E-fólio A | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Segurança em Redes e Computadores

CÓDIGO: 21181

DOCENTE: Henrique S. Mamede

A preencher pelo estudante

NOME: Ana Patrícia Gomes Valente

N.º DE ESTUDANTE: 2301426

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 06/11/2025

Introdução: O objetivo deste trabalho é aplicar, de forma prática, os princípios básicos da segurança de computadores e redes a um cenário realista de uma organização de média dimensão, analisando a sua arquitetura de rede. A partir desse cenário, é feita uma análise das possíveis vulnerabilidades e ameaças, com base nos três pilares da segurança da informação - **Confidencialidade, Integridade e Disponibilidade (CIA).**

Com base nessa análise, são propostas medidas de proteção e controlos de segurança adequados a cada tipo de risco identificado, justificando as escolhas segundo princípios como o privilégio mínimo, a defesa em profundidade e a separação de funções. Além da análise teórica, o trabalho também inclui a implementação de uma solução criptográfica, utilizando encriptação simétrica (AES) e assimétrica (RSA), de forma a garantir a confidencialidade dos dados nas comunicações entre dois pontos.

Por fim, o relatório apresenta uma reflexão global sobre a estratégia de segurança, avaliando a eficácia das medidas sugeridas e identificando possíveis melhorias futuras de forma a reforçar a proteção da rede.

1. Análise de redes e ameaças :

Na rede analisada, o acesso à Internet é feito através de um router, que envia o tráfego para uma firewall. Esta firewall é a primeira barreira de defesa e tem a função de controlar o que entra e sai da rede. Depois da firewall, existe um switch central que liga todos os dispositivos e servidores internos. Entre estes encontra-se uma zona desmilitarizada (DMZ), onde estão colocados os serviços que precisam de ser acessíveis a partir do exterior, nomeadamente:

- **Servidor Web/Aplicacional**, que aloja o site e as aplicações;
- **Servidor de Base de Dados**, onde estão guardadas as informações importantes da empresa;
- **Servidor de Email**, usado para a troca de mensagens internas e externas.

Existe também um sistema de deteção e prevenção de intrusões (IDS/IPS) que monitoriza o tráfego de rede com o objetivo de identificar e bloquear atividades suspeitas.

Os utilizadores acedem à rede através de dois tipos de ligações:

- **LAN (Local Area Network)**, usada por equipamentos fixos (computadores de secretária, impressoras, etc.) e outros dispositivos ligados por cabo;
- **WLAN (Wireless Local Area Network)**, usada por dispositivos móveis (portáteis, tablets, smartphones), que se ligam sem fios através de um ponto de acesso.

Ameaças à Confidencialidade: As principais ameaças estão relacionadas tanto com a **privacidade das comunicações** como com a **proteção dos dados transmitidos e armazenados**, nomeadamente:

• Interceção de tráfego não encriptado, especialmente em ligações sem fios (WLAN).
• Utilização de palavras-passe fracas ou repetidas que facilitam o acesso indevido.
• Acesso não autorizado a servidores devido a configurações incorretas ou permissões mal definidas.
• Envio de mensagens por correio eletrónico sem encriptação, expondo dados e metadados sensíveis.
• Utilização de protocolos inseguros (HTTP, Telnet, FTP) em sessões administrativas.
• Monitorização indevida de tráfego interno ou rastreamento de padrões de utilização.

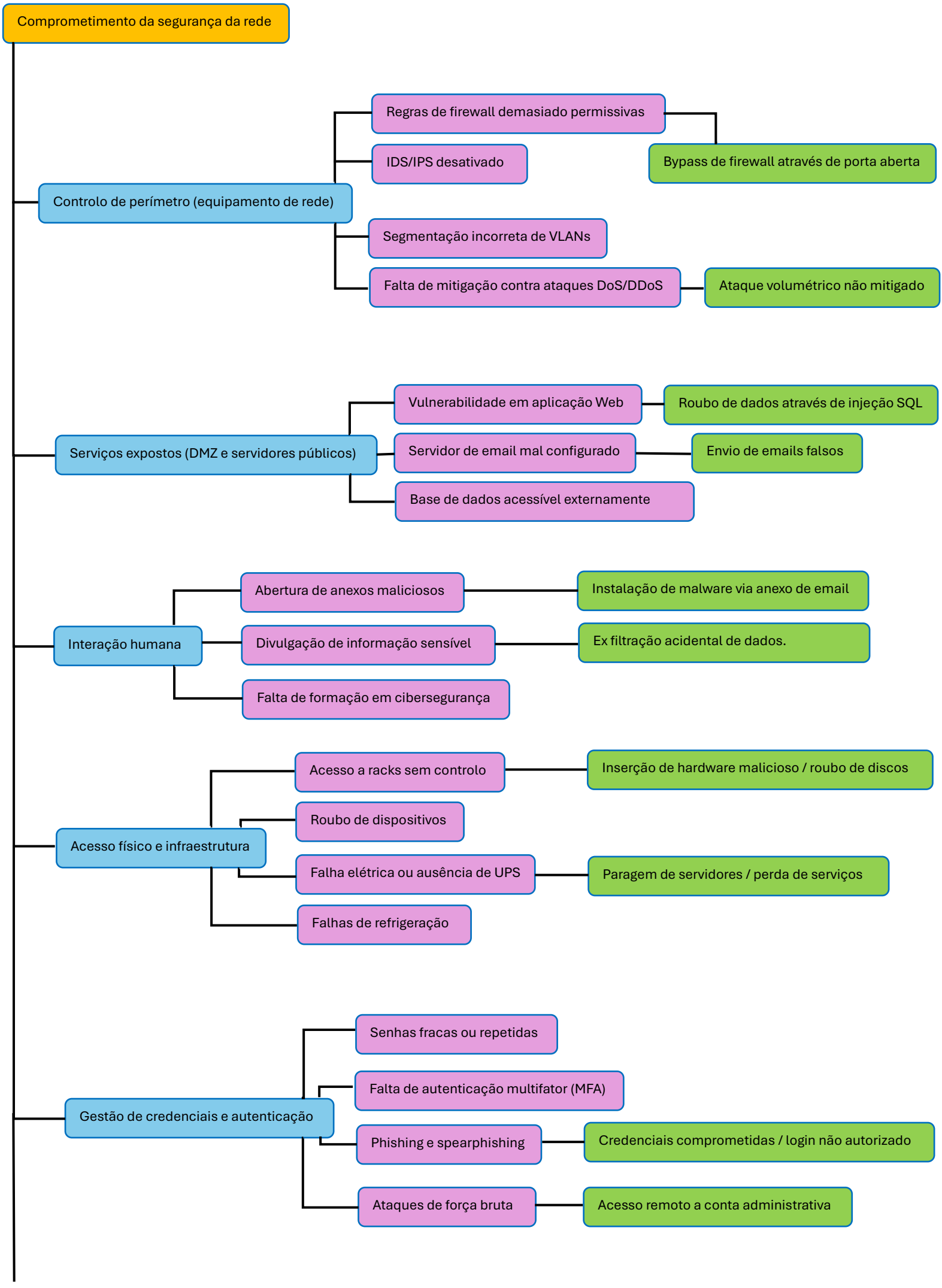
Ameaças à Integridade: A integridade pode ser afetada tanto a nível de **dados**, com alterações indevidas de informação, como a nível de **sistema**, com modificações de configurações ou políticas de segurança, nomeadamente:

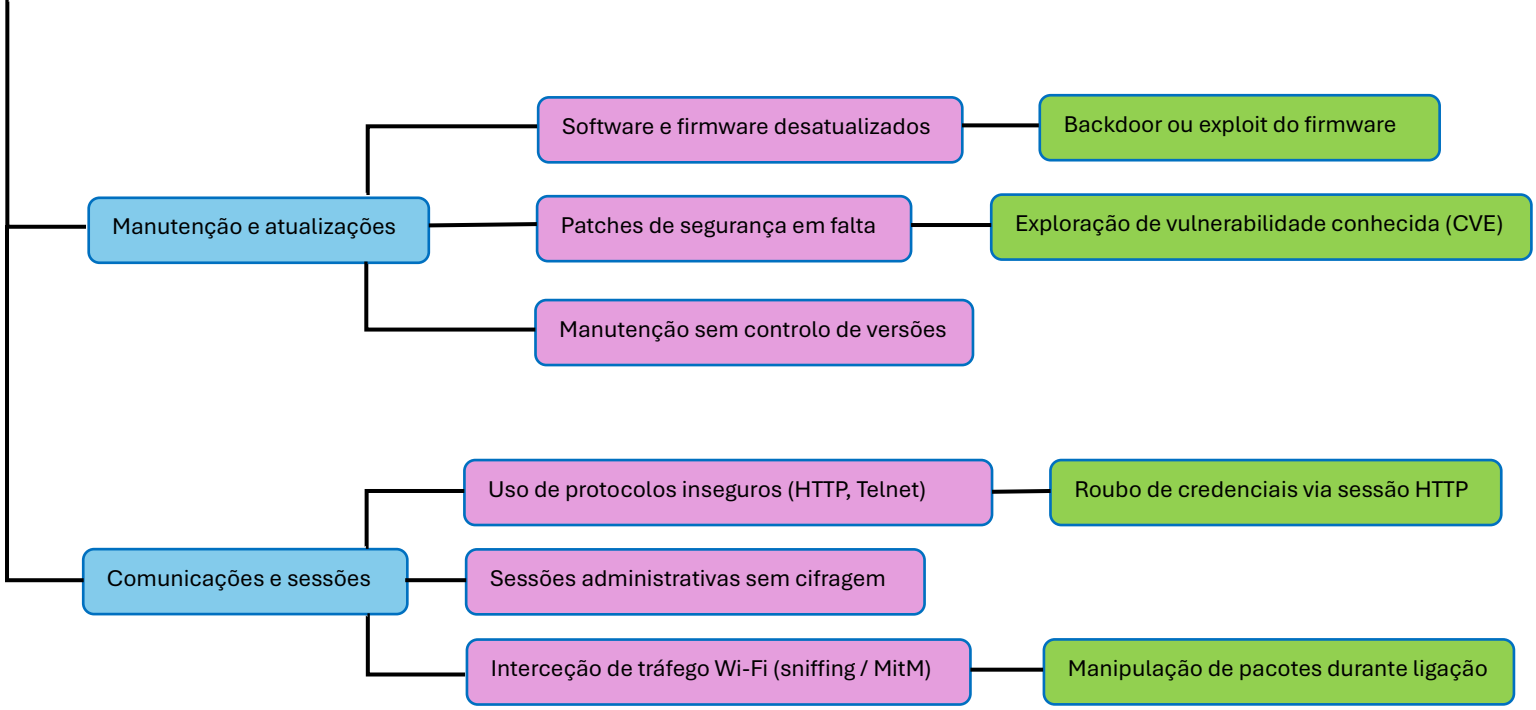
• Alteração de registos na base de dados por utilizadores com privilégios excessivos.
• Modificação de conteúdos no servidor Web devido a falhas de autenticação.
• Alteração não autorizada de regras na firewall ou configurações do router.
• Instalação de software malicioso ou de atualizações não validadas.
• Corrupção de ficheiros e logs por falta de controlo de versões.
• Manipulação de políticas de segurança ou permissões administrativas.

Ameaças à Disponibilidade: Estas ameaças comprometem o funcionamento contínuo dos serviços e o acesso à informação, como por exemplo:

• Ataques de negação de serviço (DoS ou DDoS) contra servidores na DMZ.
• Interferência intencional na rede Wi-Fi (jamming).
• Falhas de energia devido à ausência de fontes de alimentação redundantes (UPS).
• Falta de planos de contingência e de cópias de segurança (backups).
• Atualizações mal planeadas que originam interrupções temporárias dos serviços.

Diagrama de árvore de ataque:





As áreas críticas foram definidas com base nas principais camadas funcionais da infraestrutura e nos pontos de maior exposição a ameaças.

- O **controlo de perímetro** representa a primeira linha de defesa da rede;
  A **DMZ e os servidores públicos** concentram os serviços acessíveis a partir do exterior;
- A **gestão de credenciais e autenticação** abrange o acesso lógico aos sistemas;
- O **acesso físico e a infraestrutura** garantem a proteção dos equipamentos e da energia;
- A **interação humana** reflete o fator comportamental e de engenharia social;
- A **manutenção e comunicações** abrangem as operações contínuas e as ligações entre sistemas.

Em cada domínio são detalhadas as ameaças específicas que podem afetar a confidencialidade, integridade e disponibilidade da informação. As caixas a verde, representam as consequências diretas das vulnerabilidades identificadas, evidenciando o impacto potencial de cada ataque. A tabela seguinte resume esta classificação e apresenta os principais tipos de ataque que podem explorar cada falha.

Ameaça (Nível 3)	(CIA)	Tipos de ataque que exploram a falha
Regras de firewall demasiado permissivas	C, A	Port scanning, acesso remoto não autorizado, bypass de firewall
IDS/IPS desativado ou mal configurado	C, I	Exploração de vulnerabilidades
Segmentação incorreta de VLANs	C, I	Pivoting, acesso indevido a segmentos internos
Falta de mitigação contra ataques DoS/DDoS	A	Ataques volumétricos, DoS/DDoS
Vulnerabilidade em aplicação web	C, I	Injeção SQL, execução de código remoto
Servidor de email mal configurado	C, I	Phishing, spoofing, envio massivo de spam
Base de dados acessível externamente	C, I	Roubo de registos, extração de credenciais
Abertura de anexos maliciosos	C, I	Malware, ransomware, spyware
Divulgação de informação sensível	C, I	Data leakage, exposição de dados em partilhas não seguras
Falta de formação em cibersegurança	C, I, A	Phishing, erro humano, execução acidental de malware
Acesso a racks sem controlo	C, A	Inserção de hardware malicioso, roubo de discos ou componentes
Roubo de dispositivos	C, A	Acesso físico a credenciais
Falha elétrica ou ausência UPS	A	Interrupção de serviços, perda de disponibilidade
Falhas de refrigeração	A	Sobreaquecimento de servidores, paragem de sistemas críticos
Senhas fracas ou repetidas	C, I	Força bruta, credential stuffing
Falta de autenticação multifator (MFA)	C, I	Comprometimento de conta, escalada de privilégios
Campanhas de phishing / spearphishing	C	Roubo de credenciais, engenharia social
Ataques de força bruta	C, I	Password spraying, ataques a interfaces de autenticação, etc
Software ou firmware desatualizado	C,I,A	Exploração de CVEs conhecidas, execução remota
Patches de segurança em falta	C, I, A	Exploração de vulnerabilidades conhecidas
Manutenção sem controlo de versões	C, I	Instalação de software corrompido, erros de configuração
Uso de protocolos não cifrados (HTTP, Telnet)	C, I	Sniffing, Man-in-the-Middle, roubo de credenciais
Sessões administrativas sem cifragem	C	Roubo de credenciais durante sessão administrativa, *packet sniffing*
Interceção de tráfego Wi-Fi (sniffing / MitM)	C, I	Interceção de tráfego, MitM

2. Estratégia de segurança e proposta de controlo:

Estratégia geral: A segurança da rede deve basear-se em várias camadas de proteção (defesa em profundidade), de forma que, se uma falhar, outra consiga proteger. Também deve seguir o princípio do privilégio mínimo, ou seja, cada pessoa e sistema só deve ter acesso ao que realmente precisa. Além disso, é importante separar privilégios (por exemplo, funções de administrador e de utilizador normal) e evitar depender apenas de "segurança por obscuridade", como esconder portas ou nomes de ficheiros.

Controlos propostos:

- **Palavras-passe fortes e trocadas com regularidade:** Reduz o risco de adivinhação ou reutilização de credenciais. Custo e impacto reduzidos.
- **Antivírus sempre atualizado:** Evita a instalação de programas maliciosos e exploração de falhas. Custo baixo e impacto mínimo no desempenho.
- **Instalar UPS em servidores e equipamentos importantes:** Evita interrupções e perda de dados durante falhas elétricas. Custo médio, viável e com impacto nulo na utilização normal.
- **Usar sempre ligações seguras (HTTPS, VPN, etc.) e eliminar protocolos inseguros como Telnet ou FTP:** Protege os dados e credenciais contra interceção (Man-in-the-Middle). Baixo custo e sem impacto relevante no desempenho da rede.
- **Atualizações regulares de software e hardware:** Reduz o risco de exploração de vulnerabilidades conhecidas. Custo reduzido e viabilidade elevada.
- **Restringir o acesso a salas de servidores e equipamentos críticos a pessoal autorizado:** Evita que alguém roube, danifique ou manipule fisicamente os dispositivos. Custo médio (controlo de acessos e vigilância), impacto baixo.
- **Formação de funcionários e utilizadores:** Sensibilizar para phishing, engenharia social e boas práticas digitais. Diminui o risco de ataques que dependem de enganar o utilizador. Custo baixo e viabilidade alta (formações internas).
- **Rever periodicamente as regras da firewall e limitar acessos externos:** Ajuda a evitar ataques DoS/DDoS e acessos não autorizados. Custo baixo e impacto mínimo.
- **Criar procedimentos para lidar com falhas, ataques ou desastres, incluindo backups e comunicação interna:** Reduz o impacto e tempo de paragem após um incidente. Custo moderado e elevado benefício para a continuidade do serviço.

3. Implementação da solução criptográfica:

Encriptação simétrica (AES - GCM):

A encriptação simétrica **usa uma única chave secreta para cifrar e decifrar**. Esta é muito rápida e indicada para proteger dados em trânsito (mensagens, ficheiros ou pacotes).

O modo **AES-GCM** foi escolhido por ser mais seguro e eficiente do que modos mais antigos como o CBC ou o ECB, uma vez que não requer **padding** (conjunto de bytes adicionados para completar blocos) e porque permite detetar automaticamente alterações nos dados. Deve ser usado para enviar dados entre dois pontos (ex.: cliente - servidor), quando há partilha prévia de uma chave secreta, ou quando a chave é trocada em segurança por RSA ou TLS.

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

# Funcao simples para cifrar dados com AES no modo GCM
# Retorna o nonce, o texto cifrado e a tag de autentificacao
def cifrar (chave, dados, dados_autenticados=b""):


    # Gera um numero aleatório único (none) para esta decifragem
    nonce = get_random_bytes(12)

    # Cria o motor de cifragem com AES-GCM
    motor = AES.new(chave, AES.MODE_GCM, nonce=nonce)

    # Se existirem dados adicionais (ex.: cabecalhos), sao autenticados mas nao cifrados
    if dados_autenticados:
        motor.update(dados_autenticados)

        # Cifra os dados e gera a tag de autenticacao
        texto_cifrado, tag = motor.encrypt_and_digest(dados)

        # Devolve tudo que é preciso para decifrar depois
        return nonce, texto_cifrado, tag



# Funcao que decifra os dados cifrados

# Verifica também a integridade através da tag

def decifrar (chave, nonce, texto_cifrado, tag, dados_autenticados=b""):


    # Cria o motor de desencriptacao com a mesma chave e o mesmo nonce
    motor = AES.new(chave, AES.MODE_GCM, nonce=nonce)

    # Autentica novamente os dados extras (se existirem)
    if dados_autenticados:
        motor.update(dados_autenticados)

    # Tenta decifrar e verificar a tag (se falhar, da erro)
    return motor.decrypt_and_verify(texto_cifrado, tag)

# ============================================================
#  Teste rápido
# ============================================================
if __name__ == "__main__":
```

```
# Cria uma chave aleatoria de 32 bytes (256 bits)
chave = get_random_bytes(32)

# Mensagem a cifrar (tem de estar em bytes)
mensagem = b"Mensagem confidencial em transito"

# Cifra a mensagem
nonce, cifra, tag = cifrar (chave, mensagem)

# Desencripta a mensagem e verifica a integridade
original = decifrar (chave, nonce, cifra, tag)

# Mostra o resultado original apos a desencriptacao
print("Mensagem original decifrada:", original.decode("utf-8"))
```

No código, é gerada uma chave aleatória de 256 bits (32 bytes), utilizada pelo algoritmo AES no modo **GCM (Galois/Counter Mode**). Esta chave deve ser mantida em sigilo absoluto, pois toda a segurança da encriptação depende dela.

É também criado um vetor de inicialização (nonce) aleatório de 12 bytes, que garante que o mesmo texto, cifrado com a mesma chave, produz resultados diferentes. **O nonce deve ser único para cada mensagem**, pois se usasse o mesmo nonce duas vezes com a mesma chave, o mesmo texto produziria o mesmo resultado, e isso seria uma falha grave na segurança, porque permitiria detetar padrões ou até mesmo recuperar informações.

Durante a execução, **o método encrypt_and_digest()** do AES-GCM cifra o texto e gera uma tag de autenticação (esta serve para confirmar que os dados não foram alterados depois de serem cifrados). A cifra (ciphertext) protege a confidencialidade, impedindo que terceiros leiam a mensagem, enquanto a tag assegura a integridade e a autenticidade. Se alguém alterar um só byte da mensagem cifrada, a verificação falha e a decifra é rejeitada.

Na desencriptação, o código utiliza a mesma chave e o mesmo nonce para recuperar o texto original, validando a *tag*.

Vantagens (AES-GCM):

- Muito rápido (ideal para dados em trânsito).
- Garante **confidencialidade e integridade** (cria a tag de autenticação).
- Implementação simples e eficiente.

Desvantagens:

- É necessário partilhar e armazenar a chave secreta com segurança.
- **Não repetir o mesmo nonce** com a mesma chave

Encriptação assimétrica (RSA-OAEP):

A encriptação assimétrica utiliza um par de chaves:

- **Uma chave pública**, usada para cifrar os dados;
- E **uma chave privada**, usada para decifrar.

Este modelo elimina a necessidade de partilhar previamente uma chave secreta, uma vez que qualquer pessoa pode cifrar dados com a chave pública, mas apenas o detentor da chave privada consegue recuperá-los.

O RSA-OAEP (Optimal Asymmetric Encryption Padding), combinado com o SHA-256, é uma versão moderna e segura do RSA tradicional. O OAEP introduz um sistema de preenchimento aleatório (padding) que protege contra ataques de texto escolhido, tornando a cifragem mais robusta. É utilizado para cifrar pequenas quantidade de dados.

```python
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Hash import SHA256
from Crypto.Random import get_random_bytes

# ============================================================
#  Geração das chaves RSA
# ============================================================
# Cria um par de chaves (privada e pública)
# O tamanho da chave define o nível de segurança
# 2048 bits é o mínimo recomendado e 3072 bits é o mais seguro
chave_privada = RSA.generate(3072)
chave_publica = chave_privada.publickey()

# Cria dados com RSA-OAEP (usa SHA-256 como função hash) e retorna os dados cifrados
def cifrar (chave_pub, dados):

    # Cria o motor de cifragem com RSA-OAEP e SHA-256
    motor = PKCS1_OAEP.new(chave_pub, hashAlgo=SHA256)

    # Cifra os dados (normalmente pequenos, como uma chave AES)
    dados_cifrados = motor.encrypt(dados)

    return dados_cifrados



# Funcao que decifra os dados cifrados com RSA-OAEP e devolve o original
def decifrar (chave_priv, dados_cifrados):

    # Cria o motor de desencriptação com a mesma configuracao
    motor = PKCS1_OAEP.new(chave_priv, hashAlgo=SHA256)

    # Tenta decifrar os dados, e se algo estiver errado, a operação falha
    dados_originais = motor.decrypt(dados_cifrados)

    return dados_originais


# ============================================================
#  Teste rápido
# ============================================================
if __name__ == "__main__":
    # Mensagem curta (RSA serve apenas para blocos pequenos)
    mensagem = b"Mensagem confidencial"
```

```
# Cifra a mensagem com a chave pública
mensagem_cifrada = cifrar (chave_publica, mensagem)

# Decifra com a chave privada
mensagem_decifrada = decifrar (chave_privada, mensagem_cifrada)

# Mostra o resultado
print("Mensagem original decifrada:", mensagem_decifrada.decode("utf-8"))
```

No código, é gerado um par de chaves RSA: uma chave pública, usada para cifrar dados, e uma chave privada, usada para decifrar. A **função RSA.generate(3072)** cria ambas as chaves com 3072 bits, um tamanho que garante um nível elevado de segurança e proteção moderna. A chave pública pode ser partilhada desafogadamente, enquanto a chave privada deve ser mantida em sigilo absoluto, pois quem a possuir consegue decifrar as mensagens.

A **função cifrar()** recebe a chave pública e os dados a cifrar. O motor de encriptação é criado com **PKCS1_OAEP.new()**, configurado para usar o hash SHA-256. O modo OAEP acrescenta um preenchimento aleatório aos dados antes da cifragem, o que impede ataques criptográficos e reforça a segurança do algoritmo RSA. Após esta preparação, a mensagem é cifrada. Esta origina um bloco binário que apenas pode ser decifrado pela chave privada correspondente. Na **função decifrar()**, o processo é invertido. Cria-se novamente um motor de desencriptação com **PKCS1_OAEP.new()**, e desta vez usando a chave privada. O **método decrypt()** remove o padding OAEP e recupera o texto original.

Vantagens (RSA-OAEP):

- Não requer canal seguro para partilhar segredos (usa a chave pública).
- Alta segurança contra ataques de texto escolhido, graças ao padding OAEP.
- Escalável em sistemas com numerosos utilizadores (cada um tem o seu par de chaves).

Desvantagens:

- Desempenho inferior: é mais lento e ineficiente para grandes volumes de dados.
- Limite de tamanho: apenas pode cifrar mensagens até um certo número de bytes (inferior ao tamanho da chave).
- Gestão de chaves: as chaves privadas devem ser protegidas com extremo cuidado.

4. Reflexão sobre a estratégia de segurança:

No geral, a estratégia de segurança proposta procura cobrir o máximo de ângulos possíveis - técnicos, físicos e humanos. Foram incluídas medidas como a utilização de

palavras-passe fortes, antivírus atualizado, encriptação das comunicações, firewall bem configurada, UPS para evitar falhas de energia e formação dos colaboradores de forma a reduzir riscos. A ideia foi criar um sistema que não dependa de uma única defesa, mas de várias camadas que se apoiam umas às outras. Mesmo assim, sei que não existe segurança total. Nenhuma rede é 100% à prova de falhas, porque basta uma pequena distração humana ou uma atualização em atraso para abrir uma brecha.

Ainda assim, considero que esta abordagem é equilibrada, porque combina prevenção, deteção e resposta. A formação do pessoal e o uso da UPS, por exemplo, dão mais resiliência à empresa e permitem recuperar mais rapidamente caso ocorra algum incidente. No futuro, faria sentido apostar em encriptação assimétrica completa, em monitorização ativa de logs, testes de penetração regulares, políticas automáticas de rotação de chaves e auditorias de segurança. Também seria importante reforçar os backups automáticos e implementar alertas de intrusão em tempo real. Mas no fim, o maior desafio continua a ser o mesmo: as pessoas. O fator humano é, quase sempre, o elo mais fraco - e também o mais difícil de corrigir.

Referências:

encryption - Difference Between RSA-OAEP and RSA-PKCS1.5 - Stack Overflow

AES — PyCryptodome 3.23.0 documentation

RSA — PyCryptodome 3.23.0 documentation

PKCS#1 OAEP (RSA) — PyCryptodome 3.23.0 documentation

Modern modes of operation for symmetric block ciphers — PyCryptodome 3.240b0 documentation

Selecting the Best AES Block Cipher Mode (AES-GCM VS AES-CBC) | by Isuru Kariyawasam | Medium

What Is RSA OAEP? - Next LVL Programming

The RSA Encryption Algorithm (1 of 2: Computing an Example)

(30) How does RSA Cryptography work? - YouTube

AES GCM (Advanced Encryption Standard in Galois Counter Mode) - Computerphile

Attack Tree Threat Modelling

A Comprehensive Guide to Free Attack Tree Software Tools — RiskyTrees

Stallings, W., Brown, L. (2024). *Computer Security: Principles and Practice*. 5th Edition (Global Editions), Pearson.

O chatGPT foi usado como ferramenta de apoio/esclarecimento de dúvidas na elaboração deste trabalho.