Segurança e Confiabilidade

Resumo

Conteúdo

Capítulo 1

Segurança

1.1 Conceitos

1.1.1 Propriedades de Segurança

As propriedades fundamentais da segurança são:

- Confidencialidade
- Integridade
- Disponibilidade
- Autenticidade
- Prestação de contas (accountability)

1.1.2 Falhas de Segurança

Uma falha de segurança num sistema procede do seguinte modo:

$$Ataque + Vulnerabilidade \rightarrow Intusão$$

- Um ataque é uma falta intencional maliciosa introduzida no sistema com a intenção de explorar vulnerabilidades
- Uma vulnerabilidade é uma fraqueza do sistema que o torna sensível a ataques, normalmente não maliciosa, sendo inofensiva sem ataques
- Uma intrusão é uma falta operacional induzida por meio externo e intencionalmente maliciosa que provoca um estado erróneo no sistema, podendo (ou não) causar uma falha de segurança

1.1.3 Ataques

Um ataque pode ser classificado relativamente ao seu propósito:

- Ataque passivo: tenta-se obter informação existente no sistema sem afetar os seus recursos. Não requer uma ação explicita contra os mecanismos de proteção ou a integridade dos dados, focando-se na confidencialidade. Alguns exemplos são:

  - Sniffing
  - Traffic analysis
  - Snooping
  - Probing

- Ataque ativo: tenta-se alterar o funcionamento correto do sistema. Consistem em tentativas agressivas de entrar no sistema, para corromper a sua operação e/ou roubar, modificar ou mesmo destruir dados. Alguns exemplos são:
  - Personificação
  - Interposição e alteração
  - Denial of service

1.1.4 Vulnerabilidades

As vulnerabilidades de um sistema podem tomar forma em deficiências técnicas ou na atitude das pessoas.

A defesa contra falhas de segurança baseia-se na prevenção, deteção e recuperação, removendo vulnerabilidades, usando sistemas de deteção de intrusões e repondo o estado antes do ataque.

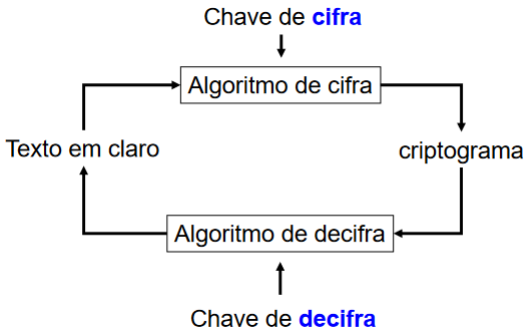
Existem vários princípios de desenho com vista a diminuir potências vulnerabilidades, como o desenho aberto, privilégio mínimo ou isolamento. Um principio em comum é que não se deve confiar apenas em segurança por ocultação, uma vez que mais tarde ou mais cedo, o desenho é divulgado.

Risco

O risco é uma métrica composta que leva em consideração o nível de ameaça a que um sistema esta exposto, o seu grau de vulnerabilidade e o impacto financeiro do ataque.

1.2 Criptografia

O propósito da criptografia é a ocultação de informação. Inerentemente, fornece indícios de que se trata de informação sensível, e o seu uso pode ser ilegal. Uma técnica também frequentemente usada é a esteganografia, onde o conteúdo sensível é ocultado dentro de outro conteúdo.



A criptanálise tem como objetivo obter o texto original, obter a chave de cifra/decifra ou obter o algoritmo de cifra usado.

1.2.1 Cifras de Substituição

Existem dois tipos de cifra de substituição:

- Mono-alfabéticas: usam apenas um alfabeto de substituição, um carácter do alfabeto original é substituído sempre pelo mesmo carácter
- Poli-alfabéticas: consistem na aplicação sucessiva e cíclica de várias cifras mono-alfabéticas

Cifra de César

É uma cifra mono-alfabética. Substitui cada letra pela k-ésima letra seguinte no alfabeto, $\mod 26$.

Cifra Vigenère

É uma cifra poli-alfabética, desde que seja selecionada uma chave com mais que um caráter.

- É selecionado um conjunto de caracteres, K, usado como chave
- Repete-se a chave em sequência até que a chave seja do tamanho do texto a ser cifrado
- Aplica-se à letra do texto em claro a substituição que corresponde à letra correspondente da chave

Por exemplo, dado $K = poema$, para cifrar o texto $elesnaosabemq$, usamos a chave $poemapoemapoe$, resultando no texto cifrado $tzienpcwmbtau$. Neste exemplo, dado que a primeira letra original era e e a letra correspondente na chave, p é a 16^{a} letra do alfabeto, avançamos e 15 letras para a frente (pois a corresponde a 0), resultando em t.

1.2.2 Cifras de Transposição

Este tipo de cifra troca a ordem dos caracteres do texto original. Por exemplo, podem ser feitas permutações fixas em blocos com um número constante de caracteres.

1.2.3 Conceitos

Cifra Perfeita

Uma cifra diz-se perfeita quando, dado um criptograma c, a probabilidade de ele corresponder a um dado texto original m e de ter sido gerado com uma dada chave k é igual à probabilidade de ocorrência do texto m. Por exemplo, a cifra de Vernam.

O cardinal do espaço de chaves tem de ser igual ou superior ao cardinal do espaço de textos em claro.

Surgem algumas dificuldades, como:

- Tem de ser usada uma chave diferente para cada texto
- O comprimento das chaves tem de ser igual ou superior ao dos textos
- As chaves não são memorizáveis

Cifra Segura

Uma cifra diz-se segura se cumprir o objetivo para que é usada, não permitindo a sua criptanálise em tempo útil e admitindo um investimento tendo em conta a relação custo-benefício.

Alguns critérios para avaliar a qualidade de cifras são:

- Quantidade de secretismo oferecida
- Dimensão das chaves
- Propagação de erros
- Dimensão do criptograma

1.2.4 Criptografia Simétrica

Neste tipo de criptografia utiliza-se a mesma chave para cifrar e decifrar. São bastante rápidos, mas necessitam de chaves secretas, cuja gestão pode ser de grande escala.

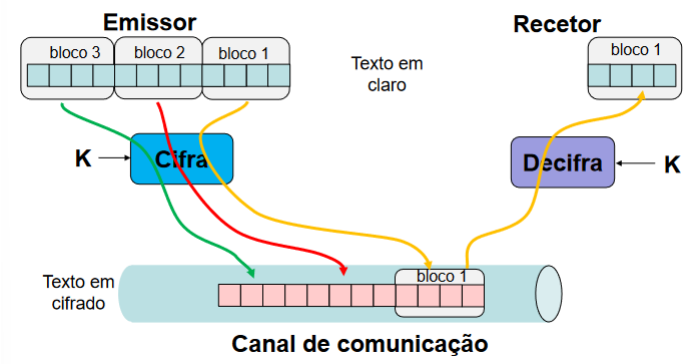
Cifra em Blocos

Atualmente o *Advanced Encryption Stantard* (AES) é o algoritmo de Rijndael:

- Recebe como entrada blocos de 128 bits de texto em claro
- As chaves podem ter 128, 192, 256 bits (quanto maior, mais seguro)
- Produz blocos de 128 bits de texto cifrado
- Iterativamente:
  - Cada bloco é dividido em 4 grupos de 4 bytes

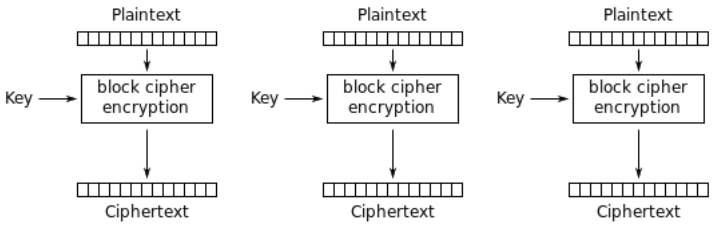
– Um bloco inteiro é modificado em cada iteração

É rápido e eficiente em CPUs pequenos e grandes.



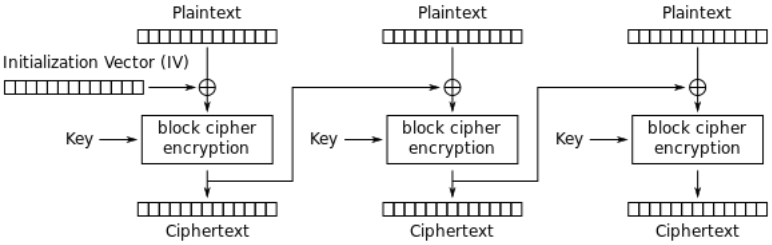
Modos de Cifra em Blocos

- *Electronic Code Book* (ECB)
  - Cifra por blocos independentes
  - Tem uma fraqueza na reprodução de padrões de texto original, dois blocos iguais produzem o mesmo criptograma



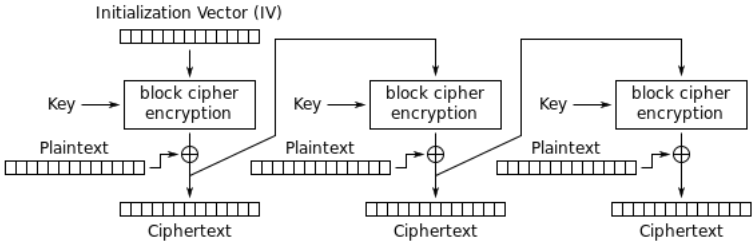
Electronic Codebook (ECB) mode encryption

- *Cipher Block Chaining* (CBC)
  - O texto em claro é XOR com o texto cifrado do bloco anterior antes de ser cifrado
  - Reduz risco de replicação de padrões
  - É usado um *initialization vector* no primeiro bloco (necessário para decifrar) e é necessário *padding*: bits para compor blocos inteiros do tamanho requerido pelo algoritmo



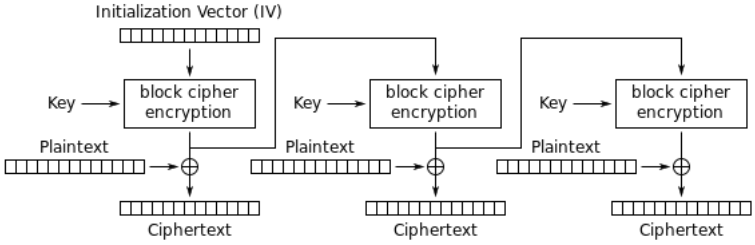
Cipher Block Chaining (CBC) mode encryption

- *Cipher Feedback* (CFB)
  - Semelhante a CBC, mas a cifra de bloco só é utilizada na direção de cifrar, o que simplifica a sua implementação
  - Também não é necessário *padding* para um múltiplo do tamanho do bloco porque o algoritmo trabalha com qualquer quantidade de bytes



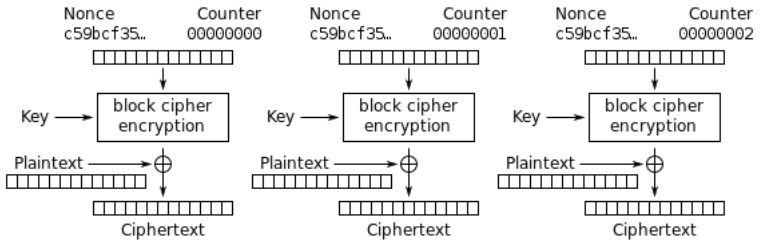
Cipher Feedback (CFB) mode encryption

- *Output Feedback* (OFB)
  - Mantém as vantagens to CFB e a mensagem não é utilizada no bloco seguinte, o que implica que as operações de cifra de bloco podem ser feitas antecipadamente permitindo que o XOR seja realizado em paralelo assim que o texto (mensagem ou mensagem cifrada) estiver disponível



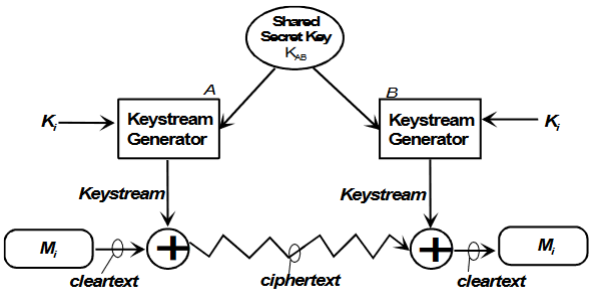
Output Feedback (OFB) mode encryption

- *Counter Mode* (CTR)
  - É o modo de cifra padrão no AES
  - É usado um nonce e um contador, que devem ser diferentes em cada operação de cifra



Counter (CTR) mode encryption

Cifra Contínua



- A cifra de fluxo processa um bit/byte de cada vez através da operação XOR
- a partir de chave partilhada é produzida uma sequência infinita de bits/bytes aleatórios a que se chama a chave de fluxo ou sequência (keystream)
- a chave de fluxo é usada uma única vez e portanto é muito difícil a criptanálise

Ao cifrar, a chave de fluxo é XOR ao correr do fluxo de texto em claro, bit a bit (ou byte a byte). Ao decifrar, o fluxo cifrado é XOR com a mesma chave de fluxo, o que retorna o fluxo original.

É necessário assegurar o secretismo, aleatoriedade e uso único da chave de fluxo, que é gerada nos dois extremos em simultâneo (estão sincronizados).

1.2.5 Criptografia Assimétrica

Neste tipo de criptografia, os dados são cifrados com uma chave pública e decifrados com uma chave privada. São baseados em problemas matemáticos para

os quais ainda não foi descoberta uma solução em tempo polinomial. Não são muito eficientes mas têm boa escalabilidade.

RSA

Neste algoritmo, o texto em claro é dividido em blocos, que são tratados como um número. As chaves são geradas da seguinte forma:

- Escolhem-se dois números primos p e q
- Considera-se $n = pq$ e $z = \phi(n) = (p-1)(q-1)$
- Escolhe-se $e < n$ tal que e é primo com z
- Calcula-se d tal que $ed \mod z = 1$
- A chave pública é $K_u = (e, n)$ e a chave privada $K_r = (d, n)$

Para cifrar calcula-se:

$$E(K_u, m) = m^e \mod n = c$$

E para decifrar:

$$D(K_r, c) = c^d \mod n = m$$

A criptanálise pode consistir em procura de chaves por força bruta ou ataques temporais, onde se consegue estimar d pelo tempo que demora uma decifração.

Diffie-Hellman

Neste algoritmo, o objetivo é obter um número secreto K, compartilhado entre A e B, mas sem o comunicar em claro.

- Escolhem-se dois números primos m e n públicos (n grande)
- A gera um número aleatório x_a e calcula $y_a = m^{x_a} \mod n$
- B gera um número aleatório x_b e calcula $y_b = m^{x_b} \mod n$
- y_a e y_b são tornados públicos
- Cada um calcula K localmente:

$$K = y_b^{x_a} \mod n = y_a^{x_b} \mod n$$

1.2.6 Funções de Síntese

A síntese ou *digest* de mensagens tem como objetivo produzir valores de dimensão constante (pequena) a partir de entradas (mensagens, ficheiros, ...) de dimensão variável. Não serve para cifrar/decifrar. Algumas propriedades são:

- Resistência à descoberta do texto original
- Resistência à descoberta de um segundo texto original
- Resistência à colisão

Para tornar colisões menos prováveis é necessário usar sínteses de tamanhos razoáveis. O ataque do aniversário é usado para encontrar um par de mensagens que colida. Para sínteses de n bits, é necessário tentar aproximadamente $2^{n/2}$ mensagens.

MD5

É um algoritmo que produz uma síntese de 128 bits, no entanto, foram encontradas falhas, pelo que a sua utilização não é recomendada.

SHA

Existem várias versões, baseadas no desenho do MD4. Recentemente foi demonstrado que é possível criar uma colisão.

1.2.7 Autenticação

MAC

O *Message Authentication Code* (MAC) usa uma chave simétrica partilhada para autenticar uma mensagem. Pode ser produzido de algumas formas:

- Cifrar mensagem e síntese da mensagem
- Cifrar síntese da mensagem
- Fazer síntese da mensagem concatenada com a chave simétrica (HMAC)
- Funções chaveadas (último criptograma gerado em modo CBC)

Assinatura

Uma assinatura digital deve cumprir as seguintes propriedades:

- Autenticidade
- Integridade
- Não-reutilização
- Não-repudiação
- Não-forjamento

Para criar uma assinatura deve-se cifrar com chave privada a síntese do texto original.

1.2.8 Criptografia Híbrida

É possível juntar a criptografia simétrica com a assimétrica. Para cifrar:

- Gerar chave secreta aleatória
- Cifrar texto em claro com chave secreta
- Cifrar chave secreta com chave pública do destinatário

Para cifrar e assinar, o texto é assinado em claro e é cifrado o texto em claro com a assinatura.

1.3 Gestão de Chaves Públicas

Existem dois tipos de chaves públicas:

- Chaves de longa duração
  - Chaves secretas partilhadas entre dois utilizadores
  - Pares de chave pública/privada
- Chaves de curta duração
  - Chaves de sessão, obtidas de forma automática
  - Mudam-se frequentemente

1.3.1 Distribuição

A distribuição de chaves públicas pode ser feita de várias formas:

- Distribuição manual, presencial ou eletronicamente, através de canais alternativos
- Distribuição embebida, através de browsers, com segurança fraca (usada para importar chaves públicas de autoridades de certificação)
- Distribuição interativa, no âmbito da execução de protocolos
- Distribuição ad-hoc, a partir de repositórios ou certificados digitais

1.3.2 Certificados Digitais

Um certificado digital contém:

- Chave pública
- Identificador do dono da chave
- Identificação da autoridade de certificação
- Assinatura digital do certificado efetuada pela Autoridade de certificação (entidade certificadora)
- Tempo de validade limitado (prazo ou certificados de revogação)

Os certificados $X.500$ possuem ainda os algoritmos usados na assinatura, números de série e versão. É possível também definir um *distinguished name* (DN), composto por uma sequência de *relative distinguished names* (pares atributo valor), que podem identificar, por exemplo, países ou organizações.

Certificação

A validação de um certificado exige a verificação da data de validade, existência de revogação e a validade da assinatura.

Uma assinatura de uma entidade certificadora pode ser validada segundo uma cadeia de certificação. Algumas organizações possíveis são:

- Monopólio (apenas uma raiz confiável)
- Oligarquia (várias raízes confiáveis)
- Certificação cruzada
- Malha
- Anárquica

A infraestrutura de chaves públicas (PKI) inclui autoridades de certificação, interligação entre cadeias de certificação e mecanismos para gerir os certificados.

Revogação de Certificados

Um certificado pode ser revogado por diversos motivos, tais como a chave privada (pessoal ou da CA) ser comprometida, ou o utilizador abandonar a organização.

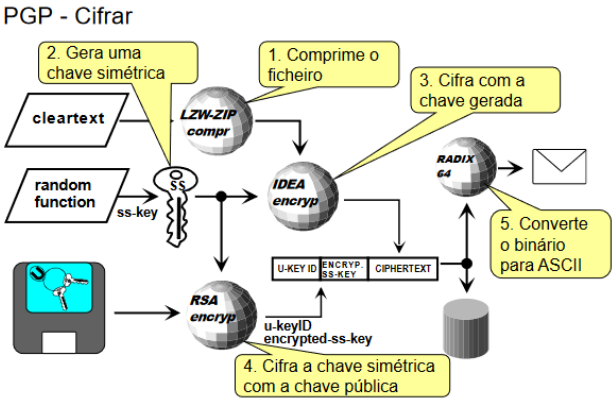
Existe um protocolo de pergunta/resposta, *Online Certificate Status Protocol* (OCSP), que mantém uma lista de certificados revogados. É incluído o ID do certificado, motivo da revogação, assinatura da CA e data de emissão.

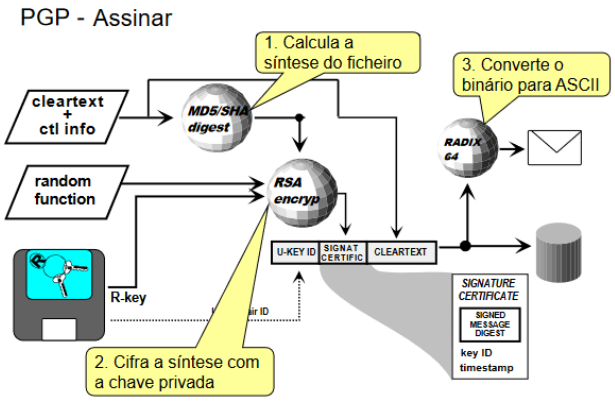
A distribuição desta lista pode ser integral ou parcial.

Pretty Good Privacy (PGP)

É um software de proteção de ficheiros e e-mail, que utiliza o modelo anárquico. Utiliza criptografia híbrida, baseada em chave pública para trocar informações de controlo (RSA) e chaves de sessão para trocar dados (IDEA).

Existe um ficheiro onde é guardada a chave privada (cifrado com chave secreta) e outro com chaves públicas. A distribuição pode ser feita por ad-hoc ou servidores HTTP/LDAP. A revogação é feita através de um certificado de revogação pelo próprio, pelo que precisa da sua chave privada.





As condições de autenticidade são:

- Recebeu certificado de chave pública do dono diretamente
- Está assinado por alguém em quem confia
- User ID tem nome completo do dono

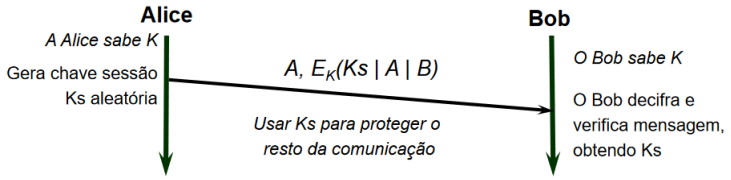
1.4 Gestão de Chaves Secretas

As chaves secretas são usadas no decorrer da execução de um determinado protocolo e mudam-se frequentemente (cada interação, cada x minutos, etc.), sendo descartadas quando a sessão termina.

Diz-se que existe *perfect forward secrecy* quando a segurança não depende da manutenção do segredo de outras chaves (de longa duração).

1.4.1 Chaves de Cifra de Chaves

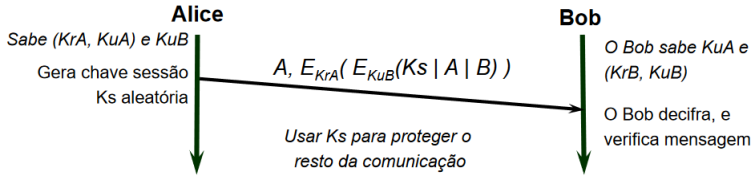
Tem como objetivo estabelecer chave de sessão recorrendo a chaves de longa duração. As chaves de longa duração são chaves secretas partilhadas entre as duas partes e a chave de sessão é trocada cifrada com a chave secreta.



Esta abordagem está sujeita a ataques de repetição. Para lidar com esse tipo de ataques, os participantes devem usar números sequenciais, pelo que devem manter contadores sincronizados. Deve também ser usado um desafio *nonce*.

Chaves Assimétricas

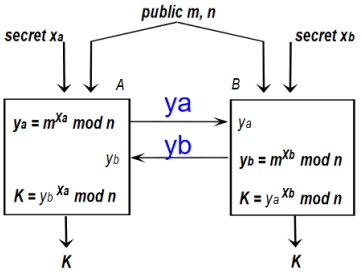
Também é possível usar este método recorrendo a criptografia híbrida. A chave de sessão é enviada cifrada com a chave pública do destinatário e é ainda assinada para garantir autenticidade (evitar ataque homem no meio).



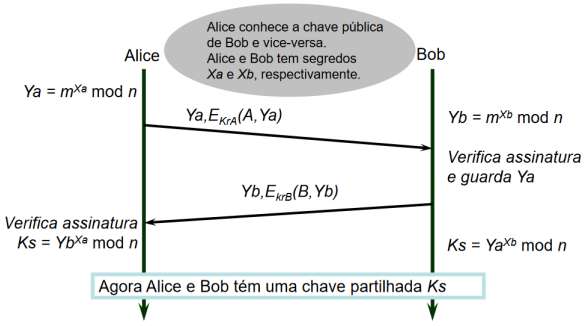
Ainda assim não garante segurança futura perfeita (*perfect forward secrecy*).

1.4.2 Diffie-Hellman

Este algoritmo garante segurança futura perfeita, dado que o cálculo da chave de sessão não envolve nenhuma chave de cifra de chaves, estando apenas sujeito a ataques de homem no meio.



A solução para este tipo de ataque consiste em validar o mac de cada mensagem, usar desafios-resposta autenticados no final do algoritmo, ou assinar valores públicos (na figura seguinte).

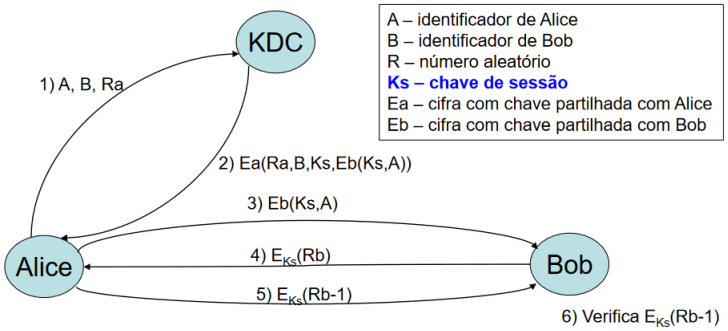


1.4.3 Centros de Distribuição de Chaves (KDCs)

Um KDC é um mediador em que todos confiam que partilha uma chave simétrica de longa duração com cada participante (n chaves).

Tem como vantagem o maior número de chaves partilhadas (n vs. $n(n-1)/2$), no entanto, é um ponto singular de falha e intrusão, um possível ponto de estrangulamento em termos de desempenho e não garante segurança futura perfeita. O funcionamento é o seguinte:

- Alice contacta o KDC e recebe:
  (a) A chave de sessão cifrada com a chave da Alice
  (b) A chave de sessão + ID da Alice cifradas com a chave de Bob
- Alice envia (b) para o Bob



O problema principal dá-se quando um atacante obtém chave de sessão antiga, consegue personalizar Alice se guardar a mensagem do passo 3.

1.5 Autenticação

Um processo de autenticação permite a um utilizador demonstrar que é a entidade associada a um dado identificador

1.5.1 Autenticação Local

Autenticação com Passwords

Este tipo de autenticação é baseado no par <nome, password>. Alguns dos ataques mais comuns são o ataque de dicionário (online ou offline), a reutilização de passwords, rapto de computadores ou engenharia social. O armazenamento de passwords pode ser feito de algumas formas:

- Texto em claro (não é uma boa solução)
- Síntese da password
- Síntese da password + salt

A melhor solução é a utilização do salt, pois evita que utilizadores que usem a mesma password tenham o mesmo hash armazenado. Ainda assim, se o adversário capturar o ficheiro e assim tiver acesso aos salts e hashes das passwords, pode usar um programa que experimenta automaticamente diversas passwords até ter sucesso ou uma tabela com estes valores pré-computados.

Autenticação com Token

- Cartão de memória
  - Guarda numa fita magnética ou numa memória interna um segredo e outra informação
  - Tipicamente usa-se em conjunto com uma password
- Cartões/dispositivos com processamento (smart cards / smart token)
  - Têm um processador embebido, capacidade de armazenar dados e alguma forma de interface
  - Suportam/correm algum tipo de protocolo de autenticação seguro com o leitor, sem divulgar as chaves armazenadas
- Passwords Descartáveis
  - É necessário um mecanismo para geração de passwords descartáveis e sincronização por tempo entre cliente/dispositivo e o servidor (password tem validade)
  - Baseado em algoritmos de desafio-resposta (a nova password é baseada num desafio fornecido pelo servidor de autenticação) ou em algoritmos criptográficos (a nova password é baseada na password anterior), como no exemplo seguinte

No algoritmo Lamport one-time password, é calculado $hash(.....hash(password)...)$ n vezes. O computador conhece n e $hash^n(password)$. Após enviar n ao utilizador, espera receber $x = hash^{n-1}(password)$ verifica se $hash(x) = hash^n(password)$.

Autenticação Biométrica

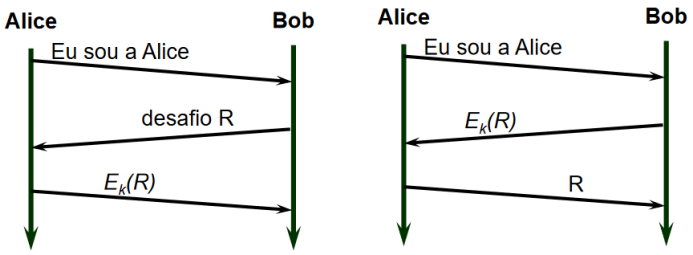
Baseia-se em características físicas estáticas (impressão digital) ou dinâmicas (voz). Funciona recolhendo um padrão associado a uma característica biométrica da pessoa e depois comparando-o mais tarde

1.5.2 Autenticação Remota

A autenticação remota pode ser classificada como unilateral, onde apenas se garante a autenticidade de quem inicia a comunicação ou mútua, onde A prova a sua identidade a B, e vice versa.

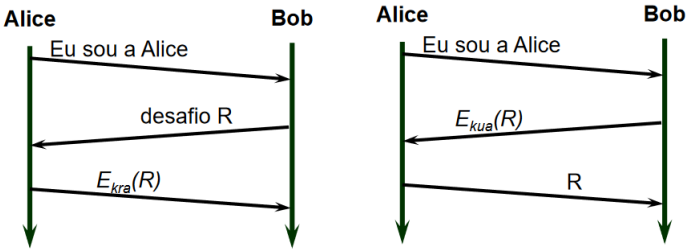
Autenticação Unilateral por Partilha de Segredo

A e B partilham um segredo K e são usados mecanismos de desafio/resposta e marcas de tempo. Por exemplo, Alice envia para Bob: timestamp, Hash(K || timestamp)

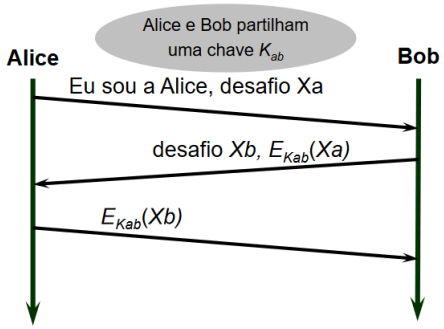


Autenticação Unilateral com Chaves Assimétricas

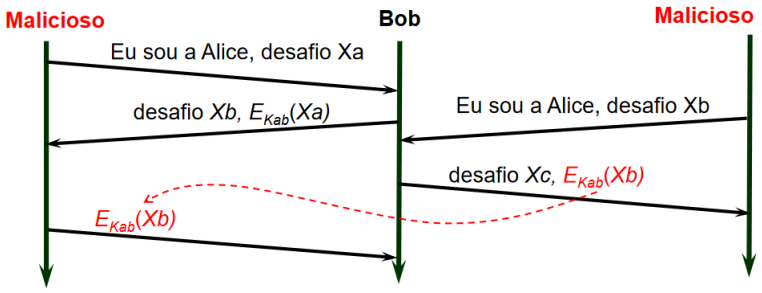
B conhece a chave pública de A, K_{ua}. Após uma troca de mensagens, B deve ficar convencido que o seu interlocutor conhece chave privada K_{ra}, logo deve ser A.



Autenticação Mútua com Segredo Partilhado



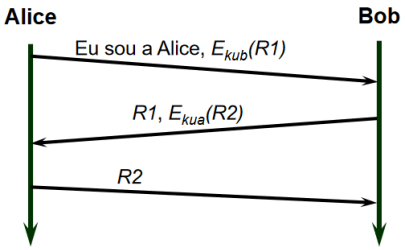
Está sujeito a um ataque de reflexo:



Existem algumas soluções possíveis:

- Duas chaves, cada uma usada para uma das direções
- Desafios com características diferentes (par vs. ímpar)
- Variante com marca de tempo

Autenticação Mútua com Criptografia Assimétrica

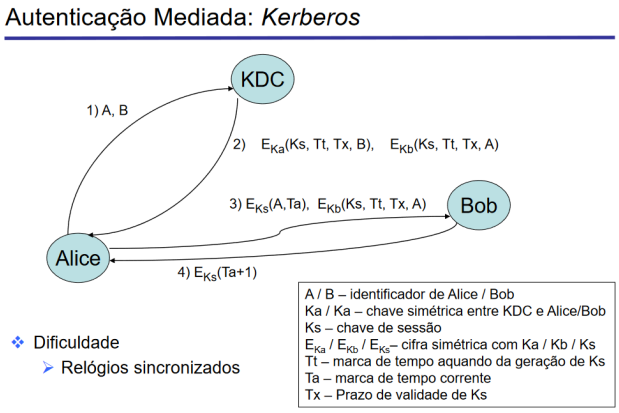


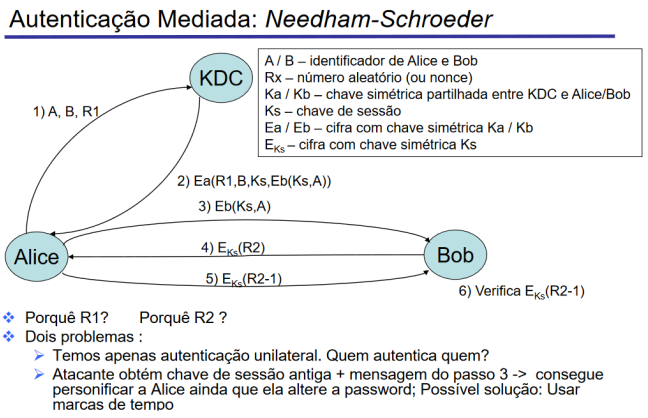
Onde $R1$ e $R2$ são *nonces*.

Autenticação Mediada

A autentica-se perante B através de um amigo comum, T. A partilha um segredo K_a com T e B partilha um segredo K_b com T.

Alguns métodos de autenticação mediada incluem:





O kerberos segue o seguinte modelo:

- Entre cliente e servidor kerberos:
  - O cliente envia uma solicitação de autenticação ao Servidor Kerberos, indicando que deseja obter um Ticket de Autenticação do Cliente (TGT)
  - O Servidor Kerberos autentica o cliente e, se bem-sucedido, envia de volta ao cliente um TGT criptografado
- Entre o cliente e o Ticket Granting Server (TGS):
  - O cliente envia uma solicitação ao TGS, juntamente com o TGT recebido do Servidor Kerberos
  - O TGS verifica o TGT do cliente e, se válido, emite um TSS criptografado, que contém uma chave de sessão para comunicação segura
- Entre o cliente e o servidor:
  - O cliente envia a solicitação original ao Servidor, juntamente com o TSS recebido do TGS.
  - O Servidor verifica o TSS do cliente e, se válido, concede acesso ao serviço solicitado

1.6 Comunicação Segura

1.6.1 Propriedades

- Confidencialidade
- Autenticidade
- Integridade

A concretização destes requisitos geralmente assenta em criptografia a/simétrica.

1.6.2 SSL

SSL é um protocolo que garante comunicação segura sobre o nível transporte. A autenticação pode ser inexistente (para interações anónimas), pode ser unilateral (do lado do servidor) ou mútua.

Parâmetros

Cliente e servidor criam um $pre - master - secret$. Os valores são calculador com funções de síntese a partir deste segredo e de *nonces*. Existem duas chaves secretas para MACs e duas chaves para cifra (servidor $\rightarrow$ cliente e cliente $\rightarrow$ servidor).

Protocolo Record

Este protocolo é responsável pela comunicação segura, fazendo fragmentação, compressão e cifra e garantindo autenticação e integridade das mensagens. Encapsula protocolos de nível superior, como HTTP, e corre sobre TCP/IP ou outros protocolos de transporte.

Utilização

Quando o HTTP corre sobre SSL, é denominado HTTPS. O servidor tem o certificado gerado por uma CA fiável contendo a sua chave pública. O servidor deve provar a sua identidade a pedido do cliente.

1.6.3 IPSec

Trata-se de uma extensão de segurança do protocolo IP. É transparente para aplicações e dá segurança em protocolos de encaminhamento usando a infraestrutura IP existente. Todo o tráfego é tratado de forma indiferenciada.

Modos

Existem dois modos possíveis:

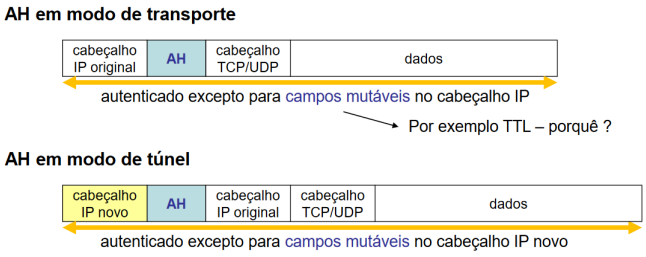
- Modo de transporte
  - Cabeçalho IP original é mantido
  - Utilizado normalmente entre duas entidades finais
- Modo túnel
  - Encapsula o datagrama
  - Usado normalmente quando uma das entidades não é final (routers)

Mecanismos

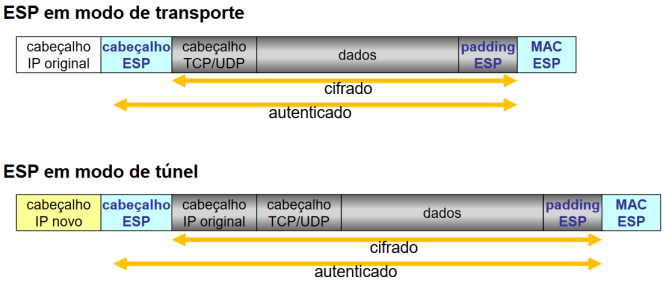
É possível fazer duas alterações a datagramas IP: cifra do corpo e adição de cabeçalhos extra (MAC).

- Authentication Header (AH) - Não deve ser usado

- Autentica o datagrama + (a maioria do) cabeçalho IP + cabeçalho extra AH
- Garante autenticação da origem e integridade do conteúdo
- Pode evitar ataques de repetição (inclui contador)



- Encapsulating Security Payload (ESP)
  - Cifra corpo do datagrama
  - Opcionalmente autentica o corpo do datagrama + cabeçalho ESP
  - Garante integridade do conteúdo
  - Pode evitar ataques de repetição (inclui contador)

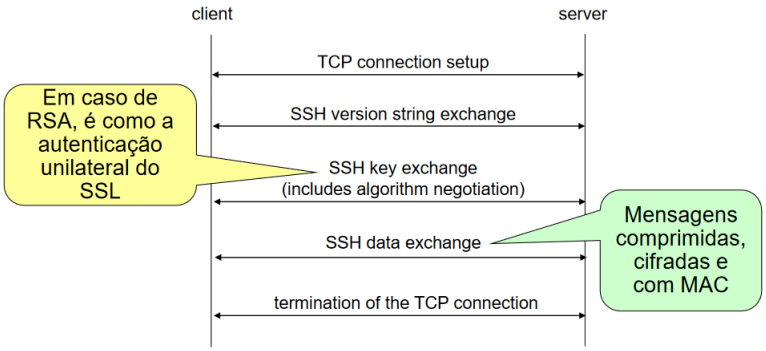


Estes mecanismos podem ser usados em conjunto:

Serviços	AH	ESP (cifra)	ESP (cifra + autenticação)
integridade	X		X
autenticação da origem de dados	X		X
detecção de replay	X	X	X
confidencialidade		X	X

1.6.4 SSH

O protocolo SSH (Secure Shell) permite sessões remotas seguras sobre TCP. Utiliza autenticação de ambas as partes e suporta distribuição de chaves manual ou automática.



1.7 Firewalls

Uma firewall proporciona o ponto único de acesso à rede interna, permitindo definir regras de controlo de acesso a máquinas/serviços da rede interna e protegendo-a de várias classes de ataques provenientes da Internet. Deve ser inserida entre a rede interna da organização e a internet.

- Todo o tráfego do exterior para o interior, e vice versa, deve passar pela firewall
- Só o tráfego autorizado (definido pela política de segurança) pode passar
- A firewall deve ser imune a penetrações

1.7.1 Controlo de Acesso

- Controlo dos serviços
  - Determina que serviços podem ser acedidos
- Controlo da direção
  - Determina a direção (de fora para dentro, e de dentro para fora) que um pedido para um serviço particular pode ser executado
- Controlo do utilizador
  - Determina que utilizadores podem aceder ao serviço
- Controlo de comportamento
  - Controla a forma como são usados determinados serviços

No entanto, não protege contra ataques que a contornam, ataques provenientes de máquinas internas nem programas/ficheiros infetados com vírus/worms.

É possível seguir uma política de segurança:

- Prudente: tudo o que não é explicitamente permitido não passa
- Permissiva: tudo o que não é explicitamente negado é permitido

1.7.2 Packet-Filtering Router

Este tipo de router aplica um conjunto de regras aos pacotes que chegam e depois reenvia ou descarta os pacotes.

Estas regras são ormalmente são baseadas nos cabeçalhos IP e do protocolo nível transporte. Caso não exista nenhuma regra associada ao pacote, aplica-se a política de omissão: descartar ou passar.

Rule	Direction	Src addr	Dst addr	Prtocol	Dest port	Action	
1	In	External	Internal	TCP	25	Permit	Ligação ao meu servidor de SMTP
2	Out	Internal	External	TCP	>1023	Permit	
3	Out	Internal	External	TCP	25	Permit	Ligação a um servidor de SMTP
4	In	External	Internal	TCP	>1023	Permit	
5	Either	Any	Any	Any	Any	Deny	Política de omissão é descartar

Benefícios:

- Simplicidade
- Transparência para com os utilizadores
- Rápidos

Limitações:

- Não evitam ataques nível aplicacional (não inspecionam o conteúdo das mensagens da aplicação)
- Não suportam mecanismos avançados de autenticação
- É relativamente fácil introduzir erros nas regras do filtro

Alguns ataques possíveis a este tipo de router incluem:

- IP address spoofing: o intruso transmite pacotes em que coloca como endereço IP do emissor um endereço da rede interna.
  - A solução é descartar pacotes provenientes da placa de rede externa com endereços de emissão internos
- Ataques source routing: o emissor especifica no pacote a opção de source route que indica o caminho a tomar na internet

- A solução é descartar todos os pacotes com esta opção

- Ataques com fragmentos pequenos: usa-se a fragmentação IP com o objectivo de se criarem fragmentos muito pequenos, de modo a forçar que os cabeçalhos apareçam em fragmentos distintos.
  - A solução é descartar todos os pacotes em que o protocolo é TCP e o tamanho dos fragmentos é pequeno (1 byte)

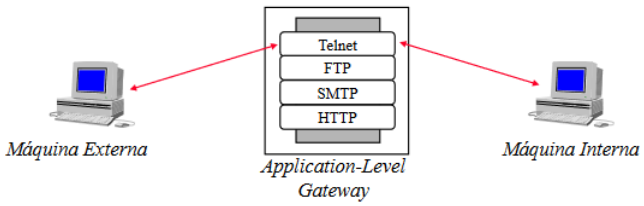
PFR com Sateful Inspection

Consiste na utilização de packet-filtering routers que mantêm estado sobre ligações que foram permitidas no passado.

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

1.7.3 Application-Level Gateway (Application Proxy)

Trata-se de uma gateway atua na camada de aplicação e suporta aplicações para as quais existe um servidor de proxy. Funciona como representante do servidor interno, agindo como intermediário entre as ligações de rede e analisando o tráfego.



- Os utilizadores apenas podem aceder aos servidores proxy, mas nunca se podem ligar diretamente à gateway ou aos servidores da rede interna

- O servidor de proxy pode ser configurado de forma a suportar apenas o subconjunto de serviços fornecidos pela aplicação que são considerados aceitáveis (seguros), e o resto dos serviços são negados

Este método assenta no conceito de uma *Trusted Computing Base*, um subconjunto do sistema que é inerentemente seguro e geralmente usado para executar mecanismos críticos de segurança (eg. firewall).

É utilizado um bastion host: uma máquina crítica do ponto de vista de segurança que executa uma versão mais segura do sistema operativo e tem instalados apenas os serviços essenciais.

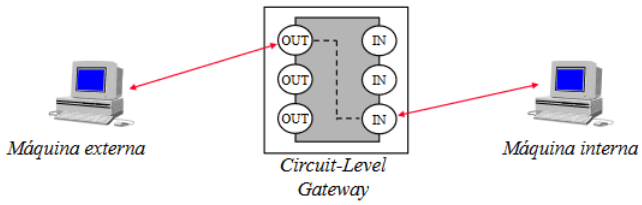
Benefícios:

- Controlo completo sobre cada serviço: é possível especificar quais os serviços suportados e com que opções
- Possibilita formas mais seguras de autenticação e de coleção de informação de auditoria
- Mais fácil de configurar e de testar que um PFR

Limitações:

- Requer que os utilizadores se habituem à nova forma de funcionar dos serviços ou a existência e instalação nas máquinas clientes de software especializado para acederem a serviços proxy
- Processamento adicional (overheads) em cada ligação

1.7.4 Circuit-Level Gateway (Circuit-Level Proxy)



A gateway atua na camada de transporte, criando duas ligações (TCP), uma entre ela e a máquina interna e outra entre ela e a máquina externa e passa o tráfego pela ligação sem fazer qualquer outro processamento.A função de segurança consiste em determinar que ligações é que são permitidas.

É usada quando quando os utilizadores internos são de confiança

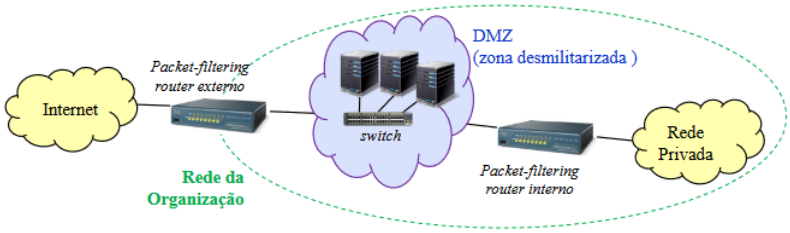
1.7.5 Firewall na Máquina

É possível criar uma firewall num computador específico (eg. IPtables ou parte do SO), que restringe os pacotes que chegam e saem da máquina.

Tem como vantagens as regras de filtragem poderem ser adequadas às necessidades locais, a proteção é feita independentemente da topologia de rede e funciona como um nível extra de proteção.

1.7.6 Arquiteturas de Firewalls

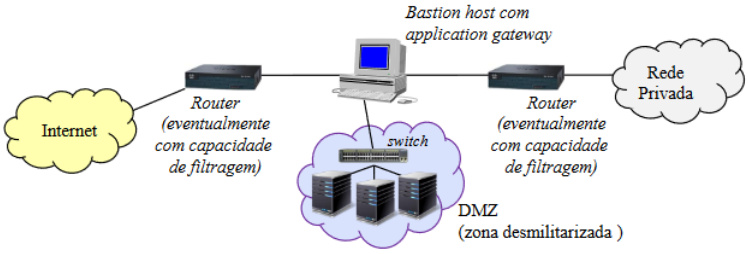
Inline Firewalls



Os computadores são distribuídos por duas (ou mais) zonas de segurança. No exterior (DMZ) são colocados os computadores que necessitam de ser acedidos pela internet (servidores) e no interior os computadores privados da organização.

Para conseguir corromper os computadores privados é preciso passar por duas firewalls, mas existe um overhead no acesso á internet nos computadores internos.

Gateway Simples (Dual Home Gateway)



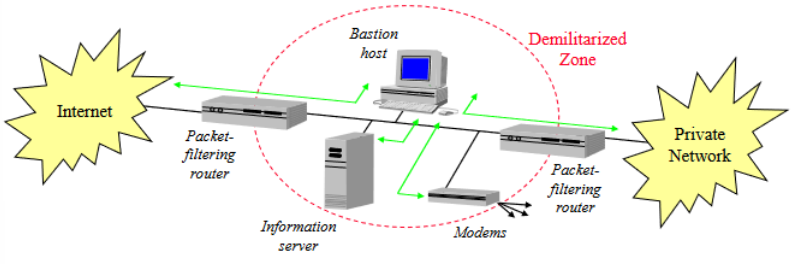
A firewall é constituida por uma máquina com duas ou mais interfaces: uma para o exterior e uma ou mais para redes interiores. Os encaminhadores são opcionais, embora permitam reforçar a segurança uma vez que podem operar como filtros de pacotes.
Vantagens:

- Simplicidade
- Economia de recursos
- Os computadores continuam a ser divididos por duas zonas de segurança, mas evitamos os atrasos extra nos computadores da rede privada

Desvantagens:

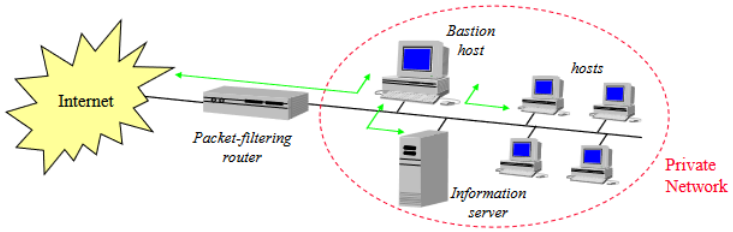
- Comprometimento da gateway-bastião (diretamente exposta a ataques do exterior e do interior)
- Limitações à localização dos serviços públicos

Screened-Subnet Firewall System



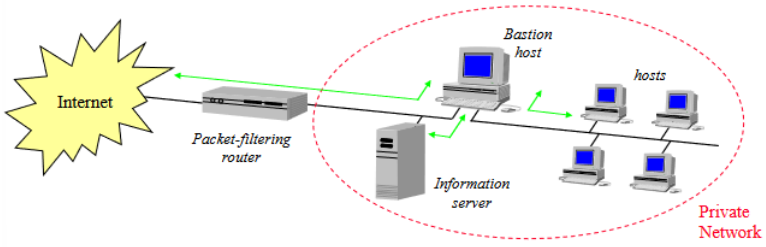
Usam-se dois PFR para definir uma zona desmilitarizada, onde são colocados os servidores públicos. O PFR permite apenas mensagens destinadas para o bastion host e o PFR interior permite apenas tráfego cuja origem é o bastion host.

Existem duas concretizações diferentes, O single-homed bastion host:



Onde o acesso às máquinas internas têm de passar pelo PFR e pelo servidor proxy e as regras do packet-filter apenas permitem acesso exterior ao bastion host. Tem a desvantagem de permitir que um intruso escute o tráfego da rede interna se as regras do PFR forem comprometidas.

E o dual-homed bastion host:



Em que o bastion host tem duas interfaces de rede e o re-envio automático de tráfego de uma placa para a outra não é permitido, pelo que os serviços de proxy têm de ser necessariamente usados para que exista acesso à rede interna.

O acesso externo ao Information Server pode ser direto ou por intermédio do bastion host.

1.7.7 Intrusion Prevention Systems (IPS)

Extensão aos sistemas de deteção de intrusões (IDS), adicionando-lhes a capacidade de bloquear o tráfego considerado malicioso

1.8 Sistemas de Deteção de Intrusões

Como visto anteriormente, numa arquitetura de segurança tradicional, o PFR tem como função básica a detecção de ataques de personificação de endereço ou outros ataques específicos. O bastion host permite, por exemplo, a execução de formas mais seguras de autenticação e a firewall limita o tráfego entre a DMZ e a rede privada.

No entanto podem surgir problemas:

- Qualquer um dos componentes de software executados na DMZ pode ter um bug de segurança
- Um ou mais dos mecanismos de segurança não está configurado corretamente

Um Sistema de Deteção de Intrusões (SDI) tem como principal função a observação e análise das atividades que ocorrem no sistema, com o objetivo de detetar intrusões. A intrusão pode ser por:

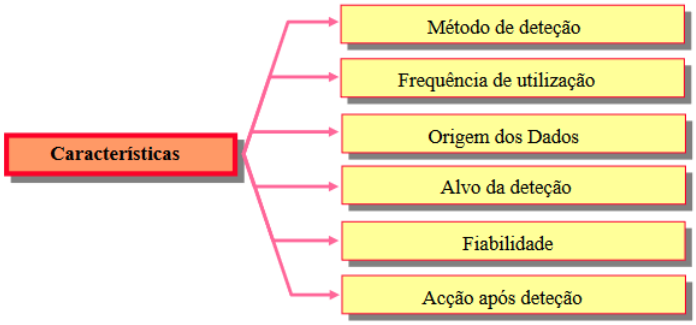
- Utilizador interno que consegue escalar os seus privilégios
- Intruso externo que consegue obter acesso a um dos computadores

Um SDI pode estar constantemente a monitorizar o sistema de forma a que os ataques sejam detetados em tempo real, ou ser executado esporadicamente quando existe suspeita de intrusão.

1.8.1 Componentes

Os principais componentes de um SDI são:

- Sensores
- Analisador
- Interface com o utilizador
- Arquivo de eventos



1.8.2 Métodos de Deteção

Deteção Baseada no Conhecimento

- Compara as atividades observadas com uma lista de padrões/assinaturas de ataque
- Assume que tudo o que não é conhecido é válido
- Baixo nível de falsos alarmes (falsos positivos)

Vantagens:

- O número e tipo de eventos a considerar está limitado às atividades descritas nos padrões
- Os cálculos tendem a ser mais eficientes do que com a deteção baseada no comportamento

Desvantagens:

- Quando surgem novos tipos de ataques, é necessário adicionar novos padrões
- Adição de padrões poderá ser difícil se não houver uma linguagem standard de especificação

Deteção Baseada no Comportamento

- Procura desvios de comportamento dos utilizadores, grupos de utilizadores, servidores, ..., face ao comportamento normal
- Um modelo é associado a cada utilizador, cuja forma é determinada através da observação do comportamento do utilizador ao longo do tempo (período de aprendizagem)
- Assume que os desvios são suspeitos
- Potencialmente um elevado nível de falsos alarmes (falsos positivos)

Vantagens:

- Permite detetar ataques zero-day
- Podem-se usar técnicas estatísticas e de aprendizagem automática bem conhecidas

Desvantagens:

- As hipóteses sobre os dados recolhidos podem não ser válidas sobre o ponto de vista estatístico
- O comportamento de certos utilizadores pode não ser consistente ao longo do tempo

1.8.3 Detetores de Vulnerabilidades

Um detetor de vulnerabilidades periodicamente percorre o sistema à procura de vulnerabilidades.

Podem ser detetadas através de análise do estado da máquina (programa x versão y tem um bug de segurança conhecido, presença de erros de configuração, ...). Os detetores podem ser:

- Locais: detetam problemas no sistema onde se está a executa
- Remotos: Procura problemas de segurança nos sistemas ligados à rede; esta atividade é conseguida através do envio de pacotes para os serviços de rede

Ambos os tipos se complementam.

1.8.4 Ação Após Deteção

Após a deteção e identificação de uma intrusão deve haver uma resposta, que pode ser:

- Resposta passiva: informa-se o administrador do ataque e espera-se que ele o corrija
- Resposta ativa: o sistema bloqueia ou afeta o progresso do ataque
  - Iniciar ação contra o intruso (terminar ligação, bloquear com firewall)
  - Alterar o ambiente
  - Recolher mais informação (eg. levar o atacante a tribunal)

Capítulo 2

Confiabilidade

2.1 Conceitos

A confiabilidade de um sistema é capacidade do sistema evitar que os seus serviços tenham uma falha mais frequentemente ou com maior severidade do que é aceitável

2.1.1 Terminologia

- Sistema: entidade que interage com outras entidades (i.e., outros sistemas)
- Ambiente: outras entidades que interagem com o sistema de interesse
- Fronteira do sistema: fronteira entre o sistema e o ambiente

Esta ideia aplica-se recursivamente: um sistema é construído a partir de subsistemas, que por sua vez também são formados for subsistemas

- Requisitos (funcionais, confiabilidade): definem o problema que o sistema resolve
- Especificação: define o que o sistema deve fazer para cumprir os requisitos
- Falha: ocorre quando o serviço prestado pelo sistema se desvia do serviço correto

Tipo de falha	Descrição
Omissão	
Omissão na receção	O sistema não recebe uma resposta a um pedido
Omissão no envio	O pedido enviado pelo sistema não chega ao seu destinatário
Paragem	O sistema para (embora funcione corretamente até parar) O sistema não envia nem recebe mensagens (omissões infinitas)
Temporal	A resposta é produzida fora do momento esperado
Valor	A resposta produzida é incorreta
Sintática	O valor incluído na resposta está mal formado (e.g., +28ge ºC)
Semântica	O valor na resposta parece válido mas é errado (e.g., -296 ºC)
Arbitrária (ou Bizantina)	O sistema tem um comportamento arbitrário (e.g., produz valores errados em alturas indefinidas)

*Relevante do ponto de vista de **segurança** -> o adversário faz com que o sistema se comporte arbitrariamente*

Modos de falha (e.g. paragem):
fail-stop (para de forma controlada) / fail-safe (para de forma a não causar danos) / fail-silent (para de forma silenciosa)

2.1.2 Atributos

- Fiabilidade: medida em que o serviço se mantém correto ao longo do tempo ($R(t)$ é probabilidade de o sistema estar a funcionar corretamente num dado ambiente operacional até ao instante t)
- Disponibilidade: medida em que o serviço se encontra acessível aos seus utilizadores ($A(t)$ é a probabilidade do sistema estar operacional no instante t)
- Segurança crítica (safety): garantia de ausência de falhas catastróficas para os utilizadores e ambiente
- Integridade: ausência de alterações incorretas ao sistema
- Manutenção: capacidade de se poderem proceder a alterações e reparações no sistema
- Confidencialidade: ausência de divulgação de informação a entidades não autorizadas

Notar que um sistema pode ter uma disponibilidade elevada mas uma fiabilidade baixa (e vice versa)

Segurança Crítica

Um sistema safety-critical é caracterizado por as falhas terem um impacto muito grande.

Na prática, como não se consegue atingir a perfeiçãor, usa-se o risco relacionado com as perdas médias por unidade de tempo que são observadas no sistema:

$$risco = \sum_i prb(failure\ i) \times loss(failure\ i)$$

Onde:

- i corresponde a todos os tipos de falhas possíveis

- $prb()$ é a probabilidade de ocorrência da falha i por unidade de tempo
- $loss()$ é valor monetário ou de vidas caso ocorra a falha i

2.1.3 Falhas

Uma falha ocorre quando:

- O sistema está num estado erróneo
- parte do estado erróneo passa para o exterior (torna-se visível na fronteira do sistema)

A causa de um erro é uma falta e a consequência é uma falha.

Classificação de Faltas

Faltas podem ser classificadas como sendo de paragem, omissão, etc. Podem também ser classificadas quanto á sua manifestação:

- Falta ativa: uma falta está ativa se causa um erro
- Falta dormente: a falta existe mas ainda não causou um erro
- Falta permanente: se a falta existe e mantém-se ao longo do tempo
- Falta transiente: se a falta aparece esporadicamente

2.1.4 Melhorar a Confiabilidade

- Prevenção de faltas: meios que permitam prevenir a introdução de faltas no sistema
- Remoção de faltas: meios que possibilitem uma redução do número e severidade de faltas
- Tolerância a faltas: meios que evitem a falha dos serviços ainda que ocorram faltas
- Previsão de faltas: meios que facilitem a estimativa das faltas presentes, que irão ocorrer no futuro e as consequências das faltas

Teste de Software

O teste de software engloba um conjunto muito vasto de técnicas, que consistem em observar o seu comportamento.

- Testes estáticos: realizados sem se executar o software
- Testes dinâmicos: realizados enquanto o programa está em execução

Outra terminologia: testes de caixa branca quando se tem acesso ao código e testes de caixa preta se apenas se tem acesso ao executável

Existem vários tipos de teste, como testes de unidade, de módulos, integração, etc. Para além disto, podem ser ainda classificados como:

- Testes funcionais : têm por objetivo garantir que o software faz o que é suposto fazer
- Testes de segurança : pretende verificar que o software não faz o que não é suposto fazer

2.1.5 Tolerância a Faltas

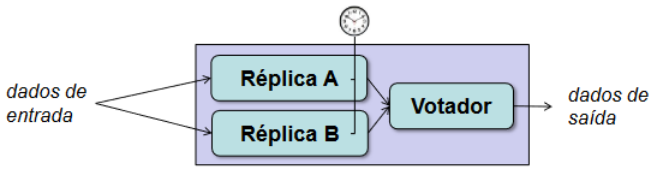
Utiliza técnicas que garantam (com elevada probabilidade) que o sistema não falha ainda que ocorram faltas durante a sua execução.

É necessário tirar partido de alguma forma de redundância, que depende do tipo de falta que se pretende tolerar. As principais forma de redundância são:

- Tempo
- Informação
- Componentes

2.1.6 Arquiteturas Comuns

Replicação Ativa



É usado um relógio para decidir quando ler valores e produzir resultados.

- Se ocorrer uma paragem, a outra unidade poderia continuar a fornecer um valor
- Se for um resultado divergente é necessário parar todo o sistema porque não se sabe qual réplica teve a falha
- Se ocorrer uma falha na especificação do sistema, em princípio não há nada a fazer se as réplicas forem iguais

Existe uma hipótese base que assume que as réplicas falham de modo independente, pelo que se deve assegurar que não ocorrem faltas de modo comum (existe o mesmo bug em ambas, falta de eletricidade, ...).

Para que as réplicas produzam os mesmos resultados, existem também duas hipótese face ao determinismo durante a execução:

- O ambiente é determinista, o que pode ser complicado se os dados forem lidos em momentos diferentes
- O componente tem uma execução determinista, produzindo os mesmos resultados ao receber as mesmas entradas e partindo do mesmo estado inicial

Ganho de Confiabilidade

Supondo que ambas as réplicas têm uma probabilidade p de falha por segundo e que o votador nunca falha (otimista).

Para falhas arbitrárias (as duas réplicas têm de funcionar corretamente para que votador possa decidir e sistema não pare):

- Probabilidade de funcionar corretamente: $R = (1-p)(1-p) = 1-2p+p^2$
- Probabilidade de falhar: $F = 1 - R = 2p - p^2$
- MTTF $= 1/(2p - p^2)$

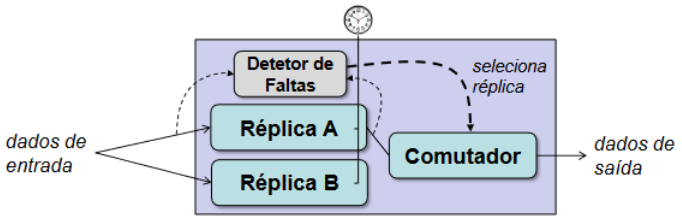
Dado que o MTTF de um só componente é $1/p$, não se conseguiu uma melhoria através de replicação, mas tem a vantagem de lidar com faltas arbitrárias.

Para faltas de paragem (apenas uma das réplicas têm de funcionar corretamente e dar uma resposta):

- Probabilidade de funcionar corretamente: $R = (1 - F) = 1 - p^2$
- Probabilidade de falhar: $F = p * p = p^2$
- MTTF $= 1/p^2$

Conseguiu-se uma melhoria no MTTF através da replicação, tem a vantagem de aumentar a disponibilidade.

Replicação passíva



O comutador ativa a réplica secundária e escolhe a sua saída seguindo a indicação do detetor de falhas. Em geral apenas tolera faltas de paragem.

O detetor de faltas monitoriza a réplica primária através de ping, por exemplo, pelo que a deteção pode não ser imediata, mas não há dados de saída incorretos.

A réplica secundária opera da seguinte forma:

- Réplica está parada (cold spare)
  - Não efetua qualquer computação e pode estar desligada
  - Primário guarda estado periodicamente e resultados de operações num log

  - Para que possa substituir o primário, a réplica tem de ser posta em execução e recuperar estado a partir dos dados armazenados

- Réplica está ativa, mas não executa nem tem estado (warm spare)
  - Não executa qualquer operação mas mantém-se pronta para substituir o primário caso seja necessário

- Réplica está ativa e tem estado, mas não executa (hot spare)
  - Não executa qualquer operação mas tem o mesmo estado que o primário e pode substituí-lo a qualquer momento

O número de réplicas necessárias para tolerar f faltas depende da classe de faltas consideradas:

	Classe de faltas	
	Paragem	Valor
Replicação passiva	f+1	----
Replicação ativa	f+1 (*)	2f+1

(*) Requer que comandos sejam entregues pela mesma ordem a todas as réplicas

Previsão de Faltas

A previsão de faltas é utilizada quando não se consegue evitar/eliminar/tolerar certas faltas, e a ideia é prever a frequência e se os efeitos dessas faltas são aceitáveis. Existem duas atividades na avaliação:

- Qualitativa que identifica, classifica, e ordena os modos de falha
- Quantitativa que determina as probabilidades da ocorrência das faltas relevantes e de que forma certos atributos (MTTF) são satisfeitos

A injeção de faltas permite por exemplo determinar os efeitos das faltas e a taxa de cobertura dos mecanismos de tolerância a faltas.

2.2 Replicação de Informação

2.2.1 Tipos de Erros

- Erros isolados: afetam apenas um bit
- Erros em grupo (burst): afetam um conjunto de bits

Os erros isolados são mais simples de detetar mas afetam mais blocos.

2.2.2 Códigos de Deteção de Erros

O objetivo é construir códigos que permitam ao recetor detetar os erros introduzidos durante a transferência de dados pela rede

- Um código com palavras de n bits tem 2^n combinações de 0s e 1s

- Se o código for estruturado de forma a que apenas um subconjunto das combinações possíveis sejam palavras válidas, então conseguiremos detetar pelo menos alguns dos erros

Distância de Hamming

A distância de Hamming entre duas palavras de código é o número de bits que as duas palavras diferem (fazer XOR das palavras e contar os 1s).

Com d erros isolados transformo (no pior caso) uma palavra na outra se a distância for d. Existe sempre deteção se o número de erros for menor que d. Algumas características dos códigos são:

- Distância do código: menor distância de Hamming entre duas quaisquer palavras válidas do código (para detetar d erros preciso dum código com distância $d + 1$)
- Separável: a informação redundante que permite detetar os erros é adicionada ao código original, logo, para uma palavra de código:

$$bits\ da\ palavra\ de\ código\ =\ bits\ que\ representam\ os\ dados\ originais\ ||$$

$$bits\ de\ informação\ redundante$$

Códigos de Paridade

Adiciona-se um bit de paridade aos dados originais, permitindo detetar apenas um erro.

Pode-se usar paridade ímpar, onde o bit extra tem o valor 0 ou 1 de forma a que o número total de bits a 1 seja ímpar, ou paridade par, onde o total de bits a 1 deve ser par.

Decimal digit	BCD	BCD odd parity	BCD even parity
0	0000	0000 1	0000 0
1	0001	0001 0	0001 1
2	0010	0010 0	0010 1
3	0011	0011 1	0011 0
4	0100	0100 0	0100 1
5	0101	0101 1	0101 0
6	0110	0110 1	0110 0
7	0111	0111 0	0111 1
8	1000	1000 0	1000 1
9	1001	1001 1	1001 0
		↑ Parity bit	↑ Parity bit

Checksum

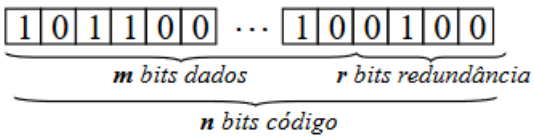
Códigos de checksum são códigos separáveis, informação que é adicionada aos dados de forma a que se possam detetar erros. Existem alguns tipos de checksum:

- Checksum de Precisão Simples
  - Adicionam-se os dados originais e ignoram-se os carrys do último bit
  - Podem surgir problemas, ignorar os carry limita a capacidade de deteção do código
- Checksum de Honeywell
  - Junta as palavras de dados duas a duas e depois faz a soma para gerar o checksum
  - Erros no mesmo bit afectam afectam pelo menos dois bits no checksum
- Checksum de Resíduos
  - Semelhante ao checksum de precisão simples, só que os bits de carry são adicionados ao checksum
  - Permite a deteção dum erro que coloque sempre um bit a um dado valor

2.2.3 Códigos de Correção de Erros

O objetivo é construir códigos que permitam ao recetor detetar e corrigir os erros introduzidos. Ao descobrir o bit com erro, basta trocar o seu valor.

Para corrigir d erros preciso de um código com distância $2d + 1$.



Nesta situação, existem 2^m palavras diferentes de dados numa mensagem e 2^n palavras diferentes de código. Garante-se que a correção de um erro é possível se para cada palavra de dados:

- Associamos uma palavra de código C (α)
- Todas as palavras de código distintas de C em 1 bit são inválidas (β)
- Por cada palavra de dados, os subconjuntos $\alpha \cup \beta$ são disjuntos

É necessário reservar $n+1$ palavras de código por cada palavra de dados. Logo, dado m, deve-se resolver a equação $(m + r + 1) \leq 2^r$ para encontrar r (redundância mínima para correção de um erro).

Cálculo da Palavra de Código

- Determinar quantos bits de redundância se deve adicionar

  • Exemplo de palavra original (7 bits) : 1 0 0 1 0 0 0
  • Para corrigirmos 1 bit precisamos de 4 bits de redundância (usar formula $m + r + 1 \leq 2^r$)

- Numerar os bits consecutivamente, começando com o valor 1 para o bit mais à esquerda

  • Palavra de código : $b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8\ b_9\ b_{10}\ b_{11}$

- Determinar os bits redundantes
  - os bits com números que são potência de 2 são bits redundantes, e os restantes são os bits da palavra original

  • Bits redundantes : $b_1\ b_2\ b_4\ b_8$
  • Bits originais : $b_3\ b_5\ b_6\ b_7\ b_9\ b_{10}\ b_{11}$
  • Palavra de código : $b_1\ b_2$ 1 b_4 0 0 1 b_8 0 0 0

- Calcular paridade
  - cada bit de redundância força a paridade num subconjunto de bits
  - cada bit pode ser usado mais do que uma vez nos cálculos da paridade
  - Método
    1. expandir o número de cada bit como soma de potências de 2

  • $b_1 \rightarrow 1$
  • $b_2 \rightarrow 2$
  • $b_3 \rightarrow 1 + 2$
  • $b_4 \rightarrow 4$
  • $b_5 \rightarrow 1 + 4$
  • $b_6 \rightarrow 2 + 4$
  • $b_7 \rightarrow 1 + 2 + 4$
  • $b_8 \rightarrow 8$
  • $b_9 \rightarrow 1 + 8$
  • $b_{10} \rightarrow 2 + 8$
  • $b_{11} \rightarrow 1 + 2 + 8$

    2. o bit de redundância garante a paridade (par ou impar) dos bits em que aparece na expansão

Bit de redundância	Paridade par
• $b_1 \rightarrow \{3, 5, 7, 9, 11\}$	XOR(1, 0, 1, 0, 0) = 0
• $b_2 \rightarrow \{3, 6, 7, 10, 11\}$	XOR(1, 0, 1, 0, 0) = 0
• $b_4 \rightarrow \{5, 6, 7\}$	XOR(0, 0, 1) = 1
• $b_8 \rightarrow \{9, 10, 11\}$	XOR(0, 0, 0) = 0

palavra de código = 0 0 1 1 0 0 1 0 0 0 0

Detetar e Corrigir o Erro

- Inicializar um contador a zero

  • Palavra recebida : 0 0 1 1 *1* 0 1 0 0 0 0 (*erro*)
  • *contador* = 0

- Verificar a paridade de cada um dos bits de redundância b_k; caso exista um erro, adicionar *k* ao contador

  • $b_1 \rightarrow$ XOR{ $b_3, b_5, b_7, b_9, b_{11}$ } = XOR(1, 1, 1, 0, 0) = 1
    o bit está **errado**, logo *contador* = *contador* + *1* = *1*
  • $b_2 \rightarrow$ XOR(1, 0, 1, 0, 0) = 0 **correcto**
  • $b_4 \rightarrow$ XOR(1, 0, 1) = 0 o bit está **errado**, logo *contador* = *5*
  • $b_8 \rightarrow$ XOR(0, 0, 0) = 0 **correcto**

- Corrigir o bit com o erro
  - caso o contador esteja igual a zero, então não houve erro
  - caso contrário, trocar o valor do bit indicado pelo contador