OFICIAL

EXIN SECURE PROGRAMMING

Curso Preparatório para Certificação Secure Programming

Área de Aprendizagem

PMG academy

www.pmgacademy.com

Official Course

Módulo 1

EXIN SECURE PROGRAMMING

PMG ACADEMY

Nível Foundation

www.pmgacademy.com

Design: By Freepik

Prof. Adriano Martins Antonio

Módulo 1

Noções Básicas Sobre Segurança na Programação

Maior Aproveitamento

Assista no mínimo 2x

Contate o instrutor

Realize os exercícios

Leia o glossário

Execute os simulados

Público-Alvo

- Todos os que desejam se preparar para o EXIN (Avaliação da Secure Programming Foundation);

- Todos que tenham interesse no desenvolvimento de aplicativos de segurança na web;

- Auditores que irão trabalhar com o Framework Secure Software.

O Que Veremos Neste Curso?

- Noções básicas sobre segurança na programação
- Gerenciamento de sessão e autenticação
- Manejo de entrada de usuários
- Autorização
- Configuração, manejo e registro de erros
- Criptografia
- Engenharia de software de segurança

- Literatura e referências
- Glossário
- Perguntas durante o treinamento

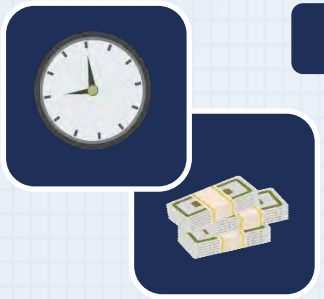
Visão Geral

E de que adianta tanto investimento em softwares para esse fim se não soubermos utilizá-los?

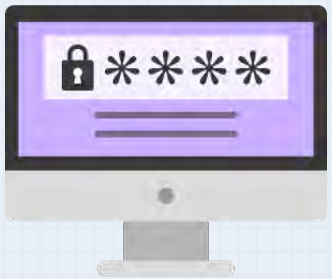
O **EXIN** Secure Programming irá justamente checar seus conhecimentos a esse respeito. A avaliação é composta por **40** questões de múltipla escolha. Por isso, neste curso você também terá questões equivalentes após cada tema abordado para treinar o seu conhecimento.

Investindo no Desenvolvimento de Softwares de Segurança

Você, que é Programador, sabe que sozinho não pode educar a sociedade para combater o crime cibernético, mas pode desenvolver programas eficientes de segurança da informação!



Softwares de segurança



- A velocidade que são desenvolvidos para atender a demanda do mercado
- Nossa dificuldade de focar na medição de riscos
- O cliente também acaba decidindo não precisar de segurança contra certos tipos de ameaça

"Imagina se alguém vai se atrever a atacar um software desenvolvido por mim!..."

Jargões da área de Segurança da Informação

Falar em 'segurança' nos remete de imediato a um cenário de

Termo	Explicação
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.

E o que dizer ‘Segurança’?

A **segurança** é definida como uma proteção contra ameaças.

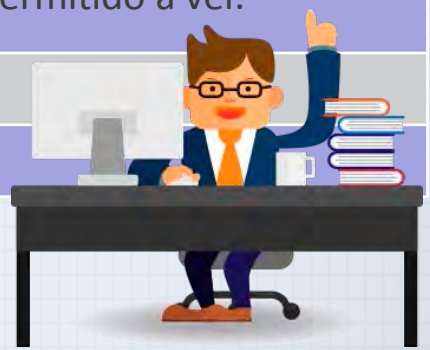
A má notícia é que é praticamente impossível criar uma lista completa de ameaças para um determinado sistema

A boa notícia é que, felizmente, muitas ameaças já são ‘manjadas’ pelos desenvolvedores de programação e, assim, dá para reverter a situação e saber como se proteger delas

Significado de ‘Stride’

'S-T-R-I-D-E’ define um tipo de ameaça diferente

	Termo	Explicação
S	Forjamento de Identidade	Fingir ser alguém ou alguma coisa (Os famosos ‘fakers’ da internet!).
T	Adulteração de dados	Modificar dados armazenados ou dados em trânsito.
R	Rejeição	Negar que você fez ou não fez alguma coisa.
I	Divulgação de informações	Ver informações que você não está permitido a ver.
D	Recusa de serviço	Deixar um sistema indisponível.
E	Elevação de privilégios	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

Superfícies de ataque

- Onde existe vulnerabilidade para acessos não autorizados

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 na Web

‘limites de confiança’

Servidor Web

Navegador Web

A linha que separa quem é quem nessa atuação é o **limite de confiança**.

- Atualmente, são usadas as solicitações **AJAX**, que funcionam como uma solicitação dentro do navegador.
- É gerado um código JavaScript que executa este navegador.
- Os **proxies MITM** (homem-no-meio) são os ataques mais conhecidos deste tipo.
- O Programador deve sempre acompanhar, sondar, detectar tudo que acontece nessa fronteira.

O protocolo HTTP - Hypertext Transfer Protocol (Protocolo de Transferência de Dados)

Um exemplo de solicitação GET:

```
GET Http://localhost/insecure/public/Login.jsp?loginadmin&passadmin
HTTP/1.1 Host: localhost
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/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:
JSESSIONID73A4496C2B6C846B5E8D8C9DDDBD9D24
```

Um exemplo de solicitação POST:

```
POST http://localhost/account/login HTTP/1.1 Host: localhost
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/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/
Cookie: _session_id5bd5199e505bce65cffdc4c5ef02b617 Content-Type:
application/x-www-form-urlencoded Content-Length: 46

user_loginadmin&user_passwordadmin&x45&y20
```

GET e POST

- **Padrão CGI (Common Gateway Interface),o qual permite o processamento de entrada do usuário e respostas geradas dinamicamente por aplicações web.**

- É destinado a operações de "apenas leitura"
- Irá modificar o estado da aplicação

Os parâmetros de entrada são incluídos no caminho de solicitação

Eles são incluídos no corpo da solicitação

CGI **Sequência de consulta**

Qualquer coisa no caminho de solicitação pode terminar no histórico do navegador, em registros server-side, em logs de proxy e nos chamados cabeçalhos "referer".

Trocando em ‘miúdos’

É feita via URL, obviamente há uma limitação no tamanho da mensagem enviada

A string não pode conter mais que 255 caracteres(embora exista diferenças entre navegadores, mas em geral o limite é 255)

Já na requisição POST não há limitações de comprimento da mensagem, já que a mesma é enviada no corpo da requisição HTTP.

Visibilidade

São os principais métodos de comunicação HTTP.

Cabeçalhos de Solicitação

Solicitações AJAX* enviam o seguinte cabeçalho:
X-Requested-With:XmlHttpRequest

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

AJAX significa AsynchronousJavaScript e XML. Em poucas palavras, é o uso do objeto ***XMLHttpRequest*** *para se comunicar com os scripts do lado do servidor.*

Acessando os Dados Solicitados

PHP

$_GET, $_POST

apache_request_headers

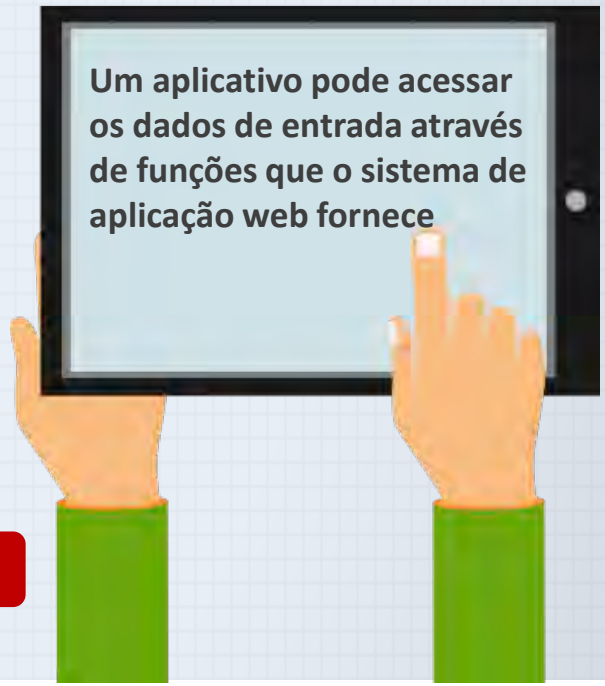
GET

POST

solicitar cabeçalhos

HttpServletRequest no JAVA

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).



REMOTE_USER

HttpRequest.Params["REMOTE_USER"].

GET REMOTE_USER=Administrator

Poluição de parâmetro HTTP(HTTP Parameter Pollution - HPP).

Exemplo de Ataque

Vamos exemplificar um ataque cujo alvo é um usuário do Yahoo Mail:

- 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**%26**par2=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 o Yahoo Mail), o parâmetro e valor acrescentados serão interpretados, gerando uma resposta diferente. E assim ações maliciosa podem ser realizadas por meio do comando errado que é interpretado como certo.

Resposta Comum de um HTTP

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
```

A resposta começa com HTTP e o número de versão do protocolo HTTP, em seguida, um código de status, seguido por uma descrição do status. Então, os cabeçalhos de resposta são enviados, e, finalmente, o corpo da resposta. Os cabeçalhos e o corpo são separados por uma linha vazia.

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 login para autenticar ao servidor web.

Os cabeçalhos de Resposta HTTP

Em certos casos, esses cabeçalhos podem determinar o que um invasor pode usar para executar um ataque mais eficaz.

Alguns dos cabeçalhos de resposta são puramente informativos, tal como o cabeçalho do servidor.

As informações detalhadas sobre qual plataforma web é usada pode ser utilizada para encontrar vulnerabilidades específicas nessa plataforma.

Alguns cabeçalhos são relevantes para a segurança, como o cabeçalho **Cache-Control**, que pode ser usado para prevenir informações sensíveis de serem armazenadas em um servidor de cache.

O Modelo de Segurança do Navegador

Modelo de segurança do navegador
(Horst & Nordhoff, 2008)

Política da Mesma Origem (SOP)

Descreve em quais **condições** os scripts e programas em sites podem acessar conteúdo em outros sites.

Princípio Básico

Os **scripts** em contexto de um local não têm permissão para acessar elementos em contexto em outro site. Isso se aplica 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.

Termos Módulo 1

TERMOS PARA O EXAME	Ficou claro? SIM ou NÃO?
Ataque	
Explorar	
Divulgação de Informações	
Mitigação	
Remendo	
Rejeição	
Risco	
Segurança	
Forjamento de Identidade	
Acrônimo STRIDE	
Adulteração de Dados	
Ameaça	
Vulnerabilidade	
Chamadas AJAX	
Superfície de ataque	

TERMOS PARA O EXAME	Ficou claro? SIM ou NÃO?
Limites de confiança	
Zonas confiáveis	
GET e POST	
Método HTTP	
Verbo HTTP	
Número de versão HTTP	
Cabeçalho 'Referer'	
Corpo de resposta	
Caminho de solicitação	
Poluição de parâmetro HTTP	
Cabeçalhos de solicitação	
Códigos de status HTTP	
Divisão de respostas HTTP	
Protocolo Simples de Acesso a Objetos (SOAP)	

Teste

Pronto para o próximo?

Clique em fechar e vá para o próximo módulo

OFICIAL

EXIN SECURE PROGRAMMING

Curso Preparatório para Certificação Secure Programming

Área de Aprendizagem

PMG academy

www.pmgacademy.com

Official Course

Módulo 2

Autenticação e Gerenciamento de Sessão

Autenticação e Gerenciamento de Sessão

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.

Senhas

As **senhas** são um segredo que é compartilhado entre duas partes

Um lado apresenta a senha, enquanto que o outro a verifica.

Presume-se que ambas as partes mantenham a senha em segredo. Se ela vazar, já não pode ser confiável para autenticação; qualquer pessoa que a possua pode falsificar a identidade da parte que deve apresentá-la, e o servidor pode não reconhecer a diferença.

Protegendo a Senha em Trânsito

A melhor maneira de proteger uma senha de ser espionada é não a enviar!

Existem protocolos para realizar autenticação de senha sem transmitir a senha para o servidor; são as **provas de conhecimento-zero** ou **autenticação por desafio e resposta**.

Criptografada

HTTPS

Entropia em Senhas

Métodos para criar senhas fortes

- Aumentar o conjunto de caracteres para as senhas.
- Aumentar o comprimento da senha.

entropia da senha (Ma, Campbell, Tran, &Kleeman, 2010)

Ao invés de espionar uma senha, um invasor pode simplesmente **adivinhar** a senha

Para evitar

- Limite o número de tentativas de autenticação. Isso pode ser permanente ou temporário.
- Implemente um atraso deliberado no processo de autenticação, para retardar um invasor.
- Bloqueie a conta após um determinado número de tentativas; limite o número total de tentativas.

Armazenamento de Senha Segura no Servidor

Armazenar senhas no servidor pode representar um risco enorme.

Ele terá acesso a todas as senhas

- As senhas nunca devem ser armazenadas como texto simples
- É melhor não armazenar as senhas, mas sim um **'hash'** (resumo) das senhas

'hash de senhas'

Uma forma de codificar a senha de tal forma que um invasor que consiga ter acesso ao servidor não tenha como recuperar esse código para decifrar a senha. Ou seja, se o usuário fizer o 'hash da senha' antes de armazená-la em um banco de dados poderá evitar a recuperação desta senha por alguém não autorizado.

Hash e Salt

- Evitar senhas simples
- Com apenas letras como caracteres
- Usar senhas diferentes para cada aplicativo

Atacante

Defesa

hash

salt

Colisão - é quando é gerado um código de 'hash' igual para dois usuários diferentes e aí o invasor tenta (tenta, tenta, tenta... até conseguir!) um 'hash' que seja equivalente a este e consegue acesso aos dois usuários de uma só vez!

Paralelismo - é quando o invasor pega um monte de senhas protegidas por 'hash' e tenta descobrir o máximo de informações sobre elas.

Chave **Supermegahiperchave** **Rainbowtables**

Procedimentos de Redefinição de Senha

Um problema adicional com senhas é que as pessoas tendem a esquecê-las. ..

- **Gerar** uma nova senha para o usuário e enviá-la para o endereço de e-mail conhecido do usuário

- **Vantagem** de que o servidor pode determinar a força da senha, e impedir que as pessoas usem a mesma senha em vários sites

- **Desvantagem** é que o e-mail contendo a senha pode vazar ou ficar armazenado por muito tempo

Um método consiste em perguntar um endereço de e-mail no momento de criação da conta. Quando um usuário indicar que eles perderam a sua senha, o aplicativo envia um link para um formulário de redefinição de senha.

Gerenciamento de Sessão

O HTTP(S) é um protocolo apátrida (sem pátria);

HTTP que amarram uma série de pedidos juntos

Após o ***login***, a sessão é criada e o servidor gera um 'identificador de sessão'. Esse 'identificador de sessão' deve ser difícil de prever, afinal, conhecer o 'identificador de sessão' daria ao invasor o acesso aos *dados da sessão* de um determinado usuário

'identificador de sessão'

acessar os ***dados de sessão*** lateral ao servidor durante o uso

No **final** do acesso, os dados no servidor podem ser limpos e o 'identificador de sessão' não apontará mais para aqueles dados.

Usando Cookies na Sessão

Identificador de Sessão

É considerado uma boa prática para que não se precise reenviar os dados de acesso em outra solicitação.

parâmetros CGI

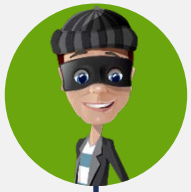
sessões sem cookies

cookies

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*.

Boas Práticas de Gerenciamento de Sessão

- O ID da sessão deve ser difícil de prever. Gere um token
- 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 token
- Imediatamente após o *login*, o ID da sessão deve ser atualizado
- *Cookies* devem ser usados e não parâmetros CGI!
- Os *cookies* devem ser enviados através de um canal criptografado.

CRSF e Furto de Cliques

Falsificação de solicitação entre sites e ***furto de cliques***.

Falsificação de solicitação entre sites (CSRFou XSRF)

Explora o fato de que pode se referir a um conteúdo em outro site e deixar o navegador fazer uma solicitação para obter esse conteúdo, sem necessitar do consentimento explícito do usuário.
Por exemplo, um invasor pode incorporar a seguinte imagem externa em seu website:

<imgsrc="https://e-voting.com/vote.cgi?candidate=George+Washington">

- O aplicativo armazena um valor aleatório nos dados da sessão. Esse valor é chamado de ***nonce*** ou ***ponto-CSRF (CSRF-tokens)***.
- O *ponto-CSRF* é, então, incluído em cada solicitação.
- Quando a solicitação é feita, o aplicativo irá primeiro verificar se o ponto na solicitação combina com aquele nos dados da sessão, antes de executar o pedido. Como resultado, o invasor não pode prever o valor correto para colocar na solicitação, portanto, um ataque bem-sucedido é bloqueado!

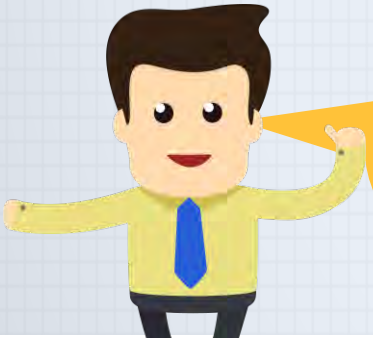
CRSF-Tokens em Solicitações POST

ponto-CSRF → **GET**

validade limitada, acaba sendo aceitável

Alguns sistemas adicionam *pontos-CSRF* automáticos para todos os formulários (solicitações POST), mas não para solicitações GET

- **Os pedidos que retornam dados JSON e JSONP podem ser lidos de forma inteligente ao carregá-los dentro de um script tag:<script src="...">**



***sequestro JSON* ou *sequestro JavaScript* (Haack, 2009**

Tenha em mente que solicitações GET podem precisar de proteção CSRF também.

Furto de Cliques (Click Jacking)

Cross-site Scripting (CSS)

Java Script

Somos capazes de mudar o olhar e o sentir dos elementos da interface do usuário em nossa página web. Podemos até empilhar elementos em cima uns dos outros em camadas. Os *furtos de cliques* exploram essas técnicas para realizar um ataque sobre a camada de interface do usuário.

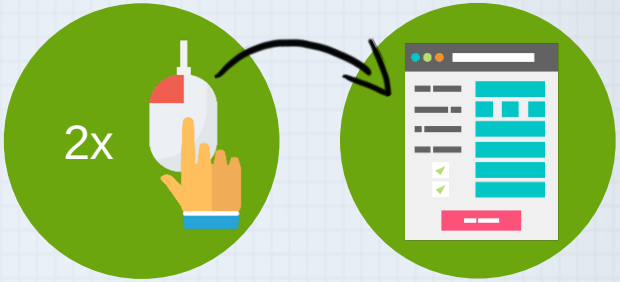
Um **invasor** incorpora o site de destino em um quadro e o sobrepõe na janela atual.

CSS

Furto de cliques é um ataque onde o invasor **usa *CSS* e/ou *JavaScript*** para fazer uma vítima clicar em algo que ela não tem intenção.

Proteção Contra Furto de Cliques (Click Jacking)

É muito difícil se defender contra esse tipo particular de ataque. Mas, o ataque de *furto de cliques básico* ainda dá para ser evitado!



- **Navegadores modernos apoiam o cabeçalho X-Frame-Options. Ao definir o cabeçalho com o valor SAME-ORIGIN ou DENY, um navegador irá recusar essa página a ser incorporada como um quadro para outro website.**

- **Navegadores antigos, às vezes, usam código Java Script para detectar se eles estão carregados em um quadro. Esse código é chamado de *código de quebra de quadro*. No entanto, sua eficácia é limitada.**

Termos Módulo 2

TERMOS PARA O EXAME	Ficou claro? SIM ou NÃO?
Autenticação	
Quebra de senhas	
Hash	
Salt	
Gerenciamento de sessão	
Falsificação de solicitação entre sites	
CSRF	
Ponto-CSRF (CSRF_tokens)	
Nonce	
Ataque de furto de cliques	
Código de quebra de quadro	

Teste

Pronto para o próximo?

Feche a tela do seu browser e vá para o próximo módulo

OFICIAL

EXIN SECURE PROGRAMMING

Curso Preparatório para Certificação Secure Programming

Área de Aprendizagem

PMG academy

www.pmgacademy.com

Official Course

Módulo 3

Manejo de Entrada de Usuário

Manejo de Entrada de Usuário

A maioria dos problemas de segurança de aplicativos pode ser atribuída a forma que...

... a entrada do usuário é tratada

Formulário web

Aplicativo

Servidor (web)

Ataques de Injeção

A **vulnerabilidade de injeção** existe quando os dados manipulados são involuntariamente interpretados por um sistema como uma instrução ao invés de dados simples.

telefônico analógico

pulsos com uma frequência de 2600 Hz

Sinalização "dentro da banda"

Injeção de SQL

Considere o código dentro de um manipulador de formulário web:

```
query= "SELECT * FROM people WHERE firstname='$post.firstname'"; connection.execute(query);
```

Carlos'; DROP TABLE people;--

SELECT * FROM people WHERE firstname= 'Carlos'; DROP TABLE people;--'

admin' OR 'a'='b

senha 'xyz',

SELECT * FROM users WHERE username='admin' OR 'a'='b' AND password='xyz'

Pois 'a'='b'é falso, e o operador "AND" tem maior procedência do que o operador "OR", isso é efetivamente o mesmo que:
SELECT * FROM users where username='admin'

Consultas Diretas e Consultas Parametizadas

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 "?". Pois, como não será mais feita análise, não é possível alterar o comando a ser executado.

Outras Formas de Injeções de Saída

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 (&).

Neutralizar Meta-Caracteres

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.

Validação de Entrada

A validação de entrada segue um princípio simples: se nós sabemos que tipo de entrada esperar, por que isso nos impediria de qualquer outra coisa? Em outras palavras: vamos filtrar a entrada!

Irá **rejeitar** tudo o que está explicitamente marcado como proibido.

Irá **aceitar** qualquer coisa que está explicitamente marcado como permitido.

Expressões Regulares

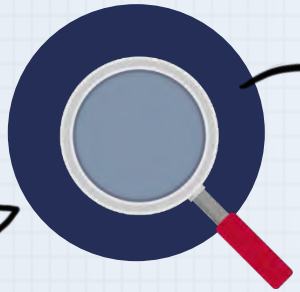
'combinação gananciosos x relutantes'

'âncoras'

Expressões regulares são úteis para combinar padrões em um texto, ou seja, uma expressão é interpretada como uma regra, que indicará sucesso se uma entrada de dados qualquer 'casar' com essa regra; obedecer exatamente a todas as suas condições.

Você sabe como elas funcionam?

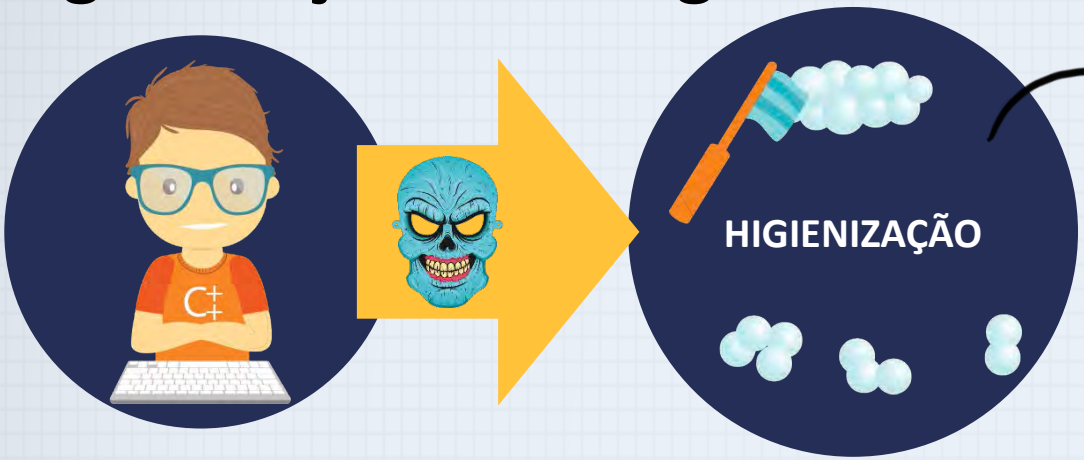
Regular



'case'

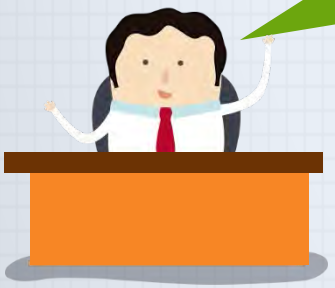
***Sugestão** de pesquisa: O site "Expressões Regulares" (2013) pode fornecer um bom começo se você não estiver familiarizado com o tema. Portanto, vale dar uma olhada lá no site e refrescar a memória!*

Higienização x Filtragem



Higienizar um nome de arquivo, pode-se substituir a sequência ../ pela sequência vazia, para prevenir que um invasor acesse caminhos fora do local pretendido.

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.

Codificações de Entrada e Normalização

codificação BASE64

codificação BASE64

Pense que nós precisamos receber a entrada de um navegador web que contém caracteres especiais que não podem ser transmitidas como estão

decodificar → validar

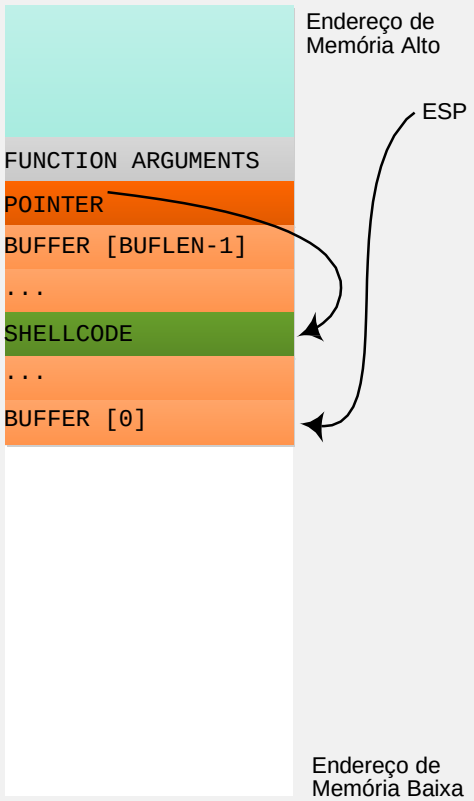
"<" em HTML:

< < < < < < < <

Primeiro normalizar (padronizar URL)

Estouro de Buffer

Ataque de transbordamento de dados (Buffer Overflow): Uma pilha transbordando (Stack Overflow)



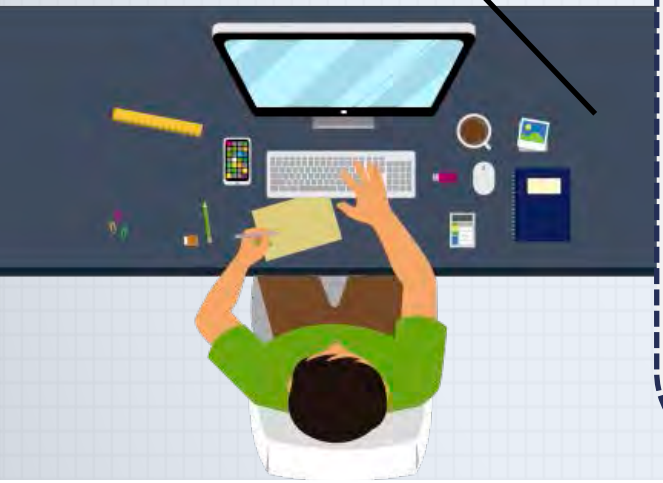

```
#include <stdio.h>
#include <stdlib.h>

#define BUFLEN 80

void doit(int offset)

{

     char buffer[BUFLEN];
     gets(&buffer[offset]);
     /* do more */
}
int main(void)
{
     doit(0);r
     eturn 0;
}
```

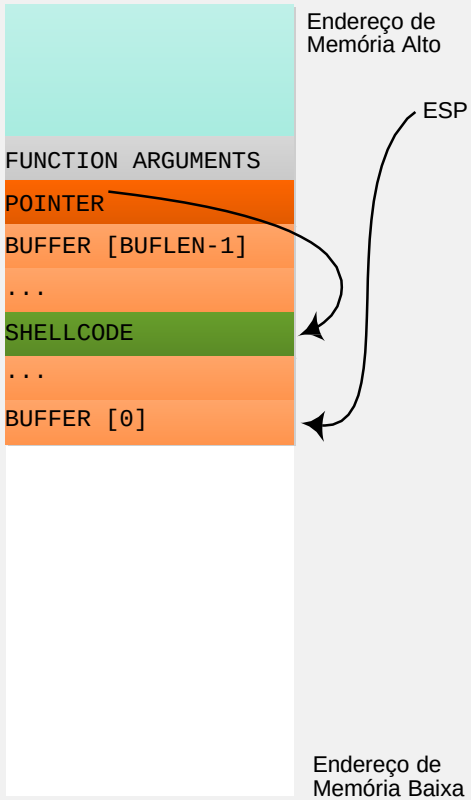
Buffer Overfow e Stack Overfow

Agora
acontece
lê uma
maior

Em prim
linguager
linguager

caracter






```
#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);r
     eturn 0;
}
```

Cross-site scripting (XSS)

cross-site scripting (XSS).

Há algumas formas que um invasor pode iniciar um ataque XSS:

- **Convencer** alguém a fazer uma solicitação, ou clicando em um link ou enviando um formulário que contém os parâmetros de entrada maliciosa, que são processados pelo navegador da vítima. Isso é chamado de **XSS refletido**, e baseia-se na vontade da vítima em cooperar.
- O mais perigoso é o **XSS armazenado**, onde um invasor pode inserir dados maliciosos no aplicativo lateral ao servidor, que será processado pouco depois, quando a vítima utilizá-lo.

Cross-site scripting no Navegador

Que não envolve um **servidor**, mas abusa do fato de que o **código Java Script** na própria página processa partes do URL da página. Esse ataque é lançado de uma forma **semelhante ao XSS refletido**.

Código do lado do servidor → **XSS** ← **Execução de scripts no navegador**

Por isso, se há um problema de injeção, a solução parece simples: **sair dos meta-caracteres e garantir a segurança.**

Mas, infelizmente, isso não é tão simples:

- HTML consiste em linguagens diversas, uma incorporada dentro da outra, onde todas possuem diferentes meta-caracteres e, portanto, diferentes regras de saída. Se for preciso tirar um valor, será, portanto, sensível ao contexto
- A maioria dos navegadores tem suas próprias particularidades sobre como eles analisam a mistura de linguagens embutidas, onde, regras claras, às vezes, não existem, e nós podemos apenas tentar evitar problemas mais óbvios de XSS.

Minimizando o XSS

Minimizações que não conseguem impedir que o XSS aconteça, mas podem reduzir o seu impacto:

- O cookie atributo ‘HttpOnly’ pode bloquear o ‘Java Script’ no acesso aos cookies
- Impedir que os arquivos carregados sejam exibidos na mesma página web que o aplicativo

Ao adicionar o cabeçalho:

X-Content-Disposition: attachment

Salvar o documento ou abri-lo em um visualizador externo?

Dessa maneira, se é aberto em outra página, será diferente da **URL** do aplicativo e o modelo de segurança do navegador irá reduzir o impacto dos ataques. Ponto para defesa!

Dificuldades na Localização de Ataques de Injeção

- **Vários tipos de ataques de injeção**
- **Quando a entrada é armazenada em algum lugar no sistema, e depois processada por outra parte do sistema, ou por um sistema completamente diferente.**
- **Um aplicativo pode registrar a entrada do usuário para um arquivo de registro, mas aquele arquivo de registro é visto por um administrador usando um aplicativo especial, que é vulnerável a um ataque de injeção superior**

Termos Módulo 3

TERMOS PARA O EXAME	Ficou claro? SIM ou NÃO?
Injeções de SQL	
Consultas diretas	
Estouro de Buffer	
Consultas parametrizadas	
Lista negra (Black List)	
Saída de meta-caracteres	
Meta-caracteres	
Expressão Regulares	
Normalização	
Codificação de Entrada	
Lista branca	
Transbordamento de dados (*Buffer Overflow*)	
Transbordamento de Pilha (*Stack Overflow*)	
Cross-site scripting (XSS)	

Teste

Pronto para o próximo?

Feche a tela do seu browser e vá para o próximo módulo

OFICIAL

EXIN SECURE PROGRAMMING

Curso Preparatório para Certificação Secure Programming

Área de Aprendizagem

PMG academy

www.pmgacademy.com

Official Course

Módulo 4

Autorização

Autorização

Autorização

É o processo de verificar se alguém tem permissão para realizar uma operação.

Autenticação

É o processo de verificar a identidade de alguém.

A autorização acontece em duas dimensões, que nós chamamos de autorização horizontal e autorização vertical. Às vezes, condições extras se aplicam afora destas duas dimensões.

Autorização Horizontal, Vertical e Outras Dimensões

Autorização horizontal

Por exemplo:

- O Editor de um jornal tem permissão específica para ler, modificar e publicar artigos dentro de uma determinada seção do jornal.

Autorização vertical

Por exemplo:

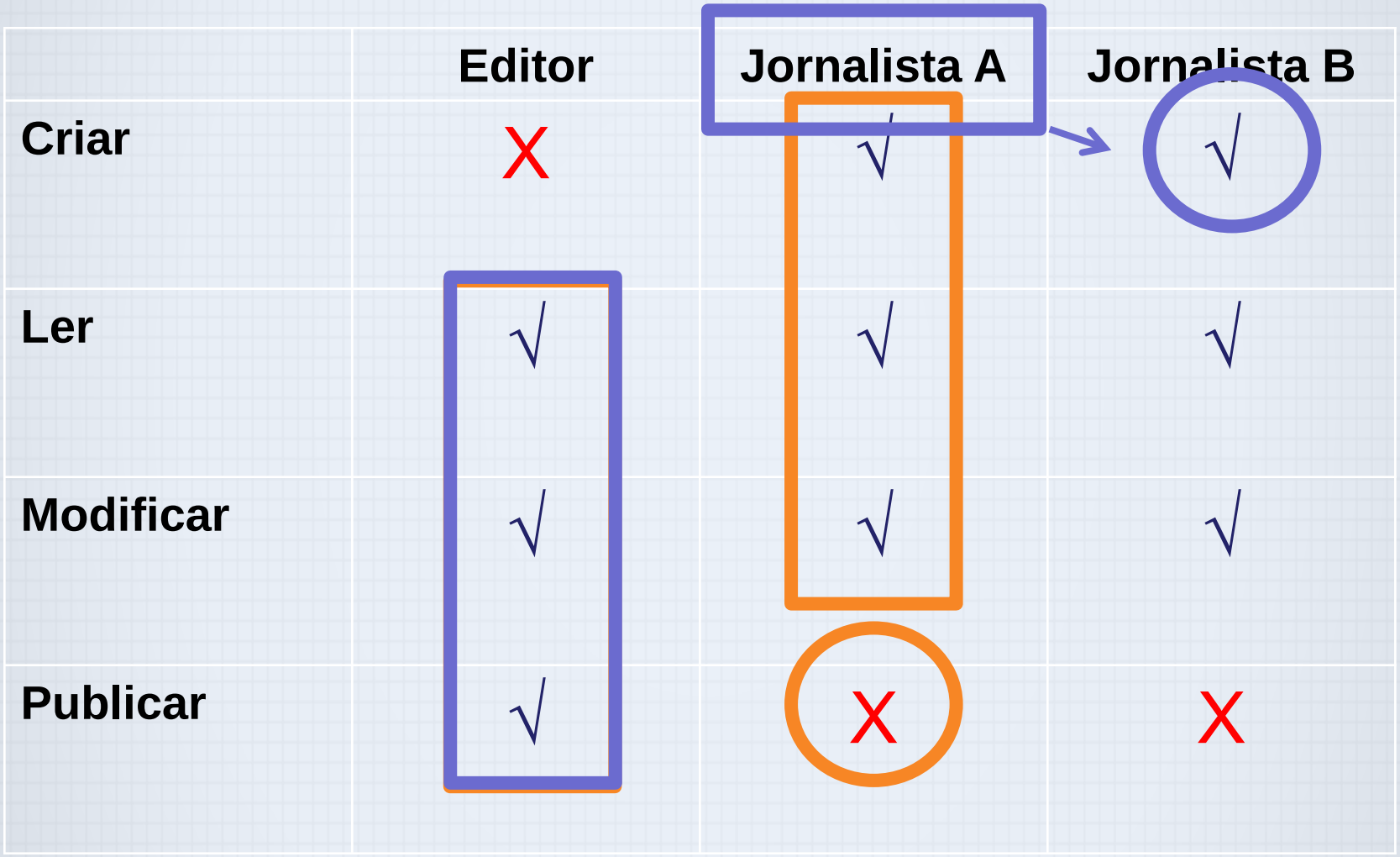
- Um editor de jornal tem permissão específica para ler, modificar e publicar artigos, enquanto um jornalista tem permissão apenas para ler e modificar artigos, mas não publicá-los.

Outras dimensões

Por exemplo:

- Um Jornalista já não pode editar um artigo depois de ter sido publicado, ou um artigo só pode ser publicado em um jornal que ainda não foi lançado.

Escalada de Privilégios

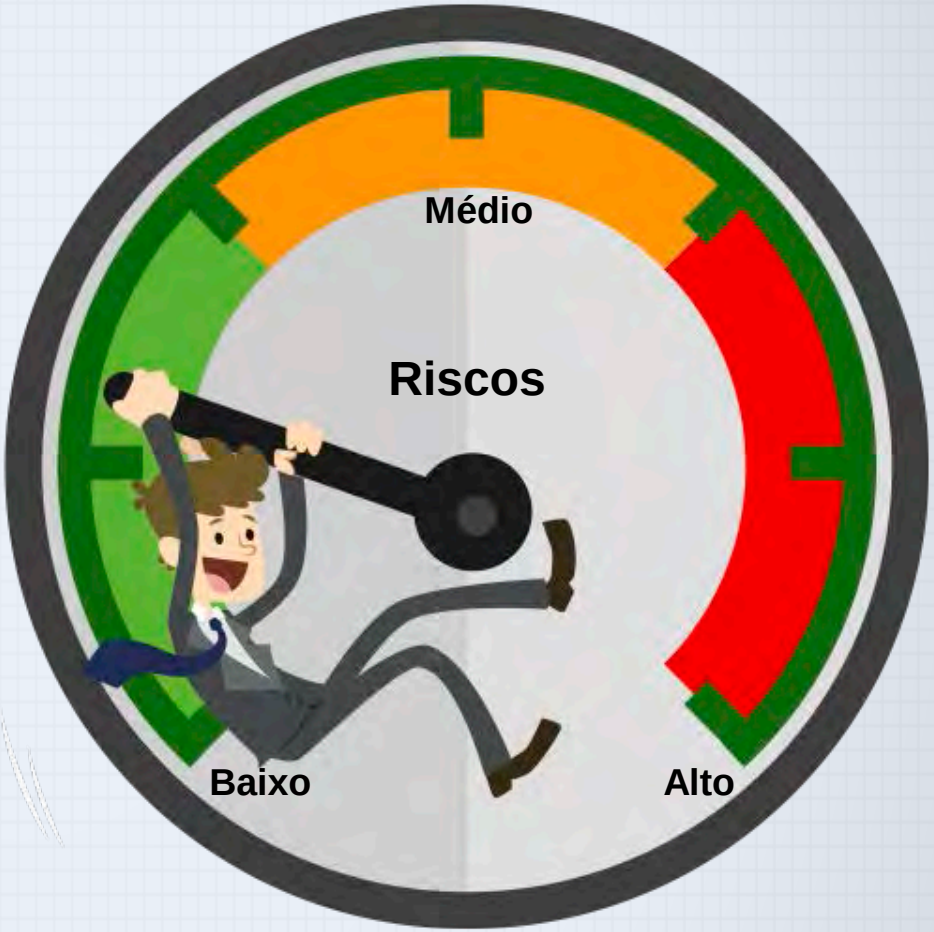


Riscos

Escalonamento 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.



Mitigação

Para impedir problemas de autorização

Mas, cuidado!

verificações de autorização

- Verificar se todos os requisitos de autorização podem ser expressos com ele, para que fique claro onde verificações adicionais de autorização precisam ser implementadas.
- Garantir que seu código seja executado com o menor privilégio possível

princípio do menor privilégio

Referências Diretas e Indiretas

Os parâmetros de solicitação de edição contêm um **número absoluto** do estudante.

Referência indireta

as solicitações contêm um índice relativo a partir de uma lista de alunos da turma

Alterar esse número para um número de estudante de outra classe pode ser capaz de editar os dados do aluno, se verificações de autorização não são executadas

Um número de fora da lista não é válido, e, os números válidos apenas apontam para os estudantes de dentro da turma.

Condições de Corrida

- Criar e configurar as permissões adequadas antes de gravar dados sensíveis.

Condições de corrida

Ataque TOCTOU

Tempo de verificação

TOC

Tempo de uso

TOU

As permissões são absolutamente vitais quando se compartilham recursos!

Operação Atômica

Exemplo de Condições de Corrida ou ataque TOCTOU

```
if (!access("/tmp/mydata.txt",W_OK)) {      // TOC: check if
wehave write permission

    // ...                                  // ATTACK WINDOW

    f=fopen("/tmp/mydata.txt","w+");        // TOU: open the
filefor writing

    write_data_to_file(f);
```

Envenenamento de Sessão

checkout

Ataques de envenenamento de sessão

Esses valores são armazenados em dados da sessão do aplicativo

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

Antes de concordar com o pagamento, contudo, o cliente acrescenta mais itens ao carrinho, repetindo a solicitação "adicionar ao carrinho".

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

Termos Módulo 4

TERMOS PARA O EXAME	Ficou claro? SIM ou NÃO?
Autorização	
Autorização horizontal	
Escalonamento de privilégios	
Acesso a objetos não-autorizados	
Autorização vertical	
Autenticação	
Verificações de autenticação	
Referência indireta	
Princípio do menor privilégio	
Condições de corrida	
Operação Atômica	
Tempo de verificação (TOC)	
Tempo de uso (TOU)	
Envenenamento de sessão	

Teste

Pronto para o próximo?

Feche a tela do seu browser e vá para o próximo módulo

OFICIAL

EXIN SECURE PROGRAMMING

Curso Preparatório para Certificação Secure Programming

Área de Aprendizagem

PMG academy

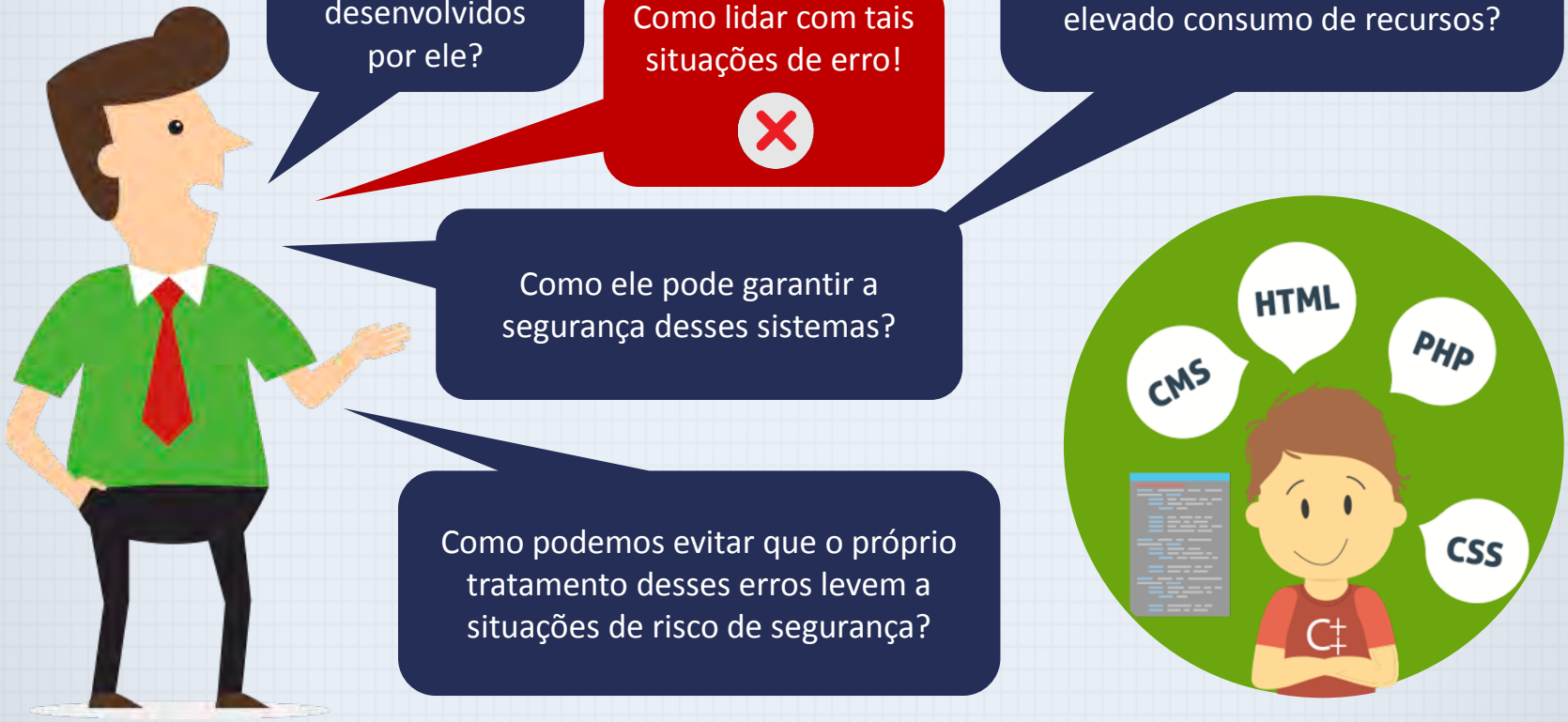
www.pmgacademy.com

Official Course

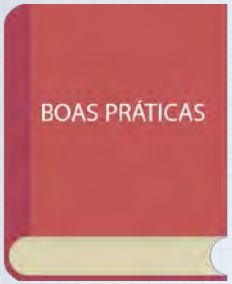
Módulo 5

Configuração, Manejo e Registro de Erros

Configuração, Manejo e Registro de Erros



Configuração e Endurecimento



Reutilização do software

Todo aplicativo usa um software de terceiros.

Ajustá-los a um aplicativo ou ambiente específico.

Checar

Adaptar

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

Funcionamento do Endurecimento

O que um processo de ***Endurecimento*** tenta fazer?

Tenta tanto reduzir a superfície de ataque, como reforçar as partes vulneráveis do sistema.

Para detectar problemas de Endurecimento*:* executar um escaneamento de vulnerabilidade.

Como Endurecimento é feito?

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

O que inclui no processo de Endurecimento?

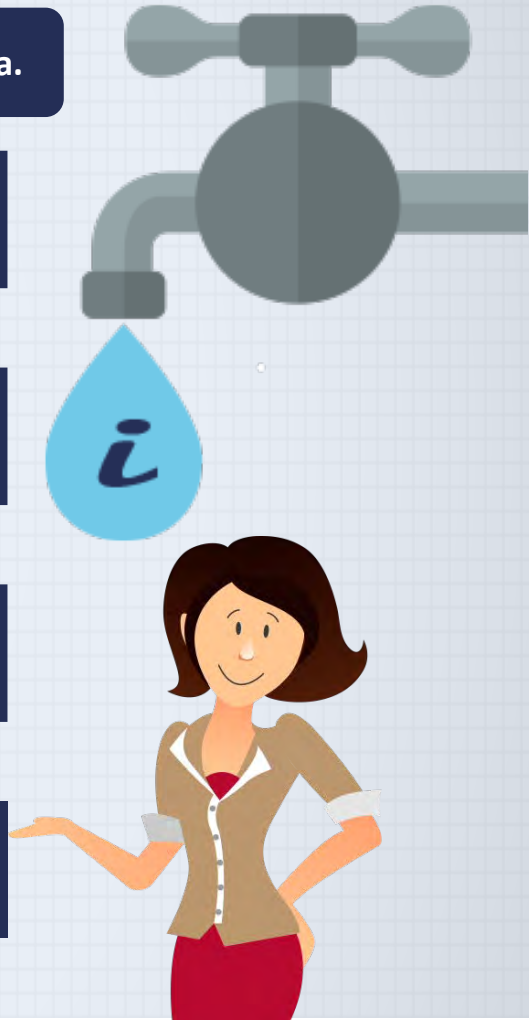
- Aplicar últimas correções de segurança.
- Restringir o acesso a características perigosas.
- Configurar o privilégio de acessos.
- Desativar ou remover os recursos de depuração.
- Remover arquivos desnecessários.
- Alterar as senhas padrão.

Vazamento de Informações

São considerados de baixo risco, porque eles não representam uma ameaça imediata.

- Cabeçalhos de resposta HTTP vazam informações sobre a versão.
- Páginas de erro que exibem rastreamento de pilha, IP internos e nomes de caminhos nos servidores.
- Comentários em páginas HTML que mostram nomes e endereços de e-mail.
- Arquivos de gerenciamento de versão podem vazar código-fonte do aplicativo.

- Comentários HTML em arquivos de modelo
- Informações sobre a versão que o aplicativo envia para fora
- Rastreamento de pilha em produção. Mostre uma mensagem de erro genérica.
- Utilize endereços de e-mail não-pessoais.



Tratamento de Erros

Um **Tratamento de erros** apropriado é essencial para a segurança sempre que um erro ocorre! E deve mesmo ser posto em prática, pois um erro não tratado pode deixar o programa em um estado de vulnerabilidade.

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**.

Registro

Excelente forma de diagnosticar problemas e detectar ataques em um sistema, desde que a informação adequada esteja registrada.

Mas, que informações devem estar registradas?

- Registrar apenas tentativas falhas de login irá detectar um ataque de adivinhação de senha, mas não dirá se o ataque teve êxito
- Combinando-o com o registro bem-sucedido de login pode surtir melhor resultado
- As informações que devem ser registradas são:
  - **O registro das tentativas falhas de login**
  - **O registro de logins bem sucedidos**

Sobre os Registros

"heartbeat"

Entradas de registro

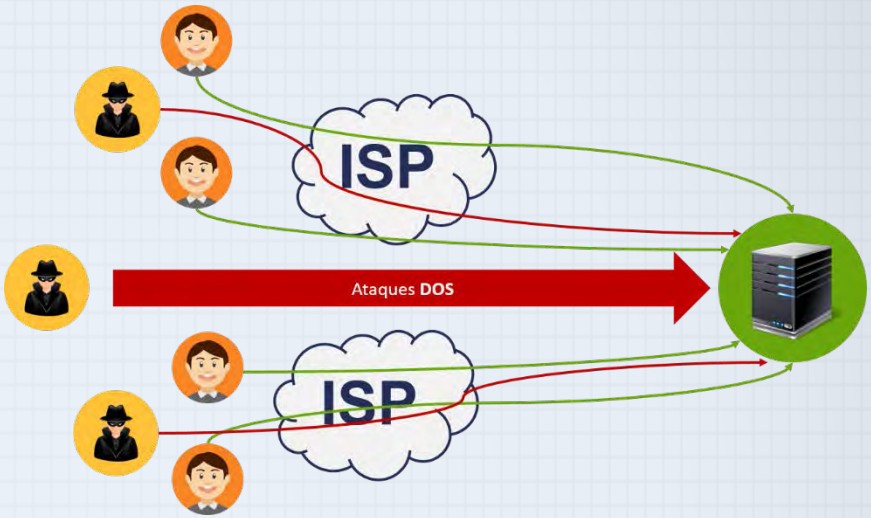
isso ajuda se os relógios de todos os sistemas estão sincronizados.

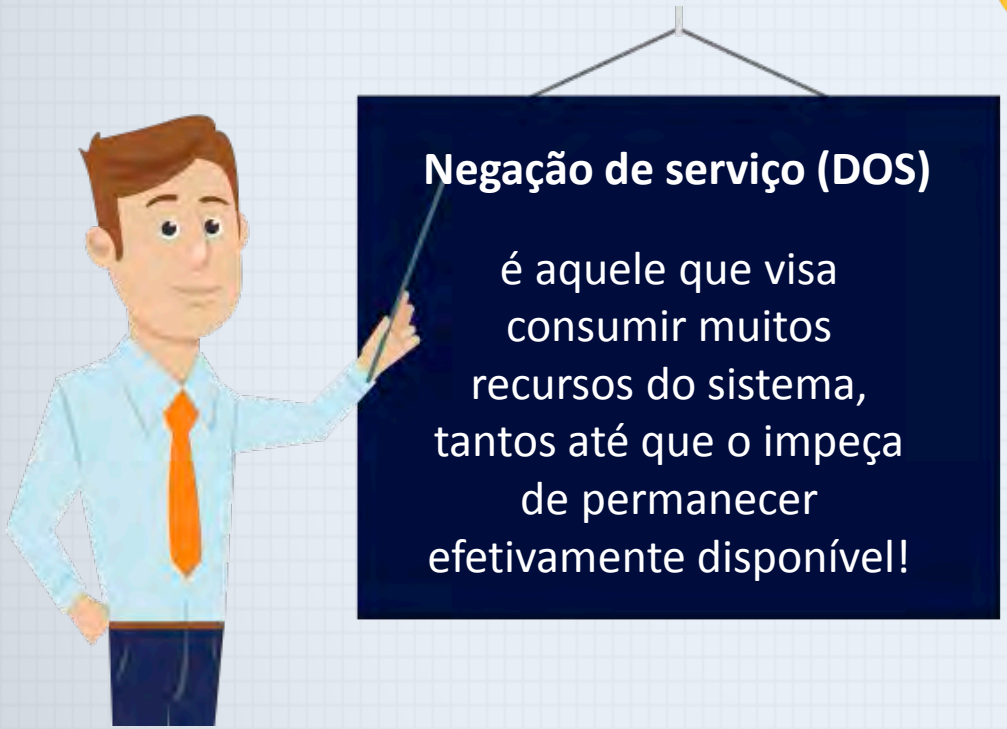
- **Os registros podem ser armazenados em**: **discos** do tipo "escreve uma vez e lê muitas vezes", ou em uma **soma de verificação criptográfica** que pode ser adicionada às entradas do registro.

- **Os registros podem ser analisados automaticamente por ferramentas**.

Negação de Serviço

DDOS: Intrusão em Massa





Negação de serviço (DOS)

é aquele que visa consumir muitos recursos do sistema, tantos até que o impeça de permanecer efetivamente disponível!

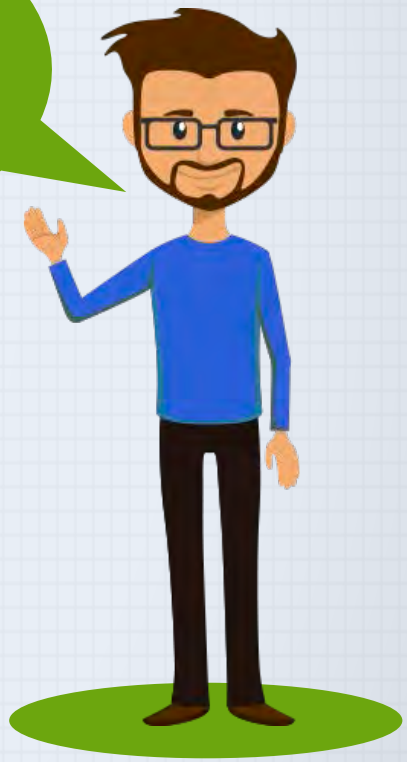
A grande quantidade de tráfego faz com que o servidor fique sem resposta

Mitigando a Negação de Serviço

Ponto para defesa!

- **Limitar a quantidade de recursos que podem ser alocados**

Mas existe possibilidade de atenuação!



Um exemplo dessa simples atenuação é **limitar** um número máximo de itens por pedido.

Se uma operação, por si mesma, provoca grande consumo de recursos (como uma consulta de banco de dados complicada), impor tais limites pode ser mais difícil. Nesse caso, o jeito é fugir da simplicidade e apelar para criatividade! Ser criativo para encontrar uma solução...

Três tentativas falhas de login

Termos Módulo 5

TERMOS PARA O EXAME	Ficou claro? SIM ou NÃO?
Configuração	
Endurecimento	
Vazamento de Informação	
Tratamento de erros	
Mitigar Negação de Serviço	
Ataque de engenharia social	
Distribuição de negação de serviço (D DOS)	
Ataque distribuído de negação de serviço (DOS)	

Teste

Pronto para o próximo?

Feche a tela do seu browser e vá para o próximo módulo

OFICIAL

EXIN SECURE PROGRAMMING

Curso Preparatório para Certificação Secure Programming

Área de Aprendizagem

PMG academy

www.pmgacademy.com

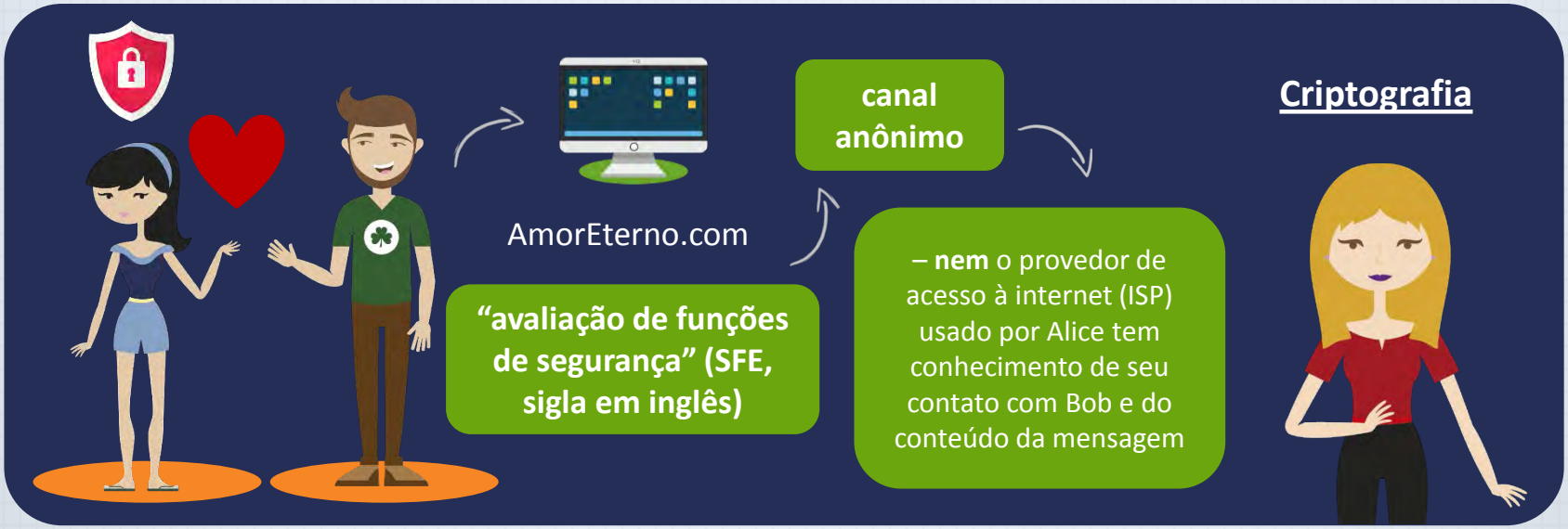
Official Course

Módulo 6

Criptografia

Era um Vez a Criptografia

- **Confidencialidade**
- **Integridade**
- **Autenticação**
- **Não rejeição**
- **Criptografia**
- **Assinaturas digitais**



Criptografia

- OS ISPs utilizados registram todas as atividades, como os sites visitados e os horários dessas ações.
- A criptografia tem métodos matemáticos que protegem a comunicação e a computação contra comportamentos maliciosos.
- A criptografia é uma forma de se comunicar com segurança em um canal inseguro (onde um invasor pode espionar.

- Bob tem uma autorização que Eva não tem.
- O conhecimento privado de Bob é chamado de chave secreta (CS).
- Alice deve ter algum conhecimento sobre a CS de Bob para criar um texto cifrado – ou mensagem criptografada – específico para Bob.

Criptografia de chave secreta

Princípio de Kerckhoffs, formulado em 1883 por Auguste Kerckhoffs (Kerckhoffs, 1883).

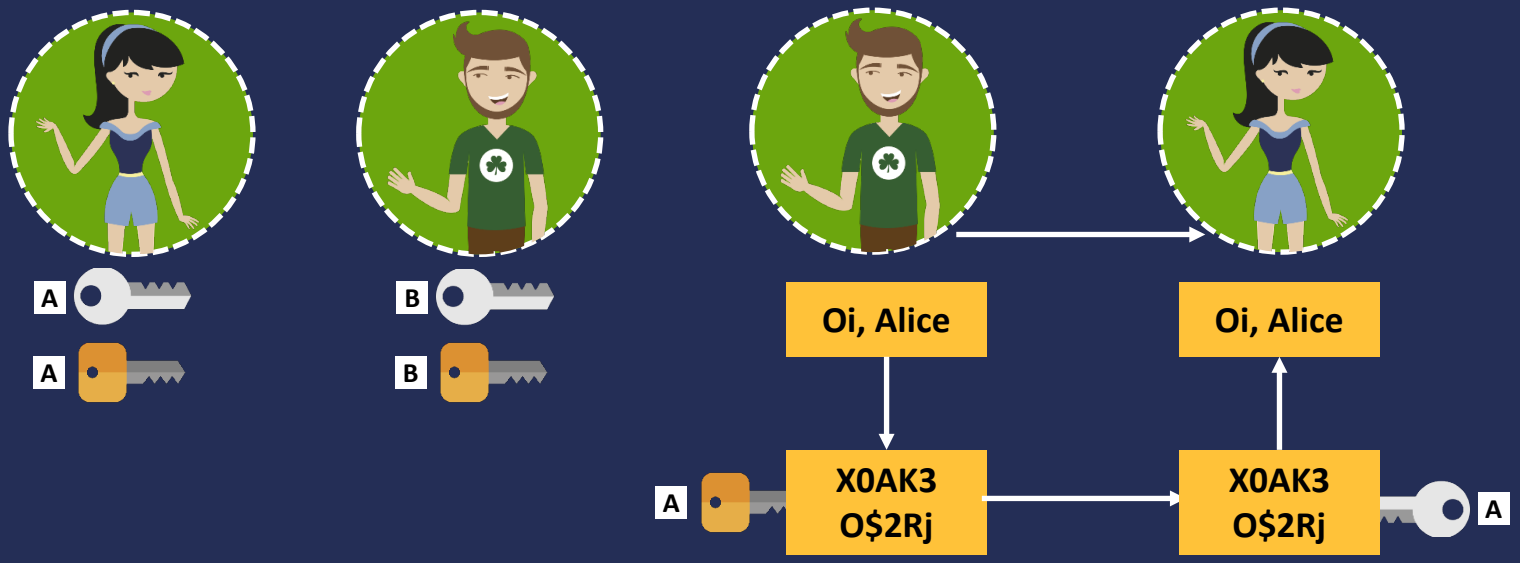
criptografia simétrica e **assimétrica**.

Criptografia Simétrica

Em um sistema de **criptografia simétrica**, todos os participantes usam a mesma chave para criptografar e descriptografar dados. Os participantes devem trocar a chave antes que os dados possam ser descriptografados.

Criptografia Assimétrica

Em um sistema de **criptografia assimétrica**, todos os participantes têm seu próprio par de chaves, que consiste em uma **chave privada** e uma **chave pública**. A chave privada deve permanecer em segredo, enquanto que a chave pública pode ser publicada sem qualquer perigo.

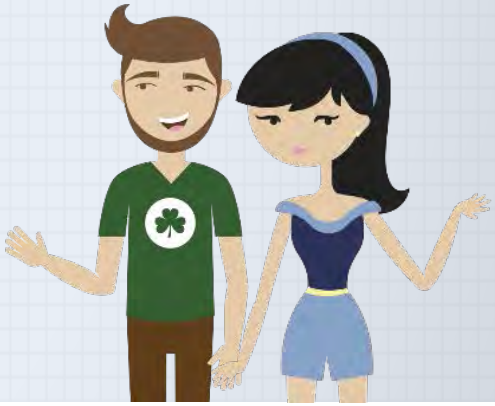


Na figura, a chave verde representa a chave pública, enquanto a chave rosa reprenta a chave privada.

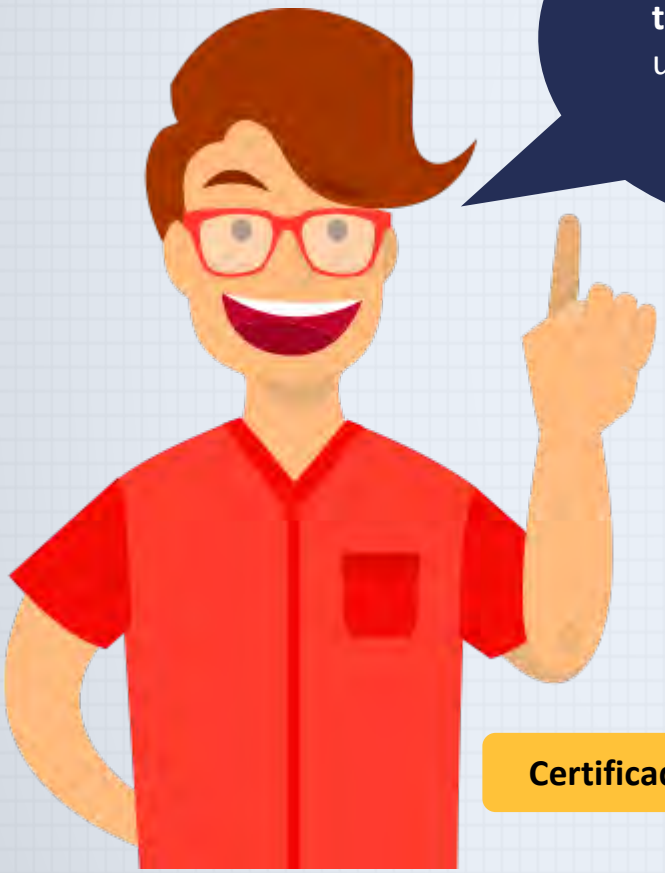
Homem do Meio

Homem-no-meio

- **Ataque funciona assim:**
  - 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 descriptografa 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 descriptografa usando sua chave privada.
  - Bob fica chocado ao ler a mensagem que ele acredita vir de Alice.



Mitigando Problemas de Homem do Meio



Uma maneira de resolver esse tipo de problema é **trocar** as chaves públicas usando um canal seguro, tal como é feito a troca de chaves simétricas.

Outra solução é usar um terceiro confiável (vamos chamá-lo de Trent) para assinar a autenticidade da chave. Isso é chamado de certificado.

Pedro

Certificado raiz – para troca de senha de forma segura.

HTTP

Vamos relembrar os conceitos básicos antes de associá-lo a criptografia?

HyperTextTransferProtocol (HTTP)

"Protocolo de Transferência de Hipertexto".

O HTTP é o protocolo utilizado para **transferência de páginas em linguagem HTML do computador para a Internet**. Por isso, os endereços dos websites utilizam a expressão "http://".

Para que a transferência de dados na Internet seja realizada, o protocolo HTTP necessita estar agregado a outros dois protocolos de rede: **TCP (*TransmissionControlProtocol*)** e **IP (*Internet Protocol*).**

HTTPS

HTTPS é um protocolo tanto para criptografar uma conexão como autenticar a parte remota.

A

É baseado em uma criptografia de chave pública

Autoridade de Certficado (CA)

assinam os certificados de sites web

Gestor de Certificados Digitais (DCM)

Deu para refrescar a memória? Agora vamos voltar ao contexto da criptografia:

Autoridade de Certificado (CA)

Autoridade de Certificado

A chave pública da autoridade de certificação é armazenada no navegador (no chamado certificado raiz),

Há outra autoridade de certificação que assinou um certificado para a autoridade de certificação

Cadeia de certificados

Vazamento Chave Privada

revogação de certificados

Erros do HTTPS

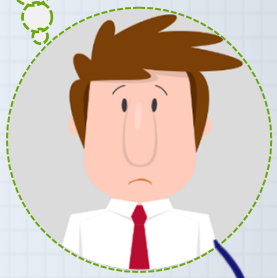
Erros

- Alguns servidores web não impõem HTTPS
- A interface do usuário do navegador é complicada quando um certificado inválido é apresentado
- A autoridade de certificado pode estar comprometida.
- Se uma chave de certificado está comprometida, o certificado não pode mais ser usado e deve ser revogado.
- A revogação é, frequentemente, mal implementado em navegadores
- A verificação de certificado pode não ser corretamente implementada (**CIPHER – CIFRA**)
- A conexão não pode usar a criptografia mais forte possível.
- Erros nos protocolos ou no software podem levar a problemas

HTTP Strict Transport Security (HSTS).

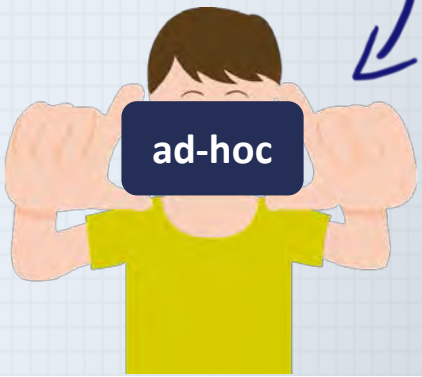
HTTPS

A infraestrutura de certificado não aumenta ou não funciona para a web



HTTP Strict Transport Security (HSTS)

Um navegador que implementar essa medida irá recordar um certificado de servidor e recusar qualquer comunicação se aquele certificado não for usado.



Outros Problemas com Criptografia:

Dicas

- Não invente seus próprios algoritmos de criptografia: eles serão quebrados
- Siga as diretrizes de implementação para os algoritmos e protocolos
- Preste atenção ao gerenciamento de chave:
  - Onde as chaves estão armazenadas
  - 0 quão forte a chave deve ser
  - Como você lida com chaves comprometidas
  - Como as chaves estão distribuídas

Termos Módulo 6

TERMOS PARA O EXAME	Ficou claro? SIM ou NÃO?
Criptografia	
Criptografia assimétrica	
Princípio de Kerckhoffs	
Chave privada	
Chave pública	
Criptografia simétrica	
Autoridade de certificação	
Cadeia de certificados	
Revogação de certificado	
Ataque homem-do-meio	
HTTP e HTTPS	
Strict Transport Security (HSTS)	
Certificado	
Aleatoriedade	

Teste

Pronto para o próximo?

Feche a tela do seu browser e vá para o próximo módulo

OFICIAL

EXIN SECURE PROGRAMMING

Curso Preparatório para Certificação Secure Programming

Área de Aprendizagem

PMG academy

www.pmgacademy.com

Official Course

Módulo 7

Engenharia de Software Seguro

Engenharia de Software Seguro

A velocidade com que o software precisa ser desenvolvido seduz os seus desenvolvedores a tomarem atalhos, que, muito provavelmente, resultam em segurança sacrificada. O que é muito ruim...

O ciclo de vida de desenvolvimento de um software comum (SDL) contém quatro fases

- **Requisitos**
- **Projeto**
- **Implementação**
- **Verificação**

Requisitos de Segurança

Requisitos de Segurança — muitas vezes, esquecidos

Não tem conhecimento suficiente para perguntar o que exatamente a segurança significa

espera que o cliente irá especificar como exatamente o software precisa ser protegido.

Comunicar sobre um tópico em que nenhum deles é especialista

E para cada ataque no STRIDE, podemos associar uma propriedade de segurança.

Propriedade de segurança STRIDE

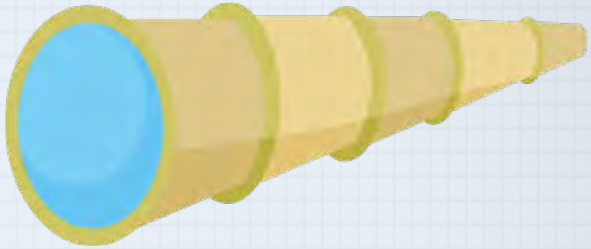
Ataque	Propriedade de Segurança	Exemplo de Medida
Forjamento de Identidade	Autenticação	Senhas
Adulteração de dados	Integridade	Manipulação de entrada
Rejeição	Não Rejeição	Registro
Divulgação de informações	Confidencialidade	Criptografia
Recusa de Serviço	Disponibilidade	Limitar o consumo de recursos
Elevação de privilégios	Autorização	Realizar verificações de autorização

STRIDE nos Ativos de TI

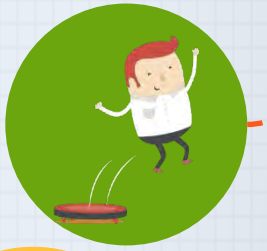
Propriedades de segurança **STRIDE**

Registros médicos

Dados financeiros

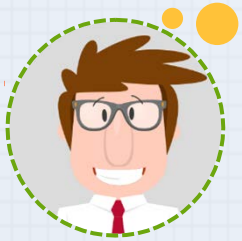


os quais são focos constantes de **ataques** maliciosos



TRAMPOLIM - As informações que auxiliam a chegar aos dados reais

A senha do usuário



Ativo

Um ativo deve conter requisitos de segurança, assim como qualquer outro ativo que dependa das propriedades de segurança (STRIDE)

Ativos de Segurança

Qualquer outro valor é um identificador de produto inválido.

Propriedades STRIDE

se sobrepõem, o que significa que, às vezes, podemos associar um requisito com uma ou mais propriedades. E isso não é um problema, pois não estamos interessados em categorizar nossos requisitos, mas na verdade queremos reunir tantos requisitos quanto possível. Portanto, um pouco de sobreposição apenas aumenta a probabilidade de encontrarmos esses tais requisitos.

Desenvolvedor

Implementa uma verificação de validação de entrada.

Testador

Testa o que acontece se um identificador de produto inválido for introduzido

Exemplos de Requisitos de Segurança

Autenticação

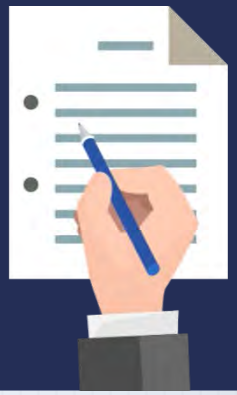
Integridade

Não rejeição

Confidencialidade

Disponibilidade

Autorização



Exemplos de Situações que Podem ser Exploradas

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

"O editor não pode editar artigos que já foram publicados."

"Um editor pode apenas editar artigos que ele criou.

Arquitetura de Segurança

Requisitos

Especificam o que o sistema deve ou não fazer

Arquitetura ou o projeto

Especifica como isso será implementado

E quais tecnologias serão usadas?

Como são as interações entre eles e o mundo exterior?

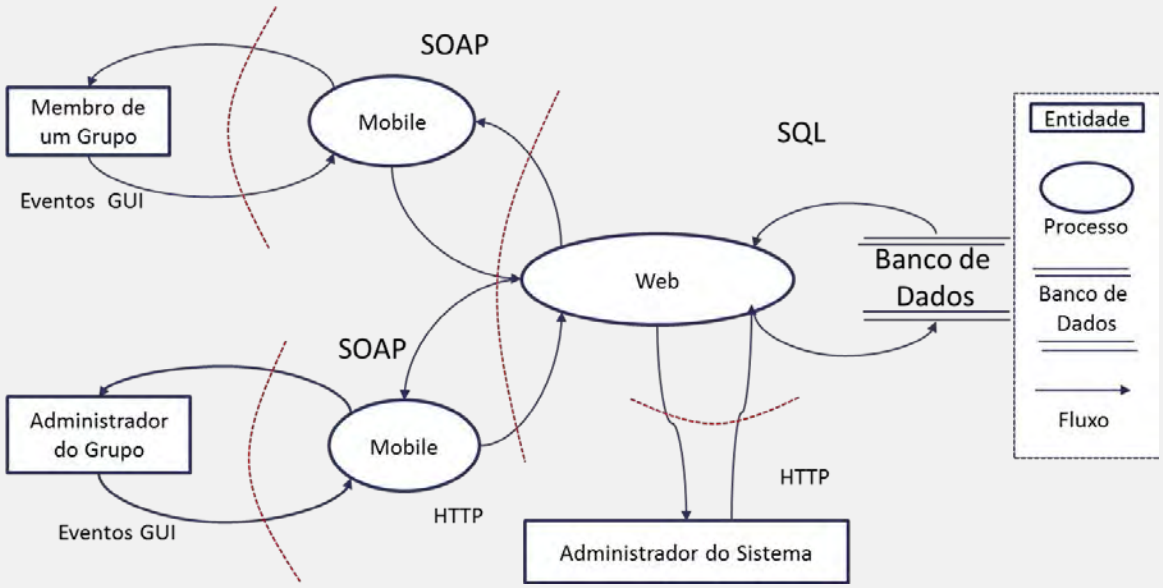
Análise de riscos de arquitetura

Microsoft →

Modelagem de ameaças

Diagrama de fluxo de dados (DFD) - pois nos mostra como os dados são processados pelo sistema e nos permite identificar onde um invasor pode atacar.

Diagrama de Fluxo de Dados (DFD)

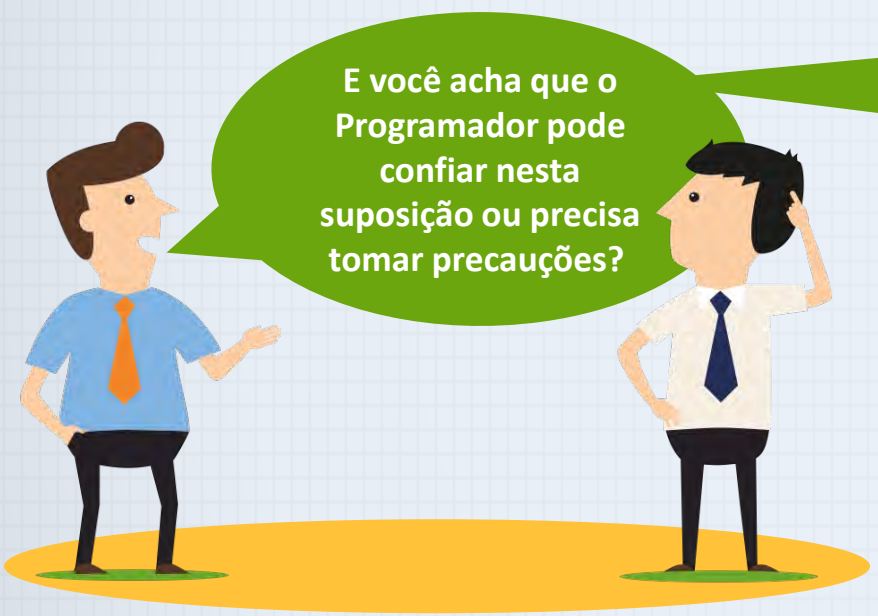


- O retângulo fechado em um diagrama de fluxo de dados é uma entidade externa
- Os ovais são os processos, o que significa que os dados que entram podem ser processados, por exemplo, o servidor SFTP
- As linhas duplas horizontais são o armazenamento de dados, por exemplo o diretório de envio.
- As setas entre esses elementos são os fluxos de dados.
- As linhas tracejadas definem os limites de confiança.

Ataques Comuns

- A possibilidade de haver outros sistemas com os quais o sistema se comunica e que, porventura, o Programador não tenha considerado antes

- Pode descobrir processos que podem representar um risco para o sistema



Ataque Comum Padrão de Enumeração e Classificação (CAPEC).

(veja "Ataque Comum Padrão de Enumeração e Classificação (CAPEC): Um Recurso Comunitário para Identificar e Entender Ameaças," 2014).

Mitigando na Arquitetura

Mitigação Técnica

- **A utilização de um determinado componente ou serviço, ou por programação de uma determinada forma**
- **Detectando o problema e tendo um cenário para reduzir o impacto**

- Analisar o impacto de segurança na aplicação da atenuação antes de decidir implementá-la

- **Exemplo:**

Implementar um serviço de **assinatura** única como uma medida de autenticação para proteger contra ataques de imitação pode ser uma boa ideia

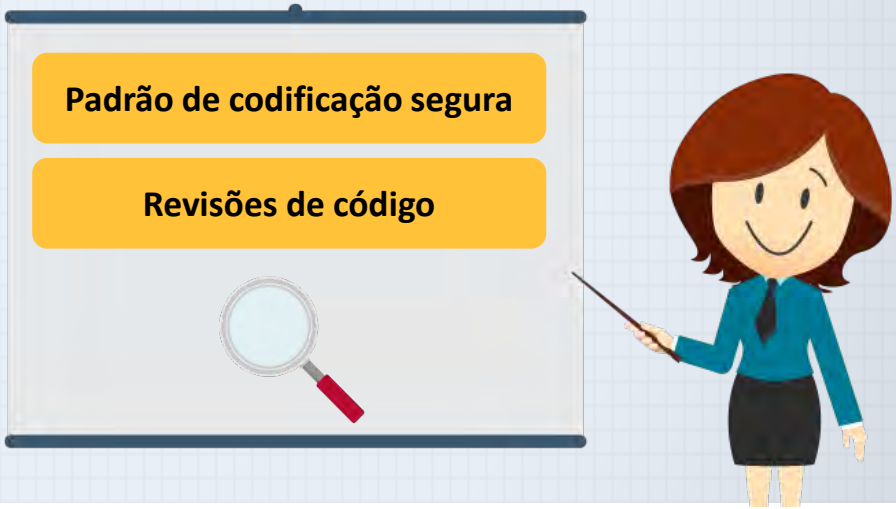
Codificação Segura

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

- Não é necessário escrever um padrão de codificação a partir do zero, pois muitos já estão disponíveis para as linguagens de programação mais comuns.

Revisão de código de segurança ≠ **Revisão de código comum**

É que a cobertura da revisão de código de segurança precisa ser muito maior.



Ferramentas de Revisão de Códigos

Muitas ferramentas automatizadas existem para ajudar com esse problema.

Ferramentas de análise estática

Analisam o fluxo de dados no aplicativo e detectam construções inseguras que são pontos de vulnerabilidade em potencial.

Desvantagens

- Precisam de personalização para um ambiente de desenvolvimento específico (linguagem de programação, sistemas, e assim por diante).
- Detectam muitos pseudo problemas (os chamados falsos positivos), enquanto que os problemas reais nem sempre são detectados...
- O serviço para ser eficaz é preciso que haja também uma revisão manual para interpretar os resultados e detectar os problemas que a ferramenta não consegue encontrar.

Testes de Segurança

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

Especialmente quando os testes podem ser automatizados, eles são um método excelente para detectar os problemas sem muito esforço. Contudo, o teste não vai encontrar todos os problemas, e se o código é apenas testado sem uma revisão de código de segurança, muitas fraquezas no código permanecerão despercebidas.

Revisões de Código

+

Testes

=

Melhores resultados

Testes de Penetração

Um especialista em segurança tenta atacar o sistema antes que os invasores tenham a chance de fazer isso

Teste de penetração

ou para obter uma imagem dos problemas de segurança mais urgentes.

Conforme uma equipe de desenvolvimento amadurece e começa a implementar mais práticas de segurança, a eficácia de um teste de penetração pode ser reduzida.

Desvantagem

Penetração é que eles só podem ser executados relativamente tarde no processo de desenvolvimento, pois os testadores precisam de um sistema de trabalho que oferece toda a funcionalidade necessária.

Escaneamentos de Vulnerabilidade

Escaneamento de vulnerabilidade

- **Endurecimento**
- **Escaneamentos específicos para cross-site-scripting**

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

Não requerem experiência em segurança para executá-los

Economizam tempo

Testes de Abuso

Os **escaneamentos** de vulnerabilidade não irão detectar as vulnerabilidades específicas do aplicativo, e testadores de penetração são improváveis de encontrá-las se eles também não têm o conhecimento específico do aplicativo.

Problema

Eles podem não ter conhecimento de segurança para criar um teste de segurança eficiente. No entanto, se a equipe **criou** os requisitos de segurança e identificou as ameaças de arquitetura, isso já é o necessário para criar um teste de segurança específico do aplicativo.

Testes de abuso

Podem ser escritos usando os mesmos sistemas de teste que são usados para outros testes automatizados, tais como **testes de aceitação, testes unitários e testes de regressão**.

Fuzzing - Parte I

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.

Google

("FFmpeg e as mil correções", 2014).

Por se tratar de uma companhia grandiosa como o Google, eles foram capazes de usar **2000 núcleos de CPU**, mas mesmo com menos recursos, a prática de **fuzzing** pode ser eficaz.

Fuzzing – Parte II

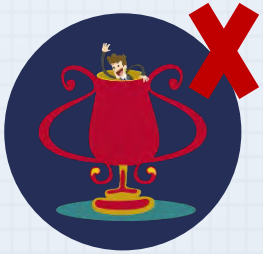
Uma boa configuração de **fuzzing:**

- Um gerador que gera entrada de teste
- Um mecanismo de entrada que garante que o sistema processe essa entrada
- Um monitor que observa como o sistema responde
- Um agente que pode reiniciar o sistema ou recuperar após um incidente.

- O teste terá pouco efeito se o seu gerador enviar mensagens XLM inválidas (embora esses testes possam apontar problemas interessantes com o **Analista XML (XML Parses)**
- Considerada a técnica mais eficaz para testar código que processa a entrada, tais como os **Analista XML**

Um teste de **fuzzing** não é objetivado

A principal razão para sua eficácia é o fato de que ele pode executar de forma autônoma por um longo tempo e tentar muitos casos de teste



Outros métodos de verificação podem ser mais eficazes

Teste de Feedback

Métodos de verificação → **Revisões de código** | **Testes de segurança**

fornecem redes seguras em diferentes pontos no processo de desenvolvimento do software.

Se você usa os resultados dessas verificações (chamadas de 'achados') para aperfeiçoar tanto o processo de desenvolvimento como as verificações, essas redes seguras ficarão mais fortes e mais eficazes ao longo do tempo.

- **Mudanças no seu código**
- **Mudanças nos requisitos**
- **Mudanças na arquitetura do sistema**

"**Embora** nós ainda não tenhamos muitos estudos de caso para apoiar o caso de negócio, os estudos de caso e os dados associados que identificamos são muito convincentes." (Meadetal.,2009)

"**Embora** os esforços de segurança de software possam requerer algum comprometimento de recursos, você pode conseguir um ROI significativo, muitas vezes, com uma pequena despesa inicial." (Microsoft SDL: Return-on-Investment, 2009)

"... **nós concluímos** que o processo SDL da Microsoft proporciona, de fato, retorno sobre o investimento para além dos custos de implementação.""(Cavit,2011)

Termos do Módulo 7

TERMOS PARA O EXAME	Ficou claro? SIM ou NÃO?
Ciclo de vida do desenvolvimento de software	
Ativos de Segurança	
Não rejeição	
Análise de Risco de Arquitetura	
Diagrama de fluxo de dados (DFD)	
Modelagem de ameaças	
CAPEC - Ataque Comum, Padrão de Enumeração e Classificação	
Revisão de código	
Enumeração Fraqueza Comum (CWE)	
Teste de penetração	
Ferramentas de análise estática	
Testes de abuso	
Escaneamento de vulnerabilidades	
Fuzzing	
Analista XML	

Teste

Pronto para o próximo?

Feche a tela do seu browser e vá para o próximo módulo