OWASP TOP 10 + Java EE

Magno (Logan) Rodrigues
OWASP Paraíba Leader
magno.logan@owasp.org

OWASP Paraíba

The OWASP Foundation
http://www.owasp.org

Quem sou eu?

- Desenvolvedor Java EE (+2 anos)
- Líder do Capítulo OWASP Paraíba
- Interesses em Segurança em Aplicações Web e Forense Computacional
- Praticante de Artes Marciais

Agenda

- OWASP
  - O que é? Como funciona?
- OWASP Top 10 + Java EE
  - Vulnerabilidades, Ataques e Contra-Medidas
- Ferramentas
  - WebGoat, WebScarab, ZAP, Mantra
- Hands-on!
  - H4ck1ng a real world web app

O que é OWASP?

OWASP

The Open Web Application Security Project

http://www.owasp.org

Open Web Application Security Project

Educar e conscientizar segurança para desenvolvedores, designers, arquitetos e organizações.

Como funciona?

- Sem fins lucrativos (ONG)
- Reconhecida internacionalmente
- Segurança de Aplicações (Web)
- Vive de voluntários!

Será que estamos seguros?

OWASP Top 10 (Edição 2010)

- A1: Falhas de Injeção
- A2: Cross-Site Scripting (XSS)
- A3: Falha de Autenticação e Gerência de Sessões
- A4: Referência Direta e Insegura à Objetos
- A5: Cross Site Request Forgery (CSRF)
- A6: Falhas de Configuração de Segurança
- A7: Armazenamento com Criptografia Insegura
- A8: Falha de Restrição de Acesso a URLs
- A9: Fraca Proteção na Camada de Transporte
- A10: Redirecionamentos e Envios Não Validados

OWASP
The Open Web Application Security Project
http://www.owasp.org

http://www.owasp.org/index.php/Top_10

A1 – Falhas de Injeção

Injeção significa…

- Enganar uma aplicação a incluir comandos nos dados enviados a um interpretador

Interpretadores

- Recebem strings e interpretam como comandos
- SQL, OS Shell, LDAP, XPath, Hibernate, etc…

SQL injection ainda é muito comum!

- Muitas aplicações ainda são suscetíveis (falha dos desenvolvedores)
- Embora seja normalmente muito simples de evitar

Impacto Típico

- Normalmente alto. Todo o banco de dados pode ser lido ou modificado.
- Pode também permitir acesso à contas de usuário ou até mesmo acesso a nível de SO.

Exemplos de Código

- String query = "SELECT user_id FROM user_data WHERE user_name = ' " + req.getParameter("userID") + " ' and user_password = ' " + req.getParameter("pwd") +" ' ";

- E se o usuário informar isto como username? ‘ OR ‘1’=’1’ --

- SELECT user_id FROM user_data WHERE user_name = ‘’ OR ‘1’=‘1’ --

- A consulta irá obter o primeiro usuário da tabela de usuários que normalmente é ... o administrador do sistema!

- Ainda existem aplicações web vulneráveis a este simples ataque!

Caso Real – Banco de Currículos do C.E.S.A.R

C.E.S.A.R

Trabalhe conosco!

Banco de Currículos

Aqui você pode se cadastrar para concorrer às oportunidades em aberto na Mais Inovadora Instituição de Ciência e Tecnologia do Brasil (Prêmio Finep, 2010).
Lembre-se de retornar sempre ao sistema para atualizar suas experiências educacionais e profissionais. Assim você aumenta suas chances de ser escolhido em uma possível seleção.

Porque no C.E.S.A.R, inovação é (a) gente!

No caso de dúvidas e sugestões, utilize nosso Fale Conosco: Banco de Currículos.

Cadastre-se agora!
Clique aqui.

Usuário já cadastrado

Para fazer o login no sistema, selecione uma das opções abaixo (e-mail ou CPF) e preencha os campos solicitados. Em seguida coloque a senha e clique em "Ok".

Preencha uma das opções

E-Mail: ' OR 1=1–x@x.com

CPF:

Senha: ••••

Ok

Esqueci minha senha Alteração de Senha

C.E.S.A.R

Banco de Currículos

Dados Pessoais | Cargos | Formação Acadêmica | Formação Profissional | Idiomas | Conhecimentos | Certificações | Dados Adicionais

Informações Pessoais

Nome: USUARIO SISTEMAS INTERNOS

CPF:

Data Nascimento: 01/01/1980 (Dia - Mês - Ano)

Gênero: Masculino

Contato

E-Mail: sisint@cesar.org.br

E-mail alternativo:

Telefone Residencial: 55 | 81 | 4445421 (País - Área - Número)

Telefone Celular: (País - Área - Número)

Localização

Endereço: rua teset

Complemento: teste

Cidade: teste

País: Brasil

UF: PE

CEP: 54215412

Última Atualização - 11/08/2011

Salvar | Salvar e Sair | Sair

pitang Agile IT

C.e.S.A.R

Trabalhe conosco!

Banco de Currículos

Trabalhe em uma das melhores empresas para trabalhar no Brasil segundo o instituto Great Place to Work. A Pitang oferece serviços de desenvolvimento de sistemas, consultoria, fábrica de software e de testes, migração de legados e outsourcing, focando nas demandas do mercado de uma forma inovadora, através do desenvolvimento de projetos exclusivos que adicionam grande valor a seus clientes. A empresa possui certificação CMMi 3, MPS.BR Nível F e ISO 9001:2008 para seus processos, que se caracterizam pela adoção maciça de práticas ágeis.

Após cadastrar seu currículo, você pode retornar ao sistema para atualizar os dados preenchidos sempre que for necessário. Por isso não esqueça: mantenha seu currículo sempre atualizado!

No caso de dúvidas e sugestões, envie um e-mail para Banco de Currículos.

Usuário já cadastrado

Para fazer o login no sistema, selecione uma das opções abaixo (e-mail ou CPF) e preencha os campos solicitados. Em seguida coloque a senha e clique em "Ok".

Preencha uma das opções

E-Mail ' OR 1=1--x@x.com

CPF

Senha ••••

Ok

Esqueci minha senha Alteração de Senha

Cadastre-se agora!
Clique aqui.

pitang
Agile IT

Inf. Pessoal | Cargos | Formação Acadêmica | Formação Profissional | Idiomas | Entrevista | Conhecimentos | Certificações

Código de acesso	1
Nome	Gustavo Henrique de C
CPF	[redacted]
E-Mail	[redacted]@hotmail.c
E-mail alternativo	
Telefone Residencial	92528077 (País - Área - Número)
Telefone Celular	8399911530 (País - Área - Número)
Data Nascimento	10 / 2 / 1983 (Dia - Mês - Ano) **Sexo** Masculino
Endereço	Av. Ministro Marcos Frei
Complemento	
Cidade	Olinda **CEP** 53030000
País	Brasil **UF** PE

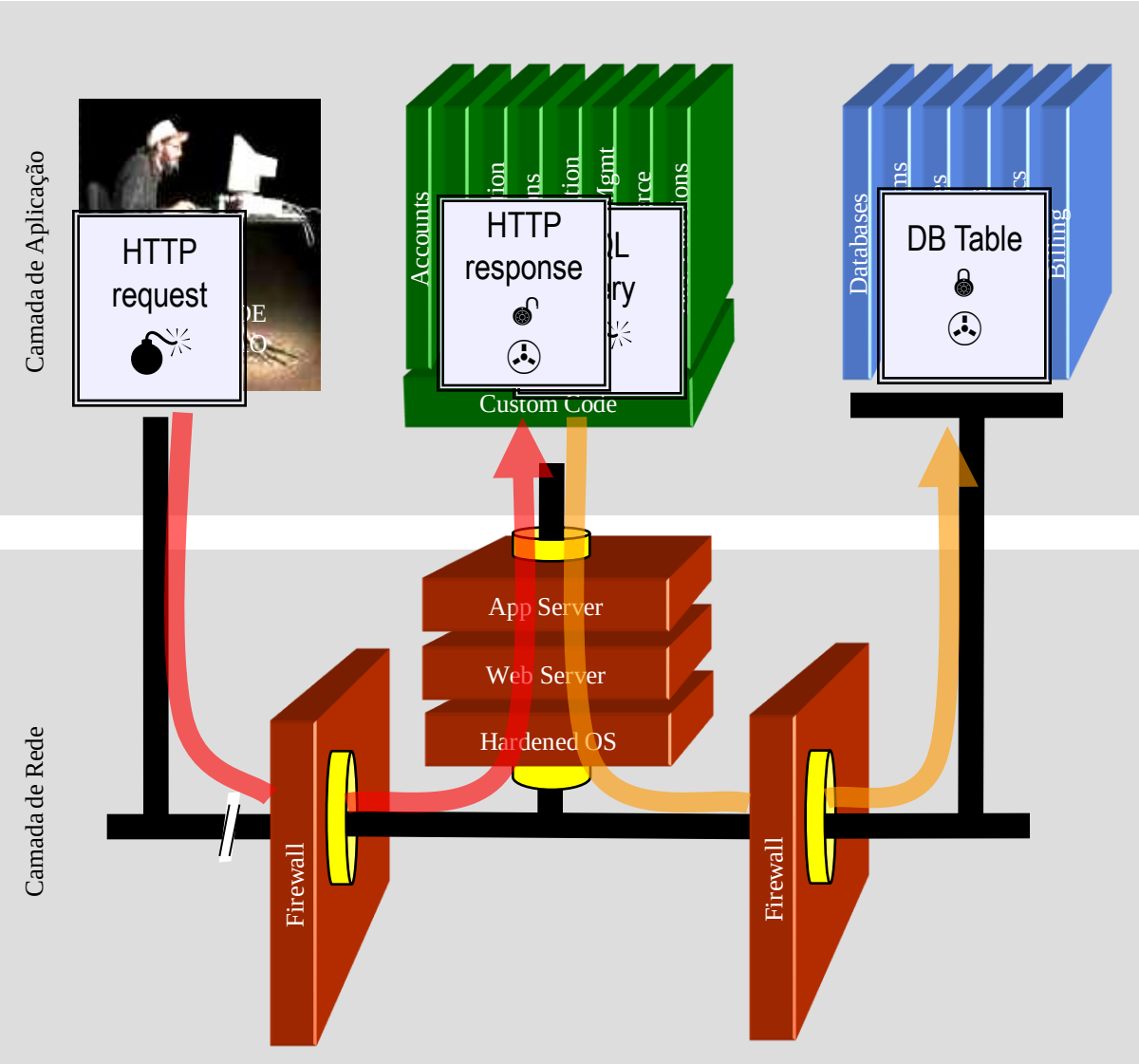
Ultima Atualização - 14/06/2011

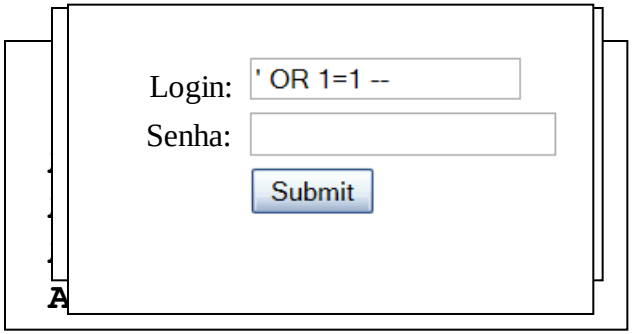
Salvar | Salvar e Sair | Sair

Exemplos de Código

- Runtime.exec(“C:\\windows\system32\cmd.exe \C netstat -p “ + req.getParameter(“proto”));
- E se o usuário informar isto como protocolo? “udp; format c:”
- Runtime.exec(“C:\\windows\system32\cmd.exe \C netstat –p udp; format c:”);
- Irá exibir as conexões ativas e portas utilizadas, mas também irá formatar a unidade C do HD
- Bastante perigosa! Pode ser utilizada para obter informações sobre o sistema

Exemplo de SQL Injection





1. Aplicação apresenta um formulário para o atacante
2. Atacante envia um ataque nos dados do formulário
3. Aplicação repassa ataque para o banco de dados em uma query SQL
4. Banco de dados executa query contendo o ataque e envia os resultados para aplicação
5. Aplicação recebe os dados e envia os resultados para o usuário

A1 – Evitando Falhas de Injeção

1. Use uma interface que suporte bind variables (prepared statements ou stored procedures)

   Bind variables permitem ao interpretador distinguir entre código e dados

   Utilize PreparedStatements fortemente tipados ou Mapeamento Objeto Relacional como Hibernate ou Spring

2. Codificar todas as entradas dos usuários antes de passar para o interpretador

   Sempre execute validação de entrada do tipo 'white list' em todas as informações fornecidas pelo usuário

   Sempre minimize os privilégios do banco de dados para reduzir o impacto de uma falha

A1 – Evitando Falhas de Injeção

- Utilize as classes Encoder e Validator da OWASP ESAPI (Enterprise Security API)

```
String input = request.getParameter("param");
if (input != null) {
        if (!Validator.getInstance().isValidString("^[a-zA-Z ]*$", input)) {
                response.getWriter().write("Inválido: " +
                Encoder.getInstance().encodeForHTML(input) + "<br>");
        } else {
                response.getWriter().write("Válido: " +
                Encoder.getInstance().encodeForHTML(input) + "<br>");
        }
```

- www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

Exercícios:

- Command Injection
- Numeric SQL Injection
- LAB SQL Injection
  - Stage 1
  - Stage 3

A2 – Cross Site Scripting (XSS)

Acontece a qualquer momento…

- Dados não processados do atacante são enviados para um navegador de um usuário inocente

Dados são…

- Armazenados em banco de dados
- Refletidos de entrada da web (formulário, campo oculto, URL, etc…)
- Enviado diretamente ao cliente JavaScript

Praticamente toda aplicação web tem este problema!

- Tente isto no seu navegador – javascript:alert(document.cookie)

Impacto Típico

- Roubar a sessão do usuário, roubar dados sensíveis, reescrever a página web ou redirecionar usuário para sites de phishing ou malware
- Mais severo: Instalar proxy XSS que permita atacante observar e direcionar todo o comportamento do usuário em sites vulneráveis e forçar o usuário a outros sites

Exemplos de Código

- *out.writeln("Você pesquisou por: "+request.getParameter("query");*

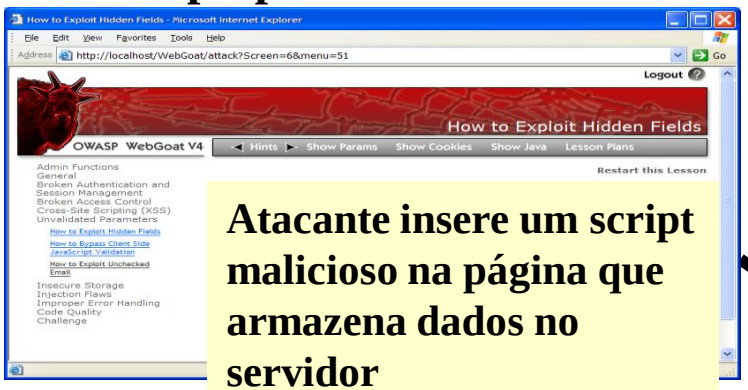
- *<%=request.getParameter("query");%>*

- out.writeln("<tr><td>" + visitante.nome + "<td>" + visitante.comentario);

```
<HTML>
<TITLE>Bem vindo!</TITLE>Hi<SCRIPT>
var pos=document.URL.indexOf("nome=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
<BR>
Bem vindo ao nosso sistema…
</HTML>
```

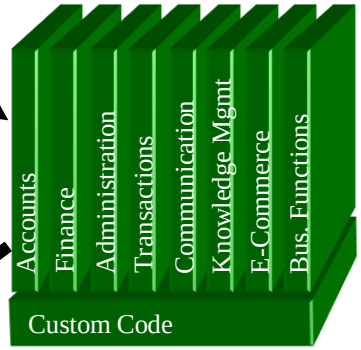
Exemplo de Cross-Site Scripting

1 Atacante prepara a armadilha – atualizar meu perfil

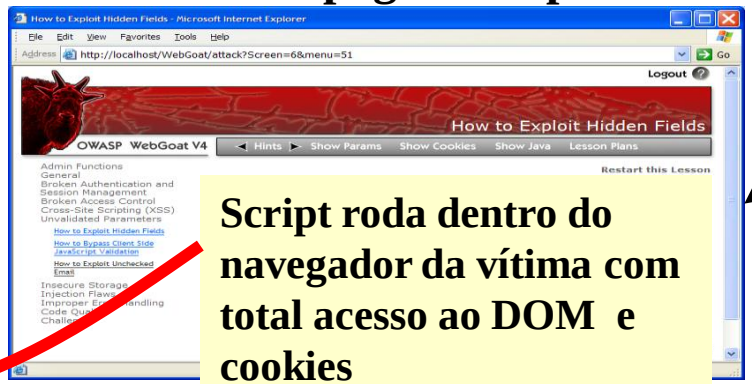


Atacante insere um script malicioso na página que armazena dados no servidor

Aplicação com vulnerabilidade de Stored XSS



2 Vítima acessa a página – o perfil do atacante



Script roda dentro do navegador da vítima com total acesso ao DOM e cookies

3 Script silenciosamente envia o cookie de sessão da vítima

A2 – Evitando XSS

(AntiSamy)

- Realize validação de entrada do tipo whitelist
  - <f:validateLength minimum="2" maximum="10"/>
  - <h:inputText required="true">

- Codifique todas as informações fornecidas pelos usuário!
  - Utilize <bean:write ...> para o Struts ou <c:out escapeXML="true" > para JSTL

- Utilize as classes Encoder e Validator da ESAPI

```
if ( !Validator.getInstance().isValidHTTPRequest(request) )  {
        response.getWriter().write( "<P>HTTP Request Inválido –
        Caracteres Inválidos</P>" );
}
```

- www.owasp.org/index.php/XSS (Cross Site Scripting) Prevention Cheat Sheet

Exercícios:

- Stored XSS
- Reflected XSS
- LAB: Cross Site Scripting
  - Stage 1
  - Stage 3
  - Stage 5

A3 – Falha de Autenticação e Gerência de Sessões

HTTP é um protocolo "stateless" (sem estado)

- Significa que as credenciais deve ser enviadas a cada requisição
- Devemos utilizar SSL para tudo que necessite de autenticação

Falhas no controle das sessões

- SESSION ID usado para controlar o estado já que o HTTP não faz
- E é tão bom quanto as credenciais para o atacante…
- SESSION ID é comumente exposto na rede, no navegador, nos logs, etc

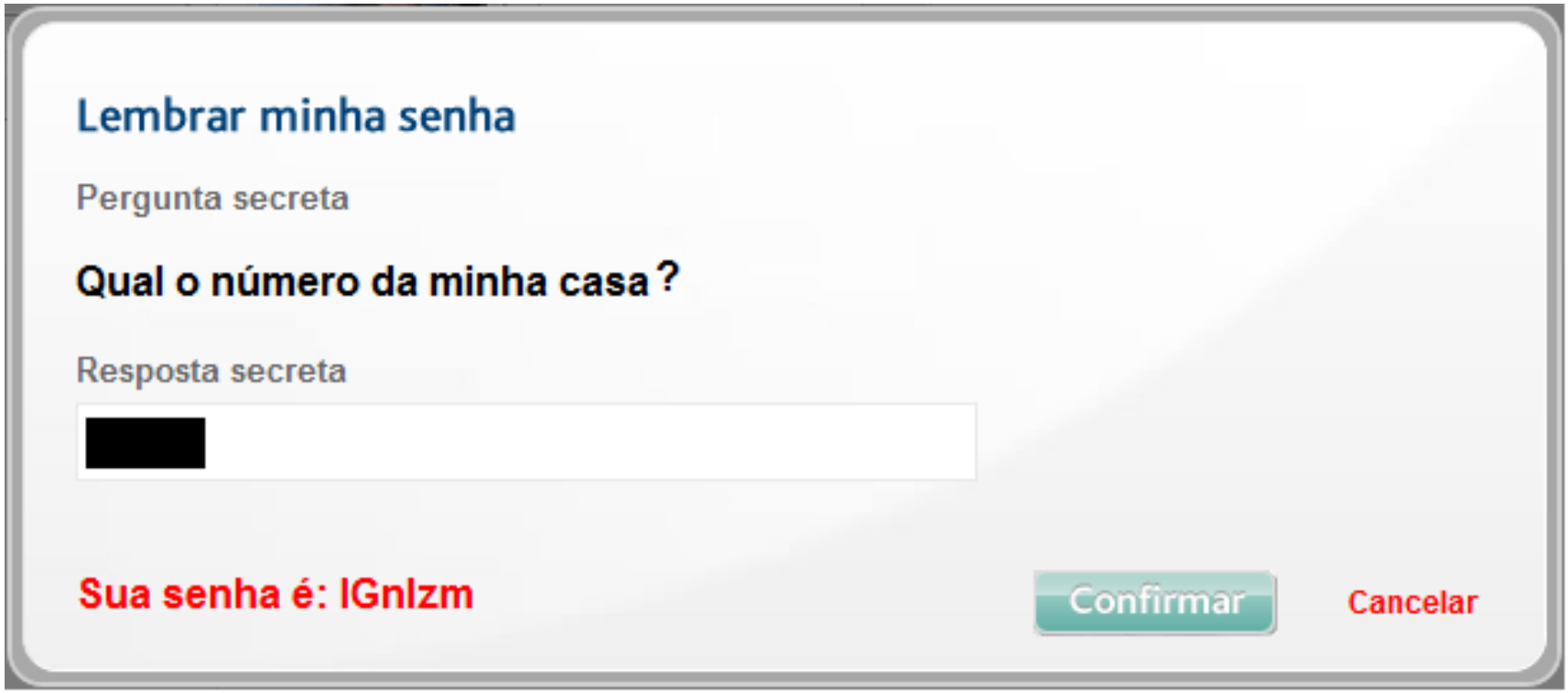
Cuidado com as alternativas!

- Mudar minha senha, lembrar minha senha, esqueci minha senha, pergunta secreta, logout, email, etc…

Impacto Típico

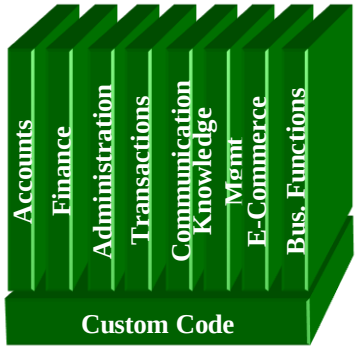
- Contas de usuários comprometidas ou sessões de usuários sequestradas

Perguntas Secretas? Nem sempre!

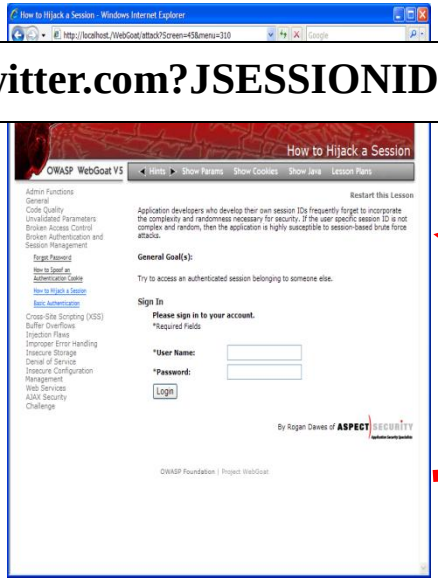


Exemplo de Falha de Autenticação

1 Usuário envia suas credenciais



www.twitter.com?JSESSIONID=9FA1DB9EA...



Site usa URL rewriting (coloca a sessão naURL) 2

3 Usuário clica no link www.hacker.com em um forum

Hacker checa os logs de referência em www.hacker.com e encontra o JSESSIONID do usuário 4

5 Hacker usa JSESSIONID e tem acesso à conta da vítima

OWASP

A3 – Evitando Falhas de Autenticação e Gerenciamento de Sessões

- Verifique sua arquitetura
  - Autenticação deve ser simples, centralizada e padronizada
  - Utilize o session id padrão fornecido pelo seu container web
  - Certifique-se que SSL proteja as credenciais e o session id
- Verifique a implementação
  - Esqueça técnicas de análises automatizadas
  - Verifique o seu certificado SSL
  - Examine todas as funções relacionadas à autenticação
  - Certifique-se que o logoff realmente destroi a sessão
  - Utilize as ferramentas da OWASP para testar sua implementação
- Siga o guia:
  - http://www.owasp.org/index.php/Authentication_Cheat_Sheet

A3 – Evitando Falhas de Autenticação e Gerenciamento de Sessões

- Adicione uma restrição de segurança no web.xml

```
<security-constraint>
    <web-resource-collection>
    <web-resource-name>Páginas em HTTPS</web-resource-name>
        <url-pattern>/profile</url-pattern>
        <url-pattern>/register</url-pattern>
        <url-pattern>/password-login</url-pattern>
        <url-pattern>/ldap-login</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENCIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

- Use as classes Authenticator, User e HTTPUtils da OWASP ESAPI

Exercícios:

- Password Strength
- Forgot Password
- Multi Level Login 1
- Multi Level Login 2

A4 – Referência Insegura e Direta à Objetos

Como você protege o acesso à seus dados?

- Isto é garantir a devida Autorização, junto com A7 – Falha em Restringir Acesso à URLs

Um erro comum…

- Listando apenas os objetos autorizados para o usuário atual
- Escondendo referências à objetos em campos ocultos (hidden)
- … E então não garantir essas restrições no lado servidor
- Isto é chamado de Controle de Acesso da Camada de Apresentação e não funciona!
- Atacante simplesmente altera os valores dos parâmetros

Impacto típico

- Usuários são capazes de acessar arquivos ou informações não autorizadas

Exemplos de Código

```
<select name="idioma"><option value="fr">Francês</option></select>
…
Public static String idioma = request.getParameter(idioma);
String idioma = request.getParameter(idioma);
RequestDispatcher rd = context.getRequestDispatcher("main_"+ idioma);
rd.include(request, response);
```

- E se o usuário modificar o valor da parâmetro para: "../../../../etc/passwd%00"
- Resultado: main_../../../../etc/passwd%00
- Irá exibir a lista de usuários do sistema!

Exemplos de Código

- Numa loja de compras online protegida contra SQL Injection e que não apresente nenhum link para carrinhos não autorizados…

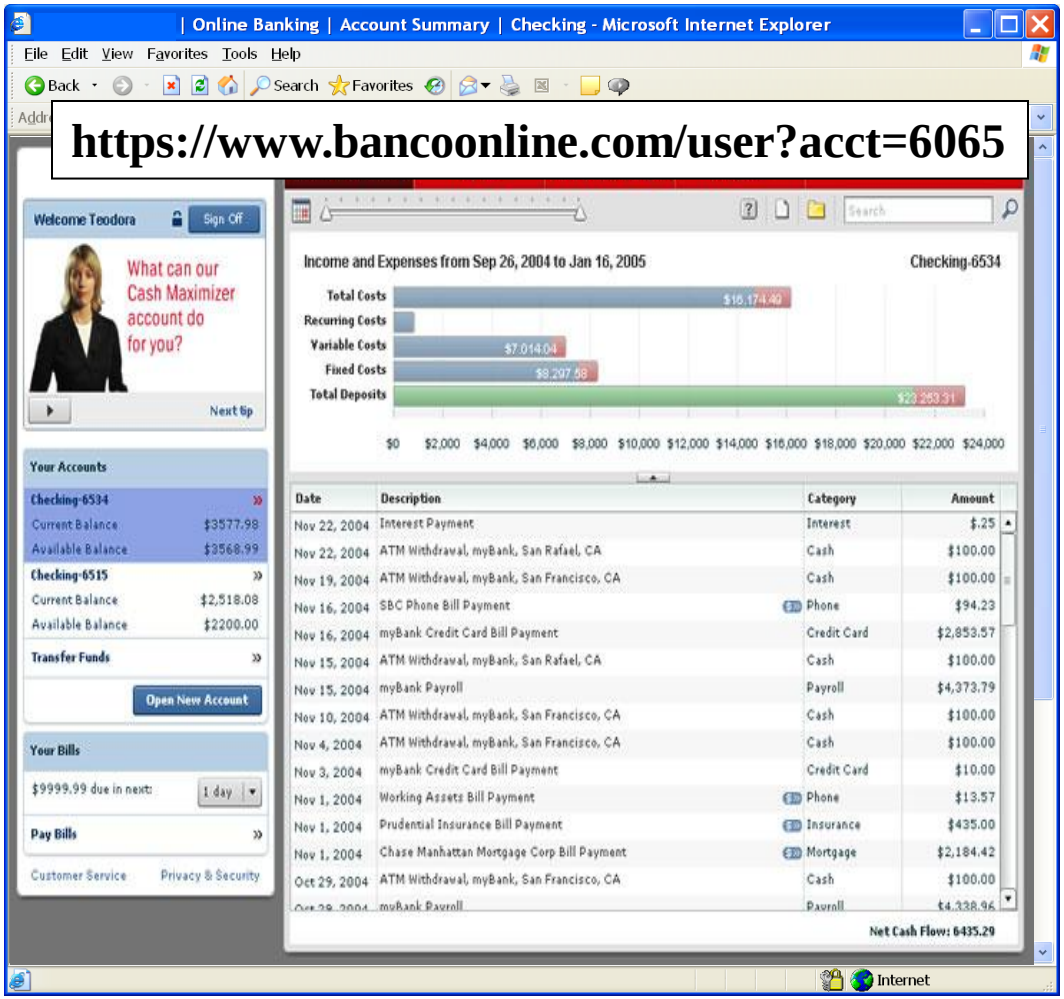
```
int cartID = Integer.parseInt( request.getParameter( "cartID" ) );
String query = "SELECT * FROM table WHERE cartID=" + cartID;
```

- Se eu modificar o ID do meu carrinho de compras, terei acesso ao carrinho de outras pessoas!

- O correto seria:

```
User user = (User)request.getSession().getAttribute( "user" );
String query = "SELECT * FROM table WHERE cartID=" + cartID + "
    AND userID=" + user.getID();
```

Exemplo de Referência Insegura e Direta à Objetos



- Atacante nota que o parâmetro da sua conta é 6065

  ?acct=6065

- Ele modifica para um número próximo:

  ?acct=6066

- Atacante visualiza as informações da conta da vítima

OWASP

A4 – Evitando Referência Insegura e Direta à Objetos

- Elimine a referência direta à objetos
  - Substitua por um valor temporariamente mapeado (ex. 1, 2, 3)
  - ESAPI fornece suporte para mapeamentos númericos e aleatórios
    - IntegerAccessReferenceMap & RandomAccessReferenceMap

http://app?file=Planilha123.xls
http://app?file=1

http://app?id=9182374
http://app?id=7d3J93

Planilha123.xls

Conta:9182374

- Valide a referência direta ao objeto
  - Verifique se o valor do parametro está devidamente formatado
  - Verifique se o usuário tem acesso ao objeto em questão
  - Verifique se o modo de acesso é permitido para o objeto (ex. read, write, delete)

Exercícios:

- Insecure Configuration
  - Forced Browsing

A5 – Cross Site Request Forgery (CSRF)

Cross Site Request Forgery

- Um ataque onde o browser da vítima é enganado a enviar um comando à uma aplicação web vulnerável
- Vulnerabilidade causada por browsers que incluem automaticamente informações de autenticação do usuário (ID da sessão, Endereço IP, …) a cada solicitação (request)

Imaginem…

- E se um hacker pudesse mexer seu mouse e fazê-lo clicar em links na sua aplicação de online banking?
- O que eles poderiam "forçar" você fazer?

Impacto típico

- Iniciar transações (transferir fundos, fazer logout, fechar conta)
- Acessar dados sensíveis
- Mudar os detalhes da conta

Padrão de Vulnerabilidade do CSRF

- O Problema
  - Navegadores web automaticamente incluem a maioria das credenciais a cada request
  - Até mesmo para requests feitos em outro site
- Todos os sites que confiam apenas em credenciais automáticas estão vulneráveis!
  - (quase todos os sites estão desta forma)
- Credenciais fornecidas automaticamente
  - Cookie de sessão
  - Cabeçalho básico de autenticação
  - Endereço IP
  - Certificados SSL do lado cliente
  - Autenticação de domínios Windows

Exemplos de Código

- <img src="http://www.example.com/logout.jsp">
- Irá realizar o logout do usuário da aplicação acessada. Nada muito perigoso não é mesmo?
- <img src="http://www.example.com/transfer.do?frmAcct= document.form.frmAcct&toAcct=4345754&toSWIFTid=434343&amt =3434.43">
- E agora? Irá realizar a transferência de uma certa quantia em dinheiro de uma conta para outra!

Exemplo de CSRF

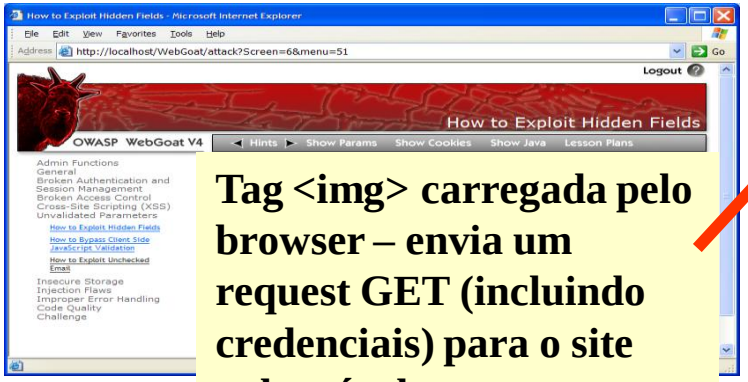
1

Atacante coloca uma armadilha em um site na internet (ou simplesmente via email)

Tag <img> oculta contém ataque contra o site vulnerável

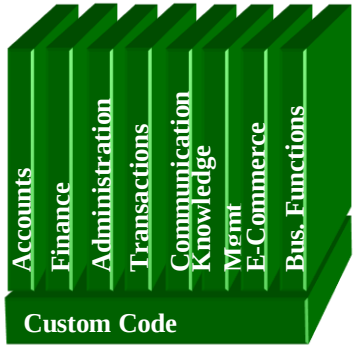
2

Enquanto logada no site vulnerável, a vítima visita o site do atacante



Tag <img> carregada pelo browser – envia um request GET (incluindo credenciais) para o site vulnerável

Aplicação com uma vulnerabilidade CSRF



3

Site vulnerável olha o request legítimo da vítima e realiza a ação solicitada

OWASP

A5 – Evitando Falhas de CSRF

- Adicione um token secreto, não submetido automaticamente, para todos os requests sensíveis
  - Torna impossível para uma atacante falsificar o request
    - (a não ser que tenha uma falha de XSS na sua aplicação)
  - Tokens devem ser criptografados e aleatórios (não sequenciais!)

- Opções
  - Armazene um token na sessão e adicione a todos os links e formulários
    - **Campo oculto:** `<input name="token" value="687965fdfaew87agrde" type="hidden"/>`
    - **URL de Uso Único:** `/accounts/687965fdfaew87agrde`
    - **Token de Formulário:** `/accounts?auth=687965fdfaew87agrde` **...**
  - Pode ter um token único para cada função
    - Use um hash do nome da função, id da sessão e um segredo (salt)
  - Pode requerer autenticação secundária para funções sensíveis (ex. eTrade)

A5 – Evitando Falhas de CSRF

- Não permita que atacantes armazenem ataques no seu site
  - Codifique devidamente toda entrada na saída
  - Isto torna todos os links/requests inertes na maioria dos interpretadores
- Elimine as falhas de XSS
- Utilize a classe User da ESAPI para gerar e validar um token

Para mais detalhes: www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet

Exercícios:

- Cross Site Request Forgery (CSRF)
- CSRF Prompt By-Pass
- CSRF Token By-Pass

A6 – Falhas de Configuração de Segurança

Aplicações Web confiam em uma fundação segura

- Tudo desde o SO ao Servidor de Aplicações
- Não esqueça das bibliotecas que estiver usando!

Seu código fonte é secreto?

- Pense em todos os locais que seu código fonte vai
- Segurança não deve requerer um código fonte secreto

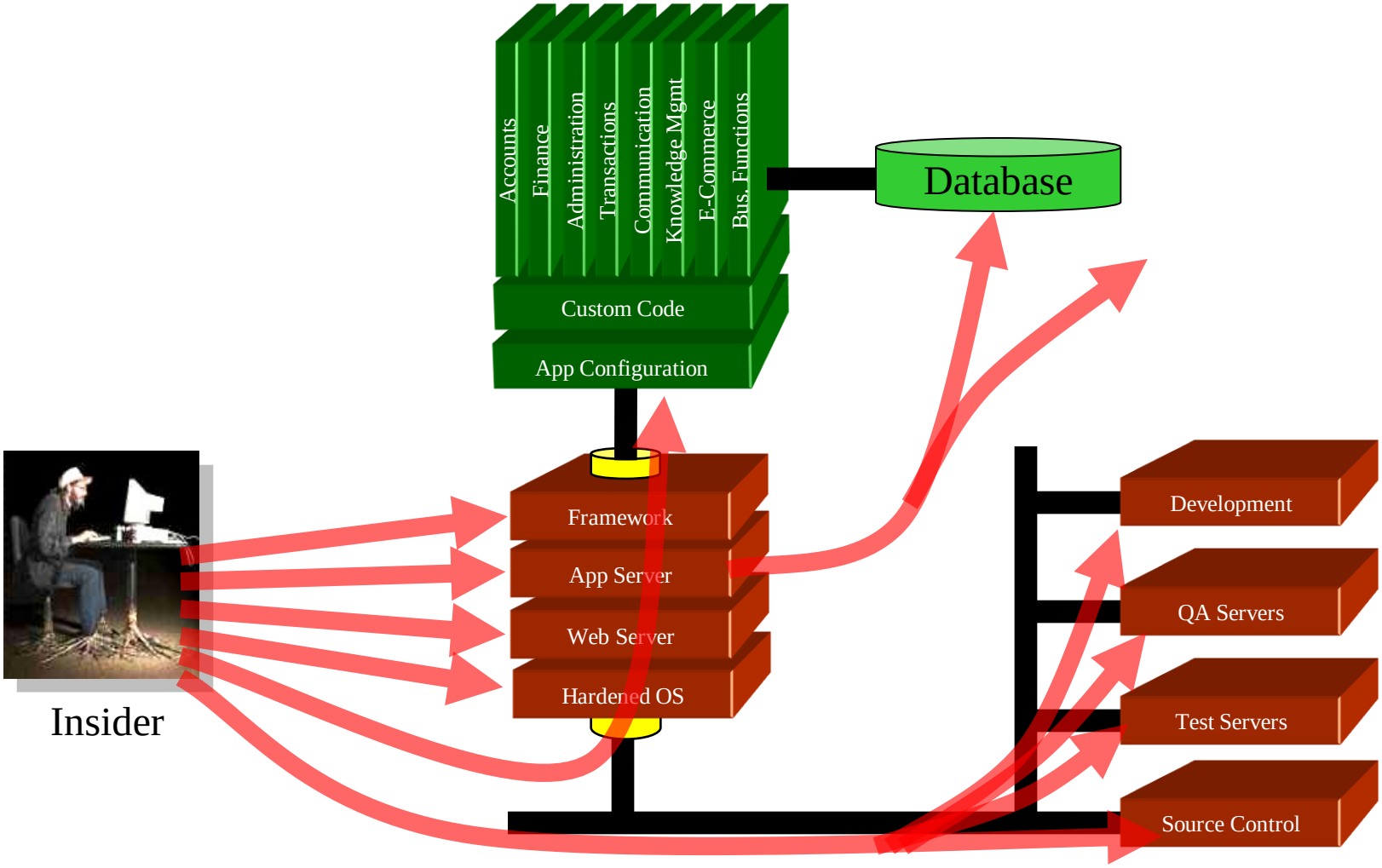
GC deve se estender para todas as partes da aplicação

- Todas as credenciais devem mudar em produção!

Impacto Típico

- Instalar um backdoor através de SO ou Servidor desatualizado
- Exploits de falhas de XSS devido a falta de patches dos frameworks da aplicação
- Acesso não autorizado a contas default, funcionalidades ou dados da aplicação ou acesso à funcionalidades devido a fraca configuração do servidor

Exemplo de Falhas de Configuração de Segurança



OWASP

A6 – Evitando Falhas de Configuração de Segurança

- Verifique o gerenciador de configurações do sistema
  - Guia do Hardening da Configuração Segura
    - Automação é MUITO IMPORTANTE aqui
  - Deve cobrir toda a plataforma e a aplicação
  - Mantenha todos os componentes atualizados!
    - Incluindo bibliotecas de software, não apenas SO e Servidor de Aplicações
  - Analise os efeitos de segurança das mudanças

- Forte arquitetura da aplicação
  - Provê uma boa separação e segurança entre os componentes

- Verifique a implementação
  - Scans encontram configurações genericas e problemas de atualização

Exercícios:

- Improper Error Handling
  - Fail Open Authentication Scheme
- Code Quality
  - Discover Clues in the HTML

A7 – Armazenamento com Criptografia Insegura

Armazenando dados sensíveis de forma insegura

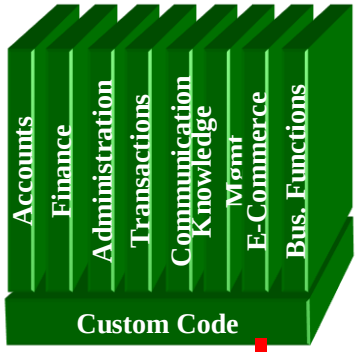
- Falha em identificar todos os dados sensíveis
- Falha em identificar todos os locais onde os dados sensíveis são armazenados
  - Banco de dados, diretórios, arquivos, logs, backups, etc..
- Falha em proteger devidamente estes dados em todos os locais

Impacto Típico

- Atacantes acessam ou modificam informações privadas ou confidenciais
  - Ex.: cartões de crédito, registros de saúde, dados financeiros (seus ou dos seus clientes)
- Atacantes obtém segredos para usá-los em novos ataques
- Embaraço da Empresa, insatisfação dos clientes e perda de confiança
- Gastos para limpar o incidente, como forense, enviar cartas de desculpas, reemitir milhares de cartões de crédito, fornecer seguro contra roubo de identidade
- Empresas são processadas e/ou multadas

Exemplo de Armazenamento com Criptografia Insegura

1 Vítima insere seu número de cartão de crédito no formulário



2 Gerenciador de erros loga os detalhes do CC porque o gateway da loja está indisponível

3 Logs são acessíveis a todos os membros da TI para depuração

4 Insider malicioso rouba 4 milhões de números de cartões de crédito

A7 – Evitando Armazenamento com Criptografia Insegura

- Verifique sua arquitetura
  - Identifique todos os dados sensíveis
  - Identifique todos os locais que esses dados são armazenados
  - Utilize criptografia para combater as ameaças, não apenas criptografe os dados

- Proteja com mecanismos apropriados
  - Criptografia de arquivos, criptografia de banco de dados, criptografia de elementos de dados

A7 – Evitando Armazenamento com Criptografia Insegura

- Utilize os mecanismos corretamente
  - Use algoritmos padrões fortes
  - Gere, distribua e proteja as chaves apropriadamente
  - Prepare-se para mudança de chaves

- Verifique a implementação
  - Um forte algoritmo padrão é usado e é o apropriado para esta situação
  - Todas as chaves, certificados e senhas estão devidamente armazenadas e protegidas
  - Distribuição de chaves segura e um efetivo plano para mudança de chaves estão implementados
  - Analise o código de criptografia para falhas comuns

A8 – Falha de Restrição de Acesso a URLs

Como você protege o acesso à URLs (páginas)?

- Isto é garantir a devida autorização junto com A4 – Referência Insegura e Direta à Objetos

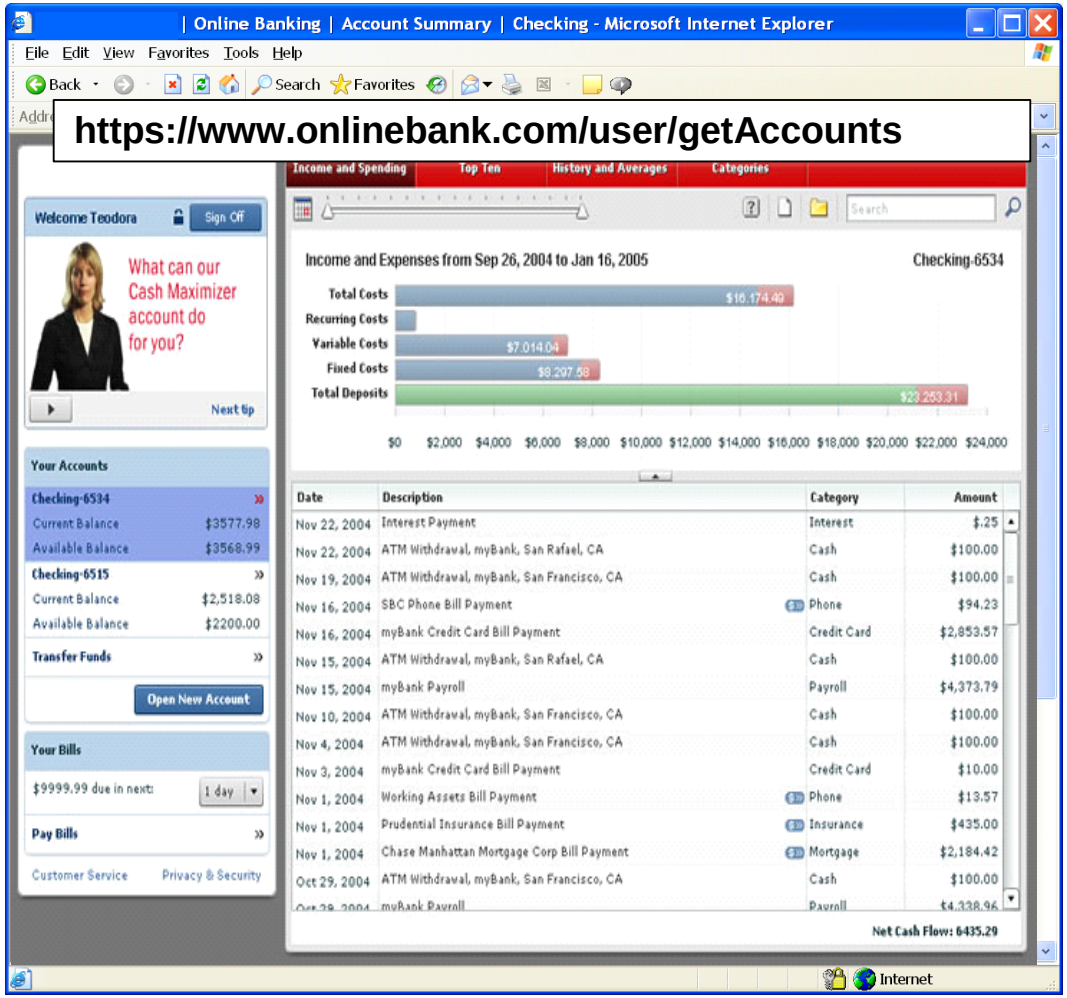
Um erro comum…

- Mostrando apenas links e menus autorizados
- Isto é chamado de Controle de Acesso de Camada de Apresentação e não funciona!
- Atacante simplesmente forja um acesso direto à páginas "não-autorizadas"

Impacto Típico

- Atacante chama funções e serviços que não estão autorizados
- Acesso à conta e informações de outros usuários
- Realização de ações privilegiadas

Exemplo de Falha de Restrição de Acesso a URLs



- Atacante nota que a URL indica seu perfil

  /user/getAccounts

- Ele modifica para outro diretório (perfil)

  /admin/getAccounts ou /manager/getAccounts

- Atacante visualiza mais contas do que apenas a sua

OWASP

A8 – Evitando Falhas de Restrição de Acesso a URLs

- Para cada URL, um site precisa fazer 3 coisas:
  - Restringir o acesso à usuários autenticados (se não for público)
  - Garantir permissões baseadas em usuários ou perfis (se privado)
  - Desabilitar completamente requests para tipos de páginas não autorizadas (Ex.: arquivos de configuração, arquivos de log, códigos fonte, etc.)

- Verifique sua arquitetura
  - Utilize um modelo simples e positivo em todas as camadas
  - Tenha certeza que realmente tem um mecanismo em todas as camadas

- Verifique a implementação
  - Esqueça abordagens de análises automatizadas
  - Certifique-se que cada URL na sua aplicação está protegida com:
    - Um filtro externo como web.xml do Java EE
    - Ou verificações internas no seu código – Utilize o isAuthorizedForURL() da ESAPI
  - Verifique se a configuração do servidor desabilita requests para tipos de arquivos não autorizados
  - Use WebScarab ou seu browser para forjar requests não autorizados

A8 – Evitando Falhas de Restrição de Acesso a URLs

- Google is your friend! Use-o! Sabe utilizar o robots.txt?!
- Utilize a ferramenta da OWASP DirBuster para mapear todos os arquivos e diretórios da sua aplicação
- Evite que estas páginas sejam acessíveis por usuários não autorizados:

  /admin/adduser.php ou /approveTransfer.do

A8 – Evitando Falhas de Restrição de Acesso a URLs

- Configure as restrições no web.xml

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Páginas de administração da Aplicação Java EE
    protegidas</web-        resource-name>
        <description>Requer autenticação dos usuários.</description>
        <url-pattern>/admin/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <description>Permite acesso às paginas de administração</description>
        <role-name>Administrador</role-name>
    </auth-constraint>
</security-constraint>
<security-role>
    <description>Administrador Java EE</description>
    <role-name>Administrador</role-name>
</security-role>
```

A9 – Fraca Proteção na Camada de Transporte

Transmitindo dados sensíveis de forma insegura

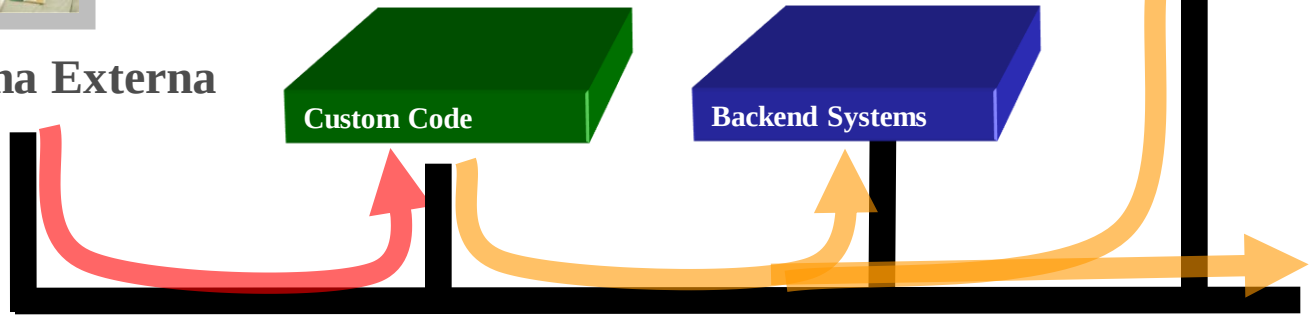
- Falha em identificar todos os dados sensíveis
- Falha em identificar todos os locais que estes dados são enviados
  - Na Internet, para o banco de dados, para parceiros de negócios ou comunicações internas
- Falha em devidamente proteger estes dados em todos os locais

Impacto Típico

- Atacantes acessam ou modificam informações privadas ou confidenciais
  - Ex.: cartões de crédito, registros de saúde, dados financeiros (seus ou dos seus clientes
- Atacantes obtém segredos para usar em ataques futuros
- Embaraço da empresa, insatisfação dos clientes e perda de confiança
- Custos de limpar o incidente (Forense)
- Empresas são processadas e/ou multadas

Exemplo de Fraca Proteção na Camada de Transporte

Vítima Externa



Parceiros

Empregados

1

Atacante externo rouba dados e credenciais da rede

Atacante Externo

2

Atacante interno rouba dados e credenciais da rede interna

Atacante Interno (Insider)

OWASP

A9 – Evitando Fraca Proteção na Camada de Transporte

- Proteja com mecanismos apropriados
  - Use TLS em todas as conexões com dados sensíveis
  - Criptografe individualmente as mensagens antes de transmitir
    - Ex.: XML-Encryption
  - Assine as mensagens antes de transmitir
    - Ex.: XML-Signature
- Utilize mecanismos corretamente
  - Use algoritmos fortes (desabilite algoritmos SSL antigos)
  - Gerencie chaves/certificados apropriadamente
  - Verifique certificados SSL antes de utilizá-los
  - Use mecanismos aprovados quando suficiente
    - Ex.: SSL vs XML-Encryption
- http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

A9 – Evitando Fraca Proteção na Camada de Transporte

- Adicione restrições de segurança no web.xml

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Páginas HTTPS</web-resource-name>
        <url-pattern>/profile</url-pattern>
        <url-pattern>/register</url-pattern>
        <url-pattern>/password-login</url-pattern>
        <url-pattern>/ldap-login</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENCIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

- Proteja o cookie de sessão setando o secure bit para 1
  - (javax.servlet.http.Cookie.setSecure(true))
  - Isto irá prevenir o envio do cookie em texto plano

Exercícios:

- Insecure Communication
  - Insecure Login

A10 – Redirects e Forwards Não Validados

Redirecionamentos em Aplicações Web são muito comuns

- E frequentemente incluiem parâmetros fornecidos pelo usuário na URL de destino
- Se eles não forem validados, atacante pode enviar a vítima para um site a sua escolha

Envios (Forwards) também são

- Eles enviam o request internamente para uma nova página na mesma aplicação
- Às vezes os parâmetros definem a página alvo
- Se não forem validados, atacante pode utilizar um envio não-validado para burlar checagens de autenticação ou autorização

Impacto típico

- Redirecionar a vítima para um site de phishing ou malware
- O request do atacante burla checagens de segurança, permitindo o acesso à funções ou dados não-autorizados

Mas qual a diferença entre eles?

- Forward
  - É executado internamente pelo servlet
  - O browser não sabe o que está ocorrendo
  - A URL na barra de endereços não muda
  - O reload irá executar a requisição original
- Ex.: rd = request.getRequestDispatcher("pagina.html");
  rd.forward(request, response);

Mas qual a diferença entre eles?

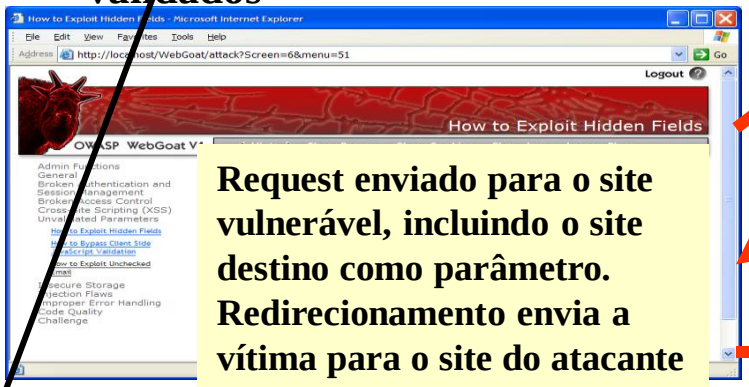
- Redirect
  - É um processo de dois passos, por isso a URL muda
  - O reload da página não repetirá a requisição original
  - É um processo muito mais lento que um *forward*, pois são necessárias duas requisições, e não uma
  - Objetos colocados no escopo do request original são perdidos durante o segundo request
- Ex.: response.sendRedirect("pagina.html");

Exemplo de Redirect Não Validado

1 Atacante envia o ataque para a vítima através de um email ou site

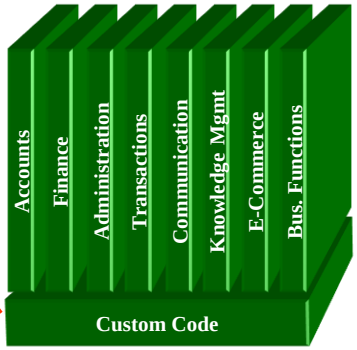
From: Receital Federal
Subject: Imposto de Renda
Nossos registros mostram que você tem uma restituição para receber. Por favor clique aqui para receber seu dinheiro

2 Vítima clica no link contendo parâmetros não validados



Request enviado para o site vulnerável, incluindo o site destino como parâmetro. Redirecionamento envia a vítima para o site do atacante

3 Aplicação redireciona a vítima pro site do atacante



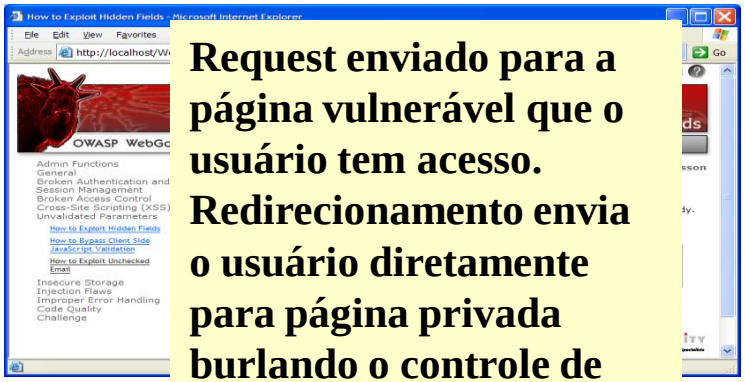
4 Site malicioso instala malware na vítima ou tenta obter informações privadas

http://www.receita.fazenda.gov.br/irpf/obterImposto.jsp?year=2011& … &dest=www.hacked.com

OWASP

Exemplo de Forward Não Validado

1 **Atacante envia o ataque para página vulnerável a qual tem acesso**



Request enviado para a página vulnerável que o usuário tem acesso. Redirecionamento envia o usuário diretamente para página privada burlando o controle de acesso

2 **Aplicação autoriza o request que continua para a página vulnerável**

Filtro

```
public void doPost( HttpServletRequest request,
HttpServletResponse response) {
    try {
        String target = request.getParameter( "dest" )
);
        ...
        request.getRequestDispatcher( target
        ).forward(request, response);
    }
    catch ( ...
```

```
public void sensitiveMethod(
HttpServletRequest request,
HttpServletResponse response) {
    try {
        // Faz coisas sensíveis
aqui.
        ...
    }
    catch ( ...
```

3 **O envio falha em validar o parâmetro, enviando o atacante para uma página não-autorizada, burlando o controle de acesso**

OWASP

A10 – Evitando Redirects e Forwards Não Validados

1. Evitar o uso de redirects e forwards o máximo possível
2. Se utilizado, não envolva parâmetros do usuário para definir a URL alvo
3. Se necessitar envolver parâmetros do usuário, então:

   Use mapeamento do lado servidor para traduzir a escolha fornecida pelo usuário com a página alvo

- Para redirects, valide a URL de destino para garantir que irá para um site externo autorizado

- A OWASP ESAPI pode fazer isso para você!
  - SecurityWrapperResponse.sendRedirect(URL)

A10 – Evitando Redirects e Forwards Não Validados

- Algumas dicas sobre proteger Forwards
  - Chame o controle de acesso para garantir que o usuário está autorizado antes de realizar o forward
  - Garanta que os usuários que podem acessar a página origem são todos autorizados a acessar a página destino
  - Analise de onde o usuário está vindo, de forma a mapear a aplicação e impedir acessos indevidos a URLs
  - Preste especial atenção aos casos parametrizados que precisam, necessariamente, utilizar URLs externas

Como resolver todos estes problemas?

- Desenvolver código seguro - "Prevenir é melhor do que remediar"
  - Siga as melhores práticas no Guia da OWASP – Construindo Aplicações Web Seguras
    - http://www.owasp.org/index.php/OWASP_Guide_Project
  - Utilize o Padrão de Verificação de Segurança em Aplicações da OWASP como um guia para saber o que uma aplicação precisa para ser segura
    - https://www.owasp.org/index.php/ASVS
  - Use os componentes padrões de segurança que se adequam a sua organização
    - Veja a ESAPI da OWASP como base para seus componentes padrões
    - https://www.owasp.org/index.php/ESAPI

Como resolver todos estes problemas?

- Revise suas aplicações
  - Tenha um time para testar e revisar suas aplicações
  - Revise você mesmo suas aplicações seguindo os guias da OWASP:
    - OWASP Code Review Guide:
      http://www.owasp.org/index.php/Code_Review_Guide
    - OWASP Testing Guide:
      http://www.owasp.org/index.php/Testing_Guide

Desafio Final – only for 1337!

- The CHALLENGE!
  - Stage 1
  - Stage 2
  - Stage 3

- Can you do it?!
- May the **source** be with you!

Ferramentas

WebGoat

OWASP WebGoat V5.3

- Uma aplicação Java EE vulnerável propositalmente mantida pela OWASP
- Desenvolvida para ensinar segurança em aplicações web
- Mas também é util para testar produtos de segurança: IPS, Firewalls, WAF, etc...
- Funciona em Windows, Linux, etc...

WebGoat v5.3

- Disponível em Inglês e Alemão
- Existe uma versão disponível em pt-BR: v5.1
  code.google.com/p/webgoat-ptbr
- Simples e fácil de utilizar, só baixar e rodar!
- Vamos começar? Execute o webgoat.bat ou .sh

Thank you for using WebGoat! This program is a demonstration of common web application flaws. The exercises are intended to provide hands on experience with application penetration testing techniques.

The WebGoat project is lead by Bruce Mayhew. Please send all comments to Bruce at WebGoat@owasp.org.

WebGoat Design Team

Bruce Mayhew
David Anderson
Rogan Dawes
Laurence Casey (Graphics)

V5.3 Lesson Contributers

Chuck Willis
Cam Morris

Special Thanks for V5.3

Christine (Maven)
Marek Jawurek (Internationalization)

To all who have sent comments

Documentation Contributers

Sherif Koussa
Aung Khant
(http://yehg.org/)
Erwin Geirnaert
(http://www.zionsecurity.com/)

Start WebGoat

WARNING
While running this program, your machine is extremely vulnerable to attack if you are not running on localhost. If tou are NOT running on localhost (default configuration), You should disconnect from the network while using this program.

This program is for educational purposes only. Use of these techniques without permission could lead to job termination, financial liability, and/or criminal penalties.

WebScarab

OWASP
The Open Web Application Security Project
http://www.owasp.org

- Framework para analisar aplicações que utilizam os protocolos HTTP e HTTPS
- Feito em Java (multiplataforma)
- Nova versão (NG) é mais amigável
- Desenvolvido por Rogan Dawes

ZAP (Zed Attack Proxy)

- Ferramenta de teste de invasão integrada
- Ideal para pessoas sem conhecimento em segurança (desenvolvedores e testers)
- Feito em Java (multiplataforma)
- Desenvolvido por Simon Bennetts e Axel Neumann

Obrigado!

Referências

- OWASP Top 10 for 2010 - http://www.owasp.org/index.php/Top_10
- The Ten Most Critical Web Application Security Risks http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf
- OWASP Top 10 Presentation - http://owasptop10.googlecode.com/files/OWASP_Top_10_-_2010%20Presentation.pptx