Programação Segura

Semana 9: Qualidade do Código, Encapsulamento, Ambiente, Normas

Profª Denise Goya
Denise.goya@ufabc.edu.br – UFABC - CMCC

Classes de Erros de Segurança

- Validação e representação dos dados de entrada
- Abuso de API
- Características de segurança
- Tempo e estado
- Tratamento de exceções
- Qualidade do código
- Encapsulamento
- Ambiente

Qualidade do Código

- Certas características podem indicar que o código não foi cuidadosamente desenvolvido ou mantido (premissas para um código seguro)
- Indícios de má qualidade do código podem levar a comportamentos indesejáveis
  - E podem dar a oportunidade a atacantes a uma exploração perigosa

Indícios de Má Qualidade do Código

- **Double Free**: pode ocasionar buffer overflow
- **Uso depois de Free**: pode levar a crash
- Memória (ou outro recurso) **alocada e nunca liberada**: pode ocasionar exaustão de recurso
- **Referência a endereço/objeto Null**: causa NullPointerException.
- **Obsolescência**: pode indicar código muito antigo ou sem manutenção
- **Variável não inicializada**: perigoso

Qualidade do Código: Double Free

- Muitas vezes ocorre por um tratamento indevido de situações de erro
  - ou porque se está confuso sobre que parte do programa deve fazer a liberação

Example Language: **C** *(Bad Code)*

```
char* ptr = (char*)malloc (SIZE);
...
if (abrt) {
  free(ptr);
}
...
free(ptr);
```

Qualidade do Código: Double Free

- Recomenda-se atribuir NULL ao ponteiro recém liberado
  - free(NULL) é válido: faz nada e impede problemas com um segundo free no mesmo endereço

```
free(x); x = NULL;
/* code */
free(x);
```

- A exploração da vulnerabilidade de *double free* é relativamente difícil: depende da implementação da alocação, mas há casos reais

Qualidade do Código: Double Free

- Exemplos reais catalogados:

Description **CVE-2002-0059**
The decompression algorithm in zlib 1.1.3 and earlier, as used in many different utilities and packages, causes inflateEnd to release certain memory more than once (a "double free"), which may allow local and remote attackers to execute arbitrary code via a block of malformed compression data.

Description **CVE-2005-0891**
Double free vulnerability in gtk 2 (gtk2) before 2.2.4 allows remote attackers to cause a denial of service (crash) via a crafted BMP image.

Qualidade do Código: Double Free

- Exemplos reais catalogados:

Description	CVE-2003-1048
Double free vulnerability in mshtml.dll for certain versions of Internet Explorer 6.x allows remote attackers to cause a denial of service (application crash) via a malformed GIF image.	

Description	CVE-2006-5051
Signal handler race condition in OpenSSH before 4.4 allows remote attackers to cause a denial of service (crash), and possibly execute arbitrary code if GSSAPI authentication is enabled, via unspecified vectors that lead to a double-free.	

Qualidade do Código: Uso após Free

- Similar ao Double free: muitas vezes ocorre por um tratamento indevido de situações de erro
  - ou porque se está confuso sobre que parte do programa deve fazer a liberação

Example Language: **C** *(Bad Code)*

```
char* ptr = (char*)malloc (SIZE);
if (err) {
  abrt = 1;
  free(ptr);
}
...
if (abrt) {
  logError("operation aborted before commit", ptr);
}
```

Qualidade do Código: Uso após Free

- Exemplos reais catalogados:

Description **CVE-2009-1837**

Race condition in the NPObjWrapper_NewResolve function in modules/plugin /base/src/nsJSNPRuntime.cpp in xul.dll in Mozilla Firefox 3 before 3.0.11 might allow remote attackers to execute arbitrary code via a page transition during Java applet loading, related to a use-after-free vulnerability for memory associated with a destroyed Java object.

Description **CVE-2010-1772**

Use-after-free vulnerability in page/Geolocation.cpp in WebCore in WebKit before r59859, as used in Google Chrome before 5.0.375.70, allows remote attackers to execute arbitrary code or cause a denial of service (application crash) via a crafted web site, related to failure to stop timers associated with geolocation upon deletion of a document.

Qualidade do Código: Uso após Free

- Exemplos reais catalogados:

Description **CVE-2010-2547**

Use-after-free vulnerability in kbx/keybox-blob.c in GPGSM in GnuPG 2.x through 2.0.16 allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code via a certificate with a large number of Subject Alternate Names, which is not properly handled in a realloc operation when importing the certificate or verifying its signature.

Description **CVE-2010-0050**

Use-after-free vulnerability in WebKit in Apple Safari before 4.0.5 allows remote attackers to execute arbitrary code or cause a denial of service (application crash) via an HTML document with improperly nested tags.

Qualidade do Código: Não Free

- Não liberar recurso alocado ou liberar de forma inadequada pode criar situações de vulnerabilidade

Example Language: **Java** *(Bad Code)*

```
private void processFile(string fName) {
  StreamWriter sw = new StreamWriter(fName);
  string line;
  while ((line = sr.ReadLine()) != null){
    processLine(line);
  }
}
```

Este método não fecha o arquivo. O método Finalize() de StreamReader chama Close(), mas sem garantia de quando

Qualidade do Código: Não Free

Example Language: **Java** *(Bad Code)*

```
try {
  Connection con = DriverManager.getConnection(some_connection_string);
}
catch ( Exception e ) {
  log( e );
}
```

Example Language: **C#** *(Bad Code)*

```
...
SqlConnection conn = new SqlConnection(connString);
SqlCommand cmd = new SqlCommand(queryString);
cmd.Connection = conn;
conn.Open();
SqlDataReader rdr = cmd.ExecuteReader();
HarvestResults(rdr);
conn.Connection.Close();
...
```

Em ambos: a conexão é aberta e se, posteriormente, ocorre alguma exceção, a conexão permanecerá aberta, consumindo recursos

Qualidade do Código: Não Free

- Exemplos reais catalogados:

Description CVE-1999-1127

Windows NT 4.0 does not properly shut down invalid named pipe RPC connections, which allows remote attackers to cause a denial of service (resource exhaustion) via a series of connections containing malformed data, aka the "Named Pipes Over RPC" vulnerability.

Description CVE-2005-3119

Memory leak in the request_key_auth_destroy function in request_key_auth in Linux kernel 2.6.10 up to 2.6.13 allows local users to cause a denial of service (memory consumption) via a large number of authorization token keys.

Qualidade do Código: Não Inicialização

Example Language: **PHP** *(Bad Code)*

```
if (isset($_POST['names'])) {
  $nameArray = $_POST['names'];
}
echo "Hello " . $nameArray['first'];
```

A inicialização da variável ocorre sob uma **condição**; seu uso posterior ocorre sempre

Qualidade do Código: Não Inicialização

- Exemplos reais catalogados:

Description **CVE-2008-0081**

Unspecified vulnerability in Microsoft Excel 2000 SP3 through 2003 SP2, Viewer 2003, and Office 2004 for Mac allows user-assisted remote attackers to execute arbitrary code via crafted macros, aka "Macro Validation Vulnerability," a different vulnerability than CVE-2007-3490.

Description **CVE-2007-2728**

The soap extension in PHP calls php_rand_r with an uninitialized seed variable, which has unknown impact and attack vectors, a related issue to the mcrypt_create_iv issue covered by CVE-2007-2727.

Qualidade do Código

- Mais tipos e exemplos relacionados:

http://cwe.mitre.org/data/definitions/398.html

ENCAPSULAMENTO INADEQUADO

Encapsulamento Inadequado

- Em **segurança**, encapsulamento é relacionado a limites bem definidos sobre quem controla o quê
- Ex:
  - Num navegador web, um código de uma aplicação móvel não pode ser abusado por outro código
  - Um servidor deve diferenciar corretamente entre
    - Dados válidos de dados não válidos
    - Dados de um usuário com os de outros
    - Dados que podem ser visualizados de dados que não

Encapsulamento: Erros Comuns

- **Comparar classes pelo nome**: diferentes classes podem ser tratadas como sendo a mesma
- **Vazamento de dados entre usuários**: compartilhar objetos singleton inadequadamente, vazando dados entre sessões
- **Vazamento de dados do sistema**: dados do sistema ou de depurador ajudam um atacante a planejar ataques

Encapsulamento: Erros Comuns

Erros em **código de aplicação móvel**:

- **Sequestro de objetos**: objetos Clonable criam novas instâncias sem chamar um construtor
- Uso de **Inner Class**: classes que são acessíveis no escopo do pacote podem expor código a alguém mal intencionado
- **Non-Final Public**: variáveis assim podem ser manipuladas por um atacante

Encapsulamento: Erros Comuns

- Comparação de classes pelo nome

Example Language: **Java** *(Bad Code)*

```
if (inputClass.getClass().getName().equals("TrustedClassName")) {
  // Do something assuming you trust inputClass
  // ...
}
```

Example Language: **Java** *(Good Code)*

```
if (inputClass.getClass() == TrustedClass.class) {
  // Do something assuming you trust inputClass
  // ...
}
```

Encapsulamento: Erros Comuns

- Vazamento entre sessões

Example Language: **Java** *(Bad Code)*

```
public class GuestBook extends HttpServlet {
 String name;

  protected void doPost (HttpServletRequest req, HttpServletResponse res) {
    name = req.getParameter("name");
    ...
    out.println(name + ", thanks for visiting!");
  }
}
```

Está armazenando um valor de um parâmetro de requisição na variável membro do servlet. Se fosse um ambiente monousuário, não haveria problema, mas se dois usuários acessarem o servlet em momentos muito próximos, pode ocorrer de uma thread exibir “name” da outra thread.

Encapsulamento: Erros Comuns

- Vazamento de dados do sistema

Example Language: **C** *(Bad Code)*

```
char* path = getenv("PATH");
...
sprintf(stderr, "cannot find exe on path %s\n", path);
```

Example Language: **Java** *(Bad Code)*

```
try {
  ...
} catch (Exception e) {
  e.printStackTrace();
}
```

Em caso de erro, informações do sistema serão exibidas

Encapsulamento: Mobile

- Sequestro de objeto

Example Language: **Java** *(Bad Code)*

```
public class BankAccount implements Cloneable{
  public Object clone(String accountnumber) throws
  CloneNotSupportedException
  {
    Object returnMe = new BankAccount(account number);
    ...

  }
}
```

Example Language: **Java** *(Bad Cod*

```
protected Object clone() throws CloneNotSupportedException {
  ...
}
```

A classe pública "BankAccount" implementa o método cloneable() sem deixá-lo *final*;

O método clone() abaixo também não foi declarado *final*.

Encapsulamento: Mobile

- Inner Class: é acessível ao pacote (e não somente à classe externa)

Example Language: **Java** *(Bad Code)*

```
public final class urlTool extends Applet {
  private final class urlHelper {
    ...
  }
  ...
}
```

urlHelper será acessível a outras classes do pacote

Encapsulamento: usar static Inner Class

Example Language: **Java** *(Bad Co*

```
public class OuterClass {

  // private member variables of OuterClass
  private String memberOne;
  private String memberTwo;

  // constructor of OuterClass
  public OuterClass(String varOne, String varTwo) {
    this.memberOne = varOne;
    this.memberTwo = varTwo;
  }

  // InnerClass is a member inner class of OuterClass
  private class InnerClass {
    private String innerMemberOne;

    public InnerClass(String innerVarOne) {
      this.innerMemberOne = innerVarOne;
    }

    public String concat(String separator) {
      // InnerClass has access to private member variables of
      System.out.println("Value of memberOne is: " + memberOr
      return OuterClass.this.memberTwo + separator + this.inne
    }
  }
}
```

Example Language: **Java** *(Good Code)*

```
public class OuterClass {

  // private member variables of OuterClass
  private String memberOne;
  private static String memberTwo;

  // constructor of OuterClass
  public OuterClass(String varOne, String varTwo) {

    this.memberOne = varOne;
    this.memberTwo = varTwo;
  }

  // InnerClass is a static inner class of OuterClass
  private static class InnerClass {

    private String innerMemberOne;

    public InnerClass(String innerVarOne) {
      this.innerMemberOne = innerVarOne;
    }
    public String concat(String separator) {

      // InnerClass only has access to static member variables of OuterClass
      return memberTwo + separator + this.innerMemberOne;
    }
  }
}
```

Encapsulamento Inadequado

- Mais tipos e exemplos relacionados:

http://cwe.mitre.org/data/definitions/485.html

AMBIENTE

Ambiente

- Fraquezas nessa classe de erros costumam ser introduzidas por condições de ambiente não esperadas.
- Os erros encontram-se fora do código fonte, mas podem ser críticos para a segurança do sistema
- Em geral, são específicos de plataforma ou ferramenta

Ambiente: Erros Conhecidos

- Otimização insegura em compilador
- Configuração inadequada no J2EE:
  - Transporte inseguro (sem SSL)
  - Tamanho do Session-ID insuficiente
  - É preciso tratar erros da Web (404, 500, etc)
  - Declaração insegura de Bean
  - Permissões fracas para acesso ao EJB

Ambiente: Erros Conhecidos

- Configuração inadequada no ASP .NET:
  - Criação de binário de debug (ajudam atacantes)
  - Sem tratador de erro personalizado
  - Senhas em arquivo de configuração

Ambiente: Erros Conhecidos

- Otimização insegura em compilador: memset
  - Microsoft Visual C++ .NET or GCC 3.x

Example Language: **C** *(Bad Code)*

```
void GetData(char *MFAddr) {
  char pwd[64];
  if (GetPasswordFromUser(pwd, sizeof(pwd))) {

    if (ConnectToMainframe(MFAddr, pwd)) {

      // Interaction with mainframe
    }
  }
  memset(pwd, 0, sizeof(pwd));
}
```

memset() ao final é uma forma de sobrescrever o conteúdo sensível (pwd) para inutilizá-lo.

Opção de **otimização no compilador** remove a chamada para memset(), pois a área não será mais acessada no código

Ambiente: Erros Conhecidos

- Otimização insegura em compilador: memset

Potential Mitigations

Phase: Implementation

Store the sensitive data in a "volatile" memory location if available.

Phase: Build and Compilation

If possible, configure your compiler so that it does not remove dead stores.

Phase: Architecture and Design

Where possible, encrypt sensitive data that are used by a software system.

Ambiente: Erros Conhecidos no J2EE

- Transporte inseguro (sem SSL)
- Com o SSL (HTTPS) é criado um canal seguro, com autenticação, integridade e confidencialidade
- Se a aplicação Web usa SSL, então a configuração deve impedir o acesso a páginas sem SSL:
  - Se o usuário digita HTTP no lugar do HTTPS e funciona, é porque está mal configurado

Ambiente: Erros Conhecidos no J2EE

- Configuração incorreta leva a transporte inseguro
  - Ex: casos reais:

Description	CVE-2009-0152
iChat in Apple Mac OS X 10.5 before 10.5.7 disables SSL for AOL Instant Messenger (AIM) communication in certain circumstances that are inconsistent with the Require SSL setting, which allows remote attackers to obtain sensitive information by sniffing the network.	

Description	CVE-2008-4122
Joomla! 1.5.8 does not set the secure flag for the session cookie in an https session, which makes it easier for remote attackers to capture this cookie by intercepting its transmission within an http session.	

Ambiente: Erros Conhecidos no J2EE

- Tamanho do Session-ID insuficiente
- Se um atacante puder advinhar o número de sessão, pode capurá-la (sequestro de sessão)

Example Language: **XML** *(Bad Code)*

```
<sun-web-app>
  ...
  <session-config>
    <session-properties>
      <property name="idLengthBytes" value="8">
        <description>The number of bytes in this web module's session ID.</description>
      </property>
    </session-properties>
  </session-config>
  ...
</sun-web-app>
```

Ambiente: Erros Conhecidos no J2EE

- 8 bytes (64 bits) é insuficiente
- Atualmente, o ideal é pelo menos 16 bytes (128 bits)

Example Language: **XML** *(Bad Code)*

```
<sun-web-app>
  ...
  <session-config>
    <session-properties>
      <property name="idLengthBytes" value="8">
        <description>The number of bytes in this web module's session ID.</description>
      </property>
    </session-properties>
  </session-config>
  ...
</sun-web-app>
```

Ambiente: Erros Conhecidos no J2EE

- É preciso **tratar erros** da Web (erros 4xx, 5xx), caso contrário, a exibição de páginas default de erro podem alimentar um atacante com informações

Example Language: **Java** *(Bad Code)*

```
Public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
  try {
    ...
  } catch (ApplicationSpecificException ase) {
    logger.error("Caught: " + ase.toString());
  }
}
```

exceção lançada e não tratada: ideal é tratar cada erro de forma adequada

Ambiente: Erros Conhecidos no J2EE

- Declaração insegura de Bean
  - expor uma interface remota para um componente (ejb bean) pode expor métodos que leiam ou alterem dados do componente

Example Language: **XML** *(Bad Code)*

```
<ejb-jar>
  <enterprise-beans>
    <entity>
      <ejb-name>EmployeeRecord</ejb-name>
      <home>com.wombat.empl.EmployeeRecordHome</home>
      <remote>com.wombat.empl.EmployeeRecord</remote>
      ...
    </entity>
  ...
  </enterprise-beans>
</ejb-jar>
```

Ambiente: Erros Conhecidos no J2EE

- Declaração insegura de Bean
  - é preciso declarar os componentes localmente sempre que possível
  - se tiver que ser remoto, certificar-se de que não há exposição de dado sensível

Ambiente: Erros Conhecidos no J2EE

- Permissões fracas para acesso ao EJB
  - não deve ceder mais direitos que o necessário

Example Language: **XML** *(Bad Code)*

```
<ejb-jar>
  ...
  <assembly-descriptor>
    <method-permission>
      <role-name>ANYONE</role-name>
      <method>
      <ejb-name>Employee</ejb-name>
      <method-name>getSalary</method-name>
    </method-permission>
  </assembly-descriptor>
  ...
</ejb-jar>
```

o método getSalary() do componente Employee deveria ser restrito a poucos papéis

Ambiente: Erros no ASP .NET

- Criação de binário de debug (ajudam atacantes)

Example Language: **XML** *(Bad Code)*

```
<?xml version="1.0" encoding="utf-8" ?>
  <configuration>
    <system.web>
      <compilation
      defaultLanguage="c#"
      debug="true"
      />
      ...
    </system.web>
  </configuration>
```

a opção de depuração deve ser ativada para "false" antes da distribuição do aplicativo

Ambiente: Erros no ASP .NET

- Sem tratador de erro personalizado:
  - páginas default de erro vazam informações
  - não se deve configurar customError mode para “Off” pois serão retornados dados da pilha em caso de erros

Example Language: **ASP.NET** *(Bad Code)*

```
<customErrors mode="Off" />
```

Example Language: **ASP.NET** *(Good Code)*

```
<customErrors mode="RemoteOnly" />
```

Ambiente: Erros no ASP .NET

- Senhas em arquivo de configuração
  - o trecho de um arquivo de configuração abaixo contém usuário e senha em claro
  - é preciso guardá-los em arquivo cifrado

Example Language: **ASP.NET** *(Bad Code)*

```
...
<connectionStrings>
<add name="ud_DEV" connectionString="connectDB=uDB; uid=db2admin; pwd=password; dbalias=uDB;"
providerName="System.Data.Odbc" />
</connectionStrings>
...
```

Ambiente

- Mais tipos e exemplos relacionados:

http://cwe.mitre.org/data/definitions/2.html

NORMAS:

ISO/IEC 15408 – COMMON CRITERIA

Common Criteria

- Padrão internacional ISO/IEC 15408
- Para segurança das aplicações e para o desenvolvimento seguro
- Arcabouço:
  - usuários definem requisitos funcionais e de segurança
  - desenvolvedores têm objetivos claros
  - produtos podem ser testados e homologados: **certificações**

Common Criteria: Conceitos

- **Perfil de Proteção** (Protection Profile – PP):
  - definido por um grupo de usuários para uma classe de dispositivos de segurança (firewalls, smartcards, etc)
  - para identificar características relevantes e estabelecerem perfis
  - desenvolvedores podem escolher perfis a implementar

Common Criteria: Conceitos

- **Alvo de Segurança** (Security Target – ST):
  - documento com as características de segurança que um produto (alvo) deve satisfazer (um ou mais documentos de perfis de segurança)

Common Criteria: Conceitos

- **Requisitos de Segurança Funcional** (Security Functional Requirements – SFRs):
  - especificam funções de um determinado produto
  - o Common Criteria tem um catálogo com SFRs

Common Criteria: Conceitos

- **Requisitos de Garantia de Segurança** (Security Assurance Requirements – SARs):
  - conjunto de medidas a serem seguidas durante o desenvolvimento e avaliação do produto

Common Criteria: Conceitos

- **Nível de Garantia de Avaliação** (Evaluation Assurance Level – EAL):
  - métrica numérica
  - sete níveis:
    - EAL 1: mais básico e barato de se implementar e avaliar
    - EAL 7: mais sofisticado e caro de se implementar

Common Criteria:

- Padronizações e normatizações internacionais são importantes, mas um certificado apenas diz que um produto passou em uma bateria de testes

- Leitura reflexiva
  - http://www.sans.org/reading-room/whitepapers/standards/common-criteria-iso-iec-15408-insight-thoughts-questions-issues-545

Metodologias de Engenharia de Software:

- Existem metodologias que ampliam outras já conhecidas para atender melhor as questões de segurança e oferecer métricas que indiquem nível de segurança.
  - Exemplo: **BSIMM** – Building Security In Maturity Model
  - Leitura introdutória:
    - https://www.bsimm.com/resources/ieeesandp-bsi-chess-arkin.pdf

ESTUDO INDIVIDUAL

Exercícios

Estude os links indicados ao longo dos slides.