

Ana Correia - 31831

Diogo Cardoso - 32466

João Silvestre - 32766

29/11/11

*Instituto Superior Engenharia de Lisboa*

**3ª Serie**

Engenheiro: José Simão

Engenharia Informática e de Computadores

Semestre de Inverno 2011/2012

Índice

[Exercício 1 4](#_Toc313893656)

[Alínea 1 4](#_Toc313893657)

[Alínea 2 4](#_Toc313893658)

[Alínea 3 4](#_Toc313893659)

[Alínea 4 4](#_Toc313893660)

[Exercício 2 5](#_Toc313893661)

[Alínea 1 5](#_Toc313893662)

[Alínea 2 5](#_Toc313893663)

[Alínea 3 5](#_Toc313893664)

[Exercício 3 6](#_Toc313893665)

[Alínea 1 6](#_Toc313893666)

[Alínea 2 6](#_Toc313893667)

[Alínea 3 6](#_Toc313893668)

[Alínea 4 6](#_Toc313893669)

[Alínea 5 7](#_Toc313893670)

[Exercício 4 8](#_Toc313893671)

[Alínea 2 8](#_Toc313893672)

[Alínea 3 8](#_Toc313893673)

[Alínea 4 8](#_Toc313893674)

[Vista 8](#_Toc313893675)

[Criação do role 8](#_Toc313893676)

[Atribuição das permissões 9](#_Toc313893677)

[Exercício 5 10](#_Toc313893678)

[Alínea 1 10](#_Toc313893679)

[Alínea 2 10](#_Toc313893680)

[Bibliografia 11](#_Toc313893681)

# Exercício 1

## Alínea 1

A diferença ente o “**dono de recursos**” e o “**cliente**” dada pela especificação do protocolo é, o “dono de recursos” é a entidade capaz de autorizar o acesso a um recurso protegido. O “cliente” é a aplicação que acede ao recurso protegido em nome do dono do recurso e com a sua autorização.

## Alínea 2

O perfil “***web application***” descreve aplicações em que a interacção feita com o *authorization server* para se obter o *access token* é feita pelo servidor web, longe da máquina do dono do recurso. Neste perfil o cliente web inicia o pedido de autorização que irá redireccionar o browser do dono do recurso para o *authorization server*, incluindo o seu identificador de cliente, um uri para para onde o *authorization server* deve redireccionar o browser do dono do recurso quando este tomar a sua decisão, entre outros dados. Este irá então autorizar ou negar o acesso do cliente ao recurso, caso autorize, o *authorization server* irá redireccionar então o browser do dono do recurso para o cliente e a obtenção do *access token* irá então ser feita no servidor web do cliente.

O perfil “***user-agent-based application***” descreve aplicações implementadas num browser, tipicamente em linguagens de scripting (JavaScript). Nestas aplicações, não é possível manter o segredo do cliente secreto, pois uma qualquer aplicação na máquina do dono do recurso poderia ter acesso ao segredo do cliente, portanto a autenticação do cliente é feita numa política de “*same-origin”*. Neste perfil o *access token* vem no *redirect* do *authentication server* e é acedido apenas no *browser* do dono do recurso.

## Alínea 3

Enquanto o “*access token*” é utilizado pelo cliente para fazer pedidos autênticados em nome do dono do recurso o “*authorization token”* é um de 4 tipos de “*access grants”* que representam a intenção do dono do recurso de dar ao cliente acesso para o recurso protegido, estes *“access grants*” podem depois ser usados para obter o “*access token*”.

## Alínea 4

Um atacante poderia conseguir alterar o *redirect* URI do *authorization token* ou no caso de um perfil “*user-agent*” poderia conseguir alterar o redirect do *access token*. Uma das formas previstas para impedir este ataque passa por permitir um registo de *redirect* URI, desta forma o atacante nunca poderia mudar o URI para algo seu. No primeiro caso, também existe outra protecção pois para obter um *access token* através do *authorization token* é preciso conhecer o segredo do cliente, que deverá estar bem seguro no servidor web do cliente.

# Exercício 2

## Alínea 1

KMAI pode emitir o seguinte certificado:

* KMAI Freguesia→ KMAI Concelho Freguesia

## Alínea 2

Para substituir o certificado a) e o da alínea anterior deve ser emitido o seguinte certificado:

* KMAI Eleitor → KMAI Concelho Freguesia Eleitor

## Alínea 3

* KMAI Eleitor → **KMAI Freguesia** Eleitor
  + **KMAI Freguesia**→ KMAI Concelho Freguesia
* KMAI Eleitor → **KMAI Concelho** Freguesia Eleitor
  + **KMAI Concelho** → KLisboa
* KMAI Eleitor → **KLisboa Freguesia** Eleitor
  + **KLisboa Freguesia** → KLapa
* KMAI Eleitor → **KLapa Eleitor**
  + **KLapa Eleitor** → KAlice
* KMAI Eleitor → KAlice

# Exercício 3

## Alínea 1

Os ataques de *buffer overflow* têm como alvo programas que apresentam falhas nas escritas de buffers e permitem assim ultrapassar os limites do buffer e escrever na memória adjacente. Estes ataques têm como objectivo atingir um, ou mais, dos seguintes alvos:

* O endereço de retorno;
* O endereço base da stack da função anterior;
* Ponteiros de funções;
* Buffers de *longjmp*.

O ataque de ponteiros de funções acontece quando o atacante consegue injectar código e modificar um ponteiro de uma função, quando o programa saltar para a função irá ser executado o código do atacante com os mesmos privilégios que o processo atacado.

## Alínea 2

Ao alterar o “*old base pointer*” o atacante pode mudar a stack da função para a qual vai ser retomado o controlo para um stack simulado por este. Este pode não só alterar os valores que a função precisará de consultar para uns que este precise mas também pode mudar o endereço de retorno dessa função para um de uma função que ele tenha conseguido injectar, dando mais uma vez controlo da aplicação ao atacante.

## Alínea 3

Esta dependência aparece com a apresentação, por parte de Emsi, de um ataque que invalidava as duas soluções iniciais (sequência de terminadores ou número aleatório) ambas independentes de qualquer valor. Ao criar uma dependência do canário sobre o valor de retorno não é possível alterar o valor de retorno sem que a validação do canário falhe, pois foi determinado que esta solução apenas seria usada com valores aleatórios. Portanto, a menos que o atacante em *runtime* descobrisse o valor do canário, este não poderia mudar o valor de retorno.

A implementação desta solução passa por, em cada execução, gerar um valor aleatório para o canário e fazer um XOR com o valor de retorno, guardando o resultante na *stack* entre o valor de retorno e o *old base pointer*. Sempre que o retorno é feito é necessário primeiro verificar que o valor do canário permanece intacto, isto previne portanto que alterem apenas o endereço de retorno ou que alterem toda a *stack frame* de uma função.

## Alínea 4

A abordagem da biblioteca Libsafe passa por criar *wrappers* para as funções potencialmente perigosas e nesses *wrappers* verificar se a função irá ultrapassar os limites do *buffer*, limites estes calculados usando o *old base pointer*. Se uma função dessas for escrever tanto que substitua o *old base pointer* então a biblioteca regista este evento e para a execução do processo. Como a verificação é feita com o *old base pointer* esta biblioteca apresenta uma grande limitação, *overflow* das variáveis locais não é protegido permitindo assim um dos ataques, a substituição de um ponteiro de uma função por outro.

## Alínea 5

Esta técnica apenas prevê a execução de código na *stack*, mas o código pode ser injectado em qualquer outra parte da memória (*heap*, *bss*, *data*), e também é possível injectar e executar código através de bibliotecas que já se encontram no sistema. Mas a maior limitação vem do facto que esta protecção é implementada como um *patch* do *kernel*, em Linux, sendo então uma opção do utilizador, se instala ou não este *patch*, e não do programador do software.

# Exercício 4

Os scripts com o código presente nestas alíneas encontram-se em anexo.

## Alínea 2

1. GRANT SELECT ON Aluno TO Ana
2. REVOKE SELECT ON Aluno TO Ana
3. CREATE ROLE Aluno  
   EXEC sp\_addrolemember 'Aluno', 'Ana'  
   EXEC sp\_addrolemember 'Aluno', 'Bernardo'  
   EXEC sp\_addrolemember 'Aluno', 'Carlos'  
   EXEC sp\_addrolemember 'Aluno', 'Diogo'  
   EXEC sp\_addrolemember 'Aluno', 'Elsa'  
   EXEC sp\_addrolemember 'Aluno', 'Fernanda'  
   GRANT SELECT ON Aluno TO Aluno
4. DENY SELECT ON Aluno TO Ana
5. REVOKE SELECT ON Aluno TO Ana

## Alínea 3

CREATE VIEW CurrentUserData AS SELECT Numero, InfoPessoal.Nome, Disciplina, Responsavel, Nota FROM

(SELECT \* FROM Aluno WHERE Nome = CURRENT\_USER) As InfoPessoal

INNER JOIN

(SELECT \* FROM Disciplina INNER JOIN Inscricao ON Disciplina.Nome = Inscricao.Disciplina) AS Notas

ON InfoPessoal.Numero = Notas.Numero\_Aluno

GO

GRANT Select ON CurrentUserData TO Aluno

REVOKE Select ON Aluno TO Aluno

## Alínea 4

### Vista

CREATE VIEW ProfessorsData AS SELECT Professor.Nome AS Professor, Inscricao.Disciplina, Aluno.Nome AS Aluno, Inscricao.Nota FROM

Professor INNER JOIN Disciplina INNER JOIN Inscricao INNER JOIN Aluno

ON Inscricao.Numero\_Aluno = Aluno.Numero

ON Disciplina.Nome = Inscricao.Disciplina

ON Professor.Numero = Disciplina.Responsavel

### Criação do role

CREATE ROLE Professor

EXEC sp\_addrolemember 'Professor', 'Joao'

EXEC sp\_addrolemember 'Professor', 'Maria'

### Atribuição das permissões

GRANT Select, Update (Nota) ON ProfessorsData TO Professor

# Exercício 5

Os scripts com o código presente nestas alíneas encontram-se em anexo.

## Alínea 1

EXEC AS USER = 'Ana';

EXEC p\_notas\_anonimas '''; GRANT CONTROL TO Aluno--'

REVERT

## Alínea 2

CREATE PROCEDURE p\_notas\_anonimas

@disciplina sysname = NULL,

@sexo sysname = NULL

WITH EXECUTE AS OWNER

AS

BEGIN

DECLARE @query nvarchar(max)

DECLARE @ParmDef nvarchar(max)

SET @query = N'SELECT dbo.Inscricao.Nota ' +

N'FROM dbo.Inscricao INNER JOIN dbo.Aluno ON dbo.Aluno.Numero = dbo.Inscricao.Numero\_Aluno ' +

N'WHERE 1=1'

SET @ParmDef = N''

IF @disciplina IS NOT NULL

BEGIN

SET @query = @query + N' AND dbo.Inscricao.Disciplina=@disciplina'

SET @ParmDef = @ParmDef + N'@disciplina nchar(10)'

END

IF @sexo IS NOT NULL

BEGIN

SET @query = @query + N' AND dbo.Aluno.Sexo=@sexo'

IF @disciplina IS NOT NULL

SET @ParmDef = @ParmDef + N', '

SET @ParmDef = @ParmDef + N'@sexo char(1)'

END

IF @disciplina IS NOT NULL AND @sexo IS NOT NULL

EXECUTE sp\_executesql @query, @ParmDef, @disciplina, @sexo

ELSE IF @disciplina IS NOT NULL

EXECUTE sp\_executesql @query, @ParmDef, @disciplina

ELSE IF @sexo IS NOT NULL

EXECUTE sp\_executesql @query, @ParmDef, @sexo

ELSE

EXECUTE sp\_executesql @query, @ParmDef

END

GO

# Bibliografia

(s.d.). Obtido em 8 de 1 de 2012, de http://tools.ietf.org/html/draft-ietf-oauth-v2-10#page-5

(s.d.). Obtido em 8 de 1 de 2012, de SpringSource: http://blog.springsource.org/2011/11/30/10317/

Wilander, J., & Kamkar, M. (s.d.). A Comparison of Publicly Available Tools for Dynamic Buffer Overflow.