**Questões de Segurança em Engenharia de Software (QSES), 2017/18**
Mestrado em Segurança Informática
Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
**Exame de 1ª época – 15/01/2018**
**Duração: 2:30**

## Grupo A

Considere o seguinte fragmento de código, adaptado da aplicação Java Vulnerable Lab. O código relaciona-se com a apresentação do conteúdo de uma entrada numa tabela de mensagens (posts). Pode assumir que o conteúdo da tabela não foi alvo de qualquer validação ou sanitização.

```
Connection db = ... ;
String postid=request.getParameter("postid");
Statement stmt = db.createStatement();
ResultSet rs =
   stmt.executeQuery("SELECT * FROM posts WHERE postid="+postid);
out.println("Title:"+ rs.getString("title") + "<br/>");
out.println("Posted by: " + rs.getString("user") + "<br/>");
out.println("Content:" + rs.getString("content") + "<br/>");
```

**1. [3 valores]** Indique e explique o tipo de vulnerabilidades de segurança que existem no código da aplicação. Dê também exemplos de valores para o parâmetro postid e/ou de conteúdo na base de dados que explorem essas vulnerabilidades.

**2. [3 valores]** De que forma o código pode ser acertado para mitigar as vulnerabilidades? Indique as alterações necessárias, ilustrando as mesmas com o correspondente código (desde que a ideia fique clara, pode usar pseudo-código aproximado se não se lembrar de todos os detalhes).

**3. [2 valores]** Suponha que queremos proteger a aplicação contra ataques do tipo Cross-Site Request Forgery (CSRF). Explique em que consistem ataques deste tipo, e indique duas medidas que podem ser tomadas em termos programação da aplicação para mitigá-los. Não precisa de explicar de que forma o código seria alterado, mas apenas a natureza das medidas em si.

## Grupo B

Considere o seguinte fragmento de código em C.

```
void someFunc(char* param) {
  char localVar[64];
  ...
  strcpy(localVar, param); // copia string param para localVar
  ...
}
```

**1. [1 valores]** Indique sucintamente três razões pelas quais código escrito em C é particularmente sujeito a ataques baseados em "buffer overflows".

**2. [2.5 valores]** Explique porque é que o código acima pode ser alvo de um ataque baseado num "stack overflow" e de que forma este pode ser explorado para uma quebra de segurança.

**3. [1.5 valores]** Indique três mecanismos de protecção contra um ataque deste tipo, explicando sucintamente como funcionam.

**Grupo C**

Considere o seguinte método isValidDate em Java para validar o input de uma aplicação em termos de datas no formato YYYY−MM−DD. O método usa a API Java de expressões regulares e chama outro método auxiliar, daysInMonth, para calcular o número de dias num mês de determinado ano.

```
1   private static final Pattern DATE_PATTERN
2     = Pattern.compile("(\\d{4})-(\\d{2})-(\\d{2})");
3   static boolean isValidDate(String s) {
4     // Verifica que string obedece a formato da expressão regular.
5     Matcher re = DATE_PATTERN.matcher(s);
6     if (! re.matches() )
7       return false;
8     // Obtém valores
9     int year = Integer.parseInt(re.group(1));
10    int month = Integer.parseInt(re.group(2));
11    int day = Integer.parseInt(re.group(3));
12    // Valida valores numéricos
13    return month >= 1 &&
14           month <= 12 &&
15           day >= 1 &&
16           day <= daysInMonth(month, year);
17  }
18  private static int daysInMonth(int m, int y) {
19    if (m == 2) {
20      return y % 400 == 0 || (y % 4 == 0 && y % 100 != 0) ? 29 : 28;
21    }
22    if (m <= 7) {
23      return m % 2 == 1 ? 31 : 30;
24    }
25    return m % 2 == 1 ? 30 : 31;
26  }
```

**1. [0.5 valores]** O método em causa implementa um esquema de validação do tipo "white-list". Porque é normalmente preferível essa opção a um esquema de validação "black-list" ?

**2. [1 valores]** Suponha que adoptávamos uma estratégia de "fuzz testing" para gerarmos inputs para testes sobre isValidDate. Porque é que poderia ser uma abordagem interessante e que tipos de valores de input poderiam ser obtidos para testes?

**3. [2.5 valores]** Considerando uma abordagem baseada em "mutation testing", e tendo em conta as seguintes mutantes sobre o código, identifique para cada mutante um teste que o mate. Justifique a sua escolha em cada caso.

| | | | |
|---|---|---|---|
| $M_1$ (linha 11) | re.group(3) | $\rightarrow$ | re.group(2) |
| $M_2$ (linha 13) | month >= 1 | $\rightarrow$ | month > 1 |
| $M_3$ (linha 16) | day <= daysInMonth(month,year) | $\rightarrow$ | day <= 31 |
| $M_4$ (linha 19) | m == 2 | $\rightarrow$ | m != 2 |

**Grupo D**

Dê respostas claras e sucintas às seguintes perguntas.

**1. [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.

**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 possível 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").