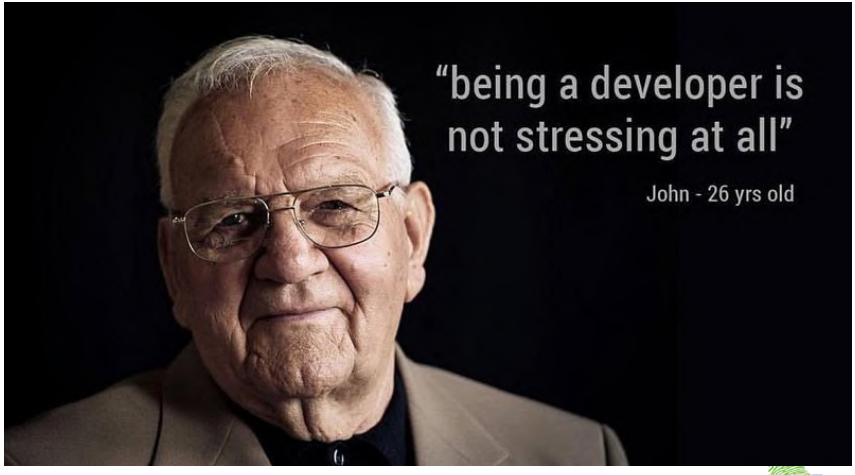
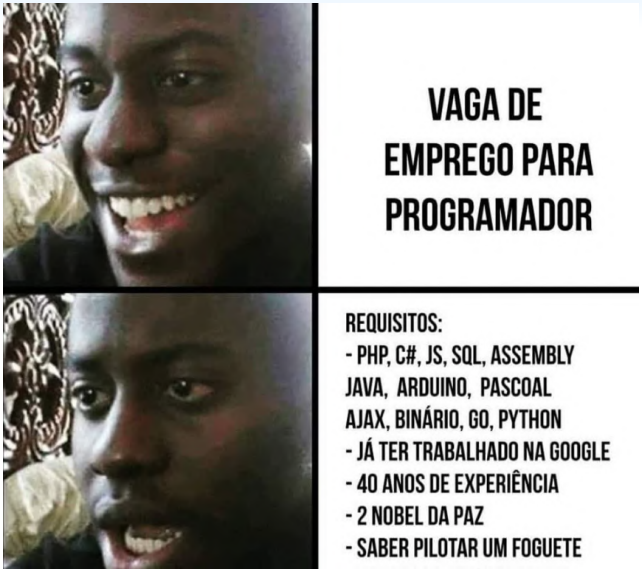
Introdução ao Desenvolvimento de Software Seguro

Frederico Schardong

Laboratório de Segurança em Computação (LabSEC)
Universidade Federal de Santa Catarina (UFSC)
Instituto Federal do Rio Grande do Sul (IFRS)


“being a developer is not stressing at all”
John - 26 yrs old



LabSEC
Laboratório de Segurança em Computação

(Algumas) Tarefas do Desenvolvedor

- ~~Levantar~~ Entender os requisitos de software
- ~~Definir~~ Seguir uma metodologia de desenvolvimento (cascata, SCRUM, XP, etc)
- Dominar múltiplas linguagens, paradigmas e tecnologias
- Criar código sem erros
- Criar código com performance adequada
- ...
- Criar **código seguro**

(Algumas) Tarefas do Desenvolvedor

- ~~Levantar~~ Entender os requisitos de software
- ~~Definir~~ Seguir uma metodologia de desenvolvimento (cascata, SCRUM, XP, etc)
- Dominar múltiplas linguagens, paradigmas e tecnologias
- Criar código sem erros
- Criar código com performance adequada
- ...
- Criar **código seguro** $\rightarrow$ código sem **vulnerabilidades**

Vulnerabilidades

- Vulnerabilidades são falhas/fraquezas de software ou hardware
- Caso exploradas (em inglês *exploit*), acarretam em incidentes de segurança:
  - Acesso à informações confidenciais
  - Escalada de privilégios
  - Ataques contra outros serviços
  - Tornar serviços indisponíveis

Exploit de Vulnerabilidades

- O *exploit* de vulnerabilidades depende de:
  1. A existência de uma vulnerabilidade

Exploit de Vulnerabilidades

- O *exploit* de vulnerabilidades depende de:
  1. A existência de uma vulnerabilidade → todos produzem código com imperfeições

Exploit de Vulnerabilidades

- O *exploit* de vulnerabilidades depende de:
  1. A existência de uma vulnerabilidade → todos produzem código com imperfeições
  2. Acesso do atacante à falha

Exploit de Vulnerabilidades

- O *exploit* de vulnerabilidades depende de:
  1. A existência de uma vulnerabilidade → todos produzem código com imperfeições
  2. Acesso do atacante à falha → se está na Internet, todos tem acesso

Exploit de Vulnerabilidades

- O *exploit* de vulnerabilidades depende de:
  1. A existência de uma vulnerabilidade → todos produzem código com imperfeições
  2. Acesso do atacante à falha → se está na Internet, todos tem acesso
  3. Conhecimento e capacidade do atacante em explorar a falha

Exploit de Vulnerabilidades

- O *exploit* de vulnerabilidades depende de:
  1. A existência de uma vulnerabilidade → todos produzem código com imperfeições
  2. Acesso do atacante à falha → se está na Internet, todos tem acesso
  3. Conhecimento e capacidade do atacante em explorar a falha → atacante tem 24x7x365 para explorar teu sistema

Repositório Mundial de Vulnerabilidades e *Exploits*

- *Common Vulnerabilities and Exposures* (CVE)
  - Banco de dados que registra vulnerabilidades e *exploits* publicamente conhecidos
  - Mantido por uma ONG financiada pelo governo americano
  - Desde 1999

Monitoramento e Notificação

- Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br):
  - Monitora e notifica vulnerabilidades e incidentes em sistemas e redes conectadas à Internet no Brasil
  - Coordena e dá o apoio necessário às partes afetadas
  - Atua em conjunto com CERT de outros países, forças policiais, provedores de internet e outras entidades

OWASP

- *Open Web Application Security Project* (OWASP)
- Publica artigos e manuais relacionados a segurança
- OWASP top 10
  - Relatório atualizado a cada 4 anos
  - Classificação de riscos e como remediá-los

1 - Controle de Acesso Quebrado

Descrição:

- Divulgação não autorizada, modificação ou destruição de dados fora dos limites do usuário

Causa:

- Adulteração de URL ou da página HTML

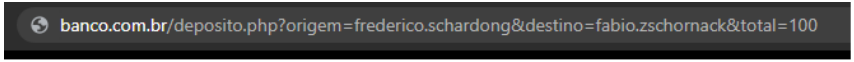
1 - Controle de Acesso Quebrado

Descrição:

- Divulgação não autorizada, modificação ou destruição de dados fora dos limites do usuário

Causa:

- Adulteração de URL ou da página HTML



LabSEC
Laboratório de Segurança em Computação

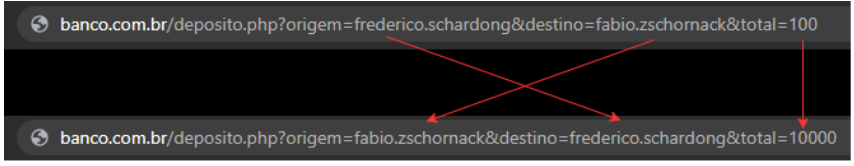
1 - Controle de Acesso Quebrado

Descrição:

- Divulgação não autorizada, modificação ou destruição de dados fora dos limites do usuário

Causa:

- Adulteração de URL ou da página HTML



LabSEC
Laboratório de Segurança em Computação

1 - Controle de Acesso Quebrado

Descrição:

- Divulgação não autorizada, modificação ou destruição de dados fora dos limites do usuário

Causa:

- Adulteração de URL ou da página HTML

Solução:

- Implementar mecanismos de controle de acesso uma vez e reutilizá-los em todo o aplicativo

2 - Falhas Criptográficas

Descrição:

- Exposição de dados confidenciais

Causa:

- Falhas relacionadas à criptografia (ou a falta dela)

2 - Falhas Criptográficas

Descrição:

- Exposição de dados confidenciais

Causa:

- Falhas relacionadas à criptografia (ou a falta dela)

Usuario	
PK	usuario_id INT
	nome VARCHAR(255)
	e-mail VARCHAR(255)
	senha VARCHAR(8)

2 - Falhas Criptográficas

Descrição:

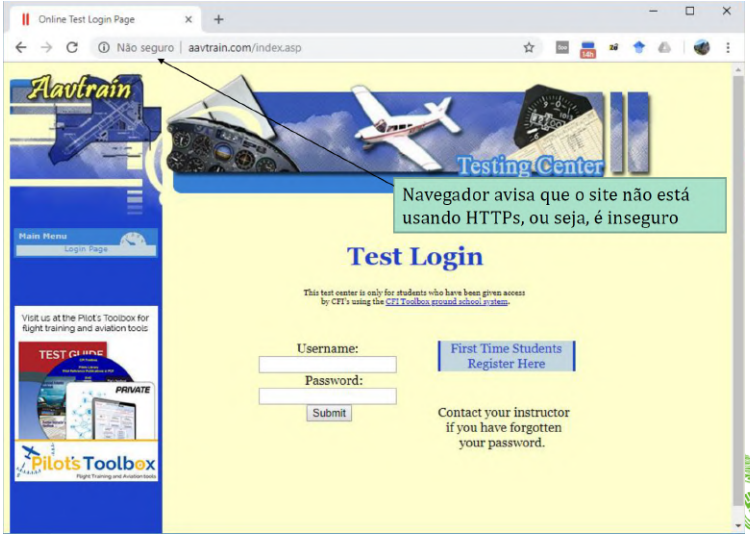
- Exposição de dados confidenciais

Causa:

- Falhas relacionadas à criptografia (ou a falta dela)

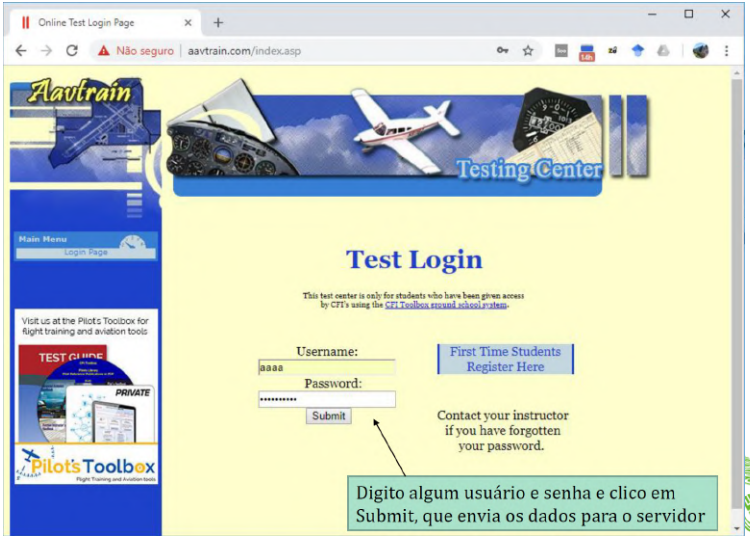
Usuario	
PK	usuario_id INT

2 - Falhas Criptográficas



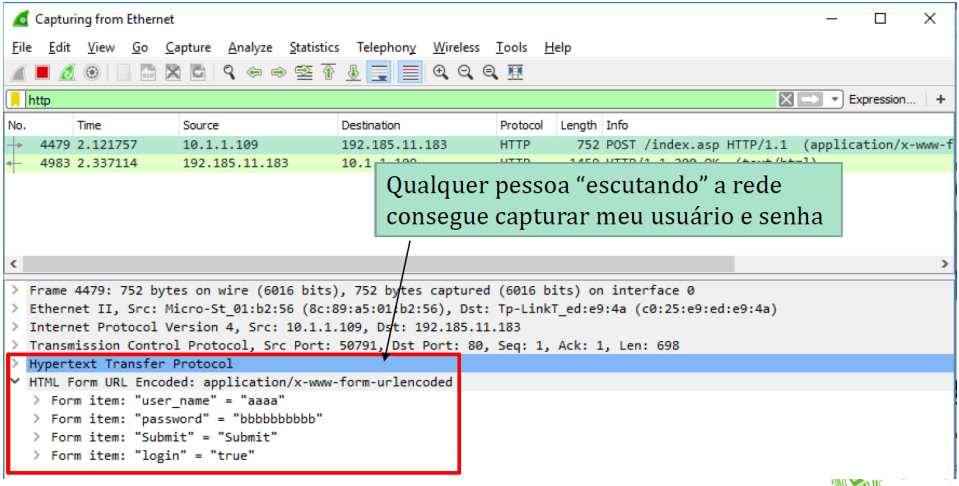
LabSEC
Laboratório de Segurança em Computação

2 - Falhas Criptográficas



LabSEC
Laboratório de Segurança em Computação

2 - Falhas Criptográficas



2 - Falhas Criptográficas

Descrição:

- Exposição de dados confidenciais

Causa:

- Falhas relacionadas à criptografia (ou a falta dela)

Solução:

- Determinar as necessidades de proteção de dados em trânsito e em repouso
- Não armazenar dados confidenciais desnecessariamente

3 - Injeção de Código Malicioso

Descrição:

- Adulteração de URL ou da página HTML

Causa:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

3 - Injeção de Código Malicioso

Descrição:

- Adulteração de URL ou da página HTML

Causa:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

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

3 - Injeção de Código Malicioso

Descrição:

- Adulteração de URL ou da página HTML

Causa:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

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

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

3 - Injeção de Código Malicioso

Descrição:

- Adulteração de URL ou da página HTML

Causa:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

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

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

```
String query = "SELECT \* FROM accounts WHERE custID='" + "' or '1' ='1" + "'";
```

3 - Injeção de Código Malicioso

Descrição:

- Adulteração de URL ou da página HTML

Causa:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

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

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

```
String query = "SELECT \* FROM accounts WHERE custID='" + "' or '1' ='1" + "'";
```

```
String query = "SELECT \* FROM accounts WHERE custID='' or '1' ='1'";
```

3 - Injeção de Código Malicioso

Descrição:

- Adulteração de URL ou da página HTML

Causa:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

Solução:

- Sanitizar dados recebidos de usuários e APIs externas

Enfim... Como criar código seguro?

1. Educando desenvolvedores, testadores, analistas e outros *stakeholders*
2. Incorporando etapas relacionadas a segurança nas metodologias de desenvolvimento de software.

Metodologias de Desenvolvimento de Software Seguro

- *Security Development Lifecycle* (SDL) da Microsoft
- *Comprehensive, Lightweight Application Security Process* (CLASP) da OWASP
- *Software Assurance Maturity Model* da Fortify
- ...

SDL

- Acesso livre e gratuítuo
- Agnóstico a plataforma e tecnologia
- Dividido em 7 etapas

Etapa 1: Treinamento de Segurança

- Estabelecer critérios para treinamentos
- Estabelecer frequência mínima de treinamento
- Estabelecer limites mínimos de treinamento de grupo

Etapa 2: Requisitos de Segurança

- Estabelecer requisitos de segurança para todo o projeto, ex:
  - Identificar líderes de segurança
  - Estabelecer fluxo de correção de bugs de segurança
  - ...
- Conduzir análise de risco (ex: ISO 27001)

Etapa 3: Design

- Estabelecer requisitos de design de segurança, ex:
  - Primitivas criptográficas
- Análise de superfície de ataque
- Modelo de ameaça

Etapa 4: Implementação

- Usar ferramentas previamente aprovadas para o desenvolvimento do sistema, ex:
  - Compiladores
  - Ferramentas de teste
- Banir o uso de funções e APIs não seguras
- Uso de ferramentas de análise de código, ex:
  - ASVS
  - SAST
  - DAST
  - Sonar qube
  - Semgrep

Etapa 5: Verificação

- Análise dinâmica do sistema por líderes de segurança
- Testes malicioso que propositalmente tentam causar falhas fornecendo informações aleatórias ou problemáticas
- Revisitar a análise de risco (etapa 2) para garantir que está aderente ao que foi implementado

Etapa 6: Resposta

- Estabelecer um plano de resposta a incidentes, identificando pessoas e canais de comunicação
- Revisão das etapas anteriores

Etapa 7: Publicação

- Executar resposta a incidente previamente planejada quando algum incidente acontecer

frederico.schardong@rolante.ifrs.edu.br