Autenticação baseada em *passwords*

Notas para a UC de “Segurança Informática”

Inverno de 11/12

Pedro Félix (pedrofelix em cc.isel.ipl.pt)

Instituto Superior de Engenharia de Lisboa

Identificação e Autenticação

- Autenticação é o processo de verificação duma alegada identidade

- Motivação
  - Parâmetro para as decisões de controlo de acessos
  - Parâmetro para as acções de personalização
  - Informação de auditoria

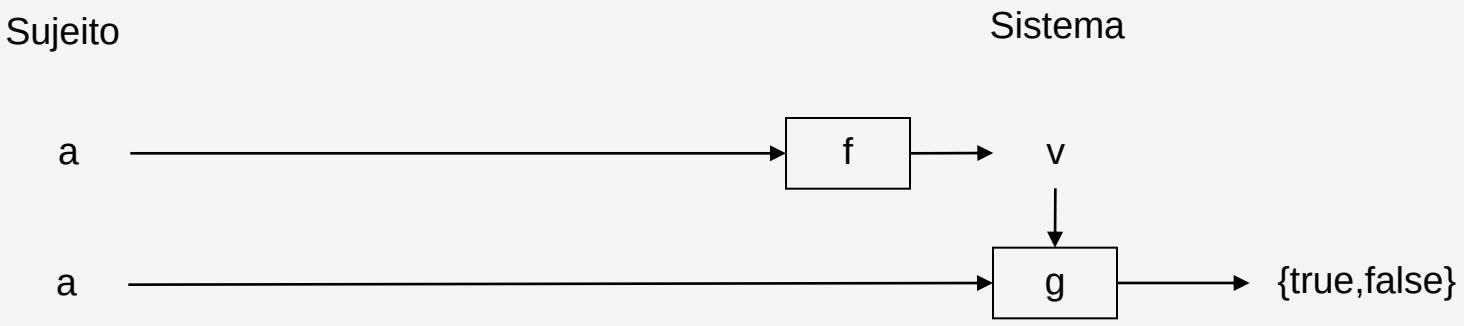
- Exemplo
  - “user” + “password”
    - “user” – identificação
    - “password” - autenticação

Informação de autenticação

- “Algo” que se conhece
  - “Passwords” e “passphrases”
- “Algo” que se possui
  - Ex.: “tokens” criptográficos, RSA SecurID
- “Algo” que se é
  - Ex.: características biométricas
- “Algo” que se faz
  - Ex.: assinatura manual
- “Onde” se está

Sistema de autenticação

- Formalização [Bishop]
  - Conjunto **A** de informação de autenticação
  - Conjunto **V** de informação de validação
  - Função **f: A → V**
  - Função **g: V → A → {true, false}**
- Exemplo
  - **f(a) = H(a)**
  - **g(v)(a) = (v = H(a))**



“Passwords”: ataques de dicionário

- Ataques do tipo 1
  - Entrada: informação de validação - **v**
  - Saída: informação de autenticação
  1. Para cada **a’** pertencente a **Dicionário**
     1. Se **f(a’) = v** retornar **a’**
  2. Retornar “falha”

- Ataques do tipo 2
  - Entrada: função de autenticação – **g(v)**
  - Saída: informação de autenticação
  1. Para cada **a’** pertencente a **Dicionário**
     1. Se **g(v)(a’) = true** retornar **a’**
  2. Retornar “falha”

Ameaça: pesquisa de “passwords”

- Pesquisa exaustiva – testar todas as “passwords” possíveis por ordem arbitrária

- Pesquisa “inteligente” – testar primeiro as “passwords” pertencentes mais prováveis
  - Dicionários
    - Palavras
  - Algoritmos
    - Combinação de palavras

Protecção contra ataques de dicionário

- Aumentar a incerteza da “password”
  - Passwords aleatórias
  - Selecção proactiva
  - Verificação *offline*
- Controlar o acesso à informação de verificação
- Aumentar o tempo de processamento da função **f**
- Aumentar o tempo de processamento ou limitar o acesso à função **g(v)**

Geração aleatória de “passwords”

- Geração aleatória de “passwords” – maximização do tempo de pesquisa.

- Problema: dificuldade de memorização
  - Escolher e memorizar transformação **t: A $\rightarrow$ A**
  - Gerar de forma aleatória e armazenar x
  - Usar **t(x)** como informação de autenticação

Selecção proactiva de “passwords”

- A “password” é escolhida pelo utilizador, contudo a sua *segurança* é verificada pelo sistema
- A aceitação da “password” está dependente da verificação realizada pelo sistema
- “Passwords” consideradas *inseguras*
  - Variantes do nome do utilizador
  - Variantes do nome do computador
  - Palavra presente em dicionários
    - Ao contrário
    - Letras maiúsculas
  - Padrões do teclado
  - Acrónimos
  - Números associados ao utilizador: BI, NC, Matrícula

Verificação *offline*

- Utilização de ferramentas para a realização de ataques de dicionário
  - Ataque às passwords dos utilizadores do sistema
  - Notificação/bloqueio dos utilizadores com “passwords” atacadas com sucesso

- Exemplos
  - L0phtCrack
  - http://lasecwww.epfl.ch/~oechslin/projects/ophcrack/index.php

Protecção contra ataques tipo 1

- Tornar a execução da função **f** mais demorada
- Exemplo
  - **f = H**, onde **H** é uma função de *hash*
  - Solução: $\mathbf{f = H^R}$
- Controlar o acesso à informação de verificação

Ataques com pré-computação

- Baseai-se no facto da função **f** ser igual para todos os utilizadores
- Seja **D** um dicionário de palavras prováveis e **M** um *array* associativo
- Pré-computação
  - Para todos **a'$_i$** em **D**, calcular e armazenar o par **(f(a'$_i$), a'$_i$)** em **M** (tal que **M[f(a'$_i$)] = a'$_i$**)

- Ataque
  - Dado **v**, retornar **M[v]**

- A pré-computação é usada para obter a "password" de qualquer utilizador

Protecção: “salt”

- Protecção contra os ataques de dicionário descritos anteriormente
- Solução: tornar a função **f** diferente para cada utilizador
- Exemplo: $\mathbf{f_U(a) = H(salt_U \mid a)}$, onde
  - $\mathbf{f_U}$ é a função associada ao utilizador **U**
  - $\mathbf{salt_U}$ é uma sequência de “bytes” gerada aleatoriamente para cada utilizador
- Neste cenário, a pré-computação depende de **salt**
  - é específica de cada utilizador do sistema
  - não pode ser utilizada para atacar todos os utilizadores do sistema

Protecção contra ataques tipo 2

- Limitar o acesso à função de autenticação **g(v)** após a detecção de tentativas de autenticação erradas
- Técnicas
  - *Backoff*
    - O tempo de execução de **g(v)** depende do número anterior de tentativas erradas
  - Terminação da ligação
    - Terminação da ligação em caso de erro
  - Bloqueamento
    - Bloqueamento da função **g(v)** após um número de tentativas erradas
  - *Jailing*
    - Acesso ao serviço com funcionalidade limitada
- Problema: garantir a disponibilidade do serviço

“Password” *aging*

- Limitar o tempo de utilização duma password
  - Tornar o tempo de utilização menor que o tempo médio de pesquisa

- Aspectos de implementação/parametrização
  - Memorização das “passwords” anteriores
  - Tempo mínimo de utilização das “passwords”
  - Tempo máximo para o utilizador proceder à mudança da “password”

Exemplo: Windows XP

- Local Security Policy/Security Settings/Account Policies/Password Policy
  - “Enforce password history” – impedir a reutilização de “passwords”
  - “maximum password age” – tempo de vida máximo da “password”
  - “minimum password age” – tempo de vida mínimo da “password”
  - “password must meet complexity requirements” – selecção proactiva
    - Não pode conter parte do nome do utilizador
    - Dimensão mínima de 6 caracteres
    - Conter caracteres de 3 de 4 conjuntos: A-Z, a-z,0-9 e alfanumérico ($,!,#,%)

Exemplo: Windows XP - filtros de “passwords”

- Um filtro é uma DLL exportando as seguintes funções
  - Iniciação
    BOOLEAN InitializeChangeNotify(void);
  - Verificação
    BOOLEAN PasswordFilter(PUNICODE_STRING *AccountName*, PUNICODE_STRING *FullName*, PUNICODE_STRING *Password*, BOOLEAN *SetOperation*);
  - Notificação
    NTSTATUS PasswordChangeNotify(PUNICODE_STRING *UserName*, ULONG *RelativeId*, PUNICODE_STRING *NewPassword*);

- Instalação
  - Colocar referência à DLL em “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Notification Packages”

Exemplo: Windows XP

- .../Account Lockout Policy
  - “Account lockout duration” – período em que o acesso é impedido após um determinado número de tentativas falhadas
  - “Account lockout threshould” – número de tentativas falhadas que resultam no impedimento do acesso
  - “Reset account lockout counter” – período de tempo após o qual é reiniciado o contador de tentativas falhadas

- Local Security Policy/Security Settings/Local Policies/Audit Policy
  - “Audit logon events”

Exemplo: Unix - Crypt

- **char *crypt(const char *_key_, const char *_salt_);**

- O Unix usa função de *hash* **Crypt** baseada na primitiva DES
  - A permutação de expansão **E** depende de **salt**
  - Entrada é usada como chave
  - Bloco inicial é 0x00...00
  - Processo repetido 25 vezes
- O ficheiro **etc/passwd**
  - Contém o *hash* da “password” de cada utilizador
  - Controla os acessos de escrita

- Novas “versões” do Unix usam métodos diferentes
  - **etc/shadow** – apenas “root” tem acesso de leitura
  - PAM (*Pluggable Authentication Module*)

Exemplo: Windows NT

- LAN Manager
  - **H(pw) = E(pw1)(C1) | E(pw2)(C2)**
  - **E** é a função de cifra da primitiva DES
  - **C1** e **C2** são constantes
  - **pw1** e **pw2** são duas chaves obtidas a partir da extensão/truncamento de **pw**, previamente convertida para “uppercase”
- NT
  - **H(pw) = MD4(pwuc)**
  - **pwuc** é a codificação de **pw** em unicode
- Exemplo (obtido através do utilitário **pwdump2**)
  - tuser1:1028:**91c7ae7122196b5eaad3b435b51404ee**:**22315d6ed1a7d5f8a7c98c40e9fa2dec**:: :
  - tuser2:1029:**91c7ae7122196b5eaad3b435b51404ee**:**61ba88d2bfe9b2e0fcff869e2fb5265c**:::

- *Local Security Settings/... / ... Do not store LAN Manager hash value on next password change*

Ameaça: *spoofing attacks*

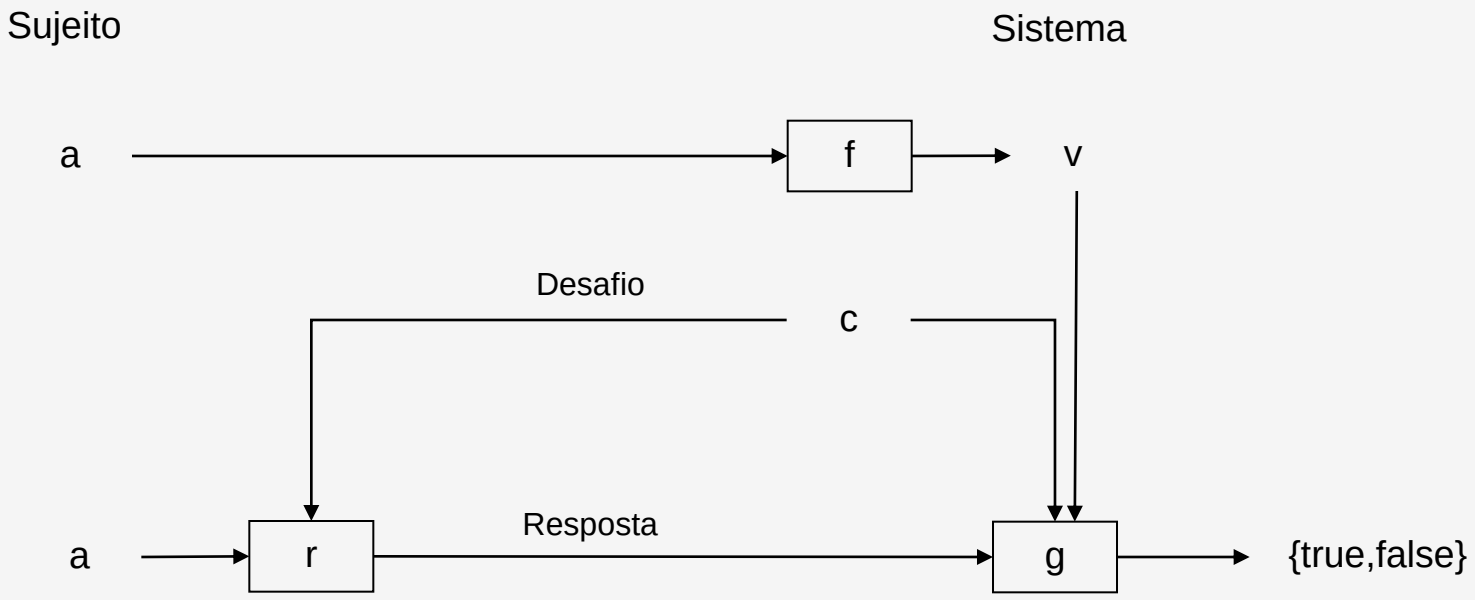
- Obtenção da “password” através da simulação da interface de autenticação ou intercepção desta

- Prevenção
  - “Trusted path” (ex. *Secure Attention Sequence* no Windows)
  - Autenticação mútua

- Detecção
  - Registo e apresentação do número e data das autenticações falhadas

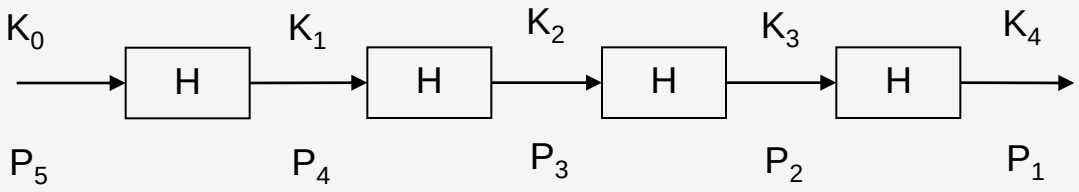
Protocolos Desafio-Resposta

- Conjunto C de desafios
- Função de resposta **r: A x C → A**
- Função de autenticação **g: V x C → A → {true, false}**



“Passwords” de utilização única

- S/Key
- Dado uma semente $\mathbf{K_0}$ e uma função de *hash* **H**
  - $\mathbf{K_i = H(K_{i-1})}$
  - $\mathbf{P_i = K_{n-i+1}}$
  - Desafio: índice i (crescente e usado apenas uma vez)
  - Resposta: $\mathbf{P_i}$



Sistemas biométricos

- Usados para identificação e autenticação
- Exemplos:
  - Face, geometria da mão, íris, voz, impressão digital, ...
- Caracterização
  - Universalidade
  - Capacidade de distinção
  - Permanência
  - Recolha
  - Desempenho
  - Aceitabilidade
  - Falsificação
- Erros
  - Identificação
  - Falsa aceitação
  - Falsa rejeição

Comparação

	Universalidade	Distinção	Permanência	Recolha	Desempenho	Aceitabilidade	Falsificação
Face	H	L	M	H	L	H	H
Impressão Digital	M	H	H	M	H	M	M
Geometria da mão	M	M	M	H	M	M	M
Íris	H	H	H	M	H	L	L
Retina	H	H	M	L	H	L	L
Voz	M	L	L	M	L	H	H

Adaptado de: D. Maltoni et al, “Handbook of Fingerprint Recognition”, Springer, 2003