

**Curso de Análise e Desenvolvimento de Sistemas**

**Disciplina de Programação para Web**

**Segurança na WEB**

Júlio Augusto Ribeiro de Sá Almeida – 0030481911022

Sorocaba

junho – 2020

**Introdução**

A Segurança na WEB é uma requisito não-funcional muito importante para qualquer aplicação que utilize a internet como parte de sua plataforma. Os usuários precisam saber que seus dados estão seguros. Existem legislações que garantem o direito do usuário de ter segurança. E a fidelização do cliente é afetada diretamente, parece que não, mas assim que há um vazamento de informações a empresa que foi alvo perde muita credibilidade.

**1. As 10 principais vulnerabilidades**

Existem vulnerabilidades nos sistemas que podem ser exploradas por usuários mal-intencionados para obter acesso não-permitido, realizar ataques de phishing, roubar informações, e até mesmo danificar o sistema.

Existe uma lista de vulnerabilidades conhecidas publicamente chamada CVE (Common Vulnerabilities and Exposures), que sempre está publicando as novas descobertas com um número de identificação, uma descrição e pelo menos uma referência pública. Seu site é o <https://cve.mitre.org/> . Ela é uma lista bem extensa. Neste trabalho serão mencionadas apenas algumas vulnerabilidades mais comuns.

As principais vulnerabilidades de segurança em aplicações web são também as mais comuns, e exatamente por isso, são as mais perigosas. Antes de tentar métodos mais complexos, um invasor certamente tentará hackear sua aplicação com estratégias mais simples. Só depois de testar as vulnerabilidades mais comuns e falhar o hacker muda para uma abordagem mais complexa, ou desiste de invadir seu sistema.

O OWASP, ou Projeto Aberto de Segurança em Aplicações Web, é uma comunidade online que cria e disponibiliza de forma gratuita artigos, metodologias, documentação, ferramentas e tecnologias no campo da segurança de aplicações web. Ele lista as 10 vulnerabilidades mais comuns nas aplicações web a cada certo tempo, a última edição foi a de 2017. A seguir estão as que apareceram nessa edição, junto de uma explicação sucinta sobre cada uma.

**1.1. Injection**

Esse tipo de falha é muito conhecida pela técnica [SQL Injection](https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_de_SQL), onde o invasor realiza consultas indevidas no banco de dados, por meio de um formulário ou URL.

Com isso, o hacker consegue logar em seu sistema com privilégios de administrador, e explorar este acesso como desejar.

**1.2. Broken Authentication**

Essa falha ocorre quando o sistema não utiliza métodos confiáveis de autenticação e armazenamento das informações do usuário, com credenciais de acesso que facilitam ataques de força bruta.

Um dos exemplos mais comuns de broken authentication é utilizar senhas muito curtas que facilitam ataques de força bruta. Normalmente, é a primeira forma que um cracker tentará utilizar para invadir seu sistema.

A solução mais imediata para este problema é criar uma política de senhas seguras, com mínimo de caracteres, combinação de letras e números, etc.

**1.3. Sensitive Data Exposure**

As ferramentas de inspecionar elementos em uma página ou aplicação web, como o [Chrome DevTools](https://developers.google.com/web/tools/chrome-devtools/?hl=pt-br), estão entre as ferramentas preferidas dos desenvolvedores, e também dos hackers mal-intencionados.

Muitas APIs não protegem devidamente as informações que são transmitidas através delas, o que permite que invasores tirem proveito dos parâmetros retornados por ela, simplesmente inspecionando a aba ‘network’ do seu sistema.

Quanto a isso, é necessário dedicar uma atenção especial ao definir como estas informações vão trafegar pela interface, e criptografá-las devidamente.

**1.4. XML External Entities (XXE)**

Esta vulnerabilidade é específica de sistemas que trabalham com a linguagem XML, como por exemplo, software emissores de documentos fiscais eletrônicos.

Invasores podem explorar processadores de XML vulneráveis, onde é possível injetar códigos maliciosos e enviar para uma aplicação que fará a leitura desse XML.

**1.5. Broken Access Control**

Essa falha ocorre em aplicações que possuem páginas, rotas com informações que apenas o administrador deveria ter acesso, porém o controle de usuários não funciona.

Neste formato, o hacker consegue autenticar-se como administrador, e a partir daí, acessar, copiar e até excluir seus dados.

Para prevenir problemas de broken access control, é necessário que a validação do perfil de acesso venha do seu servidor, e não possa ser alterada externamente.

**1.6. Security Misconfiguration**

Se trata de configurações padrão de fábrica, informações abertas na nuvem, headers http configurados incorretamente, erros verbosos que indicam com precisão o motivo do erro.

A solução desse problema é bem simples: sempre lembrar-se de excluir os arquivos e configurações default dos frameworks que utilizar. Alguns frameworks vão te lembrar disso após a instalação. Outros, não.

**1.7. Cross-Site Scripting (XSS)**

Assim como SQL Injection, essa falha é muito conhecida na internet. Através dela, invasores conseguem inserir códigos maliciosos de javascript e tags html na página.

Normalmente, o cross-site scripting requer um determinado comportamento do usuário, como acessar um link específico. No entanto, quando bem executada, este tipo de fraude é muito difícil de identificar, principalmente para usuários leigos.

**1.8. Insecure Deserialization**

Falha muito utilizada para execução remota de códigos maliciosos. Mesmo quando essa falha de segurança não resulta em um acesso remoto por parte do hacker, ele podem utilizar a brecha para uma escalação de privilégios, configurando-se como administrador.

**1.9. Using Components with Known Vulnerabilities**

Mesmo que sua aplicação esteja desenvolvida seguindo um padrão seguro, as bibliotecas de terceiros que sua aplicação utiliza podem possuir falhas que são conhecidas na internet. Isso resulta em exploits, softwares prontos para serem utilizados para uma invasão.

Assim, é necessário ficar atento a possíveis vulnerabilidades de todos os componentes, bibliotecas e APIs que você irá consumir em seu software.

**1.10. Insufficient Logging & Monitoring**

Um sistema sem logs, não consegue detectar que em algum lugar de sua API/Sistema alguém está tentando explorar alguma falha.

Muitos desenvolvedores são resistentes a implementar um módulo de logs no seu sistema, principalmente devido ao peso destes dados no banco. Se este for o problema, defina um período de auto-exclusão do log que você pode suportar. Um mês é o mínimo!

Outra solução é o monitoramento em tempo real, que permite a você e sua equipe perceber comportamentos estranhos no seu sistema, como picos ou quedas absurdas de acesso.

Assim, é importante manter painéis de informações ao vivo da sua aplicação, normalmente em monitores ou televisões sempre ligados nesta tela, com alertas sonoros caso julgue necessário.

**2. Dicas**

Existem dicas de segurança que podemos seguir para diminuir o risco que corremos. Do lado dos profissionais é necessário cuidar as configurações de rede e de servidores, garantir que as senhas sejam fortes, gerenciar os usuários e suas permissões, não deixar dados sensíveis expostos a quem não deve poder acessá-los, dentre outras.

Os usuários comuns que acessam a internet no seu dia a dia também podem seguir algumas dicas, como manter em seu computador um bom antivírus, sempre atualizado; Deixar o firewall do Windows sempre ativado; Fazer sempre as atualizações necessárias do seu Sistema Operacional; Nunca instalar programas piratas em seu computador, eles podem trazer vírus ou outros programas perigosos; Não abrir pen-drives ou CDs/DVDs de outras pessoas sem antes passar o antivírus; Seguir sempre as orientações de seu banco para acessar sua conta pela Internet; Não digitar senhas e dados pessoais em computadores públicos (lan houses, empresas, etc); Não criar senhas com datas de aniversários, sequências numéricas fáceis ou nome de pessoas. Uma sugestão é criar senhas intercalando letras, números e caracteres especiais (#, por exemplo); Ter cuidado ao utilizar o cartão de crédito em compras online. Ter certeza de que a loja virtual é segura; Não clicar em links mostrados por e-mails desconhecidos. Eles costumam instalar vírus ou cavalos-de-tróia (programas que roubam dados do computador); Não divulgar dados pessoais (endereços, números de documentos, etc.) em redes sociais (Facebook, Twitter, etc.).

**Conclusão**

Ao conhecer as principais vulnerabilidades é possível se proteger contra elas. Existe uma grande comunidade empenhada em descobrir e divulgar essas vulnerabilidades, para que logo possam ser consertadas. É importante sempre manter os programas que utilizamos atualizados para ter as últimas atualizações de seguranças e não ser vítima da exploração de uma nova falha descoberta. Temos que seguir certos cuidados tanto como profissionais que desenvolvem ou operam e mantêm sistemas quanto como usuários comuns, para não ser prejudicados por terceiros mal-intencionados.

**Referências**

The Mitre Corporation; Lista de Vulnerabilidades e exposições comuns (CVE – Common Vulnerabilities and Exporures); <https://cve.mitre.org/>

OWASP; OWASP Top 10 - 2017 The Ten Most Critical Web Application Security Risks (versão portuguesa); <https://wiki.owasp.org/images/0/06/OWASP_Top_10-2017-pt_pt.pdf>

# Rogério, Fábio; SEGURANÇA EM APLICAÇÕES WEB: AS 10 PRINCIPAIS VULNERABILIDADES; <https://blog.tecnospeed.com.br/seguranca-em-aplicacoes-web/>

# Dalamura, Marcos; Dicas de segurança na WEB; <https://www.acessa.com/tecnologia/arquivo/artigo/2019/09/18-dicas-seguranca-web/>