



Instituto Superior de Engenharia

Politécnico de Coimbra

Serviços de Rede II

Linux Pluggable Authentication Modules

Pedro Miguel Neves Martins

a2021135054(@isec.pt)

Licenciatura em Engenharia Informática
Ramo de Redes e Administração de Sistemas
Instituto Superior de Engenharia de Coimbra
Instituto Politécnico de Coimbra

Coimbra, 3 de dezembro de 2023

Conteúdo

1	Introdução	9
2	Tipos de Autenticação	10
2.1	<i>Password-based</i>	10
2.1.1	Objetivos	10
2.1.2	Princípios	11
2.1.3	Técnicas	11
2.2	<i>Two/Multi-factor Authentication (Two/Multi-factor Authentication (2/M-FA))</i>	12
2.2.1	Objetivos	12
2.2.2	Princípios	12
2.2.3	Técnicas	12
2.3	<i>Biometric Authentication</i>	13
2.3.1	Objetivos	14
2.3.2	Princípios	14
2.3.3	Técnicas	14
2.4	<i>Single sign-on (Single sign-on (SSO))</i>	16
2.4.1	Objetivos	16
2.4.2	Princípios	16
2.4.3	Técnicas	16
2.5	<i>Token-based authentication</i>	18
2.5.1	Objetivos	18
2.5.2	Princípios	18
2.5.3	Técnicas	19
2.6	<i>Certificate-based authentication</i>	20
2.6.1	Objetivos	20
2.6.2	Princípios	20
2.6.3	Técnicas	21
3	Protocolos de Autenticação	22
3.1	<i>Password Authentication Protocol (Password Authentication Protocol (PAP))</i>	22
3.1.1	Contexto histórico	22

3.1.2	Motivação	23
3.1.3	Utilização	23
3.1.4	Substituição	23
3.2	<i>Challenge Handshake Authentication Protocol (Challenge Handshake Authentication Protocol (CHAP))</i>	24
3.2.1	Contexto histórico	24
3.2.2	Motivação	24
3.2.3	Utilização	24
3.2.4	Substituição	25
3.3	<i>Extensible Authentication Protocol (Extensible Authentication Protocol (EAP))</i>	26
3.3.1	Contexto histórico	26
3.3.2	Motivação	26
3.3.3	Utilização	26
3.3.4	Substituição	27
3.4	<i>Remote Authentication Dial-In User Service (RADIUS) (Remote Authentication Dial-In User Service)</i>	28
3.4.1	Contexto histórico	28
3.4.2	Motivação	28
3.4.3	Utilização	28
3.4.4	Substituição	29
3.5	<i>Lightweight Directory Access Protocol (LDAP) (Lightweight Directory Access Protocol)</i>	30
3.5.1	Contexto histórico	30
3.5.2	Motivação	30
3.5.3	Utilização	30
3.5.4	Substituição	31
3.6	<i>Kerberos</i>	32
3.6.1	Contexto histórico	32
3.6.2	Motivação	32
3.6.3	Utilização	33
3.6.4	Substituição	34
3.7	<i>Security Assertion Markup Language (SAML) (Security Assertion Markup Language)</i>	35
3.7.1	Contexto histórico	35
3.7.2	Motivação	35
3.7.3	Utilização	35
3.7.4	Evolução	36
3.8	<i>OpenID Connect (OIDC) (OpenID Connect)</i>	37
3.8.1	Contexto histórico	37

3.8.2	Motivação	37
3.8.3	Utilização	37
3.8.4	Evolução	38
4	<i>Linux Pluggable Authentication Modules</i>	39
4.1	Evolução dos mecanismos de Autenticação <i>Linux</i>	39
4.2	Arquitetura do <i>framework Pluggable Authentication Modules</i> (PAM)	40
4.2.1	Princípios subadjacentes ao <i>design</i> do PAM	40
4.2.2	Funcionamento da arquitetura PAM	41
4.2.3	Passos genéricos da configuração do PAM	42
5	Trabalho Experimental	44
5.1	Configuração das placas no <i>Linux</i>	44
5.2	Instalação dos pacotes no <i>Linux</i>	45
5.3	Juntar o cliente ao domínio	46
5.3.1	KRB5	46
5.3.2	<i>System Security Services Daemon</i> (SSSD)	46
5.4	<i>TrueNAS Core</i>	49
5.4.1	Configuração básica do <i>TrueNAS</i>	50
5.4.2	<i>Sharing</i> da <i>Home Directory</i> dos utilizadores	51
6	Conclusão	55
	Referências	56
7	Anexos	59
7.1	Meta 1	59
7.2	Meta 2	59
7.3	Meta 3	59
7.4	Meta 4	60
7.5	Alterações no documento	60

Lista de Figuras

2.1	<i>Password-based</i>	11
2.2	<i>Two/Multi-factor Authentication</i>	13
2.3	<i>Biometric Authentication</i>	15
2.4	<i>Single sign-on</i>	17
2.5	<i>Token-based Authentication</i>	19
2.6	<i>Certificate-Based Authentication</i>	21
3.1	<i>Password Authentication Protocol</i>	23
3.2	<i>Challenge Handshake Authentication Protocol</i>	25
3.3	<i>Extensible Authentication Protocol</i>	28
3.4	<i>Remote Authentication Dial-In User Service</i>	29
3.5	<i>Lightweight Directory Access Protocol</i>	31
3.6	<i>Kerberos</i>	34
3.7	<i>Security Assertion Markup Language</i>	36
3.8	<i>OpenID Connect</i>	38
4.1	<i>Pluggable Authentication Modules</i>	42
4.2	Localização dos arquivos de configuração do PAM	43
4.3	configuração de arquivos PAM	43
5.1	Configuração das placas da rede NAT	44
5.2	Configuração das placas do <i>host-only</i>	45
5.3	Configuração das placas da NAT	45
5.4	Instalação de pacotes <i>Linux</i>	45
5.5	Edição do ficheiro KRB5.conf	46
5.6	Edição do ficheiro sssd.conf	47
5.7	Verificação do domínio <i>Active Directory</i>	47
5.8	<i>Join</i> do cliente <i>Linux</i>	48
5.9	Cliente no <i>Active Directory</i>	48
5.10	<i>TrueNAS Core</i>	49
5.11	Configuração das interfaces do <i>TrueNAS</i>	50

5.12	Configuração dos DNS Servers	51
5.13	Definição do Active Directory no TrueNAS	51
5.14	Integração do TrueNAS no Active Directory	51
5.15	Alteração do ficheiro sssd.conf	52
5.16	Alteração do fstab	52
5.17	Alteração do sudoers	53
5.18	Pasta dos utilizadores do Active Directory	53
5.19	Criação da <i>pool</i>	53
5.20	<i>Sharing</i> NFS	54
5.21	<i>Shell</i> do TrueNAS	54

Lista de Acrónimos

2/M-FA *Two/Multi-factor Authentication*

SSO *Single sign-on*

PAP *Password Authentication Protocol.*

MIT *Massachusetts Institute of Technology*

CHAP *Challenge Handshake Authentication Protocol*

PPP *Ponto a Ponto*

SAML *Security Assertion Markup Language*

EAP *Extensible Authentication Protocol*

VPNs *Virtual Private Network*

TLS *Transport Layer Security*

AP *Access Point*

EAP-TLS *Transport Layer Security*

PEAP *Protected Extensible Authentication Protocol*

EAP-SIM *Subscriber Identity Module*

ADM *Active Directory Microsoft*

AD *Active Directory*

WEP *Wired Equivalent Privacy*

RADIUS *Remote Authentication Dial-In User Service*

NAS *network access server*

LDAP *Lightweight Directory Access Protocol*

DAP *Directory Access Protocol*

KDC *Key Distributor Center*

TGT *Ticket Granting Ticket*

TGS *Ticket Granting Server*

XML *Extensible Markup Language*

IdP *Identity Provider*

SP *Services Provider*

OIDC *OpenID Connect*

JWT *JSON Web Token*

OP *OpenID Provider*

PAM *Pluggable Authentication Modules*

SSSD *System Security Services Daemon*

DNS *Domain Name System*

NTP *Network Time Protocol*

SMB *Server Message Block*

NFS *Network File System*

AFP *Apple Filing Protocol*

NSS *Name Service Switch*

SRV *Service Records*

IP *Internet Protocol*

ICMP *Internet Control Message Protocol*

fstab *file system table*

Capítulo 1

Introdução

A autenticação desempenha um papel fundamental na segurança da informação e no acesso a sistemas e serviços digitais. Existem diversos tipos de autenticação, que vão desde simples senhas até métodos mais avançados e atuais. Neste relatório, vão ser explorados os diferentes tipos de autenticação, os protocolos mais comuns usados para proteger a integridade e a confidencialidade dos sistemas e dados digitais e os mecanismos de autenticação no sistema *Linux*, desde a sua evolução até à utilização nos dias que correm. Vai ser também realizado um trabalho experimental que irá consistir na integração de um cliente *Linux* num Active Directory e na partilha das *home Directory* dos utilizadores, com recurso a um servidor *TrueNAS*.

Capítulo 2

Tipos de Autenticação

A autenticação é um processo muito usado em redes de computadores e serviços *online* que serve para verificar a identidade de um indivíduo, dispositivo ou até mesmo de um sistema. É usada para prevenir o acesso não autorizado às informações e garantir a integridade dos dados.

Existem vários tipos de autenticação que serão descritos abaixo.

2.1 *Password-based*

Sistemas *password-based* são aqueles em que o acesso a um sistema ou a uma conta é concedido após a inserção correta de uma senha pelo utilizador. Depois de digitadas as credenciais, elas são comparadas às credenciais já armazenadas na base de dados do sistema aquando da criação de uma conta, por exemplo, e, se corresponderem, é fornecido acesso ao utilizador. Contudo, devido à baixa entropia das *password*, estes sistemas estão sempre sujeitos a ataques com o objetivo de descobrir a *password*.

2.1.1 Objetivos

- **Autenticação.** O grande objetivo do protocolo *password-based* é confirmar a identidade do utilizador. A senha garante que quem está a aceder ao sistema é o verdadeiro portador da mesma.
- **Segurança.** Proteger a informação ao acesso não autorizado é crucial e as *pas-*

swords ajudam a garantir que apenas as entidades permitidas tenham acesso aos dados, garantindo a integridade dos mesmos. Além disso, a autenticação *password-based* permitem que as senhas sejam alteradas quando o utilizador assim o entender, permitindo alguma flexibilidade na sua gestão.

2.1.2 Princípios

- **Confidencialidade.** A *password* deve ser sempre mantida em segredo, o que garante que apenas o utilizador portador da mesma a possa usar.
- **Complexidade.** Qualquer senha deve ser complexa para reduzir a chance de adivinha por terceiros e evitar ataques de força bruta bem sucedidos.
- **Atualização regular.** A alteração da *password* periodicamente é recomendada, para evitar comprometê-la.

2.1.3 Técnicas

- **Hashing da *password*.** Quando um utilizador cria uma senha, o sistema aplica uma função *hash* a essa mesma senha e será armazenada na base de dados, juntamente com o identificador associado ao utilizador. Quando o utilizador torna a fazer *login*, é aplicada uma nova função *hash* à senha inserida, que vai ser comparada com a senha associada ao utilizador que está armazenada na base de dados. Se as senhas coincidirem, é concedido acesso.

[13] [5]

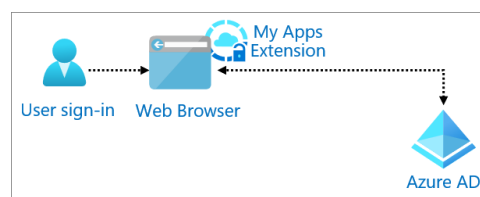


Figura 2.1: *Password-based*

2.2 *Two/Multi-factor Authentication* (2/M-FA)

A autenticação por dois fatores é um método de segurança que exige duas formas distintas para verificar a identidade de um determinado utilizador antes de lhe conceder acesso a uma conta, sistema ou aplicação. Basicamente, é colocada uma camada de segurança adicional em comparação ao sistema baseado em *password*, tornando-se mais seguro e menos propenso a ataques bem sucedidos. Pode e deve ser ativada pelo utilizador para proteção pessoal.

2.2.1 Objetivos

- **Segurança.** O principal objetivo passa por aumentar a segurança, em relação à autenticação baseada em senha, tornando mais complicado ainda para pessoas não autorizadas acederem a sistemas protegidos. Mesmo que alguém, exceto o próprio utilizador, descubra as credenciais de acesso ao sistema não é suficiente ainda para ter acesso às informações protegidas.

2.2.2 Princípios

- **Verificação de duas fases.** A autenticação por dois fatores é baseada no princípio da verificação da entidade do utilizador de duas maneiras diferentes. Este processo geralmente envolve a senha e um dispositivo que o utilizador tem.

2.2.3 Técnicas

Neste método de dois fatores, quando um utilizador tenta fazer *login* a uma conta, o sistema reconhece que a autenticação por duas fases foi ativada para a conta em questão. Quando digitada a palavra-passe, o sistema vai verificar se corresponde à senha registada na base de dados para essa conta. Se corresponder, está concluída a primeira fase e dá-se início à segunda fase.

Na segunda fase, a autenticação final pode ser feita de diversas técnicas que irão ser descritas abaixo. O sistema verifica se a autenticação é válida e se for, dá acesso à conta. Se qualquer uma das fases falhar, o acesso é negado.

- **Mensagens de texto.** Neste processo (depois de concluída a primeira fase), é enviado um código para o telemóvel do utilizador que precisa de o inserir no sistema para concluir a autenticação.
- **Aplicações de autenticação.** O utilizador pode optar por aplicações de autenticação, como o *Google Authenticator* que geram código a cada poucos segundos para ser inserido no sistema para concluir o *login*.
- **Biometria.** Outra forma de autenticação é o uso de biometria, como impressão digital, reconhecimento facial, entre outros que irão ser falados no próximo tipo de autenticação.

[8] [26]

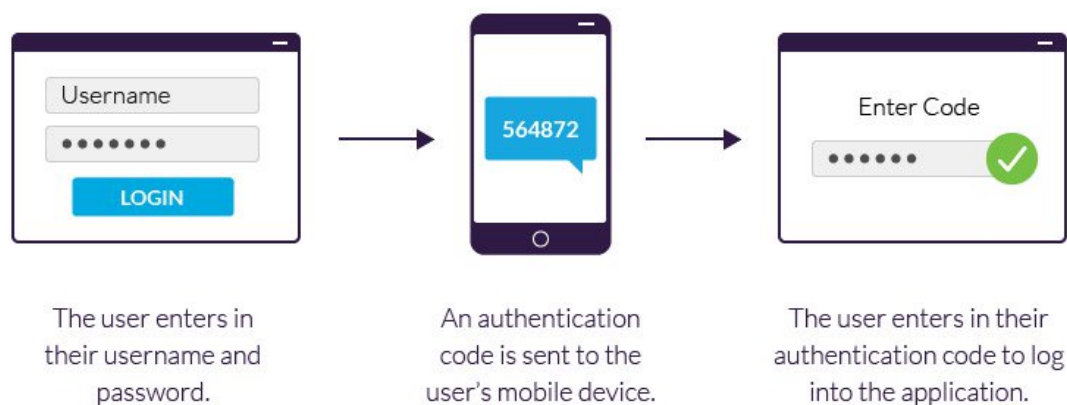


Figura 2.2: *Two/Multi-factor Authentication*

2.3 *Biometric Authentication*

A autenticação biométrica é um método de verificação de identidade que usa características físicas ou comportamentais de um indivíduo para confirmar a sua identidade. Os sistemas biométricos armazenam essas informações para verificar a identidade de um

utilizador, quando acede à conta. Este tipo de autenticação geralmente é mais seguro que alguns tipos de autenticação por dois fatores como, por exemplo, as mensagens de texto.

2.3.1 Objetivos

- **Segurança.** O principal objetivo da autenticação biométrica é garantir que apenas indivíduos autorizados tenham acesso a determinadas informações, ajudando a prevenir roubo de identidade e acessos não autorizados, uma vez que a biometria é difícil de falsificar, tornando-a mais segura que métodos baseados apenas em senha.

2.3.2 Princípios

- **Unicidade.** Cada pessoa tem características biométricas únicas, como a voz e a impressão digital, tornando difícil a possíveis atacantes reproduzi-las. Os sistemas biométricos devem ser capazes de recolher os dados acerca das características de cada indivíduo. Por vezes, com algum tipo de envelhecimento da pessoa, os dados biométricos podem-se tornar mais difíceis de recolher, mas por norma permanecem sempre reconhecíveis.

2.3.3 Técnicas

Quando um utilizador deseja ativar a autenticação biométrica num determinado serviço, o processo de recolha de dados é iniciado. Os dados biométricos do indivíduo são recolhidos e armazenados numa base de dados segura. Cada vez que o indivíduo tenta autenticar-se, é-lhe pedido a amostra biométrica novamente, que é recolhida e comparada com as informações armazenadas na base de dados. A correspondência, geralmente, é feita usando algoritmos que calculam a equivalência das amostras. Se houver uma correspondência bem sucedida, é concedido acesso ao utilizador, caso contrário, o acesso é negado.

Entre algumas técnicas de autenticação biométrica, as seguintes são as mais usadas.

- Reconhecimento de impressão digital.
- Reconhecimento facial.
- Reconhecimento de voz.
- Tecnologia IRIS.
- Reconhecimento de palma.

[4] [11]

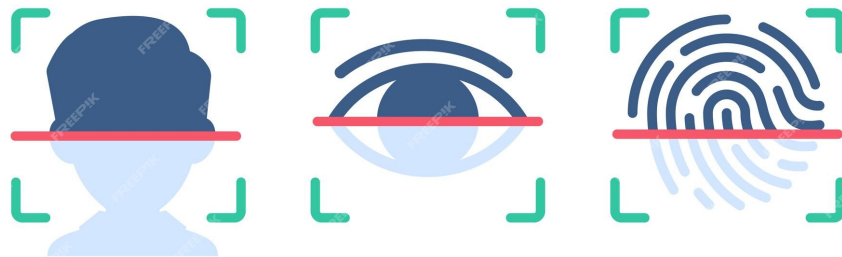


Figura 2.3: *Biometric Authentication*

2.4 *Single sign-on* (SSO)

SSO é um método de autenticação que permite o uso de um conjunto de credenciais para aceder a várias aplicações ou sistemas, simplificando o processo de autenticação e reduzindo a necessidade do utilizador de se lembrar de várias combinações de *username* e *password*. Com esta facilidade, a segurança dos dados poderia ser comprometida uma vez que se a senha do utilizador não for mantida em sigilo, um *hacker* poderia facilmente ter acesso a qualquer aplicação associada. Contudo, normalmente utilizadores de SSO usam senhas mais fortes para as aplicações que usam esse método. Além disso, os sistemas SSO têm obrigação de garantir a boa gestão da informação do utilizador.

2.4.1 Objetivos

- **Melhor experiência de utilização.** O principal objetivo do SSO é tornar mais simples o acesso a várias aplicações com um único *login*, reduzindo a necessidade de gerir as várias credenciais e aumentando, assim, a produtividade do utilizador.

2.4.2 Princípios

- **Identificação Única.** Um utilizador, depois de autenticado, tem acesso a outras aplicações sem necessidade de se autenticar às mesmas. O SSO garante que uma entidade confiável verifique a identidade do utilizador, concedendo *tokens* de autenticação para outras *apps*.

2.4.3 Técnicas

Quando um utilizador acede à aplicação que faz parte do ambiente SSO, ele fornece as suas credenciais de *login* na respetiva aplicação. Se as credenciais estão corretas, a aplicação pode iniciar a autenticação SSO. Se a autenticação for bem sucedida, a aplicação redireciona o utilizador para um *Identity Provider* (IdP), que é responsável por emitir

um *token* de autenticação, que contém informação sobre o utilizador. A aplicação recebe o *token* de autenticação e verifica se é válido e, em caso afirmativo, concede acesso ao utilizador sem a necessidade de um novo *login*.

Os tokens podem ser gerados de acordo com alguns protocolos, onde dois dos principais serão descritos abaixo.

- **SAML.** Permite a partilha de informações de autenticação entre sistemas diferentes em domínios diferentes.
- **OAuth.** Permite que uma aplicação tenha acesso aos recursos, em nome do utilizador.

[2] [25]

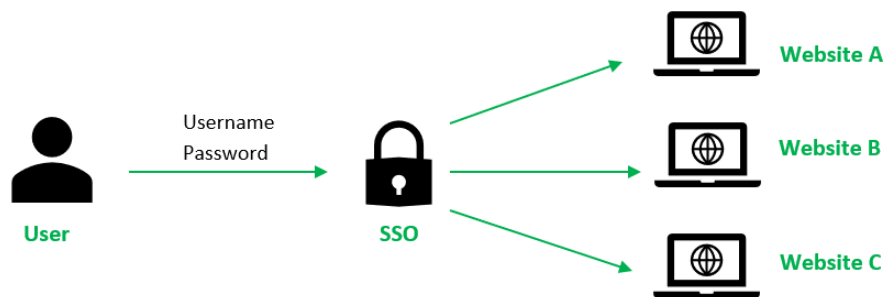


Figura 2.4: *Single sign-on*

2.5 *Token-based authentication*

Token-based authentication é uma das técnicas de autenticação do SSO. É baseada no uso de *tokens* (que podem ser um código criptografado, único ou de acesso temporário) que são fornecidos por um servidor de autenticação para verificar a entidade do utilizador, para posteriormente o mesmo poder aceder a recursos fornecidos pela aplicação à qual está a tentar aceder.

2.5.1 Objetivos

- **Segurança.** Garante que apenas utilizadores autenticados corretamente tenham acesso aos serviços.
- **Melhor experiência de utilização.** Torna-se mais conveniente e prático para os utilizadores, pois evitam estar sempre a fazer *login* sempre que querem aceder a um determinado serviço.
- **Redução de riscos.** Mesmo que um atacante saiba as credenciais de um utilizador, não é suficiente para aceder aos recursos protegidos, uma vez que não sabe o código *token*, que deve ser muito bem guardado pelo utilizador, reduzindo assim os ataques de força bruta bem sucedidos.

2.5.2 Princípios

- **Emissão controlada de *tokens*.** O servidor de autenticação deve garantir que o *token* só é fornecido em caso de autenticação bem sucedida.
- ***Tokens* temporários.** Os tokens geralmente têm um período de validade limitado (minutos ou até segundos), ajudando a minimizar o risco de uso indevido.
- **Confidencialidade.** Os *tokens* são informações sensíveis e não devem ser partilhados com terceiros.

2.5.3 Técnicas

A autenticação baseada em *tokens* tem um funcionamento idêntico à SSO. O utilizador efetua *login* numa determinada plataforma e envia uma solicitação de acesso ao servidor de autenticação, que por sua vez verifica se as credenciais do utilizador estão corretas. Se corresponderem às credenciais armazenadas na sua base de dados, o servidor gera um *token* para um determinado periodo de tempo para ser usado pelo utilizador. O *token* é registado no serviço que o utilizador está a tentar aceder. Para analisar a correspondência, o *token* é decodificado e se coincidirem, o utilizador tem acesso aos serviços.

Um dos formatos de *token* bastante conhecido é o *JSON Web Token* (JWT). É constituído por informações sobre o utilizador no formato *JSON*. Para garantir a integridade na transmissão do *token* ao utilizador, o mesmo pode ser criptografado usando uma *hash* HMAC SHA256, que é um protocolo usado mesmo para este tipo de finalidade. [7] [30]

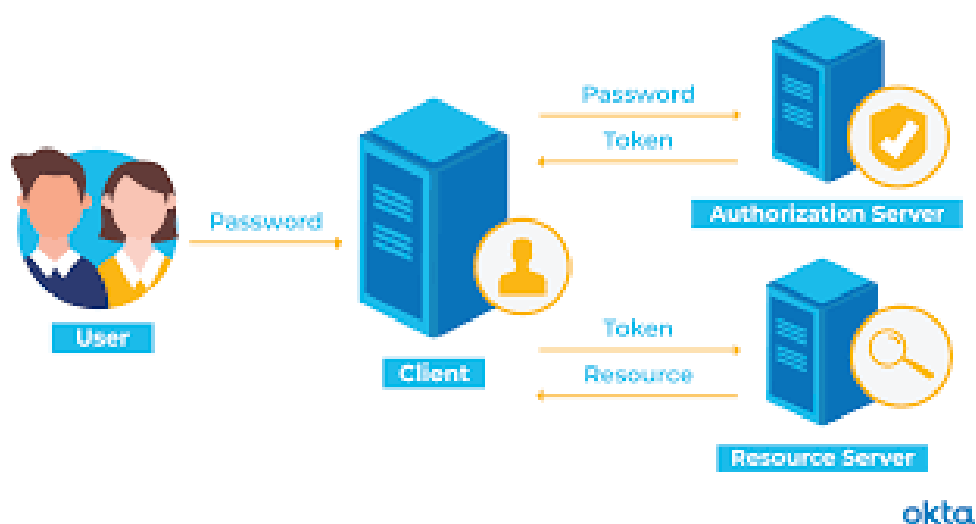


Figura 2.5: *Token-based Authentication*

2.6 *Certificate-based authentication*

Certificate-based authentication é um método de verificação da identidade de uma entidade que usa documentos eletrônicos como certificados digitais, que são emitidos por uma autoridade de certificação confiável e são usados para estabelecer a autenticidade. Os certificados digitais contêm informações sobre chave pública usada para criptografar mensagens destinadas ao titular, verificar a assinatura digital e informações pessoais.

2.6.1 Objetivos

- **Autenticidade.** Os certificados permitem que utilizadores sejam autenticados sem necessidade do nome de utilizador e *password*.
- **Segurança.** Este tipo de autenticação oferece um alto nível de segurança, pois os certificados digitais são difíceis de falsificar e podem ser protegidos por senha ou PIN.
- **Confidencialidade.** É usada com métodos de criptografia com a finalidade de proteger os dados.

2.6.2 Principios

- **Chaves Públicas e Privadas.** A autenticação baseada em certificado envolve um par de chaves criptográficas. A chave pública está no certificado e pode ser compartilhada, enquanto que a privada deve ser mantida em sigilo.
- **Autoridades de Certificado.** Emitem o certificado digital após verificar a identidade do titular.
- **Assinatura Digital.** O proprietário do certificado usa a sua chave privada para assinar informações digitalmente.

2.6.3 Técnicas

Tudo começa com a geração de chaves criptográficas para o utilizador, onde a pública é partilhada publicamente e a privada mantida em sigilo. O utilizador que se deseja autenticar, solicita um certificado digital a uma autoridade de certificação. A solicitação é feita enviando a chave pública e informações sobre o utilizador.

Após verificar a identidade do solicitante, é emitido um certificado digital. O certificado é assinado digitalmente pela autoridade de certificação, usando a sua chave privada.

Quando o utilizador se deseja autenticar num sistema, ele apresenta o seu certificado digital, que inclui a chave pública, para provar sua identidade. Esse mesmo sistema valida a assinatura digital da autoridade de certificação no certificado, usando a chave pública da mesma. Se for bem sucedido, o utilizador tem acesso ao sistema.

[14] [31] [15]

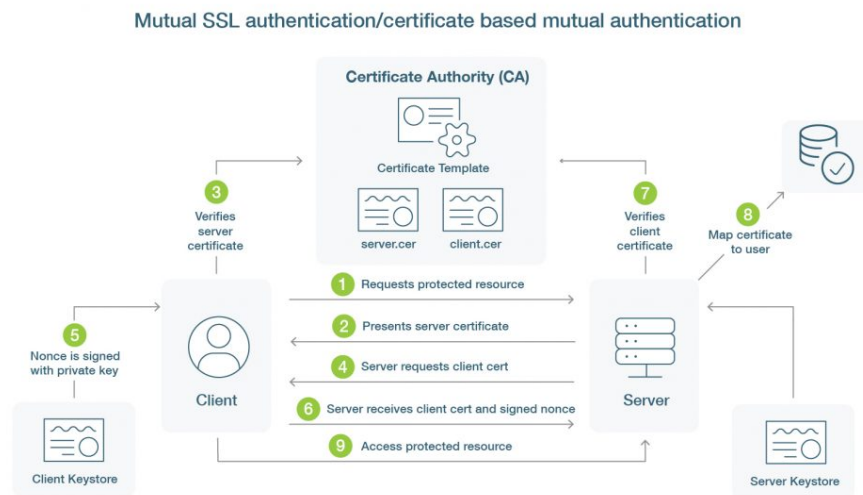


Figura 2.6: *Certificate-Based Authentication*

Capítulo 3

Protocolos de Autenticação

Protocolos de autenticação são métodos usados para verificar a identidade de um utilizador ou sistema. Ajudam a garantir que apenas utilizadores autorizados possam aceder a recursos protegidos e impedem o acesso não autorizado. Atualmente, existem diversos tipos diferentes de protocolos de autenticação, onde os mais relevantes e usados serão descritos abaixo.

3.1 *Password Authentication Protocol* (PAP)

PAP é um protocolo que usa *password* para validar o acesso do utilizador a um determinado recurso. Ao usar este protocolo, os dados não são criptografados, ou seja, são enviados ao servidor de autenticação em *plain text*.

3.1.1 Contexto histórico

O PAP foi desenvolvido na década de 60, quando as redes de computadores se estavam a começar a expandir. Nesse período, a necessidade de autenticação de utilizadores aos sistemas, bem como a proteção de ficheiros tornou-se evidente e crucial para o bom funcionamento de uma organização.

Fernando Corbató, que era investigador do *Massachusetts Institute of Technology* (MIT) na altura, implementou um programa de senhas em que as mesmas eram armazenadas em *plain text* no sistema de arquivos.

3.1.2 Motivação

A motivação por trás do desenvolvimento do PAP era fornecer um meio simples para autenticar utilizadores. Antes do PAP, muitas redes não tinham métodos eficazes de autenticação, o que as tornava vulneráveis a acessos não autorizados. E, com o passar do tempo, seria útil abordar essa preocupação de segurança e exigir que os utilizadores fornecessem um nome de utilizador e uma *password* para aceder aos recursos da rede.

3.1.3 Utilização

O PAP é um método de autenticação utilizado especialmente em conexões de acesso remoto ou então quando o protocolo CHAP não é suportado. Também se tornou comum em contextos onde a segurança não é uma preocupação crítica ou em redes virtuais mais antigas, onde o seu objetivo passa por conectar o utilizador remotamente à rede.

3.1.4 Substituição

Devido às suas limitações de segurança, o PAP foi substituído por protocolos de autenticação mais seguros, como, por exemplo, o CHAP (*Challenge Handshake Authentication Protocol*), que como iremos ver a seguir, foi projetado para resolver o problema de transmissão em *plain text*, usando criptografia para autenticar os utilizadores e por, sua vez, oferecendo um nível de segurança superior ao PAP.

[16] [3]

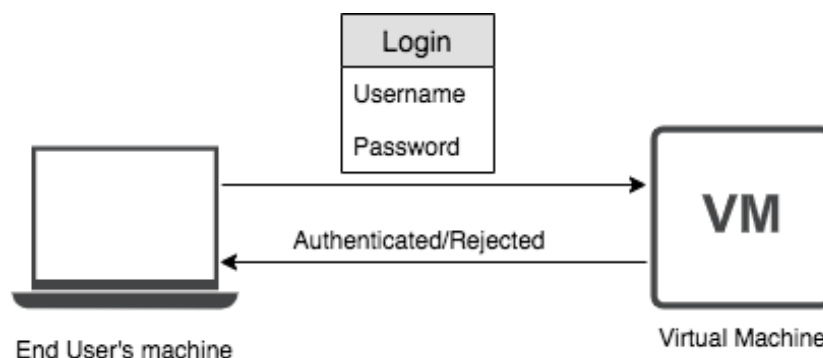


Figura 3.1: *Password Authentication Protocol*

3.2 *Challenge Handshake Authentication Protocol* (CHAP)

O CHAP é um protocolo de autenticação que os servidores de protocolo Ponto a Ponto (PPP) usam para verificar a identidade de um utilizador remoto, ao permitir que se identifiquem num sistema de autenticação, sem expor a sua senha. Com o CHAP, os sistemas de autenticação usam a *password* para criar uma *hash*, usando o algoritmo de criptografia MD5.

3.2.1 Contexto histórico

Na década de 1990, as redes de computadores estavam em crescimento e a segurança das redes era uma preocupação crescente, uma vez que a autenticação de utilizadores numa rede era muitas vezes realizada de forma insegura. As senhas eram frequentemente a única forma de autenticação, e elas podiam ser facilmente interceptadas por invasores, uma vez que elas eram enviadas aos servidores em *plain text*.

Com o conjunto destes fatores de risco que poderia colocar em causa o bom funcionamento de uma organização, havia a necessidade de uma mudança na transmissão de informações confidenciais dos utilizadores.

3.2.2 Motivação

A motivação por trás do desenvolvimento do CHAP foi melhorar a segurança da autenticação, fornecendo um método mais forte e seguro para verificar a identidade de dispositivos e utilizadores. O CHAP foi projetado para evitar ataques de *spoofing*, em que um invasor interceptava as credenciais de autenticação de uma determinada entidade e as usava para ter acesso a recursos protegidos.

3.2.3 Utilização

O CHAP funciona da seguinte maneira:

- Quando um dispositivo inicia uma conexão com um servidor (envio do *username*), o servidor envia uma mensagem de desafio ao solicitante. A entidade responde com

um valor calculado a partir do desafio e da senha, usando uma *hash* como a MD5. O servidor verifica se a resposta do dispositivo está correta, com base no seu próprio cálculo do valor esperado. Se a resposta estiver correta, o dispositivo é autenticado à rede.

3.2.4 Substituição

Embora o CHAP tenha sido eficaz e tenha melhorado a segurança da autenticação, não é imune a ataques de força bruta.

Com o avanço da tecnologia, surgiram métodos de autenticação mais robustos, como o EAP, que ganhou popularidade em ambientes mais modernos, como redes *Wi-Fi*. O EAP é flexível e pode ser conciliado com métodos de autenticação mais avançados, como autenticação de chave pública (PKI) e *tokens*, tornando-o mais adequado para ambientes onde é exigido maior segurança.

[19]

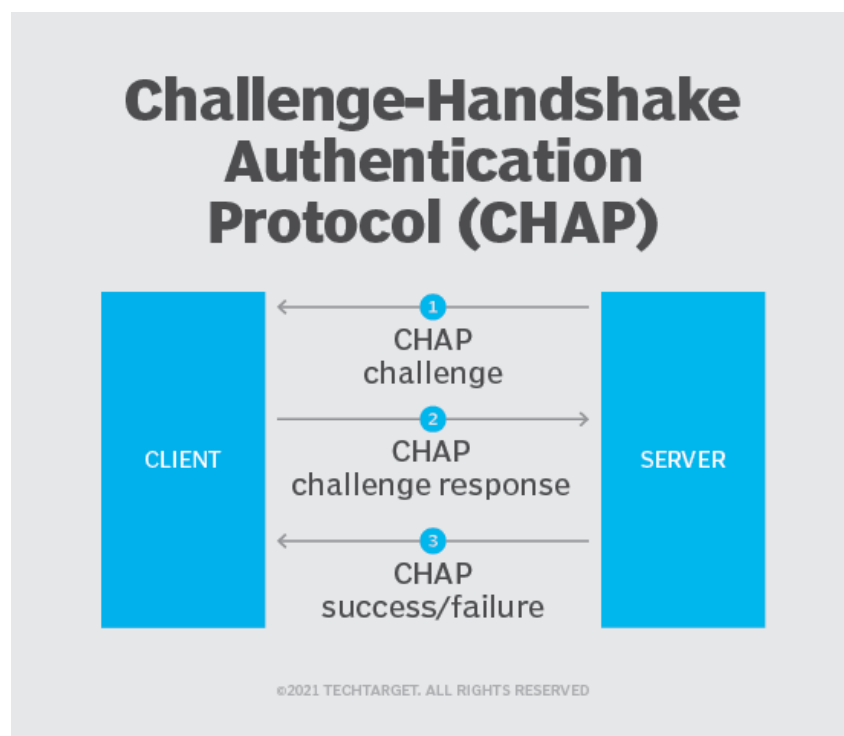


Figura 3.2: *Challenge Handshake Authentication Protocol*

3.3 *Extensible Authentication Protocol* (EAP)

O EAP é um protocolo para redes sem fio (Wi-Fi e *Virtual Private Network* (VPNs)) que é usado em redes criptografadas para fornecer uma maneira segura de enviar informações de identificação para fornecer autenticação de rede. Ele foi desenvolvido para permitir a autenticação flexível de utilizadores e dispositivos em rede, oferecendo suporte a uma variedade de métodos de autenticação, incluindo cartões inteligentes, certificados, senhas únicas e criptografia de chave pública.

3.3.1 Contexto histórico

EAP surgiu no final dos anos 90 em resposta à necessidade de estabelecer um método de autenticação seguro em redes *wireless*. Naquela época, as redes Wi-Fi estavam a começar a tornar-se populares e a segurança era uma preocupação significativa. Autenticar dispositivos e utilizadores em redes sem fio de forma segura era um desafio. O CHAP na época era um método considerado seguro, no entanto, não suportava diversos métodos de autenticação como o EAP.

3.3.2 Motivação

A motivação principal por trás do desenvolvimento do EAP seria criar um sistema capaz de suportar diferentes métodos de autenticação, ser flexível o suficiente para que as entidades escolhessem os métodos de autenticação mais adequados às suas necessidades específicas e que se adaptasse às necessidades futuras.

3.3.3 Utilização

O EAP usa o padrão 802.1x como mecanismo de autenticação numa rede *wireless*.

Os 3 componentes principais deste mecanismo de autenticação são:

- **Dispositivo sem fio.**
- *Access Point* (AP).
- **Servidor de Autenticação.**

O EAP transfere informações sobre a autenticação entre o AP e o servidor de autenticação.

O processo EAP envolve:

- Um utilizador solicita uma conexão pelo AP.
- O AP transmite os dados de identificação do utilizador para o servidor de autenticação.
- O servidor, por sua vez, pede ao AP uma prova de validade das informações para conectar o utilizador.
- O AP, por fim, obtém a verificação do utilizador e torna a enviar ao servidor de autenticação.

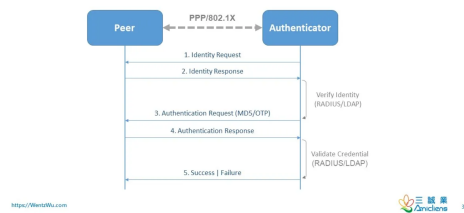
3.3.4 Substituição

Ao longo dos anos, várias extensões do EAP foram desenvolvidas para aprimorar a segurança da autenticação em redes. Algumas dessas extensões incluem:

- ***Transport Layer Security (EAP-TLS)***. Método que utiliza certificados digitais para autenticar tanto o servidor quanto o cliente. É considerado altamente seguro, mas requer uma gestão mais complexa dos certificados.
- ***Protected Extensible Authentication Protocol (PEAP)***. Método que autentica clientes usando certificados do lado do servidor. É criado um túnel *Transport Layer Security* (TLS) do servidor para o cliente, para que ele possa ser autenticado através desse túnel criptografado.
- ***Subscriber Identity Module (EAP-SIM)***. Este método é usado em redes móveis e usa a infraestrutura de cartão SIM para autenticar dispositivos. Ele usa uma chave *Wired Equivalent Privacy* (WEP) por sessão para criptografar os dados. Esse método de autenticação exige que o cliente insira um código de verificação para ativar a comunicação com o SIM.

[29] [20]

Extensible Authentication Protocol


 Figura 3.3: *Extensible Authentication Protocol*

3.4 RADIUS (*Remote Authentication Dial-In User Service*)

O RADIUS é um protocolo cliente-servidor que permite que servidores de acesso remoto autenticem utilizadores e os autorizem a ter acesso aos recursos de uma determinada rede, especialmente de acesso remoto como VPNs ou redes *dial-up*. Permite que uma organização guarde as informações do utilizador numa base de dados central comum a todos os servidores remotos.

3.4.1 Contexto histórico

O RADIUS foi desenvolvido em 1991, num momento em que as redes *dial-up* estavam a tornar-se cada vez mais populares. Antes do RADIUS, a autenticação e autorização de utilizadores na rede costumavam ser feitas com cada servidor a manter informações sobre os utilizadores na sua própria base de dados, o que não era escalável e dificultava a gestão da base de dados.

3.4.2 Motivação

A motivação para o desenvolvimento do RADIUS foi criar um sistema centralizado e padronizado para autenticação e autorização de utilizadores em redes de acesso remoto, o que permitiria uma administração mais eficiente das informações de *login* dos utilizadores e até uma maneira mais fácil de controlar o acesso à rede.

3.4.3 Utilização

Neste protocolo, os utilizadores remotos conectam-se à rede através de um *network access server* (NAS). Por sua vez, o NAS consulta o servidor de autenticação para obter

informações sobre o utilizador. Quando um utilizador remoto inicia uma conexão através de um NAS, o mesmo envia uma solicitação de autenticação ao servidor RADIUS. A solicitação pode incluir o nome do utilizador, *password* e endereço IP.

O RADIUS autentica usando dois protocolos:

- **PAP** (*Password Authentication Protocol*). O cliente RADIUS envia o ID do utilizador e a senha para o servidor de autenticação RADIUS. Se as credenciais estiverem corretas com as credenciais guardadas na base de dados, o servidor autentica o utilizador e o cliente RADIUS permite a conexão à rede.
- **CHAP** (*Challenge Handshake Authentication Protocol*). Semelhante ao PAP, mas é considerada mais segura, porque criptografa trocas de autenticação, ou seja, todas as senhas de um utilizador são enviadas criptografadas entre o cliente e o servidor RADIUS, para impedir alguém de as interceptar em *plain text*.

3.4.4 Substituição

Atualmente o RADIUS ainda é usado em diversos contextos. Porém, à medida que as redes se tornaram maiores e mais complexas, surgiram soluções de gestão de utilizadores mais avançadas, como o protocolo SAML e o OIDC, que oferecem maior flexibilidade e escalabilidade.

[23] [1]

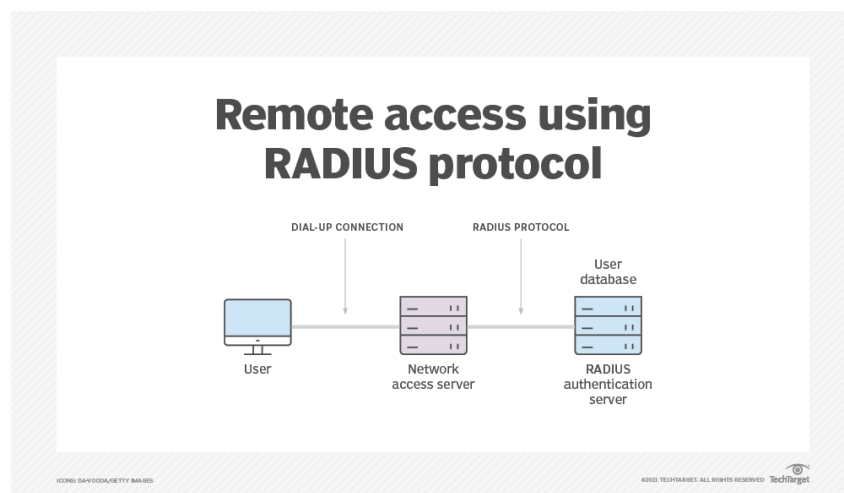


Figura 3.4: *Remote Authentication Dial-In User Service*

3.5 LDAP (*Lightweight Directory Access Protocol*)

O LDAP é um protocolo que organiza informações hierarquicamente e permite que qualquer utilizador localize dados sobre organizações, indivíduos ou outros recursos, como dispositivos, arquivos, etc. No entanto, em redes locais, um utilizador geralmente não sabe o nome de domínio de todos os objetos (servidores, discos, computadores, impressoras, etc.) disponíveis, mas o LDAP permite que um utilizador procure uma informação sem saber onde está localizada.

3.5.1 Contexto histórico

O LDAP surgiu num momento em que as organizações começaram a perceber a importância de ter um mecanismo padronizado e eficiente para armazenar e recuperar informações de diretoria, como informações de utilizadores, dispositivos de rede e outros objetos. Antes do LDAP, era usado o *Directory Access Protocol* (DAP), mas era complexo e pesado. O LDAP foi projetado para ser mais simples, eficiente e escalável.

3.5.2 Motivação

A motivação por trás do LDAP estava relacionada à necessidade de gerir informações de diretoria de maneira eficiente. As organizações precisavam de um padrão para gerir e consultar informações de maneira uniforme em vários sistemas e serviços, que se integrasse bem com outros padrões de autenticação e que fosse escalável (recomendado tanto para pequenas como grandes empresas).

3.5.3 Utilização

O LDAP fornece um local para a autenticação centralizada dos utilizadores de uma rede. Com uma única autenticação, qualquer utilizador autorizado pode validar o acesso a diversas aplicações e serviços, sem a necessidade de fazer *login* outra vez. O protocolo pode ainda ser utilizado para adicionar operações numa base de dados do servidor de diretoria. O LDAP permite ainda bloquear acessos, pesquisar, comparar e modificar entradas existentes, abandonar solicitações ou desvincular operações.

Uma configuração LDAP é organizada numa hierarquia simples, composta pelos seguintes elementos:

- **Root directory.** Fornece informações gerais sobre a diretoria, como a versão do LDAP que está a ser usado, as políticas de acesso e configurações de segurança.
- **Países.** Representa uma unidade organizacional.
- **Organização.** Usadas para agrupar *root directory's* relacionadas com a estrutura organizacional da empresa.
- **Indivíduos.** Incluem pessoas, ficheiros ou recursos partilhados.

O protocolo LDAP é usado no *Active Directory Microsoft* (ADM). O Exchange Server usa o LDAP para se comunicar com o *Active Directory* (AD).

AD é um serviço de diretoria usado para gerir domínios, utilizadores e recursos distribuídos, como objetos para *Windows*. Contém informações sobre todas as contas de utilizador numa rede inteira. Verifica a identidade dos mesmos e decide a quais recursos têm permissão para aceder com base nas políticas de autorização definidas, o que garante a segurança dos dados e dos sistemas numa organização.

O LDAP usa uma consulta baseada em *string* relativamente simples para extrair informações do AD. O LDAP pode armazenar e extrair objetos, como nomes de utilizador e senhas no AD, e compartilhar esses dados de objetos numa rede.

3.5.4 Substituição

Algumas tecnologias mais recentes e alternativas surgiram ao longo do tempo, como *OAuth* e *OIDC*, ganharam popularidade, oferecendo alternativas ao LDAP em determinados cenários, como por exemplo, permitir a autenticação onde um serviço de identidade confiável emite *tokens* de acesso para autenticar utilizadores em vários serviços. Nesse cenário, o LDAP pode ser usado para armazenar informações de utilizador, enquanto o *OAuth* é usado para autenticar os utilizadores, aproveitando as credenciais do LDAP.

[22] [28]

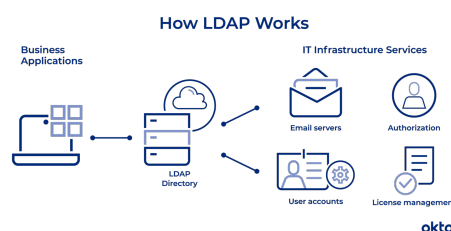


Figura 3.5: *Lightweight Directory Access Protocol*

3.6 *Kerberos*

Kerberos é um protocolo de autenticação que permite comunicações seguras com uma rede insegura, como por exemplo a Internet. Foi desenvolvido para o Projeto Athena no MIT. *Kerberos* vem da mitologia grega, onde Cerbero era um cão de três cabeças que guardava a entrada dos portões de Hades. No contexto do protocolo *Kerberos*, ele desempenha o papel de garantir a segurança da autenticação.

As três cabeças representavam:

- O cliente.
- O servidor que fornece acesso aos recursos da rede.
- O *Key Distributor Center* (KDC), que atua como serviço de autenticação confiável a outras entidades.

3.6.1 Contexto histórico

Na final da década de 1980, em que a primeira versão do *Kerberos* começou a ser desenvolvida, as redes de computadores começaram a tornar-se mais populares em ambientes corporativos. No entanto, a autenticação de utilizadores e serviços em redes ainda era um desafio. A maioria dos sistemas de autenticação da época usava senhas que eram transmitidas pela rede em *plain text*, tornando-as vulneráveis a ataques, como o *sniffing*. Isso representava um risco significativo de segurança. O *Kerberos* começou a ser desenvolvido no final da década de 1980. Contudo, a versão 5 (versão atual), foi publicada em 1993.

3.6.2 Motivação

A motivação para o desenvolvimento da tecnologia *Kerberos* era responder às preocupações de segurança da época. Ele foi projetado para fornecer autenticação segura, evitando a transmissão de senhas em *plain text* e a exposição completa das mesmas, reduzindo o sucesso de potenciais ataques.

3.6.3 Utilização

Abaixo segue uma descrição de como o protocolo *Kerberos* funciona.

- **Solicitação do servidor de autenticação.** O cliente *Kerberos* envia uma solicitação para o servidor KDC. O servidor de autenticação verifica se o cliente está na base de dados e recupera a chave privada do cliente.
- **Resposta do servidor de autenticação..** Se o nome de utilizador do cliente não for encontrado na base de dados KDC, o cliente não poderá ser autenticado e o processo de autenticação será interrompido. Caso contrário, o servidor de autenticação envia ao cliente um *Ticket Granting Ticket* (TGT) (usado para provar a identidade do utilizador em diversas partes de um sistema) e um chave de sessão (usada para criptografar e descriptografar dados durante uma transmissão).
- **Pedido de *ticket* de serviço.** Quando o cliente é autenticado, o mesmo solicita um *ticket* de serviço ao *Ticket Granting Server* (TGS) (servidor fornecedor de *tickets*). Esta solicitação deve ser acompanhada pelo TGT enviado pelo servidor de autenticação KDC.
- **Resposta do *ticket* de serviço.** Se o TGS puder autenticar o cliente, ele vai enviar as credenciais e um *ticket* para aceder ao serviço solicitado. Esta transmissão é criptografada com uma chave de sessão específica para o utilizador e para o serviço que está a ser acedido.
- **Solicitação e resposta do servidor de aplicações.** O cliente envia uma solicitação para aceder ao servidor de aplicações. Esta solicitação inclui o *ticket* de serviço recebido na etapa 4. Se o servidor de aplicações conseguir autenticar o pedido, o cliente pode aceder ao servidor.

Kerberos é usado para autenticar entidades que solicitam acesso a recursos de rede, especialmente em grandes redes para suportar SSO, uma vez que um dos princípios do protocolo é a senha ser inserida apenas uma vez a cada sessão, o que significa que os utilizadores podem autenticar-se apenas uma vez e aceder, com a devida autorização, a quaisquer recursos. Alguns sistemas que suportam o protocolo *Kerberos* são: **Serviços da Amazon Web, MacOS apple, Google Cloud, Microsoft Azure, Microsoft Windows Server e Active Directory.**

3.6.4 Substituição

Embora o *Kerberos* tenha sido eficaz em proporcionar autenticação segura, não é uma solução adequada para todos os cenários de autenticação. Em ambientes modernos, o uso de soluções baseadas em *tokens*, como *OAuth* e *OIDC*, tornou-se mais comum para autenticação em aplicações *web* e serviços em nuvem.

[21] [10]

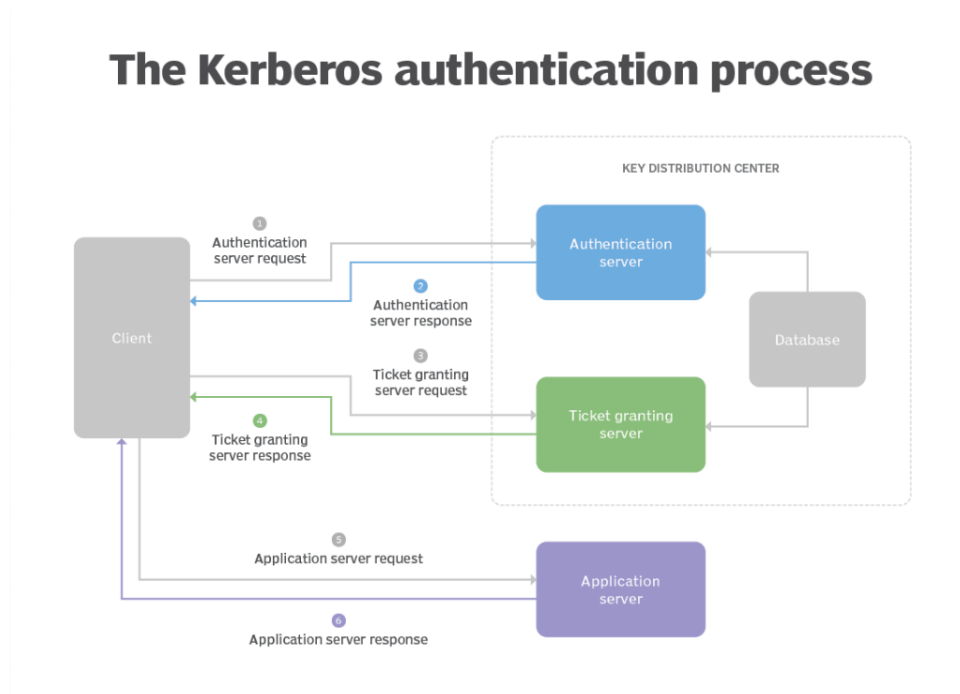


Figura 3.6: *Kerberos*

3.7 SAML (*Security Assertion Markup Language*)

O SAML é um protocolo que foi desenvolvido para permitir a troca de informações de autenticação e autorização entre sistemas, especialmente num contexto de autenticação única (SSO). O SAML é implementado com a *Extensible Markup Language* (XML) para partilha de dados. Resumidamente, o protocolo permite que diferentes sistemas compartilhem informações de autenticação e autorização de maneira segura, facilitando o acesso dos utilizadores a várias aplicações e serviços sem a necessidade de fazer *login* separadamente em cada um deles.

3.7.1 Contexto histórico

À medida que a era da Internet se consolidava no início do século XXI, surgia a crescente necessidade de estabelecer mecanismos de autenticação e autorização entre diferentes domínios. Embora o *Kerberos* tenha sido adotado para uso em redes corporativas, ele apresentava limitações quando se tratava de autenticação entre domínios distintos. Foi nesse contexto que o SAML (a partir de 2001) surgiu como uma resposta à crescente procura por autenticação e autorização seguras na Internet.

3.7.2 Motivação

A motivação por trás do SAML era criar um protocolo que permitisse que as organizações compartilhassem informações de autenticação e autorização de forma segura, sem a necessidade de compartilhar senhas, o que melhoraria a experiência do utilizador e aumentava a segurança.

3.7.3 Utilização

Um processo de autenticação SAML funciona da seguinte forma:

- **Autenticação do utilizador.** O utilizador inicia uma sessão com um IdP, fazendo *login*.
- **Envio da afirmação SAML.** Após a autenticação bem-sucedida, o IdP cria um documento SAML (em XML) que inclui informações sobre o utilizador autenticado, como nome, nível de autorização, etc. O IdP envia a afirmação SAML de volta para

o *Services Provider* (SP).

- **Validação de Credenciais.** O SP recebe a afirmação SAML do IdP e valida a assinatura digital no documento para garantir que corresponde ao utilizador autenticado. Com base nas informações contidas no documento SAML, o SP decide se concede ou não acesso ao utilizador. Se o acesso for concedido, então o utilizador vai poder aceder ao recurso que quer e que tem permissão.

SAML é uma protocolo usado para solicitar autenticação. É comum o seu uso ser com o tipo de autenticação SSO. Alguns serviços que implementam serviços SSO, usando SAML são o *Microsoft Azure AD*, *VMware vSphere*, entre outros serviços relacionados com a computação em nuvem.

Em muitos casos, o SAML e o OAuth 2.0 são usados em conjunto para fornecer autenticação e autorização abrangentes em sistemas. Por exemplo, numa configuração típica de SSO, o SAML é usado para autenticar o utilizador e o OAuth 2.0 é usado para autorizar uma aplicação a aceder a recursos em nome do utilizador após a autenticação. Resumidamente, enquanto o SAML é um protocolo de autenticação, o OAuth 2.0 é um protocolo de autorização.

3.7.4 Evolução

Embora o SAML tenha sido amplamente adotado e ainda continue a ser usado, outros protocolos surgiram, como o (OIDC), que é um protocolo mais leve e flexível em comparação com o SAML, tornando-o mais adequado para alguns casos de uso, como autenticação de aplicações móveis e APIs.

[6] [24]

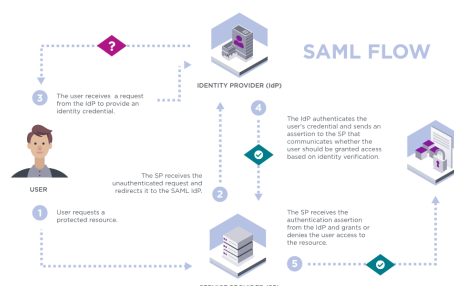


Figura 3.7: *Security Assertion Markup Language*

3.8 OIDC (*OpenID Connect*)

O OIDC é um protocolo de autenticação que permite que os utilizadores se autenticuem de forma segura em aplicações e serviços *online*. Está intimamente relacionado com o OAuth 2.0, pois é construído como uma camada de autenticação em cima do mesmo. Ambos os protocolos desempenham funções distintas, mas podem trabalhar juntos para fornecer autenticação e autorização seguras em aplicações e serviços *web*. O OIDC é um protocolo aberto, o que facilita a interoperabilidade entre diferentes aplicações.

3.8.1 Contexto histórico

O OIDC surgiu no início dos anos 2010 e visava resolver o problema do facto dos utilizadores precisarem de criar várias contas em diversos sites e serviços *online*. Com o OIDC, os utilizadores poderiam usar uma única identidade para aceder a vários serviços, simplificando o processo de *login*.

3.8.2 Motivação

A motivação por trás do OIDC foi oferecer um protocolo de autenticação (e autorização) mais seguro, projetado para atender às seguintes necessidades:

- **Segurança.** O OIDC usa o OAuth 2.0 para fornecer autorização a determinados recursos. Além disso, inclui camadas adicionais de segurança, como a troca de informações no formato JWT e a verificação de identidade por parte do *OpenID Provider* (OP).
- **Simplificação de *login*.** O OIDC visa simplificar o processo de *login* para os utilizadores, uma vez que permite que contas, como a *Google* ou *Facebook* façam login em vários serviços sem a necessidade de criar uma nova conta em cada *site*.

3.8.3 Utilização

Abaixo está descrito como funciona o processo de autenticação do OIDC.

- O utilizador inicia o processo de autenticação, fazendo *login* num determinado site. A aplicação envia uma solicitação de autenticação ao OP.

- O OP responde à solicitação de autenticação com um redirecionamento para o IdP especificado (por exemplo, *Google* ou *Facebook*), onde fornece as credenciais de *login* desse mesmo IdP.
- Após a autenticação, o IdP gera um *token* de identidade, que contém informações do utilizador autenticado, como o ID do utilizador, nome, endereço de *e-mail*, etc. É assinado digitalmente pelo IdP para garantir sua integridade e é enviado de volta para a aplicação solicitante.
- A aplicação solicitante verifica a assinatura digital no *token* de identidade. Se a verificação for bem sucedida, então o utilizador é autenticado com sucesso.

O OIDC foi amplamente adotado em diversos contextos, entre os quais:

- **SSO.** Organizações implementam, frequentemente, o OIDC, usando SSO, permitindo que os funcionários tenham acesso a vários sistemas e aplicações com uma única autenticação.
- **Aplicações móveis.** O OIDC é usado em aplicações móveis para simplificar o processo de *login* e oferecer uma melhor experiência de utilização.

3.8.4 Evolução

o OIDC surgiu como uma resposta à necessidade de autenticação e autorização seguras na *web* e simplificação do processo de *login*. Atualmente, é bastante usado, mas a constante evolução nos aspetos de segurança na autenticação pode trazer novos métodos de autenticação ainda mais eficazes no futuro.

[18] [17]

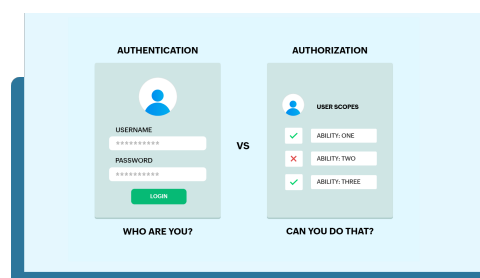


Figura 3.8: *OpenID Connect*

Capítulo 4

Linux Pluggable Authentication Modules

Os módulos de autenticação desempenham um papel fundamental na autenticação de utilizadores e na segurança do sistema, pois permitem que o *Linux* verifique a identidade dos mesmos durante o processo de *login*.

Atualmente, no *Linux*, o sistema de autenticação é gerido pelo PAM, que (como está exposto com mais detalhe de seguida) é uma estrutura flexível que permite que os administradores de sistema configurem várias formas de autenticação, como senhas, chaves, *tokens*, biometria, etc, ou seja, basicamente, permite a configuração regras de autenticação personalizadas.

4.1 Evolução dos mecanismos de Autenticação *Linux*

A evolução dos mecanismos de autenticação no *Linux* foi impulsionada pela necessidade de melhorar a segurança, a flexibilidade e a capacidade de gestão dos sistemas. Antes da introdução do PAM, a autenticação no *Linux* era tratada de maneira mais inflexível. O PAM foi criado para solucionar esses problemas e fornecer uma estrutura de autenticação mais robusta e extensível. O sistema de autenticação era gerido por diferentes bibliotecas em diferentes aplicações, ou seja, cada aplicação tinha o seu próprio método de autenticação, o que tornava inflexível a implementação de políticas de segurança consistentes e a manutenção das mesmas.

O PAM foi introduzido no final da década de 1990 para fornecer uma estrutura de autenticação que permitisse aos administradores configurarem políticas de autenticação

de maneira centralizada e que oferecesse suporte a vários métodos de autenticação, como senhas, *tokens*, impressões digitais, etc.

4.2 Arquitetura do *framework* PAM

O PAM permite aos administradores de um sistema implementar vários mecanismos de autenticação nesse mesmo sistema, através do uso de módulos plugáveis.

As aplicações que estão configuradas para utilizar o PAM podem ser integradas com novas tecnologias sem a necessidade de as modificar. Essa flexibilidade permite aos administradores selecionar qualquer serviço de autenticação no sistema para uma determinada aplicação; usar vários mecanismos de autenticação para um determinado serviço; incluir novos módulos de autenticação sem modificar as aplicações existentes e usar uma senha para autenticação em determinados módulos.

O PAM consiste em alguns componentes principais, tais como:

- **Módulos PAM.** Cada método de autenticação é implementado como um módulo PAM separado, que pode ser adicionado ou removido sem afetar a aplicação.
- **Aplicações e serviços.** Cada aplicação e/ou serviço que deseje realizar autenticação deve invocar as funções PAM pertinentes no seu respectivo código.
- **Configuração PAM.** Feita em arquivos de configuração onde os administradores podem definir políticas de autenticação, tais como quais os módulos PAM a usar, a ordem de execução e os parâmetros específicos de autenticação.
- **Interação com o utilizador.** O PAM também lida com a interação com o utilizador, como solicitação de senha ou outros dados de autenticação.

4.2.1 Princípios subjacentes ao *design* do PAM

Como já indicado aquando da definição do conceito de PAM, os princípios subjacentes ao seu *design* incluem:

- **Modularidade.** Capacidade de adicionar, modificar ou remover módulos PAM sem afetar as aplicações que os estejam a usar.
- **Políticas de autenticação.** Capacidade de definir políticas de autenticação de maneira centralizada em ficheiros de configuração.
- **Flexibilidade.** O PAM suporta vários métodos de autenticação, permitindo a adaptação às necessidades específicas de um certo sistema.

4.2.2 Funcionamento da arquitetura PAM

O funcionamento do PAM é separado em quatro grupos de gestão, entre eles:

- ***Account*.** Responsável por verificar se o utilizador tem os privilégios necessários para aceder a um determinado recurso.
- ***Auth*.** Responsável por autenticar o utilizador.
- ***Session*.** Responsável pela execução de ações no início ou no fim da sessão do utilizador.
- ***Password*.** Responsável por alterar as credenciais do utilizador, quando necessário.

Os módulos PAM criados correspondem a um dos grupos de gestão mencionados acima. A localização desses módulos está na biblioteca `/lib/security/`.

Quando uma aplicação ou serviço solicita autenticação usando o PAM, este consulta a configuração definida nos arquivos de configuração. Com base nessa mesma configuração, os módulos relevantes são executados numa determinada ordem e podem ter como resultado o sucesso, o insucesso ou então solicitar informações adicionais do utilizador, se necessário para determinar esse mesmo resultado, que garante se a autenticação é bem sucedida ou não.

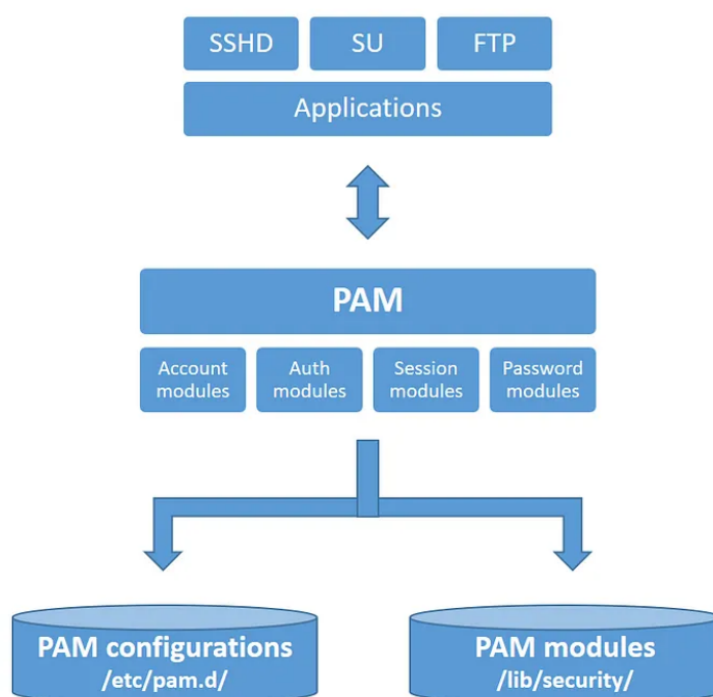


Figura 4.1: Pluggable Authentication Modules

4.2.3 Passos genéricos da configuração do PAM

Os arquivos de configuração do PAM estão localizados em `/etc/pam.d/`. Cada ficheiro contém configurações para a autenticação de um serviço específico e inclui regras que determinam como o PAM deve autenticar o utilizador.

Os ficheiros que definem como o PAM deve autenticar os utilizadores contêm regras que especificam quais os módulos PAM que vão ser utilizados para autenticação e a respetiva ordem em que são aplicados, os parâmetros e configurações específicas para cada módulo, como políticas de senha e tentativas de *login*, o comportamento em caso de sucesso ou falha da autenticação e qualquer interação necessária com o utilizador, como pedidos de senha ou outras informações.

Para configurar um arquivo PAM devemos ter em conta alguns passos.

- **Localizar o arquivo PAM e editá-lo.** Ir á diretoria `/etc/pam.d/` (onde estão localizados os arquivos de configuração PAM) e escolher o arquivo que se deseja configurar. É importante notar que é necessários privilégios de *root* para editar os ficheiros.
- **Definir regras de autenticação.** Dentro do arquivo PAM, estão várias linhas que descrevem as etapas do processo de autenticação. Cada linha refere-se a um

módulo PAM específico e inclui informações sobre como deve aplicado. Podemos adicionar, remover ou modificar essas linhas de acordo com as políticas de autenticação desejadas.

- **Especificar a ordem dos módulos.** Os módulos PAM são geralmente organizados numa determinada sequência relacionada com o processo de autenticação e podemos ajustar a ordem dos módulos para atender às necessidades de um determinado serviço.
- **Configurar os parâmetros dos módulos.** Cada módulo PAM tem parâmetros específicos que afetam seu comportamento e esses parâmetros devem ser configurados, conforme necessário para atender aos requisitos do serviço.
- **Definir o comportamento de sucesso e falha.** Para cada módulo, podemos especificar o que deve acontecer em caso de sucesso ou falha da autenticação, como permitir ou negar o acesso a recursos.

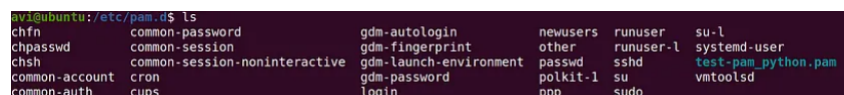


Figura 4.2: Localização dos arquivos de configuração do PAM

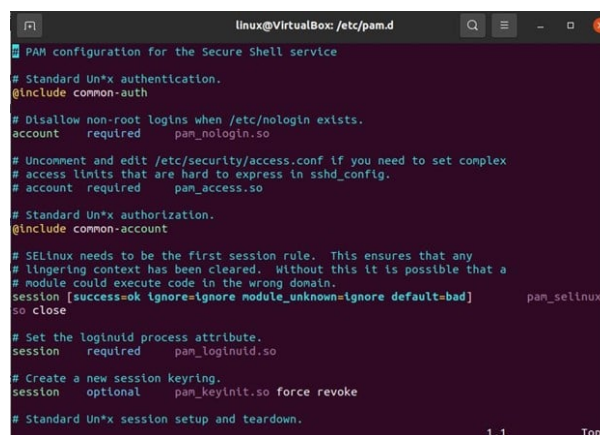


Figura 4.3: configuração de arquivos PAM

Cada linha que representa as regras de autenticação tem a seguinte sintaxe: "*type (Auth, Account, Password, Session) control (Required, Requisite, Sufficient, Optional) module arguments*".

[9] [12]

Capítulo 5

Trabalho Experimental

Neste capítulo, será abordado as diversas etapas do trabalho solicitado pelo professor, que consiste na integração de um cliente *Linux* num domínio de AD, criado no *Windows Server 2022*, com auxílio do *Windows Admin Center* nas aulas práticas de Serviços de Rede II. Também vai ser necessário o uso de um terceiro servidor, denominado *TrueNas Core* para armazenar a *home directory* dos utilizadores do AD. Irá ser mostrada nas seguintes secções, a instalação das ferramentas a usar no *Linux*, a integração do cliente *Linux* e do *TrueNas* no AD e o *sharing* da pasta *home* dos utilizadores do AD, quando dão *login* no *Linux*.

5.1 Configuração das placas no *Linux*

Para o ambiente *Linux*, usei 3 placas de rede: a **rede NAT** (SR2-NATNetwork - configurada nas aulas de Serviços de Rede II), **Host-Only Ethernet Adapter** e **NAT** (para conseguir instalar os pacotes corretamente).

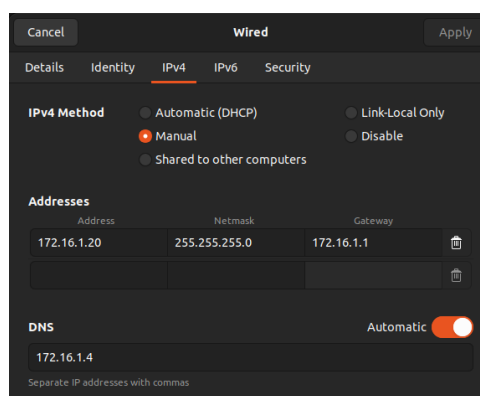


Figura 5.1: Configuração das placas da rede NAT

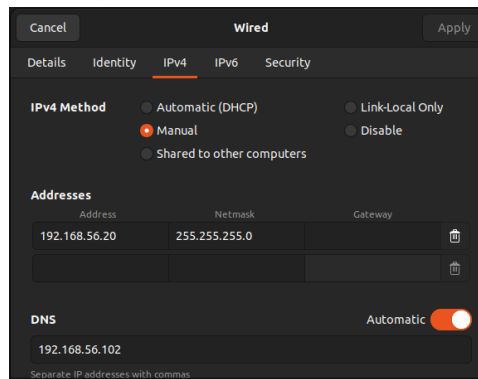


Figura 5.2: Configuração das placas do *host-only*

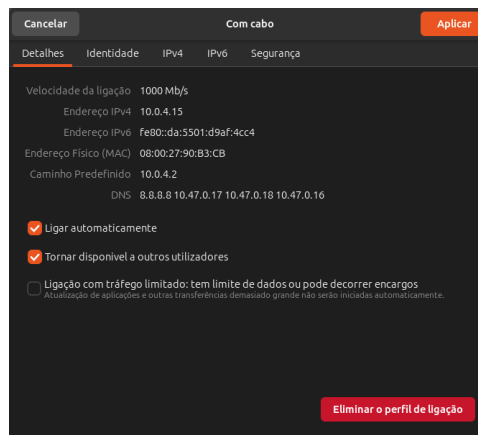


Figura 5.3: Configuração das placas da NAT

5.2 Instalação dos pacotes no *Linux*

Para conseguir integrar o *Linux* com sucesso no AD, foram instalados algumas ferramentas (*sssd*, *realmd*, *krb5-user* e *packagekit*).

```
linux@Pedro:~$ sudo apt install sssd realmd krb5-user packagekit
A ler as listas de pacotes... Pronto
A construir árvore de dependências... Pronto
A ler a informação de estado... Pronto
sssd já é a versão mais recente (2.9.1-2ubuntu2).
sssd está definido para ser instalado manualmente.
realmd já é a versão mais recente (0.17.1-2).
packagekit já é a versão mais recente (1.2.7-1).
packagekit está definido para ser instalado manualmente.
Os pacotes adicionais seguintes serão instalados:
  krb5-config libgssrpc4 libkadm5clnt-mit12 libkadm5srv-mit12 libkdb5-10
Pacotes sugeridos:
  krb5-k5libs krb5-doc
Serão instalados os seguintes NOVOS pacotes:
  krb5-config krb5-user libgssrpc4 libkadm5clnt-mit12 libkadm5srv-mit12
  libkdb5-10
0 pacotes actualizados, 6 pacotes novos instalados, 0 a remover e 35 não actuali-
zados.
É necessário obter 321 kB de arquivos.
Após esta operação, serão utilizados 1219 kB adicionais de espaço em disco.
Deseja continuar? [S/n] S
Obter:1 http://pt.archive.ubuntu.com/ubuntu mantic/main amd64 krb5-config all 2.
7 [22,0 kB]
```

Figura 5.4: Instalação de pacotes *Linux*

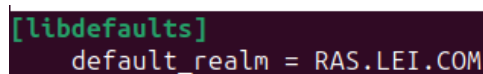
O comando `sudo apt install sssd realmd krb5-user packagekit` é usado para instalar diversos pacotes relacionados à integração de um sistema com um domínio AD no ambiente *Linux*.

- **SSSD.** É bastante usado para integrar sistemas *Linux* num ambiente AD, fornecendo autenticação, autorização e *Domain Name System* (DNS).
- **realmd.** O `realmd` é uma ferramenta que simplifica o processo de descoberta e junção a domínios.
- **krb5-user.** Este pacote instala o Kerberos no sistema, que é um protocolo de autenticação seguro e essencial para a integração do *Linux* com o AD.
- **packagekit.** Este pacote pode ser incluído para garantir que o sistema tem as ferramentas necessárias para trabalhar com pacotes instalados.

5.3 Juntar o cliente ao domínio

5.3.1 KRB5

Temos de editar o ficheiro `/etc/krb5.conf` para incluir as configurações de domínio que queria, que neste caso é "ras.lei.com".



```
[libdefaults]
    default_realm = RAS.LEI.COM
```

Figura 5.5: Edição do ficheiro KRB5.conf

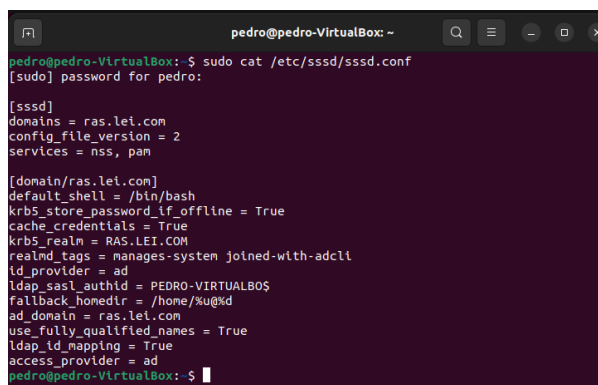
De seguida, configuramos o ficheiro `/etc/sss/sss.conf`. O SSSD fornece um conjunto de serviços para autenticação em sistemas *Linux*.

5.3.2 SSSD

- **domains:** Lista os domínios que o SSSD vai gerir. Neste caso é o "ras.lei.com".
- **config-file-version:** Indica a versão (2) do ficheiro `sss.conf`.
- **services:** Especifica os serviços fornecidos pelo SSSD, como *Name Service Switch* (NSS) e PAM.

[domain/ras.lei.com]

- **default-shell**: define a *shell* padrão para todos os utilizadores como `"/bin/bash"`.
- **krb5-store-password-if-offline**: Armazena a *password* Kerberos, se o servidor estiver *offline*.
- **cache-credentials**: Armazena em *cache*, as credenciais do utilizador.
- **krb5-realm**: Especifica as configurações do domínio como `"RAS.LEI.COM"`.
- **id-provider**: Define o IdP como `"ad"`(AD).
- **fallback-homedir**: Define a diretoria padrão em caso de falha. Neste caso é `"/home/user@ras.lei.com"`(alterada mais à frente).
- **ad-domain**: Especifica o domínio AD como `"ras.lei.com"`.
- **access-provider**: *Access Provider* configurado como `"ad"` para integrar com o AD.



```

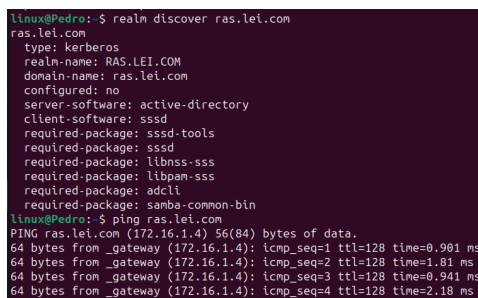
pedro@pedro-VirtualBox: ~
pedro@pedro-VirtualBox:~$ sudo cat /etc/sss/sss.conf
[sudo] password for pedro:
[sss]
domains = ras.lei.com
config_file_version = 2
services = nss, pam

[domain/ras.lei.com]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = RAS.LEI.COM
realmd_tags = manages-system joined-with-adcli
id_provider = ad
ldap_sasl_authid = PEDRO-VIRTUALBO$
fallback_homedir = /home/%u@d
ad_domain = ras.lei.com
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = ad
pedro@pedro-VirtualBox:~$

```

Figura 5.6: Edição do ficheiro sssd.conf

Como é possível ver, o Linux, já encontra domínio `"ras.lei.com"` e já tem conectividade com o mesmo.



```

linux@Pedro:~$ realm discover ras.lei.com
ras.lei.com
type: kerberos
realm-name: RAS.LEI.COM
domain-name: ras.lei.com
configured: no
server-software: active-directory
client-software: sssd
required-package: sssd-tools
required-package: sssd
required-package: libnss-sss
required-package: libpam-sss
required-package: adcli
required-package: samba-common-bin
linux@Pedro:~$ ping ras.lei.com
PING ras.lei.com (172.16.1.4) 56(84) bytes of data:
64 bytes from _gateway (172.16.1.4): icmp_seq=1 ttl=128 time=0.901 ms
64 bytes from _gateway (172.16.1.4): icmp_seq=2 ttl=128 time=1.01 ms
64 bytes from _gateway (172.16.1.4): icmp_seq=3 ttl=128 time=0.941 ms
64 bytes from _gateway (172.16.1.4): icmp_seq=4 ttl=128 time=2.18 ms

```

Figura 5.7: Verificação do domínio *Active Directory*

Para juntar o cliente *Linux* ao AD, usamos o comando `sudo realm join -U Linux-Cli ras.lei.com`. O utilizador **Linux-Cli** foi criado no AD do *Windows Server* para indicar o nome de utilizador que será usado para o *Linux* se juntar ao domínio. Este utilizador foi juntado ao grupo de *Domain Users* que tem as permissões necessárias para juntar clientes ao AD.

```
linux@Pedro:~$ sudo realm join -U Linux_Cli ras.lei.com
Senha para Linux_Cli:
linux@Pedro:~$ sudo realm join -U Linux_Cli ras.lei.com
realm: Já se juntou a este domínio
linux@Pedro:~$
```

Figura 5.8: Join do cliente *Linux*

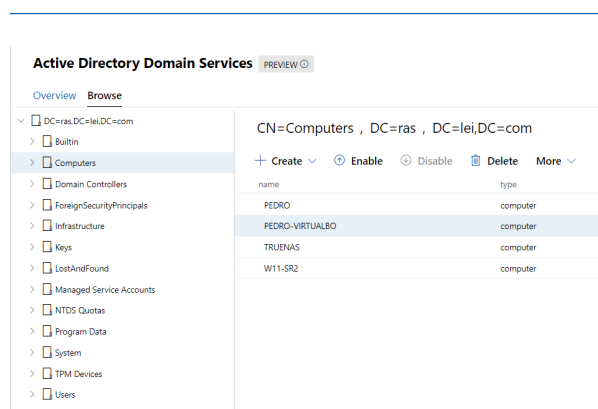


Figura 5.9: Cliente no *Active Directory*

Ao realizar o processo de junção (*realm join*) ou de descoberta (*realm discover*), o *Linux* envia consultas DNS para resolver o nome de domínio, que neste caso é **ras.lei.com**. Estas consultas ajudam a descobrir onde estão localizados os controladores de domínio associados ao domínio.

- **Consultas DNS *Service Records* (SRV).** O cliente (*Linux*) envia consultas DNS SRV (que contêm informações sobre os serviços disponíveis, como os serviços LDAP e *Kerberos* associados ao domínio) para descobrir os controladores de domínio disponíveis no domínio AD.
- **Registos A e AAAA.** O *Linux* realiza consultas DNS para obter os registos A (IPv4) e AAAA (IPv6) associados aos controladores de domínio para estabelecer conexões com os controladores de domínio identificados.
- **Registo PTR.** É comum neste processo, o cliente realizar consultas DNS reversas com o objetivo de traduzir endereços *Internet Protocol* (IP) em nomes.

Numa situação em que é testada a conectividade entre o *Linux* e um *host* no AD (por exemplo, "**ping truenas.ras.lei.com**"), o processo é semelhante ao descrito acima, acrescentando os pacotes *Internet Control Message Protocol* (ICMP) *request* e *reply* trocados entre o *Linux* e o servidor, que indicam se a conectividade foi bem sucedida ou não.

5.4 *TrueNAS Core*

O *TrueNAS Core* é um sistema operativo projetado para fornecer soluções de armazenamento de dados de forma confiável e oferecer uma ampla gama de recursos para gerir armazenamento de dados em rede.

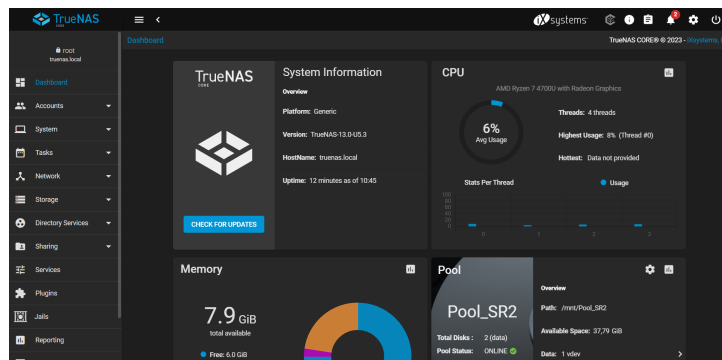


Figura 5.10: *TrueNAS Core*

O *TrueNAS Core* lida com alguns processos, tais como:

- **Accounts.** Lida com a administração de contas de utilizador e grupos. Podemos configurar permissões e acessos para diferentes utilizadores e grupos, garantindo que o acesso aos recursos de armazenamento seja feito de maneira segura.
- **System.** Aqui, podemos encontrar configurações gerais do sistema, como informações de *hardware*, configurações de rede, configurações de data e hora (*Network Time Protocol* (NTP)), atualizações de *software*, etc.
- **Tasks.** Este tópico engloba tarefas agendadas, como *backups*, *snapshots*, etc.

- **Network.** Inclui todas as configurações relacionadas à rede, como configurações de interface de rede, configurações de rotas estáticas, DNS, etc.
- **Storage.** A gestão de armazenamento é uma parte crucial do *TrueNAS*. Este tópico inclui configurações para discos, *pools* de armazenamento e outras configurações relacionadas ao armazenamento de dados.
- **Directory Services.** Este tópico lida com a integração do *TrueNAS* em ambientes de AD.
- **Sharing.** Aqui, podemos configurar a forma como os dados são compartilhados. Isso inclui configurações para serviços como *Server Message Block* (SMB), *Network File System* (NFS), *Apple Filing Protocol* (AFP), entre outros.

5.4.1 Configuração básica do *TrueNAS*

Para o ambiente *TrueNAS* instalado, usei 2 placas de rede: a **rede NAT** (SR2-NATNetwork) e a **Host-Only Ethernet Adapter**. Configurei manualmente as interfaces e os DNS *Servers* de modo a estarem na mesma rede que o servidor. Por fim, adicionei ao *TrueNAS*, o AD com o qual estamos a trabalhar.

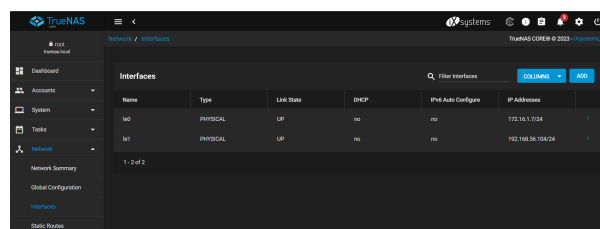


Figura 5.11: Configuração das interfaces do *TrueNAS*

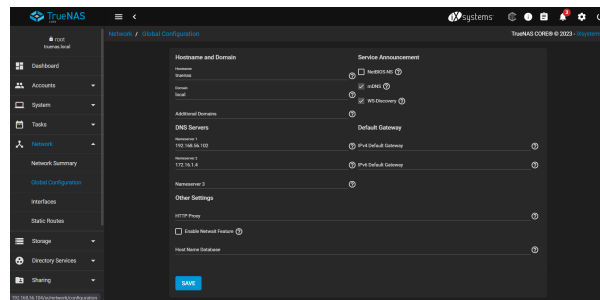


Figura 5.12: Configuração dos DNS Servers

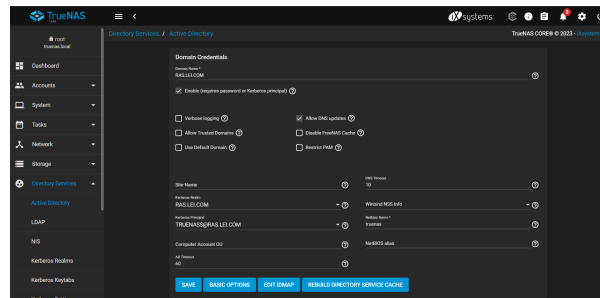


Figura 5.13: Definição do Active Directory no TrueNAS

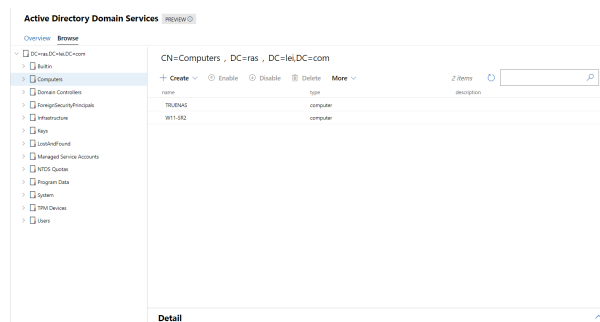


Figura 5.14: Integração do TrueNAS no Active Directory

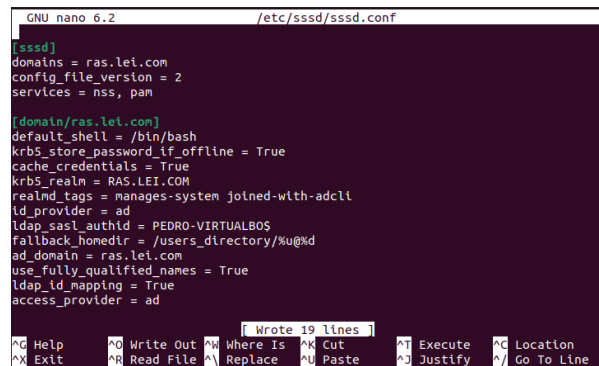
5.4.2 *Sharing da Home Directory* dos utilizadores

Nesta subsecção, o objetivo é fazer com que a *Home Directory* dos utilizadores do AD seja partilhada com o *TrueNAS*, à medida que dão *login* no *Linux*.

Algumas alterações feitas no *Linux* foram:

- **sssd.conf.** Neste ficheiro, o campo da **fallback-homedir** foi alterado para **/users-directory/%u@%d**, para que a directoria *home* padrão seja **/users-directory/**, seguido pelo nome do utilizador e o domínio a que pertence. Ou seja, quando um utilizador do AD faz *login* no *Linux*, a sua directoria *home* padrão vai ser a mencionada nesse campo.

- **file system table (fstab).** Este ficheiro determina como os ficheiros são montados e disponibilizados para uso, durante a inicialização. Foi acrescentada ao mesmo, a linha **192.168.56.104:/mnt/Pool-SR2 /users-directory nfs defaults 0 0**, onde é especificada o servidor *TrueNAS*, o caminho da diretoria partilhada (onde irá residir no *TrueNAS*, as *home Directory's* dos utilizadores), o ponto de montagem (no *Linux*) do sistema de ficheiros, onde o conteúdo partilhado do servidor será acessível e o tipo de sistema de ficheiros usado, bem como as opções de montagem padrão para o tipo de sistema em questão (NFS, neste caso).
- **sudoers.** No ficheiro **sudoers** (que define as permissões para a utilização do comando **sudo** por utilizadores ou grupos), foi definido que o utilizador **Administrator**, do *Windows Server*, teria privilégios de super-utilizador.



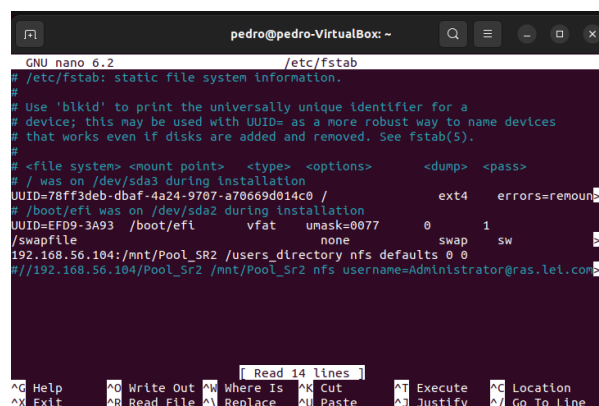
```

GNU nano 6.2 /etc/sss/sss.conf
[sss]
domains = ras.let.com
config_file_version = 2
services = nss, pam

[domain/ras.let.com]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = RAS.LEI.COM
realmd_tags = manages-system joined-with-adcli
id_provider = ad
ldap_sasl_authid = PEDRO-VIRTUALBOX
fallback_homedir = /users_directory/%u%d
ad_domain = ras.let.com
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = ad

```

Figura 5.15: Alteração do ficheiro sssd.conf



```

GNU nano 6.2 /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda3 during installation
UUID=78ff3deb-dbar-4a24-9707-a70669d014c0 / ext4 errors=remount-ro
# /boot/efi was on /dev/sda2 during installation
UUID=EF09-3A93 /boot/efi vfat umask=0077 0 1
/swapfile none swap 0 0
192.168.56.104:/mnt/Pool-SR2 /users_directory nfs defaults 0 0
#//192.168.56.104/Pool_Sr2 /mnt/Pool_Sr2 nfs username=Administrator@ras.let.com

```

Figura 5.16: Alteração do fstab

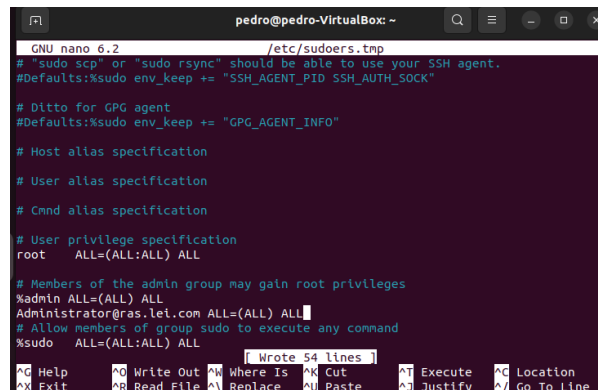


Figura 5.17: Alteração do sudoers

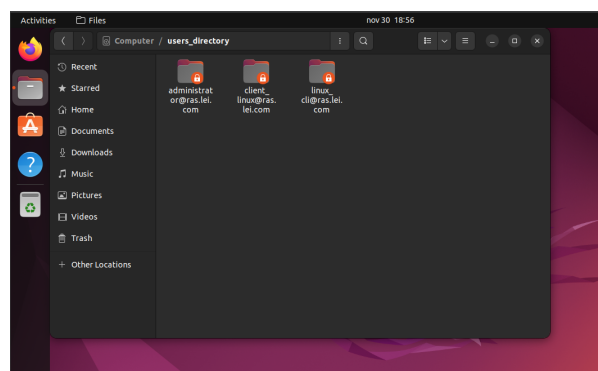
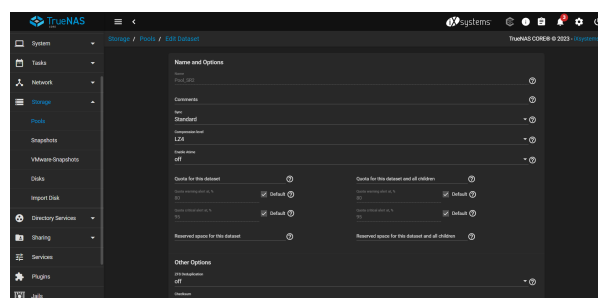
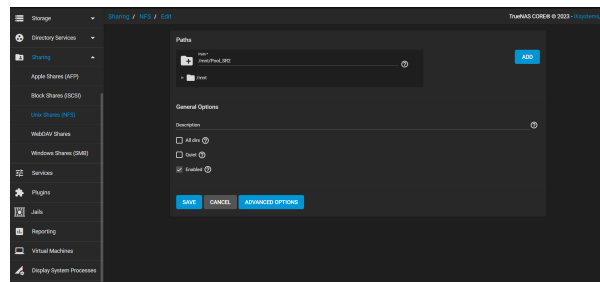


Figura 5.18: Pasta dos utilizadores do Active Directory

No *TrueNAS*, foi criada uma *pool* que tem como objetivo gerir o armazenamento das *home directory's* dos utilizadores que dão *login* no *Linux*. A utilização de *pools* é uma abordagem eficiente para gerir e distribuir dados em vários discos, proporcionando maior desempenho e confiabilidade. As *pools* também suportam várias tecnologias de armazenamento, como o caso da NFS, que é usada como forma de partilha de dados com ambientes *Linux*.

Figura 5.19: Criação da *pool*

Figura 5.20: *Sharing NFS*

Como é possível ver na *shell* do *TrueNAS*, a *home directory* dos utilizadores do AD é alojada na *pool* criada, á medida que fazem *login* no *Linux*.

```
root@truenas[~]# cd /mnt
root@truenas[/mnt]# cd Pool_SR2
root@truenas[/mnt/Pool_SR2]# ll
total 19
drwxrwxrwx  6 root   wheel           uarch   7 Dec  2 09:35 ./
drwxr-xr-x  3 root   wheel           uarch 128 Dec  2 13:26 ../
drwxrwxrwx 14 nobody wheel           uarch  17 Dec  2 14:34 administrator
@ras.lei.com/
drwx----- 3 nobody wheel           uarch   4 Dec  2 15:02 client_linux@
ras.lei.com/
drwxrwxr-x  2 root   RASLEI\domain users uarch   2 Dec  1 15:17 dataset/
-rw-rw-rw-  1 root   wheel           uarch  16 Dec  1 11:34 file.txt
drwx----- 2 nobody wheel           uarch   3 Dec  2 09:35 linux_cli@ras
.lei.com/
root@truenas[/mnt/Pool_SR2]#
```

Figura 5.21: *Shell* do TrueNAS

[27]

Capítulo 6

Conclusão

Neste projeto, foi possível consolidar conhecimentos sobre o funcionamento dos diversos tipos e protocolos de autenticação. Aprendi que a segurança da autenticação é um desafio em constante evolução, que exige adaptação a novas ameaças e tecnologias emergentes.

Com o projeto experimental, foi possível colocar em prática alguns conhecimentos adquiridos nas metas anteriores para resolução e compreensão do mesmo, bem como os conhecimentos adquiridos nas aulas de Serviços de Rede II, sobre integração de AD. Este projeto foi crucial, para melhorar as minhas aptidões em ambiente *Linux*, uma vez que já não estava em contacto tão próximo com o ambiente desde a cadeira de Sistemas Operativos, do 2º ano de licenciatura. Fiz pesquisas para compreender como se integrava um cliente *Linux* num AD. Tive algumas dificuldades a partilhar as *home directory's* dos utilizadores no *TrueNAS*, mas que foram ultrapassadas com o auxílio de pesquisas de erros que me surgiram e com a ajuda de alguns colegas, que foram cooperantes comigo.

Referências

- [1] Remote Authentication Dial In User Service (RADIUS). *C. Rigney, S. Willens, A. Rubens, W. Simpson*. URL: <https://www.rfc-editor.org/rfc/rfc2865.html>.
- [2] Tayibia Bazaz e Aqeel Khalique. *A Review on Single Sign on Enabling Technologies and Protocols*. URL: <https://shre.ink/ncyK>.
- [3] cybersecurity.asee.co. *history-of-authentication*. URL: <https://cybersecurity.asee.co/blog/history-of-authentication/>.
- [4] Farkhod Alisherov A. Debnath Bhattacharyya Rahul Ranjan e Minkyu Choi. *Biometric Authentication: A Review*. URL: <https://www.biometrie-online.net/images/stories/dossiers/generalites/International-Journal-of-u-and-e-Service-Science-and-Technology.pdf>.
- [5] descope. *password-authentication*. URL: <https://www.descope.com/learn/post/password-authentication>.
- [6] entrust. *what-is-saml*. URL: <https://www.entrust.com/resources/faq/what-is-saml>.
- [7] fortinet. *authentication-token*. URL: <https://www.fortinet.com/resources/cyberglossary/authentication-token>.
- [8] Fadi Aloul; Syed Zahidi; Wassim El-Hajj. *Two factor authentication using mobile phones*. URL: <https://shre.ink/nj9L>.
- [9] ibm. *Pluggable Authentication Modules*. URL: <https://www.ibm.com/docs/en/aix/7.2?topic=system-pluggable-authentication-modules>.
- [10] Jeffrey I. Schiller Jennifer G. Steiner Clifford Neuman. *Kerberos: An Authentication Service for Open Network Systems*. URL: <http://courses.isi.jhu.edu/netsec/papers/kerberos.pdf>.
- [11] logintc. *biometric-authentication*. URL: <https://www.logintc.com/types-of-authentication/biometric-authentication/>.
- [12] medium. *Linux PAM — How to create an authentication module*. URL: <https://medium.com/@avirzayev/linux-pam-how-to-create-an-authentication-module-cc132115bdc5>.

- [13] Pierre-Alain Fouque Michel Abdalla e David Pointcheval. *Password-Based Authenticated Key Exchange in the Three-Party Setting*. URL: https://link.springer.com/chapter/10.1007/978-3-540-30580-4_6.
- [14] networkworld. *How does certificate-based authentication work?* URL: <https://www.networkworld.com/article/2226498/infrastructure-management-simply-put-how-does-certificate-based-authentication-work.html>.
- [15] Vincent Omollo Nyangaresi; Sunday Oyinlola Ogundoyin. *Certificate Based Authentication Scheme for Smart Homes*. URL: <https://ieeexplore.ieee.org/abstract/document/9607322>.
- [16] okta. *pap-security*. URL: <https://www.okta.com/identity-101/pap-security/>.
- [17] openid. *How OpenID Connect Works*. URL: <https://openid.net/developers/how-connect-works/>.
- [18] pingidentity. *OpenID Connect (OIDC)*. URL: <https://www.pingidentity.com/en/resources/identity-fundamentals/authentication-authorization-standards/openid-connect.html>.
- [19] techtarget. *CHAP-Challenge-Handshake-Authentication-Protocol*. URL: <https://www.techtarget.com/searchsecurity/definition/CHAP-Challenge-Handshake-Authentication-Protocol>.
- [20] techtarget. *Extensible-Authentication-Protocol-EAP*. URL: <https://www.techtarget.com/searchsecurity/definition/Extensible-Authentication-Protocol-EAP>.
- [21] techtarget. *Kerberos*. URL: <https://www.techtarget.com/searchsecurity/definition/Kerberos>.
- [22] techtarget. *LDAP*. URL: <https://www.techtarget.com/searchmobilecomputing/definition/LDAP>.
- [23] techtarget. *RADIUS*. URL: <https://www.techtarget.com/searchsecurity/definition/RADIUS>.
- [24] techtarget. *SAML*. URL: <https://www.techtarget.com/searchsecurity/definition/SAML>.
- [25] techtarget. *single-sign-on*. URL: <https://www.techtarget.com/searchsecurity/definition/single-sign-on>.
- [26] techtarget. *two-factor-authentication*. URL: <https://www.techtarget.com/searchsecurity/definition/two-factor-authentication>.
- [27] ubuntu. *SSSD and Active Directory*. URL: <https://ubuntu.com/server/docs/service-sssd-ad>.

- [28] S. Kille W. Yeong T. Howes. *Lightweight Directory Access Protocol*. URL: <https://www.rfc-editor.org/rfc/rfc1777>.
- [29] Jyh-Cheng Chen; Yu-Ping Wang. *Extensible authentication protocol (EAP) and IEEE 802.1x: tutorial and empirical experience*. URL: https://ieeexplore.ieee.org/abstract/document/1561920?casa_token=bY8BJ9Xh2XcAAAAA:k5seNXMSef285c9EdFQs_PNNuuhItMHseh1kdIHwPqNBwsJ3VZcpyK8vbRxcGJtPY0tSezQ.
- [30] Adhitya Bhawiyuga; Mahendra Data; Andri Warda. *Architectural design of token based authentication of MQTT protocol in constrained IoT device*. URL: <https://ieeexplore.ieee.org/abstract/document/8272933/figures#figures>.
- [31] yubico. *what-is-certificate-based-authentication*. URL: <https://www.yubico.com/resources/glossary/what-is-certificate-based-authentication/>.

Capítulo 7

Anexos

7.1 Meta 1

Na meta 1, foi abordado a diversidade de métodos de autenticação, os princípios em que se baseiam e a sua constante procura por soluções seguras num ambiente digital. Desde as tradicionais senhas até as inovações como autenticação biométrica, cada abordagem tem suas vantagens e desafios, como foi abordado no capítulo 2.

7.2 Meta 2

Na meta 2 foi falado de diversos protocolos de autenticação que englobam conjuntos de regras e procedimentos utilizados para verificar a identidade de um utilizador ou de um sistema. Desde a PAP até ao OIDC, foram abordadas diversas evoluções dos protocolos (onde eram utilizados, porque é que surgiram, em que é que se baseavam, etc.).

7.3 Meta 3

Na meta 3, foi abordado a evolução dos mecanismos de autenticação no *Linux*, a motivação por trás do surgimento da arquitetura PAM, bem como os princípios em que se baseava e como era configurada.

7.4 Meta 4

Na meta 4, foi implementado uma autenticação de um cliente *Linux* num AD (como foi configurada a rede, quais os pacotes instalados, a configuração dos ficheiros para concluir a autenticação). Foi ainda mostrado, como poderia ser feito o *sharing* da *home directory* de cada utilizador do AD, á medida que dava *login* no *Linux*, que envolvia a criação de uma *pool* para guardar as pastas, a partilha de ficheiros NFS, a configuração do ficheiro *fstab* no Linux, etc.

7.5 Alterações no documento

O documento sofreu algumas alterações na meta 3, em relação às metas anteriores. Tentei adaptar o mesmo à *template* fornecida pelo professor. Algumas das formatações de capitulos, secções e subsecções foram também alteradas (foram adicionados capitulos ao relatório). Além disso, a partir da meta 2, foi adicionada uma página de acrónimos, que foi sendo retocada ao longo do relatório.