Apostila Preparatória para Certificação

Secure Programming Foundation

Área de Aprendizagem

www.pmgacademy.com

Official Course

www.pmgacademy.com

Design: By Freepik

Material Básico para o curso de 'Secure Programming Foundation' (SPF)
Publicação: Junho 2016

Bibliografia para este curso:

Hemel, T., & Witmond, G.
EXIN Secure Programming Foundation – Workbook
(R. Pisaturo, M. Hubregtse, & E. Kleijer, Eds.)
Utrecht, The Netherlands: EXIN Holding B.V., 2014
ISBN: 978-90-820388-6-6 http://www.vanharen.net/9789082038866

Instrutor - Prof. Adriano Martins Antonio

PARTE I

Jargões

Falar em 'segurança' nos remete de imediato a um cenário de perigo, no qual precisamos ser protegidos, não é mesmo? Na área de Programação de TI não é diferente em termos como 'ataque', 'risco' e 'ameaça' são tão usuais que – mesmo tendo significados linguisticamente parecidos – ganham definições diferenciadas. Vamos conhecer então o que cada um desses termos significa no universo dos programadores de Segurança da Informação:

Ataque: É uma tentativa de acesso sem autorização a um sistema.

Explorar: É um método de tirar proveito (com más intenções) de alguma falha específica do sistema.

Mitigação: Uma medida para diminuir o impacto de uma ameaça.

Remendo: Uma medida para corrigir uma falha de um sistema.

Risco: A probabilidade de perder onde se poderia ganhar. Ocorre, geralmente, no que diz respeito a tempo de impacto.

Ameaça: É um potencial ataque.

Vulnerabilidade: Uma fraqueza que pode ser realmente abusada.

Fraqueza: Uma construção errônea que pode reduzir a segurança.

Resumo

Cibercrime, vazamento de dados e segurança de informações recebem mais atenção que nunca nos noticiários. Governos e empresas dedicam mais e mais recursos a essas áreas. No entanto, a maioria dessa atenção parece estar focada em medidas reativas ("Como pegar os criminosos cibernéticos?") em vez de em medidas preventivas ("Como podemos tornar nossos sistemas seguros?"). Embora seja difícil medir, relatórios de pesquisa indicam que a construção da segurança vale o investimento. A educação é fundamental no processo de construção de software. Se os programadores não entendem a segurança do software que estão construindo, qualquer investimento adicional no processo é inútil.

SP Foundation

O exame Secure Programming Foundation (Fundação da Programação Segura) da EXIN testa o conhecimento do candidato sobre os princípios básicos da programação segura. Os temas deste módulo são Gerenciamento de Sessão e Autenticação; Manejo de Entrada de Usuário; Autorização; Configuração, Manejo e Registro de Erros; Criptografia; e Engenharia Segura de Software.

Contexto

O exame Secure Programming Foundation faz parte da qualificação Secure Programming. O conteúdo está relacionado com o Framework Secure Software, que pode ser baixado em www.securesoftwarefoundation.org. (Isso não é literatura de exame).

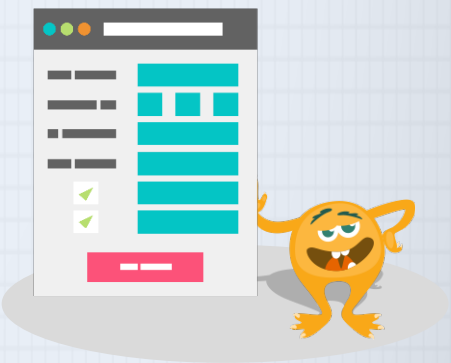
Objetivos

Os objetivos da PARTE I do cuso Secure Programming Foundation é:

- Reconhecer a tensão entre demandas de mercado e segurança.
- Explicar o jargão de segurança e STRIDE.
- Descrever questões de segurança HTTP.
- Explicar o modelo de segurança do navegador.
- Identificar problemas envolvidos no uso de senha.
- Aplicar princípios de gerenciamento de senhas.
- Explicar como o Gerenciamento de Sessão funciona.
- Reconhecer problemas no Gerenciamento de Sessão.
- Reconhecer as melhores soluções para problemas do Gerenciamento de Sessão.
- Reconhecer problemas e soluções de CSRF e do Clickjacking.
- Reconhecer os problemas dos ataques de injeção.
- Explicar as diferenças entre consultas diretas e parametrizadas.
- Aplicar soluções para ataques de injeção SQL.
- Explicar as diferenças entre filtros de lista braca (whitelist) e lista negra (blacklist).
- Aplicar validação de entrada.
- Reconhecer quando aplicar normalização de entrada e codificação.
- Identificar a ocorrência de estouros de buffer e qual seu impacto na segurança.
- Reconhecer as diferenças entre XXS refletidos e armazenados e as mitigações.
- Aplicar soluções para ataques XSS.

Causas do software inseguro:

- Sem tempo ou dinheiro.
- O cliente não o solicitou .
- Falta de conhecimento.
- Intenção maliciosa.
- Ninguém vai querer atacar nosso software.



Acrônimo STRIDE:
S - Spoofing – Forjamento. Fingir ser alguém ou algo que não é.
T - Tampering – Adulteração. Modificar dados armazenados ou dados em trânsito.
R - Repudiation – Rejeição. Negar que fez ou não fez algo.
I - Information Disclosure – Divulgação de informações. Ver informações sem permissão.
D - Denial-of-Service - Recusa de serviço. Tornar um sistema indisponível.
E - Elevation of Privilege - Elevação de privilégio. Fazer algo sem permissão.

Superfície de Ataque

No terreno da Tecnologia de Segurança da Informação, superfície de ataque é o terreno que o inimigo conquista e ergue a sua bandeira...

Zonas Confiáveis

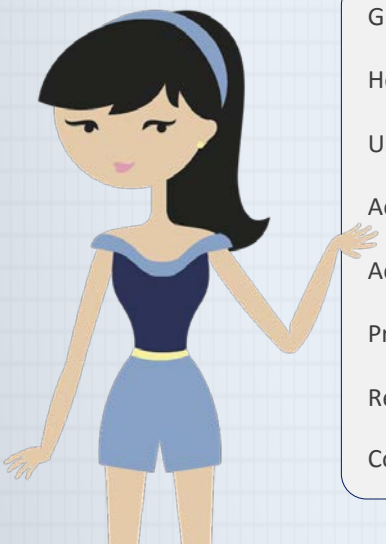
Um sistema pode ser dividido em superfícies de ataque – onde existe vulnerabilidade para acessos não autorizados – e zonas confiáveis – onde existe maior grau de segurança, probabilidade mínima de falhas de acesso do inimigo.

Os 'defensores' dessas zonas confiáveis são chamados de 'limites de confiança'.

Segurança da Web – GET request

Para exemplificar melhor os 'limites de confiança' basta pensar em um servidor web e um navegador web. A linha que separa quem é quem nessa atuação é o limite de confiança. GET é destinado a operações de "apenas leitura", enquanto que POST irá modificar o estado da aplicação.

Exemplo do problema da falta de segurança com o comendo get:



```
GET http://localhost/insecure/public/Login.jsp?login=admin&passadmin HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 Gecko/20100101 Firfox/14.0.1
Accept: text/html,application/xhtml+xml,application/xml;q0.9,*/*;q0.8
Accept-Language: en-us,en;q0.5
Proxy-Connection: keep-alive
Referer: http://localhost/insecure/public/Login.jsp
Cookie: JSESSIONID=73A4496C2B6C846B5E8D8C9DDDBD9D24
```

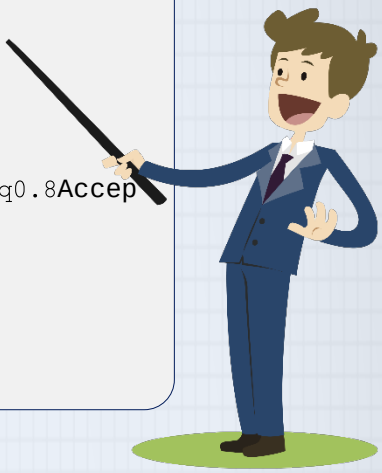
Exemplo de solicitação POST:

```
POSThttp://localhost/account/loginHTTP
/1.1Host:localhost

User-Agent:Mozilla/5.0Gecko/20100101Firefox/14.0.1

Accept:
text/html,application/xhtml+xml,application/xml;q0.9,*/*;q0.8Accep
t-Language:en-us,en;q0.5

Proxy-Connection:keep-
aliveReferer:http://lo
calhost/
```



Cabeçalhos de Solicitação

Os cabeçalhos de solicitação dão ao servidor informações adicionais sobre a conexão HTTP, o navegador, o idioma preferido e a codificação da resposta, etc.

Por exemplo, as solicitações AJAX* enviam o seguinte cabeçalho:

- X-Requested-With:XmlHttpRequest

Acessando os Dados Solicitados:

- a classe HttpServletRequest no Java fornece métodos para acessar parâmetros GET ou POST.
- NET fornece uma classe semelhante, mas em PHP tem que usar matrizes especiais $_GET, $_POST e a função apache_request_headers.
- Em‘.NET’ por exemplo, a propriedade HttpRequest.Params[name] vai encontrar o valor do parâmetro "name" pesquisando primeiro a sequência de consultas (parâmetros GET), então o formulário (parâmetros POST), em seguida os cookies e finalmente as variáveis do servidor (as definições de configuração do servidor).
- A partir de uma URL contendo um parâmetro (HTTP) que recebe um valor, adiciona¬-se %26 (codificação para &) e uma outra atribuição de valor, por exemplo, (par2=val2):
  - ✓ http://yahoo.com?par=val%26par2=val2

- Daí, a URL (e/ou link) será interpretada como:
  - ✓ http://yahoo.com?par=val&par2=val2

- O que acontece? Se a aplicação estiver vulnerável (como este exemplo do Yahoo Mail), o parâmetro e valor acrescentados serão interpretados, gerando uma resposta diferente. E assim ações maliciosas podem ser realizadas por meio do comando errado que é interpretado como certo.

Uma resposta comum HTTP:

```
HTTP/1.1 200 OK

Content-Type:
text/html;charsetutf-8Content-
Length: 1744

X-Xss-Protection: 1;
modeblockX-Content-Type-
Options: nosniffX-Frame-
Options:SAMEORIGIN

Server: WEBrick/1.3.1 (Ruby/1.9.3/2013-11-
22)Date: Mon, 04 Aug 2014 10:57:38 GMT

Connection: close
```

Status HTTP:

- Código 200 - significa que tudo ocorreu bem
- Código 404 - significa que o recurso não foi encontrado
- Código 403 - significa que o acesso foi proibido
- Código 302 - irá redirecionar o navegador a URL especificada no cabeçalho de localização
- Código 401 - irá exibir uma janela de loggin para autenticar ao servidor web.

Cabeçalhos de Respostas HTTP

Responseheader	Meaning
Content-Type	Fornece um tipo MIME*, dizendo ao navegador como lidar com o corpo da resposta. Alguns navegadores, mais notavelmente o Internet Explorer, ignoram esse cabeçalho e inspecionam seu conteúdo para determinar o tipo (farejando conteúdo).
X-Content-Type-Options	Com o valor nosniff**. previne a descoberta de conteúdo e usa o cabeçalho Content-Type.
Content-disposition	Informa ao navegador que ou exiba o conteúdo no navegador (com valor em linha), ou delega o processamento para um programa externo ou salvar em disco (com valor anexo).
Location	Com um código de status 302, o navegador irá abrir a URL especificada por esse cabeçalho.
WWW-Authenticate	Permite que o navegador pergunte por credenciais de autenticação.
Set-Cookie	Armazena informações em um cookie do navegador.
Cache-control	Informa se e como a informação é permitida ser armazenada em cache. No-cache significa que o conteúdo está sempre revalidado com o servidor, e no-store significa que os dados não são permitidos em serem armazenados em memória não volátil.

Senhas

Você deve entender sobre senhas e o acesso a contas sob a ótica da entropia e da mitigação.

A entropia da senha depende da dificuldade. A dificuldade pode ser aumentada ao acrescentar requisitos (números, letras, letras maiúsculas, símbolos), mas também ao aumentar o tamanho da senha (por exemplo, mais de 12 caracteres).

Mitigações para prevenir que invasores tentem todas as combinações (ataques brutais): número limitado de tentativas de autenticação, implementação de um atraso deliberado na autenticação para retardar invasores ou o bloqueio de contas após algumas tentativas (o que poderia levar a um ataque de Recusa de Serviço).

Injeção de Cabeçalho

Um duplo CRLF irá incluir uma linha em branco, após o qual o corpo da resposta pode ser dado. Tal ataque é chamado de um 'ataque de injeção de cabeçalho'.

O ataque irá apenas funcionar se os caracteres CRLF são passados como é em resposta HTTP.

Divisão de resposta HTTP - É uma técnica de redirecionamento de navegador que é usada para sequestrar uma sessão de navegador e roubar a informação ou introduzir o código no computador da vítima.

Modelo de Segurança do Navegador

É aplicado no acesso a elementos do documento (DOM elements), cookies, executar Xml Http Requests e acessar o armazenamento local do HTML5. Os detalhes dessa política diferem por navegador e até mesmo por versão de navegador.

O Protocolo Simples de Acesso a Objetos (SOAP) é um protocolo que envia mensagens XML sobre HTTP.

Gerenciamento de Sessão

- A ID se sessão deve ser difícil de se prever. Crie uma grande sequência de símbolos aleatórios para garantir isso.
- Depois de eftuar o logoff, a sessão deve ser (imediatamente) invalidada.
- A sessão deve ser invalidada depois de certo tempo de inatividade.

- A ID de sessão não deve ser reutilizada. Gerar uma grande sequência de símbolos aleatórios faz com que isso seja bastante improvável.
- Imediatamente após logar, a ID de sessão deve ser atualizada e a antiga ID de sessão, invalidada.
- Cookies devem ser usados, e não parâmetros CGI.
- Cookies de sessão devem ser enviados via canal criptografado.

SP Foundation

Log on

Iniciar sessão:
- ID de sessão longa;
- Não utilizar ID de sessão;
- Enviar cookies via canal criptografado.

Atualizar sessão

- Imediatamente após o log on;
- Invalidar sessão anterior.

Log out

- Invalidar sessão anterior.

Autenticação é uma medida para determinar se a identidade de um programa ou página da web com a qual nos comunicamos é autêntica.

Para garantir esse processo, o gerenciamento de sessão segura é fundamental.

O uso de cookies ao invés dos parâmetros CGI pode garantir que um identificador de sessão nunca seja usado como parte da sequência de consulta de uma solicitação.

- **Um invasor pode deixar alguém visitar um URL, como:**
  - ✓ https://bank.com/;jsessionid=ABCDEFG1234567

Fixação de sessão -> Ataque em que o invasor espera até o usuário fazer login e pode, então, acessar os dados da sessão com esse identificador da sessão

Resumo:
- O ID da sessão deve ser difícil de prever.
- Gere um grande ponto aleatório para garantir isso.
- Depois de encerrar a sessão, esta deve (imediatamente) ser invalidada.
- A sessão deve ser invalidada após certo tempo de inatividade.
- O ID da sessão não deve ser reutilizado.

- Gerar um grande ponto aleatório será muito improvável.
- Imediatamente após o login, o ID da sessão deve ser atualizado e o ID da sessão antiga deve ser invalidado.
- Cookies devem ser usados e não parâmetros CGI!
- Os cookies da sessão devem ser enviados através de um canal encriptado.

CSRF

Construir uma aplicação WEB requer muito além de apenas conhecimento em programação. É necessário construir uma aplicação segura, e isto é uma tarefa árdua, já que hoje em dia nos deparamos com diversos tipos de ataques que aproveitam de qualquer vulnerabilidade para roubar informações.

Se quisermos agir para deixar páginas seguras, é preciso conhecer os tipos de ataques que podem causar danos. Hoje vamos falar sobre o CSRF, analisando o seu conceito e funcionamento.

O que é

O CSRF (Cross-Site Request Forgery) é um tipo de ataque que tem como objetivo inserir requisições em sessões que já estejam abertas pelo usuário.

Em resumo, um ataque CSRF ocorre quando a vítima executa um script, sem perceber, no seu navegador, e este script explora a sessão iniciada em um determinado site.

Para que serve

Este ataque serve para o invasor fazer movimentações financeiras ou transações online, alterar senhas de acesso do usuário vítima, alterar dados pessoais da vítima.

As aplicações vulneráveis ao CSRF são exploradas por meio de algumas etapas, como por exemplo:

- O usuário recebe um link contendo o aplicativo malicioso
- Usuário usa o mesmo navegador e acessa um aplicativo malicioso
- O link malicioso que foi navegado pelo usuário inclui uma determinada requisição na aplicação que será alvo do ataque, e carrega os parâmetros para executar uma transação.

Como funciona e exemplos

Para entendermos melhor esta vulnerabilidade, vamos exemplificar, conforme a seguir:

O atacante pode fazer uma referência a alguma URL dentro de um outro aplicativo, convidando a sua vítima a fazer acesso a esta URL, sem que ela saiba a real intenção do seu ataque.

Uma tentativa, seria usar um link ou imagem enviado por e-mail, como por exemplo: “Visite minhas fotos”.

Exemplo:
João envia para Ana um e-mail, contendo uma referência maliciosa, por meio de uma imagem camuflada ou um link:

- ✓ <img src= http://banco.co/transferir.do?acct=ataque=1000 width= “1” height= “1”>

Ou

- ✓ <a href=http://banco.co/transferir.do?acct=ataque=1000>

Ana não irá perceber nada de errado, e irá acessar a referência e o seu navegador submeterá a requisição sem que a vítima perceba, e se Ana estiver logada em seu internet banking, a transação maliciosa será concluída com sucesso.

Algumas formas mais comuns utilizadas para explorar aplicações vulneráveis ao CSRF são:
- Por meio das TAGs do HTML, como: <script src=”http://host/?command”>
- Por meio de códigos do javascript, inserindo scripts dentro da aplicação que esteja vulnerável.

Solução
Uma forma de prevenir ataques do tipo CSRF é adotando um tipo de Token, que são inseridos no documento HTML, e que geram uma identificação aletarória, autenticando as informações armazenadas na sessão para cada vez que uma transação for executada.

Em resumo, quando algum formulário for submetido será garantido que o token estará presente, e que o seu valor será coincidente com o valor do token que estiver guardado em sessão

Algumas linguagens de programação já possuem funções específicas para gerar tokens longos e aleatórios.

Clickjacking
Clickjacking pode ser traduzido como 'furto de clique'. E é considerado um dos mais fortes tipos de ataques envolvidos contra a Internet.

Neste contexto tão tecnológico, a criatividade de invasores parece não ter um fim, com isso, novos tipos de ameaças surgem, e se o usuário não ficar atendo com as dicas de segurança pode acabar se tornando um alvo fácil.

O que é e para que serve
O Clickjacking é uma classe de vulnerabilidade que afeta maior parte dos navegadores de internet, como por exemplo, o Firefox, Chrome, internet Explorer, opera e safari. Com este tipo de ataque é possível que o invasor controle Webcam e microfone de suas vítimas.

Riscos ao usuário
O principal risco que o usuário está exposto é o Phishing, onde o usuário pode ter informações e dados relevantes roubados e usados de forma indevida. Outro risco é o uso indevido da senha de rede social e do Internet Banking do usuário afetado.

Como funciona
O invasor consegue infectar botões de um site legítimo, como o site da Netflix por exemplo, e quando o usuário clica neste botão ele acaba baixando de forma inconsciente diversos malwares ou é direcionado para página com conteúdos maliciosos.

Como podemos perceber, o atacante consegue infectar o computador do usuário sem que o mesmo perceba a ação maliciosa.

Outra forma de atacar com Clickjacking é através do facebook, onde links com frases curtas e chamativas tentam atrair a atenção do usuário, que acaba clicando sobre o link e é direcionado para página que pode baixar arquivos maliciosos para o dispositivo do cliente.

Exemplos:
Vamos exemplificar um pouco a situação, para entendermos melhor esta forma de ataque usando o Clickjacking.

O usuário mal-intencionado de alguma forma consegue criar um falso link em um botão de um site confiável.

Um usuário comum entra neste site e clica sobre o botão infectado. Sem ter conhecimento do risco, um Malware é instalado em seu computador e este usuário é direcionado para uma outra página que nada tem a ver com a página legítima.

Com isto o atacante consegue instalar na máquina do usuário programas que podem roubar senhas.

Um clássico exemplo, é quando um usuário posta anúncios maliciosos no Facebook sem saber. Seus amigos veem a Postagem maliciosa como se a própria pessoa tivesse postado. Isso indica que o usuário foi vítima deste ataque.

Dicas de defesas

Diferente de outros tipos de vulnerabilidades, o clickjacking não resulta de uma implementação errada no código fonte da linguagem de programação.

Geralmente é o mau uso de algumas características dos recursos do HTML e CSS combinados com a interação do usuário com elementos transparentes é que acabam resultando este tipo de ataque. O navegador é o principal recurso usado para este ataque.

No lado do usuário a dica para evitar esta vulnerabilidade é manter sempre o browser atualizado e evitar clicar em anúncios e websites que sejam de procedência desconhecida.

No lado de desenvolvedores, as técnicas mais adotadas são a utilização de quebra de frames e o uso de cabeçalho HTTP X-frame-options.

SQL Injection

O que é

O SQL Injection ou Injeção de SQL, é um termo que indica um tipo de ameaça, que usa falhas existentes em sistemas, para interagir com o banco de dados dos mesmos através de comandos SQL.

Pra que serve

Vivemos em um cenário onde temos diversas informações armazenadas em algum banco de dados. Quando aplicações que são acessadas pela internet usam esses banco de dados, elas se tornam alvo de ataques do tipo SQL Injection.

Este tipo de ataque serve para alterar e manipular informações do banco, comprometendo a integridade dos dados armazenados, podendo causar um grande transtorno.

Como funciona

Ao acessar uma aplicação via web, se o sistema apresentar falhas de segurança, a pessoa poderá acessar algum formulário do site e passar instruções SQL, através do local destinado para o usuário digitar informações.

Com isso, a pessoa consegue alterar diversos dados na aplicação, sem possuir o devido acesso ou autorização. Isso se trata de um ataque que pode causar muitos danos ao banco, mas que pode ser evitado com o uso de boas práticas de programação, que possibilitam otimizar o processo de segurança da informação.

As boas práticas podem ser implementadas no próprio servidor de banco de dados, como também

podem ser implementadas dentro do código fonte, independente da linguagem de programação utilizada.

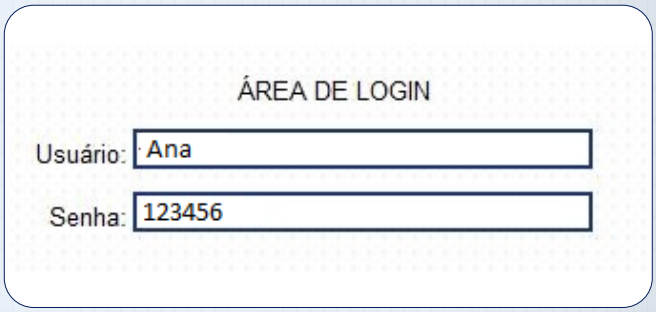
Para entender melhor o funcionamento de um ataque com SQL Injection, vamos ver alguns exemplos, para esclarecer melhor a questão.

Exemplos:
Com objetivo de exemplificar o funcionamento do SQL Injection, vamos analisar a situação apresentada a seguir.

Temos uma tela de Login. Considere que a autenticação desta tela é validada com a seguinte instrução SQL:

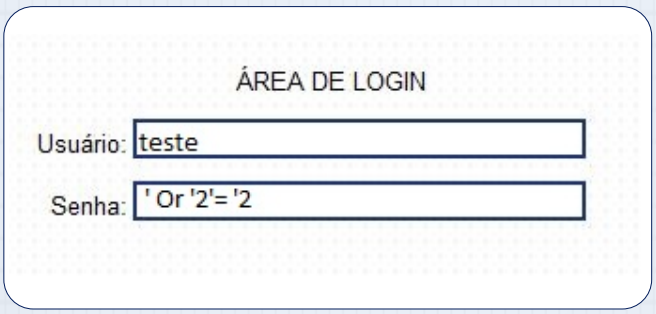
- SELECT * FROM tb_usuarios WHERE user = 'campo_usuario' AND pass = 'campo_senha'

Esta consulta busca no banco de dados um usuário que contenha as respectivas informações digitadas pelo usuário.



No caso, a consulta buscaria no banco, usuário com nome de acesso 'Ana' e senha '123456'.

Mas imagine outra situação, a pessoa mal intencionada deseja verificar a vulnerabilidade da aplicação. Se esta pessoa digitar uma informação, como exemplificado na tela a seguir, ela obterá um acesso indevido, podendo prosseguir com o seu ataque:



Neste caso, teríamos uma consulta da seguinte forma:

- SELECT * FROM tb_usuarios WHERE user = 'teste' AND pass = ' ' or '1' = '1'

O comando ' ' or '1' = '1' faz com que o usuário e senha informados sejam sempre verdadeiros, permitindo assim o acesso indevido ao sistema.

Para evitar este tipo de ataque, na linguagem de programação deve ser implementando funções que validem os dados de entrada, visando impedir a execução de comandos indevidos.

Exemplo básico de um código PHP para tratar a execução de querys e evitar o SQL injection:

```
<? php
        $usuario = $_POST['user'];
        $senha = $_POST['pass'];
        $user_escape =addslashes($usuario);
        $pass_escape = addslashes($senha);
$query_string = "SELECT * FROM tb_usuarios WHERE user = '{$user_escape}' AND senha = '{$pass_escape}'";
?>
```

A função addslashes() adicionará uma barra invertida antes de cada aspa simples e aspa dupla encontrada. Com esse tratamento, a query resultante seria:

- SELECT * FROM usuarios WHERE codigo = '' AND senha = '\' or 1=\'1'

Evitando assim que o usuário consiga o acesso indevido. Outra dica importante é evitar de exibir mensagens de erro em um servidor de aplicação que esteja em produção, pois nessas mensagens de erros e alertas podem ser exibidos caminhos de diretórios de arquivos ou outras informações importantes sobre o esquema do banco de dados, comprometendo a segurança da aplicação.

Mais informações e exemplos, veja em:
Vazamentos de senha via injeção SQL. Bijvoorbeeld:

“Violação custa quase U$1 milhão ao LinkedIn; outros U$2-3 milhões serão gastos em atualizações" (http://www.zdnet.com/article/breach-clean-up-cost-linkedin-nearly-1-million-another-2-3-million-in-upgrades/)

"LinkedIn perde U$5 milhões em ação coletiva sobre senhas vazadas" (https://nakedsecurity.sophos.com/2012/06/21/linkedin-slapped-with-5-million-class-action-suit-over-leaked-passwords/)

Mais exemplos em http://codecurmudgeon.com/wp/sql-injection-hall-of-shame/

Consultas diretas e parametrizadas

Um mecanismo chamado consultas parametrizadas faz exatamente isso:

- Primeiro, uma instrução SQL que contém espaços reservados é cedida para a rotina de banco de dados, onde é analisada.

- Em seguida, a entrada de usuário é substituída dentro desses espaços reservados.

Um formulário não parametrizado de consultas SQL são solicitações de consultas diretas.

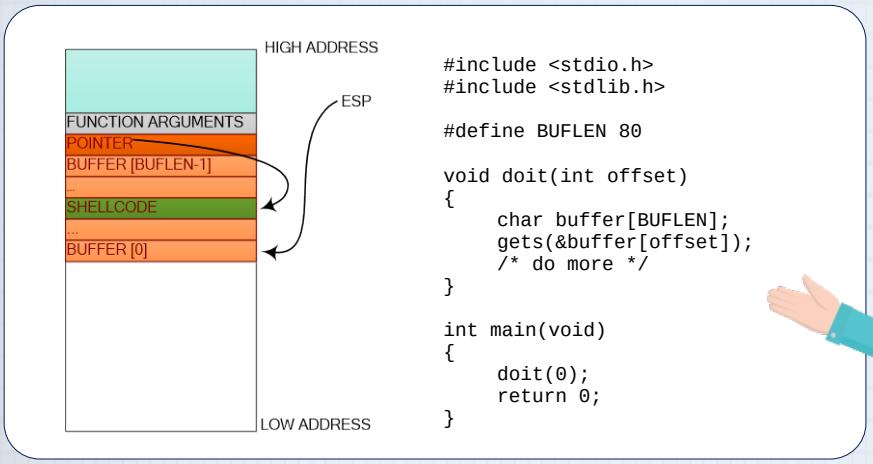
Consultas parametrizadas são, às vezes, solicitações de declarações preparadas.

Vejamos esse exemplo:

```
query=connection.prepare("SELECT * FROM people WHERE firstname
= ?")
connection.execute(query,$post.firstname)
```

A função de preparar irá analisar a consultar. Depois da análise, o valor de $post.firstname será colocado no espaço reservado.

Estouro de Buffer




```
#include <stdio.h>
#include <stdlib.h>

#define BUFLEN 80

void doit(int offset)
{
     char buffer[BUFLEN];
     gets(&buffer[offset]);
     /* do more */
}

int main(void)
{
     doit(0);
     return 0;
}
```

Outras formas de Injeção e de Saída

Não só SQL é usado em ataques de injeção, as injeções podem acontecer em qualquer sistema que interpreta (parte) da entrada como declarações. Exemplos:

- Injeção de comando, onde os dados são interpretados por um comando, como:
  - ✓ johndoe; rm-rf/

- Injeção-XML, onde os dados são interpretados como etiquetas, por exemplo:
  - ✓ peter</name><role>admin</role></user><user><name>susan

Nesse caso, mecanismos, como as consultas parametrizadas, podem não existir. Portanto, nós devemos recorrer à técnica de saída de meta-caracteres. Essa técnica pode também ser usada para sistemas SQL onde consultas parametrizadas não são implementadas.

Meta-caracteres são caracteres que possuem um significado especial.

Em SQL, por exemplo, o caractere de aspas simples (') irá indicar o início ou o fim de uma sequência.

Em XML, os colchetes (< e >) são meta-caracteres, assim como o E comercial (&).

A saída é uma técnica para neutralizar esses meta-caracteres.

Em XML, para escrever um E comercial, usa-se a sequência &.

A saída irá impedir que os meta-caracteres sejam interpretados e, assim, evitar ataques de injeção.

Expressões regulares

Expressões regulares são úteis para combinar padrões em um texto...

Combinações: gananciosos x relutantes:

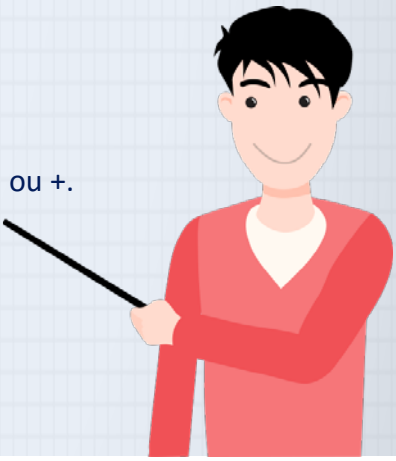
fragmento[1] XML

.*<name>(.*)</name>.*

<doc><name>Susan</name><name>Peter</name></doc>.

Existem duas maneiras de combinar padrões que contêm o operador * ou +.

Susan</name><name>Peter



Âncoras em expressões regulares:

[A-Za-z]{4}\d{4}\.txt

HACK2014.txt ; reboot

A âncora ^, no começo, indica o começo da sequência, enquanto que no final a âncora $ indica o fim da sequência.

Se tivéssemos usado a expressão regular:
^[A-Za-z]{4}\d{4}\.txt$

Então a sequência maliciosa simplesmente não combinaria.

Higienização e Filtragem

Higienização = arrumar a entrada maliciosa.

Apesar do fato que a entrada maliciosa possa ainda ser introduzida (a sequência ..../ será substituída por ../, que ainda é maliciosa), há um problema mais fundamental: se nós sabemos que a entrada é maliciosa, por que continuar processando-a?

Obter um código mais legível ou simples preguiça são desculpas inaceitáveis do ponto de vista da segurança.

Portanto:

- Se validarmos a entrada, certifique-se que só a entrada aceita é processada.
- Higienização tem o seu uso, mas tenha cuidado.

Interpretação Errada

Se duas partes de um sistema processam a mesma entrada, mas de forma diferente, uma questão de segurança pode aparecer. Isso pode acontecer em vários níveis.

Na codificação Shift-JIS, os bytes 0xEA 0x5C compõem o caractere 鶴 (CJK UNIFIED IDEOGRAPH-9DED). O banco de dados vai interpretar esses bytes como aquele caractere. Nossa função de saída interpretará essa sequência como dois caracteres individuais, onde o último é uma barra invertida (\). Agora, olhe como a seguinte sequência é processada:

- sequência [0xea]' or 1=1--
- em bytes [0xea]\' or 1=1--
- saída [0xea]\\' or 1=1--database sees [0xea]\\' or 1=1--

É importante assegurar que o sistema inteiro usa a mesma codificação de caracteres, URF-8 é uma boa escolha. Se isso não for possível, forneça funções de conversão e certifique-se que elas são seguras.

Em até um nível menor, pode haver uma diferença em como as sequências são representadas. A linguagem de programação Java (e muitas linguagens de script) aceita o byte nulo (0x00) como uma sequência de caracteres válida. Nas linguagens de programação C e C++, o byte nulo é usado para indicar o fim da sequência. Isso pode levar a resultados inesperados se no código a seguir, o nome do arquivo variável contiver o byte nulo:

- iffilename.endsWith(".png") {File f= new File(filename) }

Então, vamos pensar que o invasor injeta a sequência /etc/passwdNulpng (se ela foi enviada como um parâmetro CGI, a sequência real seria semelhante a etc/passwd%00.png).

A primeira linha do código verifica se o nome do arquivo termina com .png, que é realmente o caso. Logo, o valor é passado para o construtor de arquivo, que abre um arquivo, usando o código que está escrito no C.

No mundo-C, a sequência termina no byte nulo, e, portanto, o programa irá abrir o arquivo /etc/passwd.

XSS

O que é XSS?

XSS é um tipo de vulnerabilidade que pode ser encontrada em aplicações web, e que permite inserir códigos no lado do cliente (altera a página no computador do usuário).

Este ataque pode ser subdividido em três tipos de categorias: Refletido, Armazenado, baseado em DOM.

Para que serve

XSS é uma vulnerabilidade que pode causar desde um simples alerta na tela até um sequestro de sessão ou redirecionamento para outros sites de tipos maliciosos.

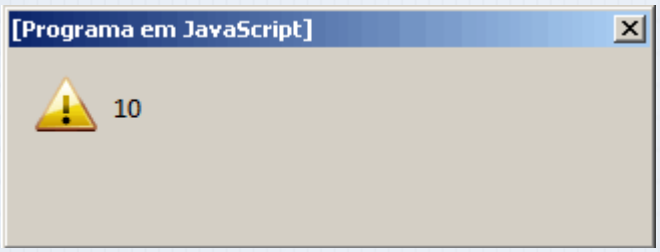
Como funciona e exemplos

XSS Reflected:

O ataque refletido é quando o servidor da página web reflete aquilo que enviamos sem filtrar o conteúdo que o usuário digitou. Suponha que um usuário comum acesse uma página e digite seu usuário e senha:

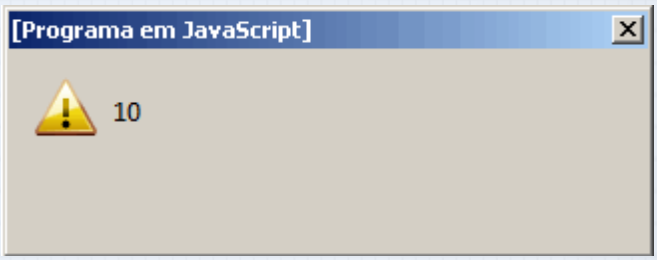
1) Usuário acessa site http://exemplo.com.br
2) página solicita que entre com usuário e senha
3) Usuário digita um user não existente, chamado "João"

4) A página exibirá na tela a mensagem: "O usuário João não está cadastrado em nossa base"
Porém se um usuário mal intencionado acessar a página, ele tentará verificar a vulnerabilidade digitando um script:
1) Usuário invasor acessa site http://exemplo.com.br
2) página solicita que entre com usuário e senha
3) Usuário digita um user não existente, chamado "<script>alert(10)</script>"
4) A página exibirá a mensagem de usuário não existente, e exibirá uma caixa de mensagem gerada pelo script.



XSS Stored:
É quando o código que será injetado não foi filtrado e está armazenado, assim, quando alguma página WEB for exibir o conteúdo armazenado, o XSS será disparado.

Se um usuário malicioso deseja verificar a vulnerabilidade da página, o procedimento será:
1) Acessar algum site, http://exemplo.com.br
2) O site solicita que entre com o usuário e senha para fazer o cadastro
3) Usupreenche no nome a informação: <script>alert(10)</script>, e no campo senha coloca: teste.ário
4) Após o cadastro, será exibida a mensagem 'seja bem-vindo', e exibirá uma caixa de alerta com o script digitado.

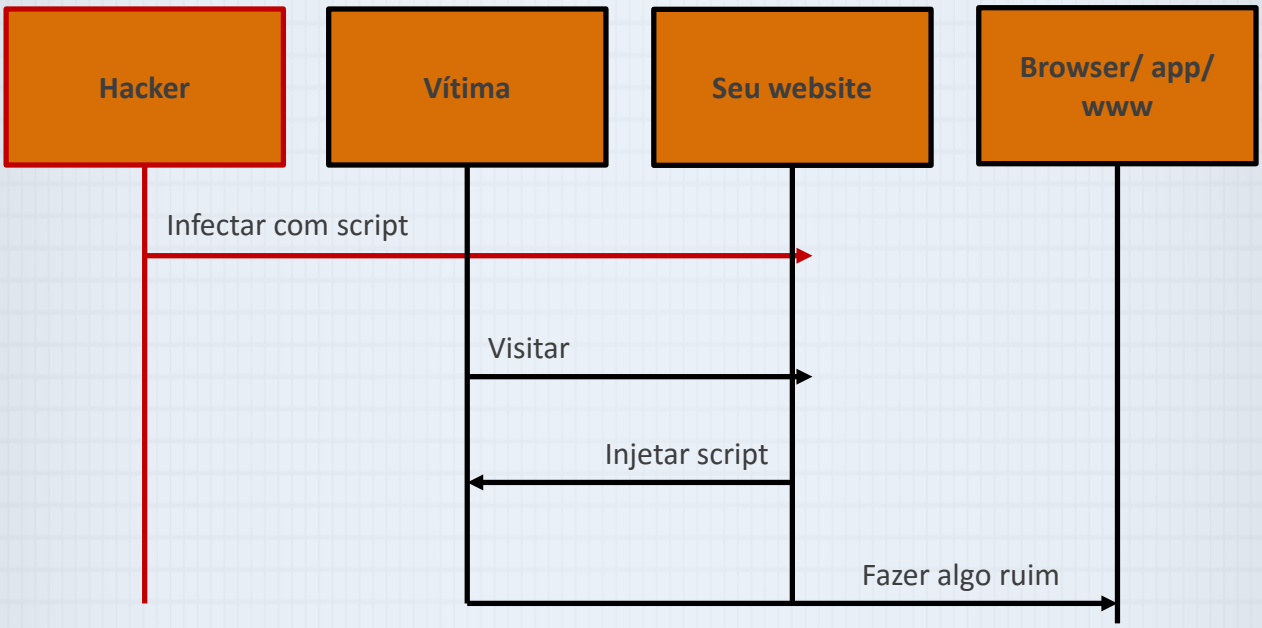


XSS DOM Based:
Este tipo é dependente das vulnerabilidades em algum componente da página, onde o script irá alterar o HTML da página usando manipulação DOM (Document Object Model).

As consequências destes ataques podem implicar em roubo de informações confidenciais que estejam em um cookie, o invasor pode realizar ataques de pishing, dentre diversas outras ações que podem causa danos na segurança do usuário.

Ações como filtrar o dado que o usuário está digitando, verificar os caracteres '<', '>' , '-' ao imprimir o dado, são ações que ajudam a combater este tipo de ataque.

Ilustração de como funciona:



Em um típico ataque XSS, o hacker infecta um website normal, bem intencionado, com um script malicioso (do lado do cliente).

Quando um usuário visita a página, o script é baixado em seu navegador e executado, o que usulmente resulta em fazer algo ruim ou não desejado na web. O maior risco é que o invasor pode fingir ser a vítima e fazer qualquer coisa no lugar dela dentro de uma aplicação. No caso de um navegador, isso pode significar muita coisa.

Parte II

Objetivos

Os objetivos da PARTE II do cuso Secure Programming Foundation é:

- reconhecer a diferença entre autorização horizontal e vertical.
- reconhecer a diferença entre referências diretas e indiretas.
- reconhecer envenenamento de sessão e condições de corrida.
- justificar a necessidade de endurecimento.
- reconhecer métodos de endurecimento.
- reconhecer diferentes vazamentos de informação.
- explicar a importância do registro para a segurança.
- explicar o princípio de 'Falhar com Segurança'.
- reconhecer ataques por recusa de serviço e mitigações.
- explicar a importância do Princípio de Kerckhoffs, Gerenciamento de Chaves e Aleatoriedade.
- descrever a criptografia de chave pública, ataques 'Man-in-the-Middle' e certificados.
- reconhecer as ameaças a SSL/TLS/HTTPS.
- aplicar HTTPS corretamente.
- identificar requisitos de segurança em falta.
- reconhecer ambiguidades e pressupostos ocultos em determinadas condições e contextos.
- reconhecer ameaças que são inerentes a uma arquitetura específica.
- reconhecer soluções adequadas a ameaças e as imperfeições nessas soluções.
- reconhecer o escopo, objetivo e vantagens da revisão de código para as práticas de desenvolvimento.
- lembrar diferentes métodos para testes de segurança.
- reconhecer o melhor teste para um determinado cenário.
- identificar formas de melhorar o desenvolvimento de software e processos de testes, incorporando resultados de testes.

Autorização Horizontal e Vertical

Autorização horizontal:

Diagrama azul: Jornalista A trabalha em um artigo que pertence ao Jornalista B. Em uma autorização horizontal, podemos checar se uma certa operação pode ser realizada em um certo objeto. Por exemplo, o editor do jornal é o único com permisão de ler, modificar e publicar artigos dentro de uma certa seção do jornal. (Acesso a objetos não autorizados).

Cuidado: Jornalista A não assume a identidade do Jornalista B (problema de autorização), mas mexe com um artigo que não é seu.

Autorização Vertical:
Cenário amarelo: Jornalista A publica seu próprio artigo, o que não seria permitido. Em uma autorização vertical, podemos checar se uma pessoa tem autorização de realizar certa ação, independentemente do objeto sobre o qual a ação é realizada. Por exemplo, um editor de jornal tem permissão de ler, modificar e publicar artigos, quando um jornalista só tem permissão de ler e modificar artigos, mas não publicá-los. (Escala de privilégios.)

	Editor	Jornalista A	Jornalista B
Criar	X	√	√
Ler	√	√	√
Modificar	√	√	√
Publicar	√	X	X

Riscos
Escalada de privilégios e acesso a objetos não autorizados = quando as verificações de autorização são esquecidas de serem implementadas.

Muitas vezes, uma matriz de autorização é utilizada para que as listas com as funções fiquem em um eixo e as permissões em outro. Mas, isso mostra apenas os requisitos de autorização vertical. E as chances de que a autorização horizontal e outras condições de autorização sejam ignoradas são bem relevantes.

Atenuações
O método óbvio para impedir problemas de autorização é simplesmente implementar as verificações de autorização.

Uma segunda medida é garantir que seu código seja executado com o menor privilégio possível. Isso é chamado de princípio do menor privilégio.

O acesso a objeto não autorizado, muitas vezes, acontece porque um objeto é referenciado através

de um identificador absoluto, tal como um número.
Referência indireta = quando um aplicativo para de referenciar objetos através de um identificador absoluto, mas aceita apenas um índice relativo em uma lista de identificadores permitidos e não há possibilidade de um invasor acessar objetos de outras pessoas através do identificador.

Ilustração de uma referência direta:

- Um sistema de administração de estudantes permite que os professores editem dados do estudante.
- Um professor só tem permissão de editar dados de estudantes de sua classe.
- Os parâmetros de solicitação de edição contêm um número absoluto do estudante.

Condições de Corrida (ataque TOCTOU)

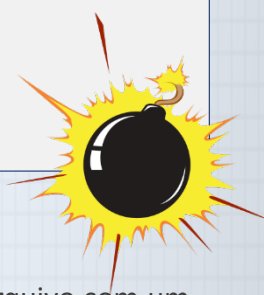
Ao criar um arquivo temporário, ele não precisa existir ainda, então o processo pode ser criá-lo e configurar as permissões adequadas antes de gravar dados sensíveis.

Se esses passos acontecem em duas ações separadas, um outro processo poderia realizar operações sobre o mesmo recurso entre o tempo de verificação (TOC) e o tempo de uso (TOU).

Assim, existe uma janela de tempo entre esses dois passos durante o qual um invasor pode acessar o recurso e anular as condições que são verificadas. Tal problema é chamado de condições de corrida ou ataque TOCTOU.

Exemplo:

```
if (!access("/tmp/mydata.txt",W_OK)) {
    // TOC: check if we have write permission
    // ...
    // ATTACK WINDOW
    f = fopen("/tmp/mydata.txt","w+");
```



- Invasor cria o arquivo /tmp/mydata.txt e inicia o programa.

Logo, após o programa ter executado a verificação de acesso, o invasor substitui o arquivo com um link simbólico que aponta para outro arquivo, por exemplo /etc/passwd, o arquivo de senha.
O programa irá gravar os dados para o arquivo de senhas, um arquivo que o invasor, de outra forma, não seria capaz de editar.

‘As permissões são absolutamente vitais quando se compartilham recursos! ‘; podemos deixar agora assim: ’As permissões são absolutamente vitais quando se compartilham tempo!’

Mas, para evitar essas condições de corrida o melhor é combinar o TOC e o TOU em uma única operação sem interrupção; técnica essa que podemos chamar de operação atômica. E assim não perdemos mais tempo!

Envenenamento de Sessão

Ataque de envenenamento de sessão em aplicativos web.

Exemplo de cenário em uma loja web:

- Um cliente seleciona os itens a serem colocados no carrinho e então passa para a página de checkout.
- Na página de checkout, o montante total é calculado e os custos de frete são determinados baseados no número total de itens.
- Esses valores são armazenados em dados da sessão do aplicativo.
- Antes de concordar com o pagamento, contudo, o cliente acrescenta mais itens ao carrinho de compras, repetindo a solicitação "adicionar ao carrinho".
- Então, o cliente solicita a página final de pagamento, ignorando a página de checkout e um novo cálculo dos custos totais e do custo do frete.

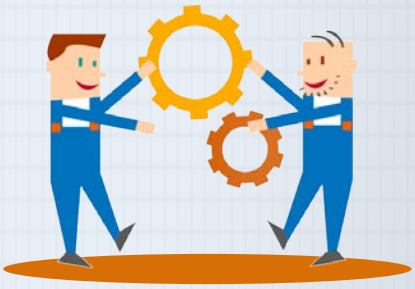
Se os dados da sessão contiverem os valores anteriores, o cliente paga menos do que deveria.

Esse problema é conhecido como envenenamento de sessão ou poluição de sessão.

A causa subjacente é a má gestação do estado da sessão.

Configuração e hardening

Hardening - processo de tornar seguro um software, utilizando outras formas que não seja programando.



Como Hardening pode ser feito?

- Alterando as definições de configuração;
- Aplicando atualização de segurança ou ;
- Protegendo partes vulneráveis do software.

O que um processo de hardening inclui?

- Aplicar as últimas correções de segurança.
- Restringir o acesso a características perigosas (por exemplo, através de um firewall).

- Configurar o privilégio de acessos (permissões de arquivo em seu servidor, funções em seu banco de dados).
- Desativar ou remover os recursos de depuração.
- Remover arquivos desnecessários (arquivos de backup, arquivos de gerenciamento da versão).
- Alterar as senhas padrão.

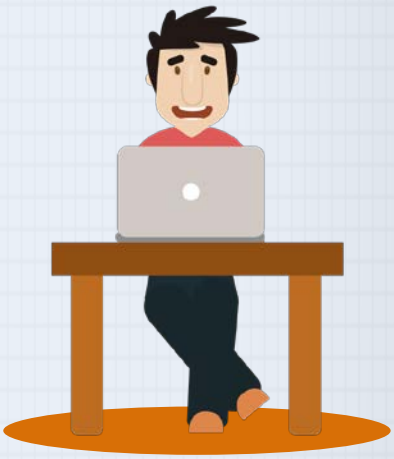
O que um processo de Hardening tenta fazer?

- Um processo de hardening tenta tanto reduzir a superfície de ataque, como reforçar as partes vulneráveis do sistema.

Para detectar problemas de hardening, deve-se executar um escaneamento de vulnerabilidade. E a boa notícia é que existem muitos escaneamentos de vulnerabilidade disponíveis no mercado, tanto de forma gratuita ou paga.

Componentes de Terceiros, Configuração e Endurecimento

- **Vamos conferir alguns exemplos:**



- Aplicando os patches de segurança mais recentes;
- Restringindo o acesso a funcionalidades perigosas;
- Configurando privilégios de acesso;
- Desativando ou removendo bugs;
- Removendo arquivos desnecessários;
- Alterando senhas pré-definidas.

Vazamentos de Informação

Vamos conferir alguns exemplos:

- Cabeçalhos de resposta HTTP vazam informação sobre a versão, permitindo que o invasor busque por vunerabilidades específicas.
- Páginas de erros que exibem o rastreamento de pilha mostram exatamente como o erro aconteceu, dando ao invasor dicas de como isso poderia ser explorado.
- Endereços IP internos e nomes de patches em servidores, ajudam os invasores a definir novos alvos depois que um sistema fica comprometido, ou se um invasor utiliza o navegador de alguém para realizar um ataque a partir do interior.
- Comentários em páginas HTML mostrando nomes e endereços de email ajudam a executar um ataque de engenharia social. (Nesses ataques, o invasor explora a fraqueza humana para convencer as pessoas a realizarem ações específicas que auxiliam a invasão).
- Arquivos de gerenciamento de versão podem vazar códigos de fonte da aplicação.
- Vazamentos de informação ajudam o invasor a atacar mais eficientemente.

Manejo e Registro de Erros

- Está relacionado com o conceito de Falhar com Segurança!
- Garantir então que sejam registrados todas as informações necessárias para detectar um ataque.

Um Tratamento de erros apropriado é essencial para a segurança sempre que um erro ocorre!

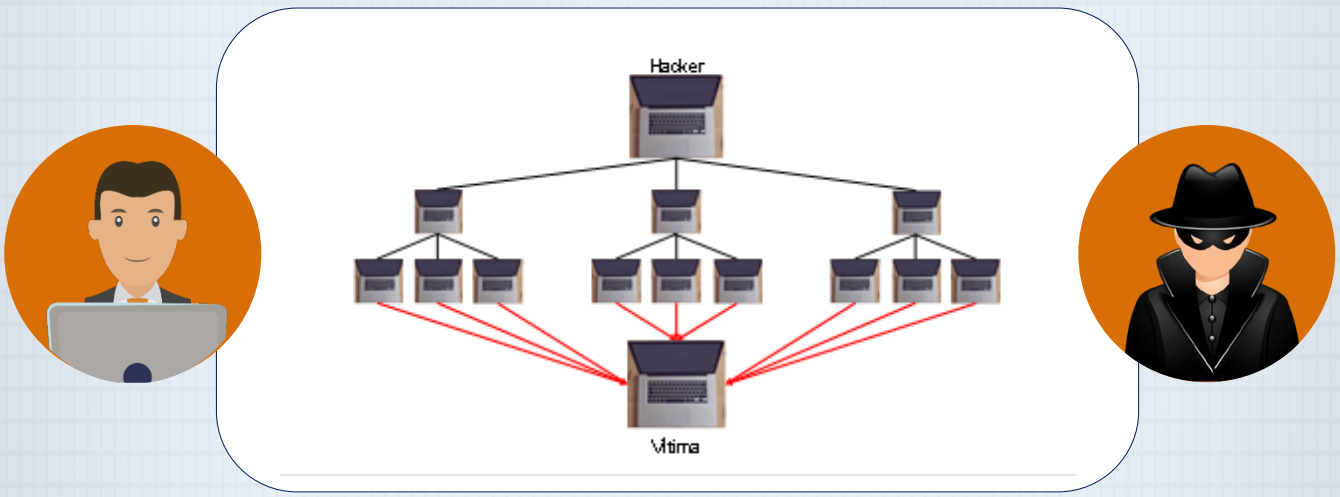
A solução é pegar todos os erros potenciais e se certificar que o código não irá ser deixado em um estado inseguro. Esse é o chamado princípio de falha da segurança.
Em linguagens com manipulação de exceções, você pode se beneficiar da construção try-catch-finally, se for suportado.

Logging

Logging não é só uma excelente forma de diagnosticar problemas, mas também serve para detectar ataques em um sistema, desde que a informação adequada esteja registrada.

Mas, que informações devem estar registradas?

Recusa de Serviço (Distribuído)



Em resumo os ataques DdoS funcionam da seguinte forma: o invasor garante que muitos outros invasores façam requisições a partir de um único servidor web, fazendo com que este pare de responder. Por exemplo, através da criação de computadores escravos para aumentar seu número de solicitações.

Princípio de Kerckhoffs

"Um bom algoritmo de criptografia não deve depender do sigilo do próprio algoritmo. A única parte que precisa permanecer em sigilo é a chave."

Auguste Kerckhoffs, 1883

Criptografia de Chave Pública

A criptografia pode ser usada para implementar: confidencialidade, integridade, autenticação e não rejeição, usando diferentes técnicas tais como codificação e assinaturas digitais.

Criptografar é uma forma de se comunicar com segurança em um canal inseguro (onde um invasor pode espionar).

- **Criptografia simétrica:** Todos os participantes usam a mesma chave para codificar e decodificar dados. Os participantes devem trocar a chave antes que os dados possam ser decodificados.
  - ✓ Como nós podemos trocar a chave de forma segura se nós ainda não temos estabelecido um canal seguro?

- **Criptografia assimétrica (criptografia de chave pública):** A criptografia assimétrica tenta resolver o problema de troca de chaves.

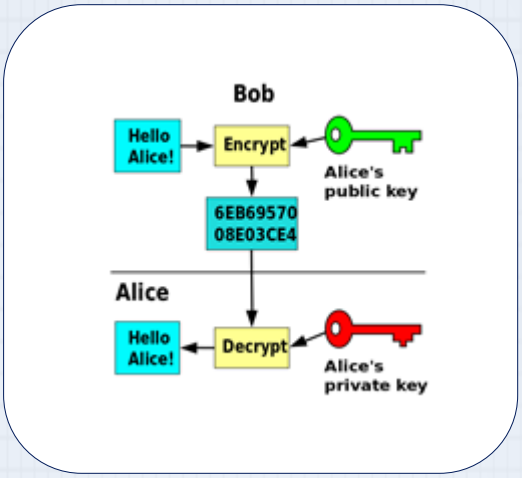
- Ataque homem-no-meio:
  - ✓ Eva se certifica que Alice acredita que a chave pública de Eva é a de Bob.
  - ✓ Alice criptografa uma mensagem usando a chave pública de Eva.
  - ✓ Eva intercepta a mensagem e decodifica-a usando sua chave privada.

- Depois de ler e modificar a mensagem, ela a criptografa novamente usando a chave pública real de Bob.

- Bob recebe a mensagem e a decodifica usando sua chave privada.

- Bob fica chocado ao ler a mensagem que ele acredita vir de Alice.

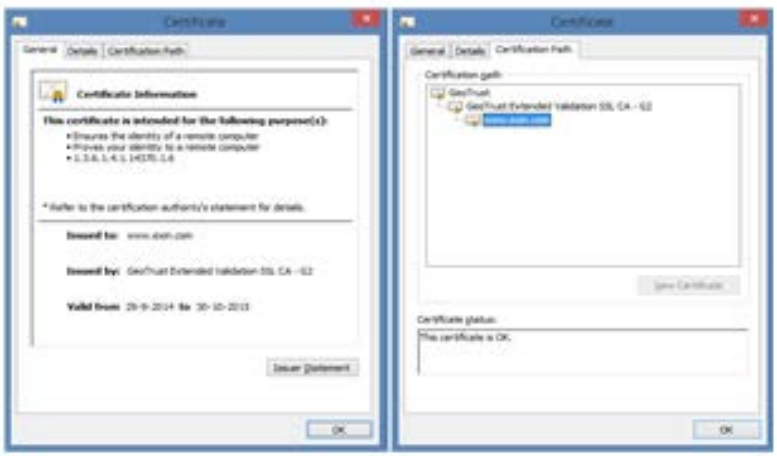
Entenda resumidamente como funciona a criptografia conforme ilustração:



HTTPS

O HTTPS é uma medida contra acessos não-autorizados. Exemplos são os certificados de HTTP para garantir autoridades confiáveis.

Exemplo de certificado de segurança para o uso do HTTPS, assim, reduzindo o risco de ataques XSS:



Ameaças a SSL/TLS/HTTPS

Com o avanço da tecnologia, cada vez mais pessoas realizam operações pela internet, como por

exemplo, compra online, transações bancárias, pagamento de contas, dentre outras. Todas estas atividades precisam de garantia de segurança, de forma que possamos realizar todas estas tarefas com confiança.

É justamente por causa desta necessidade de segurança, que foram desenvolvidos protocolos para minimizar riscos nas informações dos usuários. Ainda que estes protocolos sejam robustos, existe a possibilidade de ataques e ameaças. Vamos ver mais sobre isso adiante.

O que é

Mas afinal, o que são os protocolos SSL / TLS / HTTPS?

SSL (Secure Socket Layer): É um certificado que possibilita que dois computadores se comuniquem de forma segura. O SSL implementa privacidade, integridade e autenticação. O SSL é usado para proteger protocolos TCP/IP. Como por exemplo: HTTPS, SSH, FTPS, etc.

TLS (Transport Layer Security): foi desenvolvido depois do SSL, e seu uso é mais frequente nos programas de e-mails, e usa mecanismos de criptografia mais fortes.

Para saber se você está navegando em um site seguro, basta observar na barra de endereços do navegador, se o link da página possui as letras https(https://www.xxxxx.com.br) ou se na barra de endereços contém um símbolo de um cadeado.

Requisitos de Segurança

Dois lados diferentes precisam se comunicar sobre um tópico em que nenhum deles é especialista. O que fazer?

Ataque	Propriedade de segurança	Exemplo de medida
Imitando	Autenticação	Senhas
Adulterando	Integridade	Manipulação de entrada
Rejeição	Não rejeição	Logging
Divulgação de informações	Confidencialidade	Codificação
Negação de serviço	Disponibilidade	Limitar o consume de recursos
Elevação de privilégios	Autorização	Realizar verificações de autorização

Qualquer outro valor é um identificador de produto inválido.

Desenvolvedor -> implementa uma verificação de validação de entrada.
Testador -> testa o que acontece se um identificador de produto inválido for introduzido.

Exemplo de requisitos para cada propriedade STRIDE:

Autenticação
- Postar comentários pode apenas ser feito depois da autenticação.
- Somente usuários registrados podem abrir uma caderneta de poupança.
- Apenas os administradores podem mudar as informações do usuário.

Integridade
- Modificação de dados de pagamento na solicitação para o corretor de pagamentos deve ser detectada e, nesse caso, o pagamento deve ser rejeitado.
- Depois de uma transferência de dinheiro entre duas contas, o montante total nas duas contas deve ser igual ao montante antes da transferência.

Não rejeição
- Todas as edições do artigo devem ser registradas: a data e a hora, o editor e o número de linhas que foram alteradas.

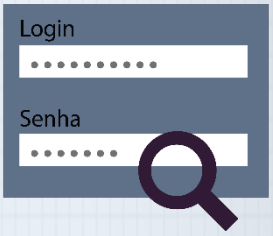
Confidencialidade
- As informações de cartão de crédito no aplicativo devem apenas serusadas para processar o pagamento e nunca ser acessíveis por qualquer pessoa que não seja o usuário.

- Não deve ser possível detectar se alguém é um membro de nosso site web "alcoólicos anônimos".

Disponibilidade
- A operação de "relatório de incidente" nunca deve estar indisponível por mais do que 30 minutos.
- O processamento de pagamento de salários não pode esperar mais do que cinco dias úteis.

Autorização
- Apenas membros platina e ouro podem fazer reservas de assento.
- A votação para a pesquisa pode ser apenas feita na sexta-feira, entre 16h-18h, hora local.
- Um usuário pode apenas ver as informações do cliente de sua própria organização.

Exemplo1
"Funcionários dentro da mesma organização devem ser capazes de inserir transações durante o expediente.“



Exemplo2
"Um editor pode apenas editar artigos que ele criou.
"O editor não pode editar artigos que já foram publicados.“

Ameaças a SSL / TLS / HTTPS

Se a tecnologia avança para fornecer segurança, de outro lado temos também o avanço de novas técnicas de tentativas para invadir dados e acessar aplicações. Invasores já utilizam métodos de codificação que burlam a criptografia HTTPS. Veremos a seguir mais sobre os tipos de ameaças que podem afetar estes protocolos de segurança.

Como funciona
Invasores tentam se infiltrar em sistemas através de códigos maliciosos, além disso, kits de ataque, mais conhecido como Kits de exploit, são usados para atacar vulnerabilidades que sejam detectadas nos sistemas de organizações. Dentre os kits de ataque, temos o Nuclear, o Rig e Angler.

A tentativa é 'passar' escondido pelo mecanismo de segurança da informação, realizando alterações no padrão URL, adicionando imagens dentro de outro arquivo, modificando páginas de destino.

Exemplos de ataques aos protocolos SSL/TLS
Existem diversas formas de ataques aos protocolos SSL /TLS, dentre eles, podemos citar o SSL Renegotiation Attack e o Compression Ratio Info-Leak Mass Exploitation.

SSL Renegotiation Attack: Este se trata de um tipo de vulnerabilidade que ocorre justamente no processo de renegociação dos protocolos SSL e TLS, permitindo então que o invasor injete textos em todas as solicitações que o usuário realizar em uma conexão segura.

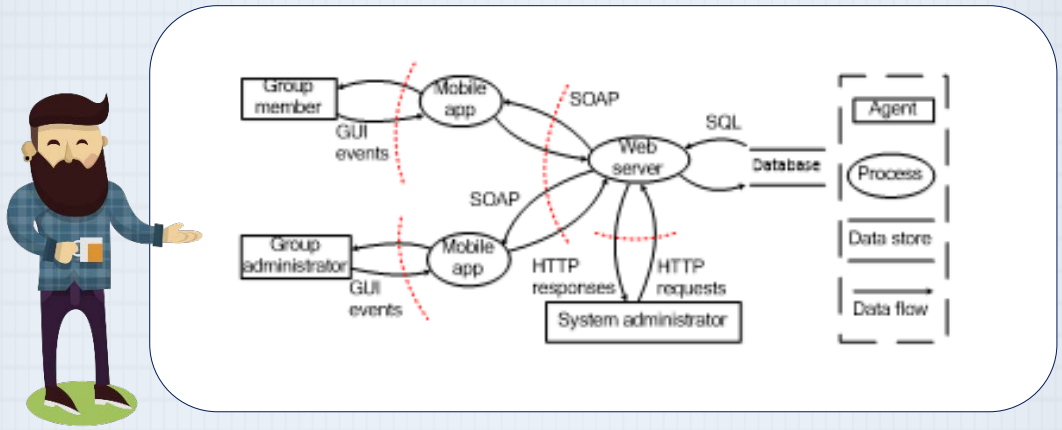
Compression Ratio Info-Leak Mass Exploitation: Outro tipo de ataque que podemos citar é o Compression Ratio Info-Leak Mass Exploitation, conhecido como 'CRIME', em sua forma abreviada. Este tipo de ataque possibilita que o invasor consiga acesso aos conteúdos guardados em cookies da Internet quando é utilizado TLS. Com este tipo de ataque é possível que o invasor ''sequestre'' uma sessão web, conseguindo expor as informações e comprometendo integridade do acesso do usuário.

A dica para evitar esses ataques é sempre observar as melhores práticas para configurar o servidor HTTPS, mantendo o navegador WEB sempre atualizado e protegido contra vulnerabilidades.

Exemplo de Design de Segurança e Limites de Confiança

Um bom número de problemas de segurança (potenciais) pode ser identificado na fase de arquitetura. Em um processo chamado de **análise de riscos arquitetônicos**, nós analisamos a arquitetura e vemos se qualquer padrão de ataque conhecido se aplica. A Microsoft chama isso de processo de **modelagem de ameaças.**

Este diagrama de fluxo de dados e os limites de confiança fazem parte de um design seguro.



Os 'defensores' dessas zonas confiáveis são chamados de 'limites de confiança'. Esses 'guardiões' trabalham como uma linha que separa essas zonas de confiança, resguardando-as de brechas e possíveis ataques:

- "E se alguém manipulasse os dados no armazenamento de dados?"
- "Pode um invasor imitar esse processo e enviar solicitações em nome dele?"

Princípios de codificação segura

Na fase de codificação, temos que escolher as construções de programação corretas, a fim de tornar o código seguro.

Padrão de codificação segura - existe para nortear as linguagens de programação mais comuns.

As revisões de código são uma boa maneira de capturar os erros de codificação inevitáveis.

A diferença entre uma revisão de código de segurança e uma revisão de código comum é que a cobertura da revisão de código de segurança precisa ser muito maior.

Muitas ferramentas automatizadas existem para ajudar a analisar o fluxo de dados no aplicativo e detectar construções inseguras que são pontos de vulnerabilidade em potencial.

Para isso existem as chamadas ferramentas de análise estática.

Testes de Segurança

Muitos problemas de segurança podem também ser detectados através da execução de **testes**.

Revisões de código + testes = melhores resultados

Teste de penetração -> um especialista em segurança tenta atacar o sistema antes que os invasores tenham a chance de fazer isso.

- Uma grande desvantagem dos testes de penetração é que eles só podem ser executados relativamente tarde no processo de desenvolvimento.

Escaneamento de vulnerabilidade - ferramentas criadas para verificar, de forma automatizada, a existência de problemas de segurança em um determinado sistema.

Vão desde escaneamentos gerais de hardening a escaneamentos específicos para cross-site-scripting.

Teste de abuso: Testes de abuso são criados pela equipe que desenvolve o aplicativo

Fuzzing é uma técnica de teste em que o sistema recebe uma entrada que é ou gerada automaticamente ou modificada aleatoriamente, para ver o comportamento do aplicativo.

Teste de Feedback: Os métodos de verificação, tais como revisões de código e testes de segurança fornecem redes seguras em diferentes pontos no processo de desenvolvimento do software.

Parte III

Glossário

GLOSSÁRIO	
TERMO	**SIGNIFICADO**
***Aleatoriedade**	A palavra aleatoriedade é utilizada para exprimir quebra de ordem, propósito, causa, ou imprevisibilidade em uma terminologia não científica. Um processo aleatório é o processo repetitivo cujo resultado não descreve um padrão determinístico, mas segue uma distribuição de probabilidade.
***Analista XML (validação de entrada)**	Função do profissional que cuida da validação de entrada com XML.
***Ataque de força bruta**	Um Ataque de Força Bruta caracteriza-se por uma tentativa continuada de obter acesso a um serviço do sistema (ssh, smtp, http, etc.), tentando diversas combinações de nome do utilizador e senha. Para levar a cabo este ataque, o atacante pode usar um software que gere diversas combinações de caracteres ou basear-se numa lista de palavras (dicionário).
***Ataque temporizado**	É um side channel attack no qual um atacante tenta comprometer um sistema criptográfico analisando o tempo gasto para executar determinados algoritmos criptográficos.
***Checksums (Verificações de somatório)**	É um código usado para verificar a integridade de dados transmitidos através de um canal com ruídos ou armazenados em algum meio por algum tempo.
***Ciclo de Vida de Desenvolvimento (S-SDLC)**	Refere-se aos estágios de concepção, projeto, criação e implementação de um SI.
***Cifra**	Método usado para criptografar mensagens.
***Cracking (Quebra de códigos)**	Quebra de códigos de um sistema de segurança de forma ilegal ou sem ética.

***Endereço MAC**	É um endereço físico associado à interface de comunicação, que conecta um dispositivo à rede.
***Framebusting (Falha de estrutura)**	Trata-se de uma falha na estrutura de um código.
***Framework Secure Software**	A estrutura de software é uma plataforma ou conceito onde o código comum, com funcionalidade genérica pode ser especializado ou substituído por desenvolvedores ou usuários .
***Hacking**	Modificação de um programa ou dispositivo para dar, ao usuário, o acesso a recursos não disponíveis anteriormente, como adaptações de acessibilidade.
***Malware**	É o termo usado para referenciar qualquer tipo de software destinado a infiltrar em computador alheio de forma ilícita, com o intuito de causar algum dano ou roubo de informações.
***Meta informações**	São dados sobre outros dados.
***Parsing (validação de entrada)**	É o processo de analisar uma sequência de entrada (lida de um arquivo de computador ou do teclado, por exemplo) para determinar sua estrutura.
***Phishing**	Em computação, phishing, termo oriundo do inglês (fishing) que quer dizer pesca, é uma forma de fraude eletrônica, caracterizada por tentativas de adquirir dados pessoais de diversos tipos; senhas, dados financeiros como número de cartões de crédito e outros dados pessoais.
***Protocolo Simple de Acesso a Objetos (SOAP)**	É um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída. Ele se baseia na Linguagem de Marcação Extensível (XML) para seu formato de mensagem, e normalmente baseia-se em outros protocolos da camada de aplicação, mais notavelmente em chamada de procedimento remoto (RPC) e Protocolo de transferência de hipertexto (HTTP), para negociação e transmissão de mensagens.
***Sistema de Nomes de Domínio (DNS)**	É um sistema de gerenciamento de nomes hierárquico e distribuído para computadores, serviços ou qualquer recurso conectado à Internet ou em uma rede privada.

***Vazamentos de core dump**	Consiste no estado da memória em funcionamento de um programa de computador no exato momento onde, geralmente, o programa termina com falha de sistema (crash).
Adulteração	A adulteração de dados é uma forma comum de ataque, ofuscada por outros tipos de ataque. Consiste em modificar, falsificar.
Análise de risco de arquitetura	É um processo de análise de riscos arquitetônicos, por meio do qual analisamos a arquitetura e vemos se qualquer padrão de ataque conhecido se aplica. A Microsoft chama isso de processo de modelagem de ameaças.
Análise estática	Processo efetuado por meio de ferramentas automatizadas que existem para ajudar a analisar o fluxo de dados no aplicativo e detectar construções inseguras que são pontos de vulnerabilidade em potencial.
Ataque de homem do meio	É uma forma de ataque em que os dados trocados entre duas partes são de alguma forma interceptados, registrados e possivelmente alterados pelo atacante sem que as vítimas se apercebam.
Autenticação	É um processo que busca verificar a identidade digital do usuário de um sistema, normalmente, no momento em que ele requisita um log in (acesso) em um programa ou computador.
Autoridade de certificados	Em criptografia, Autoridade de Certificação (acrónimo: CA, do inglês Certification Authority, ou AC) é o terceiro confiável que emite um certificado.
Autorização	Autorização, em segurança da informação, é o mecanismo responsável por garantir que apenas usuários autorizados consumam os recursos protegidos de um sistema computacional.
Chave privada	Similar a uma senha, é utilizada como elemento secreto pelos métodos criptográficos. Seu tamanho é geralmente medido em quantidade de bits. Serve para abrir as mensagens (desencriptação).

Chave pública	Similar a uma senha, é utilizada como elemento secreto pelos métodos criptográficos. Seu tamanho é geralmente medido em quantidade de bits. Serve para fechar (encriptar) as mensagens.
Clickjacking (Roubo de click)	O clickjacking é uma técnica de informática fraudulenta. O roubo de click é uma armadilha preparada para que o usuário pense que está fazendo uma ação num determinado site, mas na verdade os cliques executados nessa ação estejam sendo usados pelo atacante, para executar operações maliciosas.
Consultas diretas	São formulário não parametrizado de consultas SQL.
Criptografia	Trata-se de um conjunto de regras que visa codificar a informação de forma que só o emissor e o receptor consiga decifrá-la.
Criptografia assimétrica	Criptografia de chave pública, também conhecida como criptografia assimétrica, é uma classe de protocolos de criptografia baseados em algoritmos que requerem duas chaves, uma delas sendo secreta (ou privada) e a outra delas sendo pública.
Criptografia simétrica	Os algoritmos de chave simétrica (também chamados de Sistemas de Chave Simétrica, criptografia de chave única, ou criptografia de chave secreta) são uma classe de algoritmos para a criptografia, que usam chaves criptográficas relacionadas para as operações de cifragem e decifragem.
Cross-Site Request Forgery (Falsificação de Solicitação Cruzada entre Sites – CSRF/XSRF)	É o ataque em que o invasor pode inserir código Java Script na saída, que então é executado no navegador.
Diagrama de fluxo de dados	O diagrama de fluxos de dados (DFD) é uma representação gráfica do "fluxo" de dados através de um sistema de informação, modelando seus aspectos de processo.
Divulgação de informações	Ver informações que você não está permitido a ver.

Elevação de privilégio	Fazer algo sem permissão.
Encadeamento de certificados	Lista de uma assinatura de autoridade assinando outra chave de autoridade de certificação.
Endurecimento	Ato ou efeito de endurecer(-se).
Escalonamento de privilégios	É um computador exploit que permite a um usuário privilégios de acesso concedidos a outro usuário, potencialmente criando uma vulnerabilidade em que um hacker pode reconfigurar um sistema e executar operações ilegais.
Explorar	É um método de tirar proveito (com más intenções) de alguma falha específica do sistema.
Falsificação de senha	É a tentativa de ganhar acesso a um sistema usando uma identidade falsa.
Forjamento	Ato ou efeito de forjar; forjadura.
Fuzzing	É uma técnica de teste em que o sistema recebe uma entrada que é ou gerada automaticamente ou modificada aleatoriamente, para ver o comportamento do aplicativo.
Gerenciamento de sessão	Gerenciar uma sessão é a forma de garantir um processo de autenticação eficaz.
Greedy matching e non-greedy matching	expressão regular correspondente com a opção de não-ganancioso.
Hashing	Em ciência da computação, uma tabela de dispersão é uma estrutura de dados especial, que associa chaves de pesquisa a valores. Seu objetivo é, a partir de uma chave simples, fazer uma busca rápida e obter o valor desejado. É algumas vezes traduzida como tabela de escrutínio.
Injeção SQL	Um ataque de injeção SQL explora as vulnerabilidades na validação de entrada para executar comandos arbitrários no banco de dados.
Limite de confiança	A linha que separa as zonas confiáveis de um sistema.
Lista branca	A abordagem da lista branca irá aceitar qualquer coisa que está explicitamente marcado como permitido.

Lista negra	A abordagem da lista negra irá rejeitar tudo o que está explicitamente marcado como proibido.
Logging	Logging não é só uma excelente forma de diagnosticar problemas, mas também serve para detectar ataques em um sistema, desde que a informação adequada esteja registrada.
Mitigação	Consiste numa intervenção humana com o intuito de reduzir ou remediar um determinado impacto nocivo. Também significa referência relativa à um determinado ato. Na Gerência de Projetos de Software, o plano de mitigação inclui procedimentos para amenizar ou eliminar a ocorrência dos riscos impactantes no projeto.
Modelagem de ameaça	Modelos de ameaças.
Não rejeição	Não há como "dizer não" sobre um sistema que foi alterado ou sobre um dado recebido.
Nonce	É um número arbitrário que só pode ser usado uma vez .
Parametrização	Parametrização é o processo de decisão e definição dos parâmetros necessários para uma especificação completa ou relevante de um modelo ou objeto.
Princípio de Kerckhoff	O que tem sido chamado de lei ou principio de Kerckhoffs tem origem num conjunto de condições que August Kerckhoffs, engenheiro holandes radicado na França, formulou como desejáveis à criptografia militar, no século XIX. Ou, alternativamente, em critérios que duas dessas condições lhe fazem presumir. Essas duas condições foram apresentadas como parte de uma lista de seis, formulada em um artigo publicado em duas partes, nas edições de janeiro e fevereiro de 1883 da revista Journal des Sciences Militaires. De acordo com este princípio, a segurança do sistema deve mudar de volta a chave, tendo que assumir o resto dos parâmetros conhecidos do sistema criptográfico. O princípio foi reformulado, talvez de forma independente, por Claude Shannon: o inimigo conhece o sistema. Sob tal formulação é conhecido como o máximo de Shannon e é o princípio mais comumente adotada por cryptologists em oposição ao chamado de segurança através da obscuridade.

Recusa de Serviço (DoS)	Um “ataque por recusa de serviço” (em inglês “Denial of Service”, abreviado em DoS) é um tipo de ataque destinado a tornar indisponível, durante um tempo indeterminado, os serviços ou recursos de uma organização. Trata-se, na maior parte do tempo, de ataques contra os servidores de uma empresa, para que não possam ser utilizado e consultados.
Registro	Dar propriedade ou nomear posse.
Rejeição	Negar que você fez ou não fez alguma coisa.
Revisão de código	O próprio nome “revisão de código” deixa claro o objetivo da prática. A ideia é que o código escrito por um desenvolvedor, antes de ser promovido ao ambiente de produção, seja revisado por outro membro da equipe.
Revogação de certificados	É o ato de revogar um certificado emitido.
Scripts Cruzados entre Sites (XSS)	É o ataque em que o invasor pode inserir código Java Script na saída, que então é executado no navegador.
STRIDE (Spoofing identity – Tampering with data – Repudiation – Information disclosure – Denial-of-Service – Elevation of privilege, ou seja, Forjamento de Identidade – Adulteração de dados – Rejeição – Divulgação de informações – Recusa de Serviço – Elevação de privilégio)	Acrônimo com a reunião das letras iniciais (em inglês) de cada palavra que define um tipo de ameaça diferente.
Superfície de ataque	Parte do sistema onde existe vulnerabilidade para acessos não autorizados.
Transbordamento de buffer	Quando os dados podem ser escritos fora do intervalo de memória por um invasor, sendo capaz de manipular o estado interno do programa.
Transbordamento de pilha	Trata-se de uma tecnologia de compilador que adiciona um recurso de detecção de transbordamento de buffer à pilha do aplicativo.
Zona de confiança	Parte do sistema onde existe maior grau de segurança, probabilidade mínima de falhas de acesso do inimigo.

Literatura e referências

- Cavit, D.(2011). Return onInvestment (ROI)andSecureApplicationDevelopment:Canaholisticapproachsavemoneyandincreaseproductivity?MicrosoftCyberTrustBlog. Retrieved October15,2014,from http://blogs.microsoft.com/cybertrust/2011/02/15/return-on-investment-roi-and-secure-application-development-can-a-holistic-approach-save-money-and-increase-productivity/

- CommonAttackPattern Enumeration and Classification (CAPEC): ACommunityResourcefor IdentifyingandUnderstandingAttacks.(2014).MITRE.RetrievedOctober15,2014,fromhttps://capec.mitre.org/

- CommonWeaknessEnumeration(CWE):ACommunity-Developed Dictionaryof SoftwareWeaknessTypes.(2014).MITRE.RetrievedOctober15,2014,fromhttp://cwe.mitre.org/

- DataBreachToday.(2014).RetrievedOctober15,2014,fromhttp://www.databreachtoday.com/news Devanbu,P.T.,&Stubblebine, S. (2000).Softwareengineeringforsecurity:a roadmap. InProceedingsof the

- Conferenceon the Futureof Software Engineering (pp. 225–239).ACM.Retrievedfrom http://dl.acm.org/citation.cfm?id=336559

- FFmpegandathousand fixes.(2014). Google Online SecurityBlog. Retrieved October15,2014,fromhttp://googleonlinesecurity.blogspot.com/2014/01/ffmpeg-and-thousand-fixes.html

- Framework SecureSoftware.(2014). Secure Software Foundation. Retrieved October 15, 2014, fromhttps://www.securesoftwarefoundation.org/FrameworkSecureSoftware.html

- Haack, P. (2009).JSON Hijacking.You'vebeenhaackedweblog.RetrievedOctober15,2014, from http://haacked.com/archive/2009/06/25/json-hijacking.aspx/

- Haahr,M. (1999).Introductiontorandomnessandrandomnumbers. random.org. RetrievedOctober15,2014,fromhttp://www.random.org/randomness/

- Hellman,M. E. (1979). The Mathematicsof Public-Key Cryptography. ScientificAmerican,241(2),146–157.doi:10.1038/scientificamerican0879-146

- Hernan,S., Lambert, S., Ostwald, T., &Shostack, A. (2006).Threat modeling-Uncoversecuritydesignflawsusing the STRIDE acronym.MSDNMagazine, 68–75.

- Horst, G.,&Nordhoff, M. (2008). BrowserSecurityModel.InChair of NetworkandDataSecurity.Bochum,Germany.
- Ives,B., Walsh, K.R.,&Schneider,H.(2004). The dominoeffectof password reuse. Communications oftheACM,47(4),75–78.doi:10.1145/975817.975820
- Johnstone, M. N. (2010). Threat modelling with STRIDEandUML.In Proceedings ofthe8th AustralianInformationSecurity Management Conference(pp.18 27).Perth,WesternAustralia.Retrievedfromhttp://ro.ecu.edu.au/ism/88/
- Kahneman,D.(2002). Mapsofboundedrationality: Aperspectiveonintuitivejudgmentandchoice. NobelPrizeLecture, (December), 449–489. Retrieved from
- http://wisopsy.uni-koeln.de/uploads/media/kahnemann_Nobelpreisrede_20.pdf
- Kelly, H. (2014).The “Heartbleed”securityflaw thataffectsmost oftheInternetThe“Heartbleed”securityflawthataffectsmostoftheInternet.CNN. RetrievedOctober15,2014,fromhttp://edition.cnn.com/2014/04/08/tech/web/heartbleed-openssl/
- Kerckhoffs, A.(1883). Lacryptographiemilitaire.Journal DesSciencesMilitaires,IX,5–83,Jan.1883, 161–191, Fév.1883.
- Langley,A. (2014).No,don’tenablerevocationchecking.ImperialVioletweblog.RetrievedOctober15,2014,from https://www.imperialviolet.org/2014/04/19/revchecking.html
- Ma,W.,Campbell,J., Tran, D.,&Kleeman, D. (2010). PasswordEntropyand Password Quality. In 2010FourthInternationalConferenceonNetworkandSystemSecurity (pp.583–587).IEEE.doi:10.1109/NSS.2010.18
- Mead, N. R.,Allen, J.H.,Conklin, W. A.,Drommi, A.,Harrison, J.,Ingalsbe,J.,… Shoemaker, D. (2009).
- Makingthebusinesscaseforsoftwareassurance(No. CMU/SEI-2009-SR-001).Pittsburgh, PA.Retrievedfromhttp://resources.sei.cmu.edu/asset_files/SpecialReport/2009_003_001_15008.pdf
- Mercuri,R.T. (2003). Analyzing securitycosts.Communicationsof the ACM, 46(6),15 18.doi:10.1145/777313.777327

SP Foundation

- MicrosoftSDL:Return-on-Investment.(2009).Redmond,WA.Retrievedfrom https://www.isecpartners.com/media/11970/isec%20partners%20-%20microsoft%20sdl%20return-on-investment.pdf
- Regular Expressions.(2013).Regular-Expressions.info.Retrieved October15,2014,fromhttp://www.regular-expressions.info/
- Rice, D. (2008). Geekonomics: The realcostofinsecuresoftware.PearsonEducation,Inc.
- SecureSDLCCheatSheet (Work in Progress).(2012).OWASP.RetrievedOctober 15,2014, from https://www.owasp.org/index.php/Secure_SDLC_Cheat_Sheet
- XSS(Cross SiteScripting)PreventionCheatSheet.(2014).OWASP.RetrievedOctober15, 2014,fromhttps://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet.

SP Foundation

Requisitos do exame

Os requisitos do exame estão detalhados na especificação do exame. A tabela abaixo lista os tópicos dos módulos (requisitos do exame). O peso dos diferentes tópicos contidos no exame é expresso como uma porcentagem do total.

Requisitos do Exame	Peso (%)
1. Introdução	10 %
1.1	Consciência de Segurança
1.2	Princípios Básicos
1.3	Segurança da Web
2. Gerenciamento de Sessão e Autenticação	15 %
2.1	Senhas
2.2	Gerenciamento de Sessão
2.3	Cross-Site Request Forgery (Falsificação de Solicitação Cruzada entre Sites – CSRF/XSRF) e Clickjacking
3. Manejo de Entrada de Usuário	22.5 %
3.1	Ataques de Injeção
3.2	Validação de Entrada
3.3	Estouros de Buffer
3.4	Cross-Site-Scripting (Scripts Cruzados entre Sites – XSS)
4. Autorização	7.5 %
4.1	Autorização
4.2	Envenenamento de Sessão e Condições de Corrida
5. Configuração, Manejo e Registro de Erros	15 %
5.1	Componentes de Terceiros, Configuração e Endurecimento
5.2	Vazamentos de Informação
5.3	Manejo e Registro de Erros
5.4	Recusa de Serviço

6. Criptografia	10 %
6.1	O Princípio de Kerckhoff, Manejo de Chaves e Aleatoriedade
6.2	Criptografia de Chave Pública
6.3	HTTPS
7. Engenharia de Software Seguro	20 %
7.1	Requisitos de Segurança
7.2	Design Seguro
7.3	Codificação Segura
7.4	Testes de Segurança

Especificações do teste

1. Introdução (10%)
1.1 Consciência de Segurança (2,5%)

O candidato pode:
1.1.1 Reconhecer a tensão entre as exigências do mercado e a segurança.

1.2 Princípios Básicos (2,5%)

O candidato pode:
1.2.1 Explicar o jargão de segurança e STRIDE.

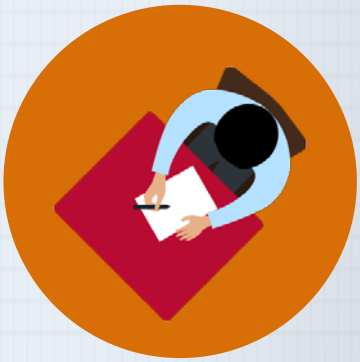
1.3 Segurança da Web (5%)

O candidato pode:
1.3.1 Descrever questões de segurança HTTP.
1.3.2 Explicar o modelo de segurança do navegador.

2. Gerenciamento de Sessão e Autenticação (15%)
2.1 Senhas (5%)

O candidato pode:
2.1.1 Identificar problemas envolvidos no uso de senha.
2.1.2 Aplicar princípios de gerenciamento de senhas.

2.2 Gerenciamento de Sessão (7,5%)

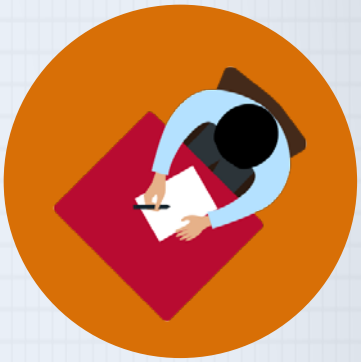


SP Foundation

O candidato pode:
2.2.1 Explicar como funciona o gerenciamento de sessão.
2.2.2 Reconhecer problemas em gerenciamento de sessão.
2.2.3 Reconhecer as melhores soluções para problemas em gerenciamento de sessão.

2.3 Cross-Site Request Forgery (Falsificação de Solicitação Cruzada entre Sites – CSRF/XSRF) e Clickjacking (2,5%)

O candidato pode:
2.3.1 Reconhecer problemas e soluções de CSRF e Clickjacking.



3. Manejo de Entrada de Usuário (22,5%)
3.1 Ataques de Injeção (7,5%)

O candidato pode:
3.1.1 Reconhecer os problemas de ataques de injeção.
3.1.2 Explicar a diferença entre consultas diretas e parametrizadas.
3.1.3 Aplicar soluções para ataques de injeção SQL.

3.2 Validação de Entrada (7,5%)

O candidato pode:
3.2.1 Explicar a diferença entre filtros de lista branca (whitelist) e lista negra (blacklist).
3.2.2 Aplicar validação de entrada.
3.2.3 Reconhecer quando aplicar normalização de entrada e codificação.

3.3 Estouros de Buffer (2,5%)

O candidato pode:
3.3.1 Identificar onde ocorrem estouros de buffer e como eles impactam a segurança.

3.4 Cross-Site-Scripting (Scripts Cruzados entre Sites – XSS) (5%)

O candidato pode:
3.4.1 Reconhecer a diferença entre ataques XSS refletidos e armazenados e as mitigações.
3.4.2 Aplicar soluções para ataques XSS.

4. Autorização (7,5%)
4.1 Autorização (5%)

O candidato pode:
4.1.1 Reconhecer a diferença entre autorização horizontal e vertical.
4.1.2 Reconhecer a diferença entre referências diretas e indiretas.

4.2 Envenenamento de Sessão e Condições de Corrida (2,5%)

O candidato pode:
4.2.1 Reconhecer envenenamento de sessão e condições de corrida.

5. Configuração, Manejo e Registro de Erros (15%)

5.1 Componentes de Terceiros, Configuração e Endurecimento (5%)

O candidato pode:
5.1.1 Justificar a necessidade de endurecimento.
5.1.2 Reconhecer métodos de endurecimento.

5.2 Vazamentos de Informação (2,5%)

O candidato pode:
5.2.1 Reconhecer diferentes vazamentos de informação.

5.3 Manejo e Registro de Erros (5%)

O candidato pode:
5.3.1 Explicar a importância do registro para a segurança.
5.3.2 Explicar o princípio de 'Falhar com Segurança'.

5.4 Recusa de Serviço (2,5%)

O candidato pode:
5.4.1 Reconhecer ataques por recusa de serviço e mitigações.

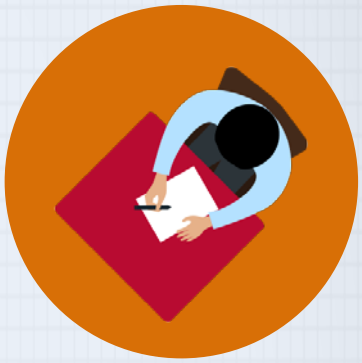
6. Criptografia (10%)
6.1 O Princípio de Kerckhoff, Manejo de Chaves e Aleatoriedade (2,5%)

O candidato pode:
6.1.1 Explicar a importância do Princípio de Kerckhoff, Manejo de Chaves e Aleatoriedade.

6.2 Criptografia de Chave Pública (2,5%)

O candidato pode:
6.2.1 Descrever a criptografia de chave pública, ataques de homem do meio e certificados

6.3 HTTPS (5%)



O candidato pode:
6.3.1 Reconhecer as ameaças a SSL/TLS/HTTPS.
6.3.2 Aplicar HTTPS corretamente.

7. Engenharia de Software Seguro (20%)
7.1 Requisitos de Segurança (5%)

O candidato pode:
7.1.1 Identificar requisitos de segurança em falta.
7.1.2 Reconhecer ambiguidades e pressupostos ocultos em determinadas condições e contextos.

7.2 Design Seguro (5%)

O candidato pode:
7.2.1 Reconhecer ameaças que são inerentes a uma arquitetura específica.
7.2.2 Reconhecer soluções adequadas a ameaças e as imperfeições nessas soluções.

7.3 Codificação Segura (2,5%)

O candidato pode:
7.3.1 Reconhecer o escopo, objetivo e vantagens da revisão de código para as práticas de desenvolvimento.

7.4 Testes de Segurança (7,5%)

O candidato pode:
7.4.1 Lembrar diferentes métodos para testes de segurança.
7.4.2 Reconhecer o melhor teste para um determinado cenário.
7.4.3 Identificar formas de melhorar o desenvolvimento de software e processos de testes, incorporando resultados de testes.