

USER page } front-end  
- edit page

afonsu - backend  
ze - front-end

## Criptografia → não é a solução de todos os problemas, apenas parte

Outrora uma "arte" onde o segredo e o obscurantismo eram princípios fundamentais. Atualmente, uma ciência com princípios rigorosos e normas internacionais. É uma componente central em muitos mecanismos de segurança.

Segurança da informação → criptografia protege a disponibilidade, integridade e confidencialidade da informação em 3 cenários

Proteção da segurança da informação:

em trânsito A => B, online / síncrono

em trânsito A => B, offline / assíncrono (mensagens)

em repouso A => A, disk encryption

Garante<sup>1</sup> confidencialidade: informação acessível apenas a emissor A e receptor B. Cifras simétricas e assimétricas, acordos de chave.

tecnologia usada para garantir confidencialidade

Garante<sup>2</sup> autenticidade (e integridade): receptor B tem a certeza de que os dados vieram sem alteração de A (autenticidade garante integridade)

3. Não repúdio - assinaturas digitais

Criptografia moderna (confidencialidade)

1. Definições precisas e rigorosas de segurança: modelos matemáticos da realidade → estamos a usar problemas para a qual não temos solução como base da crie

2. Quando a segurança de uma construção se baseia num pressuposto que não conseguimos provar, esse pressuposto deve ser simples. (Brownfield upon)

3. as construções criptográficas justificadas formalmente, pelo princípio #1.

### Cifras simétricas

Algoritmos que usam a mesma chave para encriptação e descriptação

A chave secreta tem, tipicamente, 128 bits.

Pode ser

$C = \text{Enc}(K, n, m)$

$m = \text{Dec}(K, n, c)$

$K$  é único  $\rightarrow n$  é irrelevante  
no sequência ou sobre a  
aleatório many-time key nonce não é relevante  
 $c$  já usado + 1 vez  $\rightarrow n$  é sempre único  
variantes de

### One time pad segurança

OTP garante confidencialidade contra "eavesdroppers",  $m = K$  porque a chave é do tamanho do texto limpo.

Cifras sequenciais  $\rightarrow \text{Enc}(K, m) = m \oplus \text{PRG}(K)$

Gerador pseudo-aleatório para a chave. Se não houver nonce  $\text{PRG}(K)$  é sempre igual. Em duas cifrações pode ser usada a mesma chave

Cifras de bloco  $\rightarrow$  usado para construir PRGs

$\text{PRG}(K) \rightarrow$  cria uma string do tamanho do texto limpo a partir de um  $K$  pequeno

Apesar do nome não são cifras, permitem construir cifras. Dado um bloco e uma chave, podem outros blocos

Um algoritmo determinístico opera sobre blocos com uma transformação invariável que é especificada por uma chave simétrica.

O algoritmo de encriptação tem 3 componentes:

$K$  = chave secreta (tem de ser igual dos dois lados e desconhecida a todos os outros)

$m$  = mensagem a transferir

$n$  = nonce (not more than once, non-repeating), pode ser público, mas nunca deve ser usado com a mesma chave.

$c$  = output do algoritmo, criptograma

$c = \text{Enc}(K, n, m)$

de um  $k$  original são derivadas  $k_i$  chaves,  
aplica-se sequencialmente encriptação  
com a chave  $k_{i+1}$  sob a cripta feita por  $k_i$

$$R(k_1, P) \rightarrow R(k_2, P_1) \rightarrow R(k_3, P_2) \dots$$

→ AES advanced encryption standard

Provavelmente o algoritmo criptográfico + utilizado.

Estado = matriz  $4 \times 4$  bytes

1 Round = 4 transformações. Cada round usa uma  
chave derivada da chave da cifra.

O melhor ataque teria que fazer, no mínimo,  $2^{16}$   
operações

Alguns métodos de cipher blocking não são garantidamente seguros. Por exemplo, Electronic Code book é ineficaz, e inferior a Cipher block chaining mode, mas melhor que ambos é o counter mode.

• dentro de um initialization vector

### Poder do adversário

Em que condições conseguimos garantir confidencialidade? Adversário consegue ler, alterar ou inserir mensagens?

As cífras apresentadas não protegem contra ataques ativos. Se o adversário alterar o criptograma o receptor pode não perceber.

É sempre preciso acudir a ataques ativos:

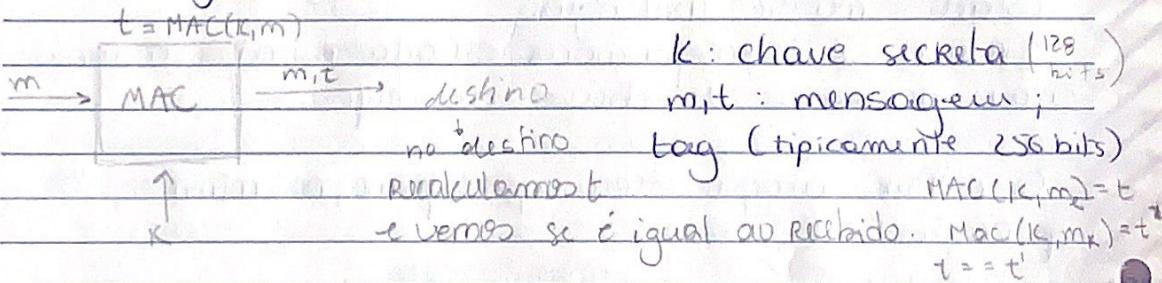
integridade mensagem não é alterada  
autenticidade A e B partilham uma chave  
segreta. B apenas aceita mensagens de A.

Autenticidade implica integridade.



## Message authentication codes

MAC: algoritmo  $\Rightarrow$  público e standard.



Usamos MACs para autenticidade e integridade.  
Tags não criptam m.

Tag = checksum criptográfico, apenas calculável com k

Se no destino a tag calculada e a recebida não coincidirem algo se passou

## Garantias MAC

MAC não ~~permite~~ permite detectar ataques em que as mensagens não são entregues ou são entregues múltiplas vezes.

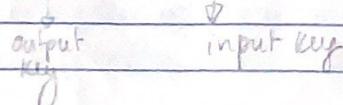
Para prevenir tal, garantimos que o valor de m é sempre único. Como? Pré-processar m, com o número da sequência (LOGICAL OR).

$$t = \text{MAC}(k, m \parallel n)$$

## Construção MAC

$$\rightarrow \text{HMAC} \quad t = H(\text{ckey} \parallel H(\text{icay} \parallel m))$$

MAC a partir de função de hash, hoje em dia SHA-256.



$x = \rightarrow$  DIFF  
 $= \leftarrow$  CCR  
 $\geq$  LITE

$$\rightarrow \text{Poly1305} \quad t = f(m, R) + s$$

função de hash algébrica. polinômio definido por  $m$ , avaliado no ponto  $R$ .

Chave secreta  $(R, s) \Rightarrow 256$  bits. Pode ser usada apenas uma vez.

### Confidencialidade e autenticidade

Como obter ambas? Precisamos de usar: cifra (confidencialidade) + MAC (autenticidade). Precisamos de 2 chaves secretas.

Como combinar cifra com MAC?

$C = \text{Enc}(k_1, m) \rightarrow$  encrypt and MAC      SSL       $t = \text{MAC}(m, k_2)$   
 $C = \text{enc}(k_1, m) \rightarrow$  encrypt then MAC      IPSEC       $t = \text{MAC}(c, k_2)$   
 $C = \text{Enc}(k_1, m || t) \rightarrow$  MAC then encrypt      SSL       $t = \text{MAC}(m, k_2)$

### AEAD

mais usado,  
só desencriptamos  
depois de ver se  
o tag é só certo.

Authenticated encryption with associated data.

Abstração correta para a implementação de um canal seguro com criptografia simétrica.

Garante confidencialidade e autenticidade.

$\text{Enc}(n, k, m, \text{data}) \Rightarrow (c, t)$

$\text{DEC}(n, k, c, t, \text{data}) \Rightarrow m$  (só se  $c, t$  autênticos)

### Aleatoriedade

Na análise técnica, assume-se aleatoriedade perfeita, ou seja, cada bit é independente de todos os outros. Na realidade, é difícil gerar aleatoriedade.

## Criptografia de chave pública

Revolução nos anos 70, anteriormente só haviam chaves simétricas e pré-partilhadas.

Surgiu, então, cífras de chave pública, assinaturas digitais, acordos de chave.

### Gestão de chaves

#### Com criptografia simétrica:

N agentes (conjunto bem definido), N chaves  
sistema fechado → Key distribution center → um centro ligado  
a todos os agentes

#### Chaves de sessão

→ é preciso pré-definir  
manualmente chave para cada agente  
e partilhar-se uma chave  
com o seu destino

Os sistemas modernos distinguem:

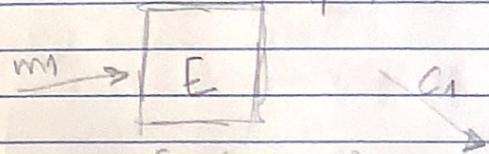
chaves de curta vs. longa duração.

efêmeras,  
dados limitados

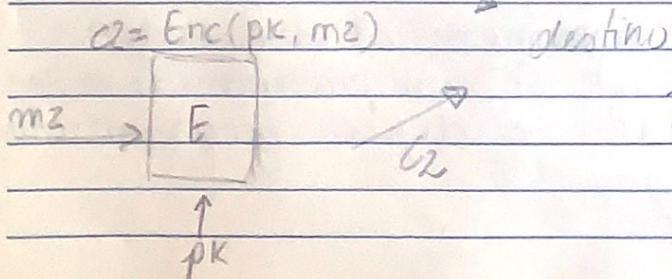
requisitos fortes de  
segurança no  
armazenamento

### Cífras de chave pública

Ferramenta para gerir chaves simétricas (apenas!)  
 $c_1 = \text{Enc}(pk, m_1)$



E, D: algoritmos (encrypt, decrypt) → D públicos e standard.



busca para  
pk chave pública  
sk chave secreta  
→ usada para  
descriptar só entendível  
e capaz de decifrar

Muito mais infelizes do que cifras simétricas.  
As chaves têm milhares de bits e é apenas viável  
cifrar mensagens muito pequenas.

O AEP encryption  $\rightarrow$  inútil

Construção conceptual:

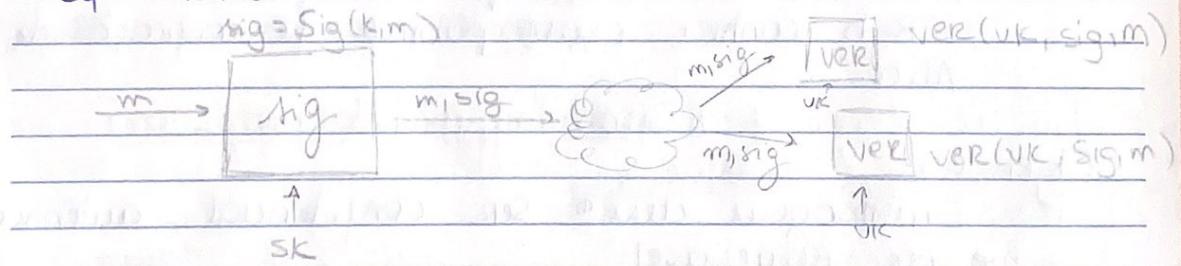
começar por um objeto matemático

"permutação unidirecional com alçapão"

codificar  $x$  de maneira inteligente.

Assinaturas digitais

Equivalente eletrônico às assinaturas manuscritas



$\text{sig}, \text{ver}$  : algoritmos ( $\text{sign}, \text{verify}$ )  $\Rightarrow$  públicos e standard.

$sk$  = chave de assinatura (secreta - conhecida a 1 pessoa)

$vk$  = chave de verificação (pública) usada para decifrar  
(pode ser decifrada por vários)

Assinaturas digitais garantem a não falsificação,  
reabilitação e refutação. Garantem a autoria  
de um documento. Supondo que nada foi compro-  
metido (chave de assinatura íntegra).

Como construir assinaturas digitais?

Assinaturas com base em RSA  
historicamente mais populares  
predominantes em aplicações de assinatura  
eletrônica

ECDSA

Assinaturas com base em curvas elípticas  
mais eficientes, chaves públicas mais ~~grandes~~  
pequenas.

### Cenário Email seguro

Pressupostos:

Alice conhece chave pública de Bob

Bob conhece chave pública de verificação de  
Alice

Objetivos:

mensagem deve ser confidencial, autêntica  
e não-repudiável.

Solução: sign-then-encrypt:

Alice assina a mensagem

Alice cifra a mensagem com chave pública  
de Bob

Cuidado com milha-dados quando se combina  
cifras com assinaturas

## Cenário acordo de chaves

### Pressupostos

alice conhece chave(s) publica(s) de Bob

bob " " " " de Alice

### Objetivos

chave a estabelecer deve ser confidencial

chave a estabelecer deve ser autêntica,  
confirmada

comprometer chaves de longa duração não  
comprime sessões passadas.

A solução para este problema baseia-se no  
protocolo diffie - hellman.

### Protocolo diffie - hellman

parametros públicos  $(G, g, \circ)$

conjunto G:

podem ser valores entre  $[0, p]$  primo e grande  
" " pontos numa curva elíptica.

operação  $\circ$ :

mapeia dois elementos do conjunto num  
terceiro.

gerador  $g$ :

permite codificar um inteiro gerado alea-  
toriamente, de forma irreversível

$$\text{Bob} \quad K = (g^y)^x = g^{xy} = (g^n)^x \quad \begin{matrix} n \in [0, q] \\ y \in [0, q] \end{matrix} \quad \text{Alice}$$

$$K = (y^n)$$

$$K = g^n \quad \begin{matrix} \xrightarrow{\quad} \\ \xleftarrow{\quad} \end{matrix} \quad X = g^y$$

$$K = X^y$$

## Man-in-the-middle attack

o man-in-the-middle altera  
x e y e é indetectável,  
a não ser que sejam  
usadas chaves longe  
duração para validade

Os ataques Man-in-the-middle são possíveis  
sempre que utilizamos parâmetros públicos na  
rede sem sabermos a sua origem.

## Estabelecimento de canais seguros

Obriga a:

\* Protocolo Diffie-Hellman  
Autenticado usa chaves securas

autenticação de chaves públicas  
protocolo de acordo de chaves  
tecnologia simétrica

## Public Key Infrastructure

### Porquê PKI

A criptografia de chave pública pressupõe  
chaves públicas autênticas. Caso contrário: ataques  
man-in-the-middle

Garantir a autenticidade confirmando manual-  
mente que a chave pública pertence à entidade  
correta ou utilizando sistemas de autenticação  
de chaves públicas como PGP / GPG

usar

PKI - Quando é necessária cobertura legal/regulamentar:

Normas técnicas que algoritmos utilizam  
Regulamentações como devem ser utilizadas as  
normas técnicas; responsabilidades/direitos dos participantes  
e suas garantias formais e penalizações em  
caso de violação das regras

## Certificados de chave pública

→ objetivo:

alice envia a Bob uma chave pública pk através de canal inseguro. Bob tem de obter garantia que Alice deu pk a chave secreta correspondente. → evitar Man-In-the-Middle attack

→ solução trivial

Bob tem canal autenticado com Trusted-Third-Party (TTP). Alice demonstrou previamente a TTP que é dona de pk (como?). Bob pergunta à TTP se pk pertence a Alice.

Problemas → problemas

- {  
1. como se constrói o canal entre o Bob e a TTP?  
2. e se a TTP estiver offline?  
3. como garantimos que Bob e Alice confiam na TTP?  
4. o que significa confiança?

Os certificados de chave pública usam assinaturas digitais para resolver os problemas 1 e 2.

TTP := autoridade de certificação (CA)

Alice prova a CA que possui pk, assinando um pedido de certificado que contém pk usando a chave secreta

CA fornece a Alice a pk

CA fixa/verifica todos os dados relevantes para certificado: identidade de Alice + chave pública de Alice; informação específico da CA; validade

CA assina um documento eletrónico com esta informação

Tecnicamente um certificado é um array de bytes, codificado de uma forma específica. Os bytes estão numa estrutura de dados ASN.

→ Abstract Syntax Notation

Linguagem de especificação herdada das normas para protocolos.

DER (distinguished encoding rules) são um conjunto de diretórios que especificam como foi feita a codificação.

Como é que os certificados resolvem os problemas 1 e 2?

Alice tem um certificado que veio de CA, que envia a Bob durante a comunicação e depois Bob verifica a assinatura da CA. Assim, certificados podem ser transmitidos por canais inseguros.

Mas como verifica Bob que a assinatura de Alice?

## Verificações de Certificados

(estudos de facto e comunicação com o mundo)

1. Verificar que corregão de identidade da Alice
2. Verificar que está dentro da validade
3. Verificar meta-info
4. Verificar que CA é de confiança (verificação dependente do sistema)
5. Obter chave pública da CA

Os certificados de chave pública podem conter extensões (attachments), algumas das quais podem ser assinaladas como críticas. Todas as extensões têm um identificador de objeto (OID).

Se uma extensão estiver sinalizada como crítica e não for reconhecida, todo o certificado é inválido e rejeitado.

## Public Key Infrastructure

Uma PKI é ~~é~~ o conjunto de tudo o que é necessário para garantir que o utilizador de um certificado recebe garantia de uma chave pública.

## Transações operacionais / gestão

Vários protocolos ~~são~~ podem ser usados para armazenar/transportar os certificados de chave pública. No entanto, os mesmos têm de ser codificados em formatos que garantem interoperabilidade.

## Funcionamento PKI

Como ~~sabe~~ o utilizador quem são as CA e como ~~são~~ são verificadas as assinaturas nos certificados?

Todas as chaves públicas numa PKI estão codificadas em certificados, quer as dos utilizadores comuns, quer as das autoridades de certificação.

O utilizador usa a chave pública que está num CA para validar a assinatura que está no certificado da Alice.

A chave pública é autêntica se o Bob conhece de antemão o certificado da CA que emitiu o certificado da Alice, ou se Bob confia na CA que autenticou o certificado da Alice.

## Cadeias de Certificação

Os certificados Raiz são auto-assinados, ou seja, quem gera o certificado coloca-se a si mesmo como ~~assessor~~ sujeito desse certificado.

A CA gera um par de chaves ( $sk, pk$ ) e inclui no certificado  $pk$ , assinando-o mesmo com  $sk$ .

Qualquer pessoa pode fazer um certificado auto-assinado, mas só confiamos naqueles que recebemos através de um canal seguro.

Podemos ter muitos níveis na árvore de certificados. Enfim Bob recebe o certificado de Alice, analisa a assinatura do CA, que por sua vez foi assinado por outra CA, esta última sendo da nossa confiança. (Tipicamente até chegar a um certificado de Root).

### Gestão PKI: Registo

Além das Autoridades de Certificação, temos também as autoridades de registo. Estas autoridades estão em contacto direto com utilizadores/titulares.

Verificam a autenticidade dos dados colocados no Registo

### Gestão PKI: Certificação

Autoridades de Certificação tipicamente não interagem com os utilizadores. Essencialmente, um computador isolado, muito bem protegido, que faz assinaturas digitais.

### Gestão PKI: Revogação

O que acontece se a chave for perdida ou o certificado deixar de ser válido? Como invalidar um certificado válido?

A solução clássica são Certificate Revocation Lists essencialmente uma black-list dos certificados revogados. Infelizmente, depende do utilizador e por isso não funciona.

Soluções Reais:

Trusted Service Provider Lists white-list de certificados usada em comunidades pequenas/fechadas.

Online Certificate Status Protocol pergunta a várias autoridades se o certificado está black-listed.

Certificate Pinning empregadas grandes empresas (web servers / browsers) geram suas próprias white-lists e transmitem as mesmas aos utilizadores.

## Autenticação

Anteriormente foi mencionada autenticação de origem de mensagens. O destinatário tinha garantia que a mensagem teve origem em um emissor específico, independentemente da idade da mensagem.

Em Autenticação de entidades queremos ter a certeza que neste momento específico estamos a falar com um destinatário específico. Tipicamente, é feita a autenticação para depois ser feita uma autorização.

## Solução Criptográfica

Protocolo Desafio-Resposta:

Bob cria um desafio aleatório envia o mesmo a Alice, Alice assina digitalmente a solução ao desafio e devolve a mesma, por fim, Bob verifica a resposta.

Para adaptar este protocolo a utilizadores humanos, é requisitado um tipo de verificação que permite provar a identidade do utilizador.

(por exemplo, password, telemovel, biometria)

algo que sabe / tem / intrinsecamente

→ Algo que se sabe

Um segredo que apenas um humano específico conhece (um código secreto, nome do primeiro animal de estimação,...).

Na prática, verifica apenas conhecimento do segredo, ~~que~~

Um ataque a passwords pode ocorrer em vários pontos do processo de autenticação (Alice, teclado Alice, computador Alice, canal entre Alice e servidor, servidor). É também possível a Alice revelar a sua password acidentalmente.

Phishing convencer a pessoa a fornecer a password, ~~usando~~ usando o servidor incorreto, mas com igual aparência.

Keyloggers registo de keystrokes.

### Armazenamento de passwords: server-side

A solução naïve é armazenar listas de usernames e passwords, que apesar de ser inseguro, acontece mais frequentemente do que esperado.

Importante: o servidor não precisa de saber a password, precisa apenas de a reconhecer. Podemos, portanto, guardar uma hash da password.

Ataques Dicionário coleção de passwords possíveis tentar as possibilidades todas até encontrar a correta ou construir uma hash-table para valores frequentes.

Difficultam-se ataques de dicionário usando salt! Armazenar salt e hash, e juntá-los o salt durante a encriptação das passwords.

O salt será conhecido se houver uma data breach.

Outra mitigação a estes ataques é o uso de um hash lento que não afeta a performance de autenticação de forma tangível mas afeta a construção de dicionários.

→ Algo que se possui

Aplicado às pessoas. É utilizado um dispositivo fixo que é apenas possuído pela pessoa que está a autenticar.

A token é na mesma comprometível, tal como a password, no entanto, é improvável que os tokens sejam comprometidos.

smartcards processador embutido em cartão de plástico (cartões multibanco, cartão SIM, cartão cidadão)

one time tokens uma pequena máquina que tem um processador mais simples que smartcard com um ~~processador~~ écrã pequeno. O mecanismo mostra um código, que tem um espelho no servidor, e é usado como 2 fator. Vulnerável a Man-in-the-middle attacks.

one time password substituto de one time tokens, menor garantia de independência de fatores. mandar código para telemóvel.

## → Biometria

Prova de identidade através característica intrínseca da pessoa: impressão digital, caligrafia (física ou comportamental). Não é facilmente forgável ou de possível transferência, mas apresenta problemas de privacidade.

Na autenticação presencial é mais fácil implementar verificação Biométrica. Na autenticação remota há várias hipóteses:

servidor recebe toda a informação do registo

servidor guarda características pré-processadas

servidor recebe apenas resultado

• servidora confia no computador da Alice para fornecer amostras. Alice confia no servidor

• servidor confia na máquina da Alice para receber amostras, extrair templates. Alice confia no servidor

• servidor confia no computador da Alice para processar autenticação. Alice não precisa confiar no servidor

O registo de um utilizador é muito mais complicado do que para uma password.

uma taxa de falsos positivos baixos está sempre associada a uma taxa de falsos negativos alta, o que prejudica usabilidade, pelo que deve manter-se um nível igual dos dois

Os ataques maliciosos à biometria são de 2 tipos:

intercepção perda de confidencialidade

usurpação criação de característica falsa que engana

O SENSOR  $\rightarrow$  Alice sequestra

## Autenticação e sessões web

Uma sessão é uma sequência de pedidos/respostas a um ou mais aplicativos. Pode ser longa ou curta.

Uma autenticação por sessão.

### HTTP Auth

Na pré-história era usado HTTP auth. Servidores HTTP mantinham ficheiros de (hash de) passwords em pastas.

Não se deve utilizar, pois o log-out implica fechar o browser.

### Tokens de sessão

No momento da autenticação cria-se um testemunho do lado do cliente e é enviado em todos os pedidos relacionados.

Há 3 métodos para fazer isto: cookies, informação embutida em links clicáveis e campos escondidos em formulários.

Atualmente é usada uma combinação destes mecanismos.

Tokens devem ser imprevisíveis e invalidados no momento de log-out.

### Ataques comuns:

Ruivos de token XSS, leaves dropping, falha logout token fixation atacante "converte" utilizador a fazer login com o mesmo token.

# Segurança de Redes

## A Rede

Rede simples básica onde cada máquina/host tem um endereço IP. Complexidade nos endpoints. Funciona como correio, entrega pacotes.

A comunicação na rede segue protocolos que dão a sintaxe, semântica e especificações de cada pacote.

## Protocolo IP

Comunicação sem ligação. Não há garantias de entrega, nem tentativas de recuperação.

Pacotes podem ser fragmentados

O cabeçalho IP apesar de ter um campo source address este nunca é verificado, podendo ser alterado sem o conhecimento do receptor. Não temos portanto garantia que o source address é válido. Não temos garantias de segurança nenhuma.

## Routing de pacotes

Globalmente o routing é baseado no Border Gateway Protocol. Protocolo baseado em propagação e encaminhamento a vários destinos. Baseado na boa vontade, não é protegido.

## Modelo CIA

Confidencialidade, Integridade e Disponibilidade.

Ataques potenciais em todos os níveis da rede:  
CORHAR cabos, manipular routers, .... A superfície  
de ataque é muito vasta.

## Adversários

→ Sniffing

Atacantes podem observar comunicações, inserir pacotes, observar e inserir pacotes, <sup>on path</sup> e controlar todas as comunicações. → man in the middle

Para ter proteção na Internet é necessário utilizar criptografia.

## Wiretapping

Processo de escutar um canal de comunicações onde entidades terceiras estão a trocar informação.

O atacante liga-se à rede comunicação e recolhe toda a informação trocada.

Sniffing / packet sniffing é uma forma de wiretapping em redes de comunicação, onde se re-colhem e armazenam pacotes trocados entre utilizadores legítimos de uma sub-rede.

## MAC flooding (melhor sniffing)

Um switch mantém uma tabela dos endereços MAC. Se enviamos um flood de pacotes com MAC addresses diferentes, o switch será obrigado a re-escrever as entradas na tabela. Quando os pacotes "legais" chegam ao switch ele vai fazer um broadcast das MAC addresses

desconhecidos e por isso o atacante infiltra essas comunicações.

Se apanharmos um MAC Address podemos depois usar o mesmo.

### ARP poisoning

Quando é enviado um pacote ARP a perguntar o IP de um certo MAC address, é possível o

atacante responder com o seu IP e assim os pacotes serão redirecionados para ele, em vez de para a máquina com o MAC original.

Isto implica a possibilidade de man-in-the-middle attacks.

### Rogue DHCP

O protocolo DHCP funciona da seguinte forma, cliente faz um anúncio de que precisa um IP, o servidor responde com uma proposta de endereço IP. O cliente recebe do servidor DHCP o endereço IP.

Um Rogue DHCP SERVER pode convencer um cliente de que o Route/gateway está correto, permitindo um Man-in-the-middle attack.

DNS spoofing → alterar a tabela DNS

DNS cache poisoning → bombardear o servidor DNS tentando com respostas de DNS falsas.

### TCP session Hijacking

Usurpação de uma ligação legítima (já estabelecida) por parte do atacante diferente de spoofing, porque não se faz passar por um user para ter acesso, primeiro escuta depois ataca.

As soluções criptográficas permitem resolver o problema em parte, garantir segurança entre dois end-points. No entanto, não evitam ataques à infra-estrutura, malwares, Dos.

### Problema

Como protegemos as nossas máquinas ligadas à rede?

Definimos uma fronteira clara entre organização (que está sob o nosso controlo) e a internet. Limitamos a quantidade de tráfego na rede interna, focando sempre em proteger a rede interna de ameaças externas. firewalls + NAT + proxies + Network intrusion detection systems

### Firewalls

Podem ser locais ou na rede. Firewalls locais correm na máquina dos utilizadores e firewalls na rede interceptam comunicações de/para muitas máquinas

Tipicamente, filtram os pacotes, olhando para os endereços IP, protocolo TCP,... e tentam eliminar comunicações maliciosas, protegendo máquinas internas de acesso externo.

Um proxy é uma barreira também mas é tipicamente direcionado para uma aplicação específica (web p.e.)

## Políticas de controlo de acessos

Uma firewall implementa uma política de controlo de acessos. Distingue tráfego recebido de enviado. Permitir todos os acessos de dentro para fora, e serviços específicos do exterior têm acesso à rede interna.

Default allow permitir todos os acessos exceto problemas específicos

Default deny permitir apenas alguns acessos comuns a todos os sistemas  
→ provoca sempre instabilidade inicial, adicionando-se exceções

## Filtragem de pacotes

Cada pacote é verificado contra as regras e são tomadas as ações forward ou drop.

Filtragem pode ser com ou sem estado. A filtragem sem estado analisa cada pacote individualmente não procura saber o contexto em que o mesmo foi enviado. → sistema mais eficiente mas mais permissivo.

Filtragem com estado, mantém o registo das ligações ativas para verificar se pacote "faz sentido" neste contexto.

## Vantagens/Desvantagens NAT

Reduz exposição ao exterior, uso de 1 único endereço IP.

Pode perturbar o funcionamento de alguns protocolos. Fácil de fazer bypass para adversário através

## Proxies de aplicação

Obriga tráfego a passar pelo proxy. um Man-in-the-Middle de bem.

SMTP → fazer scan de virus    SSH → log de autenticação  
HTTP/HTTPS → bloquear URLs proibidas

## Intrusion Detection Systems

A ideia é monitorizar o tráfego e tentar identificar um ataque. Pode ser host-based ou na rede.

IDS / IPS a nível da rede ~~host~~ mantém uma tabela de ligações ativas e acompanham codos ma. Não é necessário, portanto, alterar máquinas individuais. Contudo, são muito exigentes em termos de processamento e menos precisos (lidar com pacotes cifrados).

IDS / IPS a nível do host implica fazer uma detecção mais agressiva em algumas máquinas. Não se generaliza por causa do custo associado.

## Análise de logs

Ferramentas que analisam e interpretam logs dos servidores / máquinas. É apenas reativo, só detecta após o ataque, não o previne. Atacante pode também alterar os logs.

## Pen-testing - ethical hacking

Em vez de esperar por um ataque, simular situações de ataque e ver como o sistema reage. é caro e podemos aumentar com o sistema.

## Honey-pots

Usar um chamariz, acelerando a detecção expondo um sistema atraente para atacantes. Qualquer tentativa de acesso é ataque ou erro. Melhor aplicável a ataques automatizados

## Malware

### Exemplos tomada de controlo

Atacar um serviço vulnerável exposto na rede  
Ransomware - servidor web malicioso injeta malware nos clientes

Software enganador - convence o utilizador a instalar  
Deturpar equipamento no fabrico / transito  
atacar fornecedores de software.

## Botnets

Uma botnet é uma rede de computadores com um sistema de comando e controlo. ~~maneiras~~ Cada computador é um bot.

Tipicamente usados para realizar uma tarefa a pedido de compradores

Têm 2 tipos de arquitetura

centralizada com muitos servidores centrais para robustez

peer-to-peer auto organizada, hierárquica

Dois tipos de fluxo:

push servidor envia comandos

pull bot pergunta se há comandos a realizar

Estes sistemas foram criados para ajudar à Recuperação / Resiliência do sistema no caso de um ataque

### Detectar e combate de botnets

Detectar malware na máquina comprometida, trânsito de rede de comunicação com C2, e expor máquinas (honeypots) para securi comprometidas e monitorizá-las.

Para combater, limpar máquinas comprometidas, isolare o C2 ou atacar a botnet e desativar.

### Combater malware

Existem 2 grandes formas de combater/detectar malware

Intrusion Detection System detecta ocorre depois do ataque ter sido concretizado

Intrusion prevention system intervenção rápida para evitar ataque

Muitas ferramentas misturam os dois

HIDS / NIDS → IDS aplicado a rede

## ERROS de deteção

Falso positivo: alerta para um problema que não existe

Falso negativo: ausência de alerta para um problema existente

$$Pr(\text{falso positivo}) = Pr[D | \text{not}(I)]$$

$$Pr(\text{falso negativo}) = Pr[\text{not}(D) | I]$$

Quando a frequência relativa de ataques é muito baixa é difícil ter um sistema eficaz e eficiente.  
→ 100 M pedidos com 5 maliciosos → taxa relativa baixa

## Como se detecta malware?

Signature-based reconhecer assinaturas padrões em ataques conhecidos (black listing)

Anomalias um sistema analisa a utilização de recursos e ~~detectar~~ detectar um acesso à rede anómalo ou um acesso a ficheiros sensíveis.

Integridade / especificação especificar o que é correto manter registo que permite detectar alterações a ficheiros críticos. Exige muito esforço e constante atualização.

## Antivírus

Os vírus não podem ser invisíveis, o código está armazenado algures. Mas onde e como procurar? Com que frequência? E os vírus que mudam de forma?

Os vírus para evitar detecção, criptografam com uma nova chave sempre que se propaga para outro ficheiro. Portanto, nunca vemos encontrar 2 fingerprints iguais.

que descripto o vírus

O código continua a ser igual, mas recentemente malwares ~~não~~ têm a capacidade de mudar o código de decrifração. Chamava-se mutation engine código que quando se propaga é mutado.

Alguns antivírus modernos executam programas numa sandbox e analisar então a utilização de recursos e comportamentos suspeitos.

### Detecção na rede

Monitorização de toda a rede e não só da fronteira com o exterior.

Analisar os protocolos em execução e assinaturas.

### NIDS vs HIDS

NIDS um sistema único permite proteger N sistemas, mais simples de gerir e instalar, mais difícil acesso para os atacantes.

HIDS tem acesso à semântica das atividades maliciosas protege de ameaças que não veio da rede (USB), não afeta o sistema todo, é possível configurar para cada máquina.

# Transport Layer Security

## Handshake TLS 1.3

As sessões TLS começam por usar criptografia de chave pública.

Chaves de longa duração são utilizadas para assinaturas e não para transportar chaves de sessão. Comprometer uma chave de longa duração não compromete acordos de chave passados (Diffie-Hellman autenticado).

Utilização do TLS implica PKI. O servidor autentica a troca Diffie-Hellman com uma assinatura digital.

Servidor ~~envia~~ envia o seu certificado de chave pública. Cliente verifica validade do certificado (Root CAs pré-instaladas). Cliente usa chave pública no certificado para verificar assinatura Diffie-Hellman.

## Integração TLS /HTTP

As mensagens HTTP são transmitidas usualmente como payloads TLS depois do canal ser estabelecido.

Web proxy: proxy precisa de saber o cabeçalho HTTP para estabelecer ligação.

Virtual hosts: mesmo IP com múltiplos DNS → como sabe o servidor que certificado devolver? Não existe ainda solução para este problema que proteja a privacidade do nome de domínio.

SSL strip → http://paypal.com => https://www.paypal.com  
pode haver alguém entre o servidor http e https, browsers modernos apresentam avisos.

As autoridades de certificação podem também ser atacadas, sendo o efeito de tal desastre

### Privacidade

O TS revela ainda muita meta informação. A análise de tráfego permite muitas vezes reconhecer com quem o servidor está a interagir e as operações em curso.