Henrique P

Projeto no github - > https://github.com/poledna0/rust-ssh-cert-auth

O projeto requer a linguagem Rust instalada. Para instalar, acesse o site oficial da linguagem:

https://rust-lang.org/learn/get-started/

Você também precisará de algumas dependências adicionais. Em sistemas baseados em Debian/APT, use o seguinte comando:

sudo apt install libsqlite3-dev libssl-dev pkg-config

Depois de clonar o repositório ou abrir a pasta, execute os três comandos a seguir em terminais diferentes:

cargo run --bin vault

cargo run --bin signer

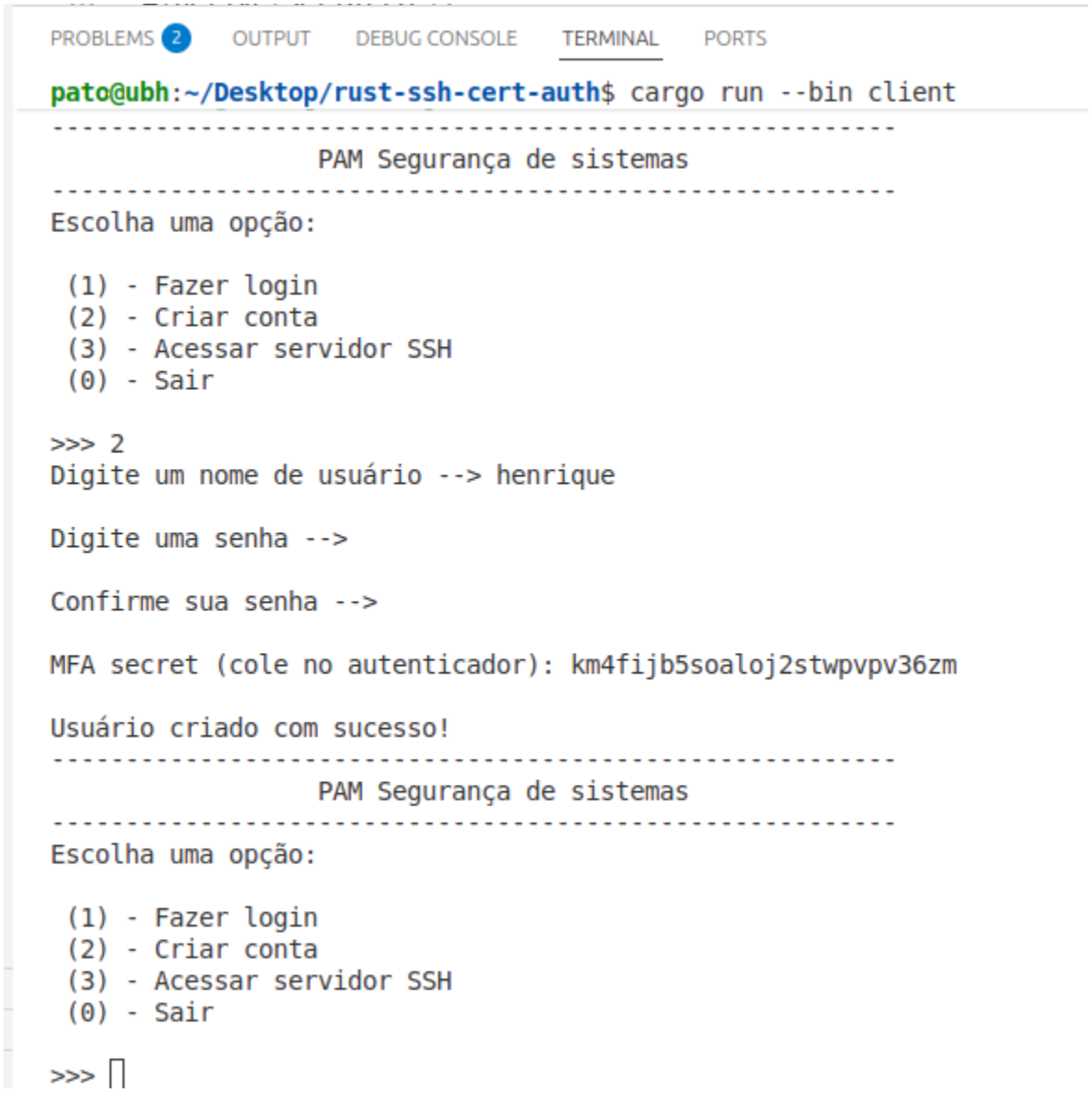
cargo run --bin client

O Vault e o Signer não têm interação direta com o usuário (somente para ver logs, pois fazem prints sobre o que estão fazendo). O único componente necessário para interagir é o Client.

Abaixo, está um exemplo da tela de criação de conta de usuário.

Podemos ver que, depois de registrar as informações no banco de dados, o sistema mostra uma chave Base32 para o client adicionar no autenticador. Isso ocorre porque essa chave Base32 será utilizada para gerar o TOTP, que são os números de 6 dígitos baseados em tempo.

As linhas 28–32 do código do client contêm a implementação desse Base32 randômico, que é enviado junto com as informações de cadastro em uma requisição POST para o signer. As linhas 168–172 da struct contêm as informações que serão enviadas.



Resumindo tudo ate agora, cliente digita o nome e a senha. O código gera uma chave Base32 para ser configurada no autenticador, faz o hash da senha do cliente e envia todas essas informações para o signer. O signer, por sua vez, possui um banco de dados que armazena as seguintes informações: hash da senha, nome de usuário, ID e o Base32.

O processo de login solicita o usuário e a senha. Se as credenciais estiverem corretas, o sistema pedirá o código MFA.

Após o código MFA ser verificado, o sistema solicita que o usuário cole a chave pública dele. Se todas as verificações estiverem corretas, a chave é enviada primeiro para o signer e, em seguida, o signer a encaminha para o vault, e depois volta o certificado para o signer e dele para o cliente, e possui uma funçãoo para escrever em um arquivo o certificado_nomedousuario.pem

O código do vault é relativamente simples e utiliza o ssh-keygen para sua funcionalidade.

```rust
// executa ssh-keygen
let output: Result<Output, Error> = Command::new("ssh-keygen")
    .args([
        "-s", ca_key_path().to_str().unwrap(), // chave CA
        "-I", &format!("{}-cert", username),   // identificador
        "-n", username,                        // principal
        "-V", "+10m",                          // validade
        "-z", &std::process::id().to_string(),  // serial único
        pub_path.to_str().unwrap(),            // arquivo .pub
    ])
    .output();
```

```
------------------------------------------------------------
Escolha uma opção:

 (1) - Fazer login
 (2) - Criar conta
 (3) - Acessar servidor SSH
 (0) - Sair

>>> 1

--- LOGIN ---
Digite o nome de usuário --> henrique
Digite a senha -->
Usuário e senha OK.
Código MFA (6 dígitos) --> 668827
MFA OK!
Agora cole sua chave pública SSH:
Chave pública --> ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIMmNR6J9J8g136hyhe8SLX2JgpbyMVJt+9d/rqTSng6v pato@ubh

Certificado do usuario henrique salvo em: /home/pato/Desktop/rust-ssh-cert-auth/client/certificado-client/certificado_henrique.pem
------------------------------------------------------------
                  PAM Segurança de sistemas
------------------------------------------------------------
Escolha uma opção:

 (1) - Fazer login
 (2) - Criar conta
 (3) - Acessar servidor SSH
 (0) - Sair

>>>
```

É necessário configurar o servidor que desejamos acessar para que ele confie na Autoridade Certificadora (CA) utilizada pelo sistema. Isso é feito adicionando a chave pública da CA ao repositório de certificados confiáveis do servidor. Este passo é crucial para permitir que o servidor valide as identidades dos clientes e do vault.

O comando para acessar o arquivo foi:

pato@ubh:~$ vim /etc/ssh/sshd_config

```
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# Allow client to pass locale and color environment variables
AcceptEnv LANG LC_* COLORTERM NO_COLOR

# override default of no subsystems
Subsystem       sftp    /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server
# Permitir autenticação por certificado da CA
TrustedUserCAKeys /home/pato/Desktop/rust-ssh-cert-auth/vault/src/ca_key.pub
PubkeyAuthentication yes

-- INSERT --
```

A funcionalidade de acesso ao servidor é implementada pela função fn chamar_ssh_com_inputs().

Essa função atua como um wrapper em Rust para o comando openssh do sistema operacional. Ela foi criada para ser mais alto nível e fácil de usar, simplificando o processo ao solicitar e coletar do usuário as quatro entradas necessárias para a conexão SSH, Usuário SSH , Host, Caminho da Chave Privada ,Caminho do Certificado

Após coletar os dados, a função invoca a conexão SSH usando um comando simples que passa a chave e o certificado, abstraindo o comando inteiro.

```
------------------------------------------------------------
                 PAM Segurança de sistemas
------------------------------------------------------------
Escolha uma opção:

 (1) - Fazer login
 (2) - Criar conta
 (3) - Acessar servidor SSH
 (0) - Sair

>>> 3
Usuário SSH: henrique
Host (ex: localhost ou 192.168.1.10): localhost
Caminho da chave privada (ex: ~/.ssh/id_ed25519): /home/pato/Desktop/rust-ssh-cert-auth/client/chave-pv-cliente/id_ed25519
Caminho do certificado (ex: certificado_henrique.pem): /home/pato/Desktop/rust-ssh-cert-auth/client/certificado-client/certificado_henrique.pem

Conectando via SSH...

Welcome to Ubuntu 25.10 (GNU/Linux 6.17.0-6-generic x86_64)

 * Documentation:  https://docs.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

11 updates can be applied immediately.
11 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Mon Nov 17 14:46:19 2025 from 127.0.0.1
$ bash
henrique@ubh:~$
```

E por ultimo, a comprovação da duraçao do certificado, que é curta:

```
pato@ubh:~/Desktop/rust-ssh-cert-auth/client$ ssh-keygen -Lf /home/pato/Desktop/rust-ssh-cert-auth/client/certificado-client/certificado_henrique.pem
/home/pato/Desktop/rust-ssh-cert-auth/client/certificado-client/certificado_henrique.pem:
        Type: ssh-ed25519-cert-v01@openssh.com user certificate
        Public key: ED25519-CERT SHA256:OShH6/qoZaMD3Am7a3NcrOMCb7OlUHA/EfDz3G6PhME
        Signing CA: ED25519 SHA256:V1Nb93TsR4gLLSbJB5tKPWi9CJsSIH3R6ZON2/kQ9mY (using ssh-ed25519)
        Key ID: "henrique-cert"
        Serial: 19683
        Valid: from 2025-11-17T22:19:00 to 2025-11-17T22:30:30
        Principals:
                henrique
        Critical Options: (none)
        Extensions:
                permit-X11-forwarding
                permit-agent-forwarding
                permit-port-forwarding
                permit-pty
                permit-user-rc
pato@ubh:~/Desktop/rust-ssh-cert-auth/client$
```