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 2ª época – 27/01/2018
Duração: 2:30

Grupo A

Considere o seguinte fragmento de código, adaptado de uma servlet da aplicação Java Vulnerable Lab. O código relaciona-se com a validação da senha de um utilizador.

```
... void processRequest
(HttpServletRequest request, HttpServletResponse response) ... {
    Connection conn = ... ;
    String user = request.getParameter("username");
    String pass = request.getParameter("password");
    Statement stmt = con.createStatement();
    ResultSet rs =
      stmt.executeQuery("select * from users where username='"
         + user + "' and password='" + pass + "'");
    if (rs.next()){
       Cookie username = new Cookie("username", user);
       Cookie password = new Cookie("password", pass);
       response.addCookie(username);
       response.addCookie(password);
       ...
    } else {
      out.println(user + ": invalid password");
      ...
    }
```

1. [3.5 valores] Indique e explique o tipo de vulnerabilidades de segurança que existem no código da aplicação. Para as vulnerabilidades que dependam de valores de input à servlet, exemplifique valores de input malicioso.

2. [3.5 valores] De que forma o código pode ser acertado para mitigar as vulnerabilidades? Indique as alterações necessárias, ilustrando as mesmas com o correspondente código. Se não se lembrar de todos os detalhes, pode usar pseudo-código aproximado, desde que a ideia fique clara.

3. [1.5 valores] Para *uma* das vulnerabilidades identificadas acima explique os tipos de ameaças que acha que se podem materializar segundo a taxonomia STRIDE[1].

4. [1 valores] Explique como é que uma ferramenta de análise estática poderia detectar pelo menos uma das vulnerabilidades empregando "taint analysis".

Grupo B

Dê respostas claras e sucintas às seguintes perguntas.

1. [1 valores] Porque é especialmente complexa a validação de um sistema construído a partir de um conjunto de componentes de que executam de forma concorrente (ex. programas com múltiplas "threads")?

2. [1 valores] Uma empresa de software tem um processo de software que é bastante intenso e em termos de validação de requisitos funcionais de uma aplicação por forma a garantir uma alta fiabilidade. Porque é que isso não significa necessariamente um alto grau de segurança?

[1] S: Spoofing; T: Tampering; R: Repudiation; I: Information disclosure; D: Denial of service; E: Escalation of previlege

Grupo C

Considere o seguinte fragmento de código em C.

```
void processFile(char* baseName) {
  char fileName[128];
  //  Define caminho completo para ficheiro.
  sprintf(fileName, "/home/qses/data/%s.txt", baseName);
  if (access(filename, R_OK) == 0) { // Testa se pode ser lido
    // Abre ficheiro para leitura.
    FILE* fp = fopen(filename, "r");
    ... // lê ficheiro
    // Fecha ficheiro
    fclose(f);
  }
}
```

1. [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.

2. [1.5 valores] Indique três mecanismos de protecção contra um "stack overflow", explicando sucintamente como funcionam.

3. [1 valores] Explique porque é que o código tem uma vulnerabilidade TOCTOU ("Time Of Check - Time Of Use") e de que forma esta pode ser explorada.

Grupo D

Considere o seguinte método isValidTime em Java para validar o input de uma aplicação em termos de horas no formato HH:MM:SS.

```
1  private static final Pattern HOUR_PATTERN
2    = Pattern.compile("(\\d{2}):(\\d{2}):(\\d{2})");
3  static boolean isValidTime(String s) {
4    // Verifica que string obedece a formato da expressão regular.
5    Matcher re = HOUR_PATTERN.matcher(s);
6    if (! re.matches())
7      return false;
8    // Obtém valores
9    int hour = Integer.parseInt(re.group(1));
10   int min  = Integer.parseInt(re.group(2));
11   int sec  = Integer.parseInt(re.group(3));
12   // Valida valores numéricos
13   return hour >= 0 && hour < 24 &&
14          min >= 0  && min  < 60 &&
15          sec >= 0  && sec  < 60;
16 }
```

1. [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 6)	! re.matches()	$\rightarrow$	true
M_2 (linha 9)	re.group(1)	$\rightarrow$	re.group(2)
M_3 (linha 14)	min < 60	$\rightarrow$	min < sec
M_4 (linha 15)	sec >= 0	$\rightarrow$	sec > 0

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