentication Modules), auditoria de eventos, SELinux, AppArmor, IMA (Integrity Measurement Architecture), entre outros.

4. Explore as portas USB presentes:

```
┌──(root㉿kali)-[~]
└─# exit

┌──(aluno㉿kali)-[~]
└─$ usbview
/sys/bus/usb/devices/ must be present, exiting...
```

5. Veja que o programa “USB Viewer” é aberto (**NÃO SE ESQUEÇA DE PRINTAR ESTE PASSO!**).
6. Perceba também que o programa não apresenta dispositivos USB. Isso acontece porque estamos em um ambiente virtual!
7. Feche o programa “USB Viewer” e o Terminal.

Parabéns! Agora você sabe como visualizar o TPM e USB presente no Kali Linux! Considere testar no seu computador pessoal!

Atividade 4.10 – Gerenciando TOTPs com o OTPClient no Kali Linux

Nesta atividade, vamos dar continuidade a atividade anterior para estabelecer uma conexão SSH usando 2FA no Kali Linux através do Google Authenticator.

Vamos começar inicializando nosso Kali Linux via RDP ao IP: 192.168.98.40, com usuário “aluno” e senha “rnpesr”.

1. Abra o Terminal e execute o seguinte comando para inicializar o OTPClient:

```
┌──(aluno㉿kali)-[~]
└─$ otpclient
[WARNING] your OS's memlock limit may be too low for you (current
value: 8388608 bytes).
This may cause issues when importing third parties databases or dealing
with tens of tokens.
For information on how to increase the memlock value, please have a
look at https://github.com/paolostivanin/OTPClient/wiki/Secure-Memory-
Limitations
```

2. Clique em “OK” caso a janela “Warning: memlock value too low” apareça. O OTPClient abrirá. Clique em "Create new database".
3. Será aberta uma tela para salvar arquivo. Acesse a pasta “Documentos” e deixe o nome da base de dados como "NewDatabase.enc". Clique em "OK".
4. Inserira e confirme a senha "rnpesr" ao visualizar a seguinte mensagem, a seguir clique em “OK”:

```
Choose an encryption password for the database

Please note that there is no way to recover a forgotten password.
```

5. Caso a janela “Unlock Login Keyring” apareça, insira as credenciais do Kali “rnpesr” e clique em “Unlock”. Clique em “OK” na janela que apareceu com o comentário (*This is the first time you run OTPClient...*). Será aberta a janela do OTPClient.
6. Abra um segundo Terminal e inicialize o Google Authenticator no Kali Linux:

```
┌──(aluno㉿kali)-[~]
└─$ google-authenticator
```

7. Ative o TOTP e veja que uma chave de segredo é apresentado no Terminal:

```
Do you want authentication tokens to be time-based (y/n) y
Warning: pasting the following URL into your browser exposes the OTP
secret to Google:
  https://www.google.com/chart?chs=200x200&chld=M|
0&cht=qr&chl=otpauth://totp/
root@kali%3Fsecret%3DRPWPLFITJZIYYKKMFUNF2ASJVQ%26issuer%3Dkali

...

Your new secret key is: RPWPLFITJZIYYKKMFUNF2ASJVQ
```

8. Volte ao OTPClient e clique no botão superior esquerdo “+”. Clique em “Manually”.
9. Insira qualquer nome nos campos “Account” e “Issuer”.
10. No campo “Secret”, insira a chave de segredo apresentado no passo 7.

11. Clique na linha do token que começa com “TOTP” e veja que o Token do Google Authenticator conta com os parâmetros válidos “Type”, “Account, “Issuer”, “OTP Value” e “Validity” (**NÃO SE ESQUEÇA DE PRINTAR ESTE PASSO!**).
12. Feche o OTPClient. No primeiro Terminal, aperte “Ctrl + C” e navegue até a pasta Documentos. Em seguida, apague os arquivos criados.

```
┌──(aluno㉿kali)-[~]
└─$ sudo -i
[sudo] senha para aluno:

┌──(root㉿kali)-[~]
└─# cd /home/aluno/Documentos

┌──(root㉿kali)-[/home/aluno/Documentos]
└─# ls
NewDatabase.enc  NewDatabase.enc.bak QRcode.png

┌──(root㉿kali)-[/home/aluno/Documentos]
└─# rm *
zsh: sure you want to delete all 3 files in /home/aluno/Documentos
[yn]? y
```

13. Feche os Terminais.

Parabéns! Agora você sabe como usar o OTPClient para gerenciar softwares de Tokens no Kali Linux!