**Questões de Segurança em Engenharia de Software (QSES), 2019/20**
Mestrado em Segurança Informática
Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
**Exame de época normal – 21/01/2020**
**Duração: 2:30**

## Grupo A

Considere o Fragmento 1 de código Java abaixo, inspirado na aplicação Java Vulnerable Lab. O código relaciona-se com a validação de "login" para um utilizador. Assuma que:
— o input do nome do utilizador (user) pode ser malicioso;
— o conteúdo da tabela USERS pode ser malicioso;
— o método hash calcula o valor de hash para uma password, sendo este armazenado na coluna PHASH da tabela USERS;

**Fragmento de código 1:**

```
1   String user = request.getParameter("user");
2   String pass = request.getParameter("pass");
3   try {
4     Connection db = ... ;
5     Statement stmt = db.createStatement();
6     ResultSet rs = stmt.executeQuery("SELECT PHASH,NAME FROM USERS WHERE USER=" + user);
7     if (rs.next() && rs.getString(1).equals(hash(pass))) {
8       out.println("Hello " + rs.getString(2));
9       ...
10    }  else {
11      out.println("Login failure for user: " + user);
12      ...
13    }
14  }
15  catch (SQLException e) {
16    out.println("Database error!<br/>");
17    e.printStackTrace(out);
18  }
```

**Questões:**

**1. [5 valores]** Identifique as vulnerabilidades no fragmento de código, explicando para cada uma:

- o tipo de vulnerabilidade e de que forma pode ser explorada;
- as alterações necessárias ao código para eliminá-la — pode usar pseudo-código aproximado se não se lembrar de alguns detalhes, desde que o significado seja claro.

**2. [2.5 valores]** Explique em que consistem ataques do tipo "cross-site request forgery" (CSRF) e como o uso de "anti-CSRF tokens" podem mitigar ataques desse tipo. Explique também de forma sucinta um mecanismo possível para a implementação de "anti-CSRF tokens".

**Grupo B**

Considere o Fragmento 2 de código em C abaixo. A função processData lê successivamente identificadores de dados (id) e usa-os para processar ficheiro de dados (id.txt) na pasta PATH_FOR_FILES. Assuma que PATH_FOR_FILES é uma string constante definida que pode ter um tamanho arbitrário.

**Fragmento de código 2:**

```
1    char id[32];  // buffer para identificador
2    char* path;   // buffer para path de um ficheiro
3    FILE* file;   // ficheiro
4    while(1) {
5     gets(id);      // lê identificador
6     printf(id);    // imprime identificador
7     if (strcmp("quit",id) == 0)
8       break; // sái do ciclo
9     path = (char*) malloc(128); // aloca memória para path do ficheiro
10    sprintf(path, "%s/%s.txt", PATH_FOR_FILES, id); // define path do ficheiro
11    file = fopen(path, "r"); // abre ficheiro para leitura
12    if (file != NULL) {      // testa se abertura foi bem sucedida
13      ...                    // processa ficheiro de alguma forma
14      fclose(file);          // fecha ficheiro
15    } else {
16      printf("File cannot be read!\n"); // imprime mensagem de erro
17    }
18    free(path);  // liberta memória alocada para path
19    printf("Done processing %s\n", path);  // imprime mensagem
20  }
```

**Questões:**

**1. [5 valores]** Identifique as vulnerabilidades no fragmento de código, explicando para cada uma:

- o tipo da vulnerabilidade e o seu possível efeito durante a execução;
- as alterações necessárias ao código para eliminá-la.

**2. [2.5 valores]** Explique em traços gerais a estratégia para um ataque do tipo "stack-smashing" no código (original), assumindo que não existe qualquer tipo de mecanismo de proteção no uso da stack ou espaço de endereçamento. Indique de seguida se são necessárias variações do ataque face a (1) uso de ASLR, (2) uso de canários na "stack", ou (3) "NX-bit" que desabilita código executável na stack.

**Grupo C**

Dê respostas claras e sucintas às seguintes perguntas.

**1. [2 valores]** Em relação ao uso de "fuzz testing":

(a) Qual é o objectivo geral?

(b) Indique três técnicas base empregues na geração de valores.

(c) Qual é a distinção entre as variantes "black-box" e "white-box" de "fuzz testing" ?

**2. [1 valores]** Uma empresa de software faz bom uso de várias mecanismos de segurança (ex. uso de criptografia nas comunicações, esquemas avançados de autenticação, etc), e está bastante confiante na segurança das suas aplicações . Qual é a falácia deste raciocínio?

**3. [1 valores]** O que são "zero-day exploits"? Porque são particularmente gravosos? Faça a relação com o conceito de janela de vulnerabilidade ("vulnerability window").

**4. [1 valores]** Indique uma vantagem e uma desvantagem do uso de ferramentas de análise estática face a ferramentas de "pen-testing" para validação da segurança de uma aplicação.