



Hackers do Bem – Fundamental
Prof. Fábio Carneiro de Castro
18/02/2026

Atividade Prática – Módulo 4

Aulas 3 e 4

Gabriel dos Santos Schmitz

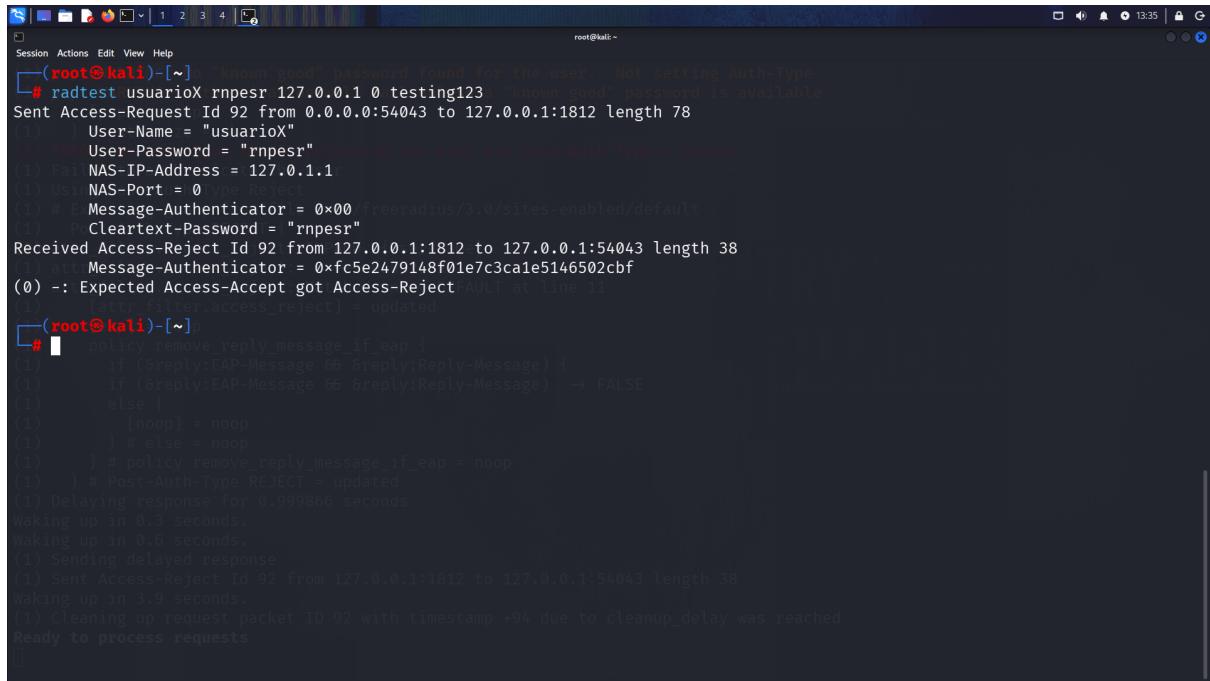
1 Introdução

Este documento apresenta as evidências práticas das atividades do Módulo 04 (Aulas 3 e 4) do Programa Hackers do Bem – Nível Fundamental, por meio dos prints solicitados, demonstrando a correta execução das tarefas propostas.

Conforme orientações do curso, este documento reúne, em um único arquivo PDF, os seguintes prints obrigatórios: Atividade 4.6 (passo 11), Atividade 4.7 (passo 9), Atividade 4.8 (passo 14), Atividade 4.9 (passo 5) e Atividade 4.10 (passo 11), todos acompanhados de breve descrição explicativa para facilitar a avaliação pelo instrutor.

2 Atividades

Atividade 4.6. Implementando um servidor RADIUS com FreeRADIUS no Kali Linux



The screenshot shows a terminal window on a Kali Linux desktop environment. The title bar says "root@kali: ~". The terminal content is as follows:

```
(root@kali)-[~]o "known good" password found for the user. Not setting Auth-Type
└─# radtest usuarioX rnpesr 127.0.0.1 0 testing123 "known good" password is available
Sent Access-Request Id 92 from 0.0.0.0:54043 to 127.0.0.1:1812 length 78
(1)   User-Name = "usuarioX"
(1)   User-Password = "rnpesr" Rejected the user via Post-Auth-Type = Reject
(1)   NAS-IP-Address = 127.0.0.1
(1)   USN = 
(1)   NAS-Port = 0
(1)   # E Message-Authenticator = 0x00 freeradius/3.0/sites-enabled/default
(1)   Cleartext-Password = "rnpesr"
Received Access-Reject Id 92 from 127.0.0.1:1812 to 127.0.0.1:54043 length 38
(1)   att Message-Authenticator = 0xfc5e2479148f01e7c3cale5146502cbf
(0) -: Expected Access-Accept got Access-RejectFAULT at line 11
(1)   attr_filter.access_reject = updated
└─# ┌─ 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 0.999866 seconds
Waking up in 0.3 seconds.
Waking up in 0.6 seconds.
(1)   Sending delayed response
(1)   Sent Access-Reject Id 92 from 127.0.0.1:1812 to 127.0.0.1:54043 length 38
Waking up in 3.9 seconds.
(1)   Cleaning up request packet ID 92 with timestamp +94 due to cleanup_delay was reached
Ready to process requests
```

Fig. 1: Servidor FreeRADIUS em execução e teste de autenticação utilizando o comando radtest

Sobre:

Nesta atividade, foi realizada a implementação de um servidor de autenticação centralizada utilizando a ferramenta **FreeRADIUS** no sistema *Kali Linux*, com o objetivo de explorar o funcionamento do protocolo RADIUS (*Remote Authentication Dial-In User Service*).

Inicialmente, foi acessado o ambiente via RDP e obtidos privilégios de superusuário por meio do comando `sudo -i`. Em seguida, foi realizada a verificação das interfaces de rede com o comando `ifconfig`, destacando a interface de loopback `127.0.0.1`, utilizada para testes locais de autenticação.

Posteriormente, foram analisados os arquivos de configuração do serviço no diretório `/etc/freeradius/3.0/`, bem como o arquivo `clients.conf`, onde foi identificado o cliente padrão `localhost` com o segredo compartilhado `testing123`, utilizado para testes.

Na sequência, foi criado um usuário para autenticação no arquivo `users`, definindo o usuário `usuario1` com a senha `rnpesr`. Após a configuração, o servidor foi iniciado em modo de depuração com o comando `freeradius -X`, permitindo acompanhar os logs em tempo real.

Para validar o funcionamento do servidor, foi utilizado o comando `radtest usuario1 rnpesr 127.0.0.1 0 testing123`, que realiza uma requisição de autenticação ao servidor RADIUS. O retorno *Access-Accept* confirmou que a autenticação foi realizada com sucesso, indicando que o servidor estava corretamente configurado.

Em seguida, foi realizado um teste com credenciais inválidas, utilizando um usuário inexistente. Nesse caso, o servidor retornou a resposta *Access-Reject*, evidenciando a negação de acesso. A análise dos logs mostrou que a falha ocorreu devido à ausência de uma senha válida cadastrada, impedindo a definição do método de autenticação.

Dessa forma, a atividade demonstrou na prática o funcionamento do processo de autenticação, autorização e controle de acesso utilizando o protocolo RADIUS, bem como a importância da correta configuração de usuários e clientes para o funcionamento seguro do serviço.

Atividade 4.7. Explorando o Google Authenticator no Kali Linux



Fig. 2: Configuração do TOTP com Google Authenticator no Kali Linux por meio de QR Code

Sobre:

Nesta atividade, foi realizada a integração do Google Authenticator ao Kali Linux para implementação de autenticação em dois fatores (2FA) utilizando TOTP (*Time-based One-Time Password*), fortalecendo o controle de acesso ao sistema.

Inicialmente, foi acessado o ambiente Kali Linux via RDP e obtidos privilégios de superusuário com o comando `sudo -i`. Em seguida, configurou-se corretamente o fuso horário do sistema com o comando `timedatectl set-timezone America/Sao_Paulo`, garantindo a sincronização de tempo necessária para o funcionamento adequado dos códigos TOTP.

Posteriormente, foi executado o comando `google-authenticator`, iniciando o processo de configuração do mecanismo de autenticação. Foi selecionada a opção de utilização de tokens baseados em tempo (TOTP), sendo então gerado um QR Code diretamente no terminal, juntamente com uma chave secreta única.

Utilizando um dispositivo móvel, o QR Code foi escaneado por meio do aplicativo Google Authenticator, permitindo a geração de códigos temporários de seis dígitos, atualizados a cada 30 segundos. Em seguida, foi inserido no terminal o código exibido no aplicativo, confirmando a correta sincronização entre o servidor e o dispositivo.

Durante a configuração, foram habilitadas medidas adicionais de segurança, como a atualização do arquivo de configuração `/root/.google_authenticator`, a proibição de reutilização de tokens, a ampliação da janela de tolerância de tempo para compensar possíveis diferenças de sincronização e a limitação de tentativas de autenticação, reduzindo a exposição a ataques de força bruta.

Dessa forma, a atividade demonstrou a implementação prática de autenticação multifator no Kali Linux, adicionando uma camada extra de segurança baseada em algo que o usuário possui (o dispositivo móvel), além da senha tradicional, elevando significativamente a proteção do sistema.

Atividade 4.8. Incorporando o Google Authenticator ao Mozilla Firefox

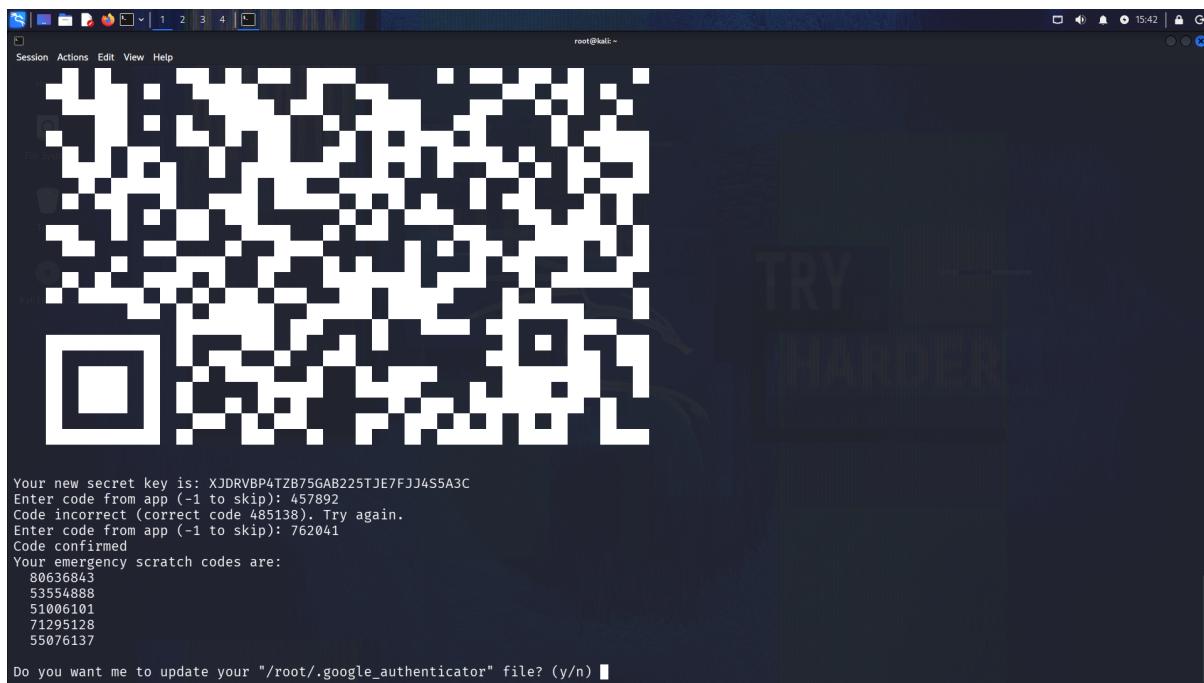


Fig. 3: Geração de códigos TOTP no Mozilla Firefox utilizando extensão Authenticator

Sobre:

Nesta atividade, foi realizada a integração do mecanismo de autenticação TOTP ao navegador Mozilla Firefox por meio de uma extensão, permitindo a geração de códigos de autenticação sem a necessidade de um dispositivo móvel.

Inicialmente, no computador pessoal, foi acessado o site de complementos do Mozilla Firefox e instalada a extensão “Authenticator”. Após a instalação, o ícone da extensão passou a ficar disponível na barra de navegação do navegador, possibilitando o gerenciamento de tokens de autenticação diretamente pelo browser.

Em seguida, no ambiente Kali Linux, foi obtido acesso como superusuário utilizando o comando `sudo -i`. Posteriormente, foi executado o comando `google-authenticator` para iniciar a configuração do TOTP. Ao selecionar a opção de tokens baseados em tempo, foi gerada uma chave secreta única associada ao usuário.

A chave secreta apresentada no terminal foi copiada para o computador pessoal e inserida manualmente na extensão Authenticator do Firefox. Para isso, foi utilizado o recurso de adição manual, preenchendo os campos de emissor e segredo com as informações correspondentes.

Após a configuração, a extensão passou a gerar códigos temporários sincronizados com o sistema do Kali Linux, atualizados periodicamente. Dessa forma, foi possível reproduzir o funcionamento do Google Authenticator diretamente no navegador, sem a necessidade de um smartphone.

Dessa maneira, a atividade demonstrou a utilização de ferramentas alternativas para implementação de autenticação em dois fatores, evidenciando a flexibilidade do uso de TOTP e a possibilidade de integração com diferentes plataformas, mantendo o mesmo nível de segurança no processo de autenticação.

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

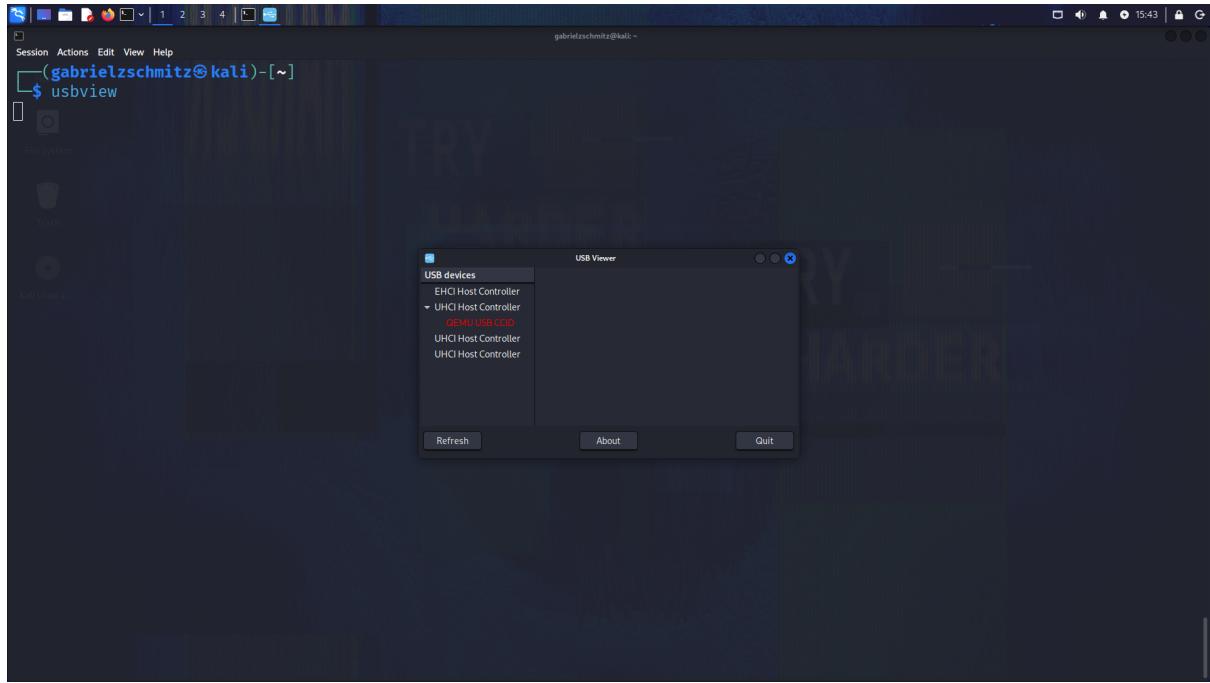


Fig. 4: Visualização das portas USB no Kali Linux utilizando o USB Viewer

Sobre:

Nesta atividade, foi realizada a verificação da presença do módulo TPM (Trusted Platform Module) e das portas USB no sistema Kali Linux, com o objetivo de identificar possíveis dispositivos de segurança e meios de entrada de tokens físicos.

Inicialmente, foi obtido acesso privilegiado no sistema por meio do comando `sudo -i`. Em seguida, utilizou-se o comando `journalctl -k --grep=tpm` para buscar registros do kernel relacionados ao TPM. A saída indicou a mensagem “*No TPM chip found, activating TPM-bypass!*”, evidenciando que a máquina virtual não possui um módulo TPM físico. Dessa forma, o sistema ativa um modo de bypass, permitindo o funcionamento sem os recursos de segurança fornecidos por esse hardware.

Também foi possível observar mensagens do `systemd`, indicando os componentes de segurança e funcionalidades habilitadas no sistema, como PAM, auditoria e mecanismos de integridade, demonstrando que o sistema continua operando mesmo na ausência do TPM.

Na sequência, foi realizada a análise das portas USB utilizando o comando `usbview`. Ao executar a ferramenta, foi exibida a interface gráfica do programa “USB Viewer”, responsável por listar os dispositivos USB conectados ao sistema. No entanto, não foram identificados dispositivos, o que é esperado em um ambiente virtualizado, onde não há conexão direta com hardware físico.

Dessa forma, a atividade demonstrou como verificar a presença de recursos de segurança em nível de hardware e identificar dispositivos de entrada, destacando as limitações inerentes a ambientes virtuais em relação ao acesso a componentes físicos como TPM e dispositivos USB.

Atividade 4.10. Gerenciando TOTPs com o OTPClient no Kali Linux

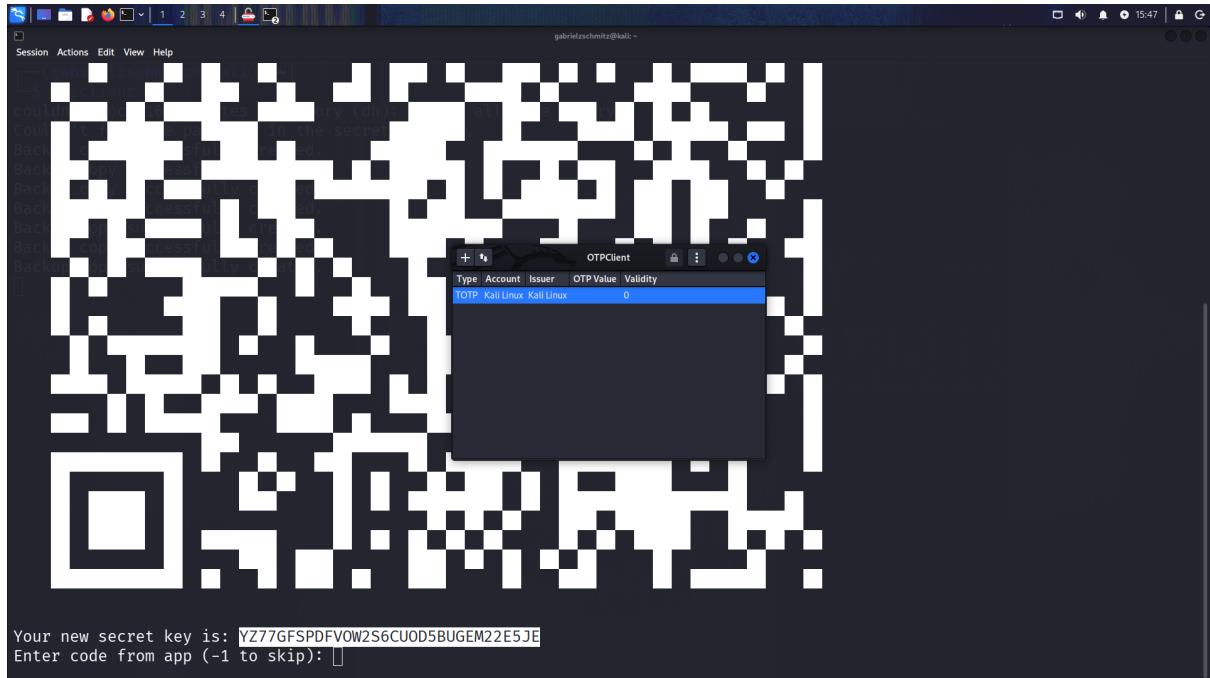


Fig. 5: Gerenciamento de tokens TOTP no OTPClient com base na chave gerada pelo Google Authenticator

Sobre:

Nesta atividade, foi explorada a utilização do OTPClient para gerenciamento de tokens TOTP (Time-based One-Time Password) no Kali Linux, dando continuidade à implementação de autenticação em dois fatores (2FA).

Inicialmente, o OTPClient foi iniciado por meio do comando `otpclient`. Durante a execução, foi exibido um aviso relacionado ao limite de memória segura (*memlock*), o qual não impede o funcionamento da ferramenta e foi confirmado para prosseguir. Em seguida, foi criada uma nova base de dados criptografada, armazenada no diretório `/home/aluno/Documentos/` com o nome `NewDatabase.enc`, protegida por uma senha definida durante o processo.

Posteriormente, em um segundo terminal, foi executado o comando `google-authenticator` para gerar uma chave secreta associada a um token TOTP. Ao selecionar a opção de geração baseada em tempo, foi apresentada uma chave secreta, utilizada para configuração manual no OTPClient.

De volta ao OTPClient, foi adicionada uma nova entrada por meio da opção “Manually”, na qual foram definidos os campos de identificação do token, como *Account* e *Issuer*. A chave secreta gerada anteriormente foi inserida no campo *Secret*, permitindo a geração de códigos TOTP válidos diretamente na aplicação.

Ao selecionar o token criado, foi possível visualizar seus parâmetros, incluindo tipo (TOTP), conta, emissor, valor do código gerado e tempo de validade, confirmando o correto funcionamento do mecanismo de autenticação baseado em tempo.

Por fim, a base de dados criada foi removida do sistema, juntamente com os arquivos auxiliares gerados durante o processo, utilizando o comando `rm *`, garantindo a limpeza do ambiente.

Dessa forma, a atividade demonstrou como gerenciar tokens de autenticação de dois fatores localmente, sem a necessidade de dispositivos externos, reforçando práticas de segurança no acesso a sistemas.