# Tópicos Especiais em Segurança da Informação

## TP9 - Estudo sobre vulnerabilidades de segurança

Arthur do Prado Labaki

12-07, 2022

GBC 235

# Informações adicionais

Nem todos os exercícios estão com imagem aqui no relatório, mas todas as imagens integradas nesse relatório, quanto códigos, planilhas ou gifs de demonstração estão em meu repositório no GitHub abaixo.
**Link do meu GitHub**

# Resolução do item 1)

O conceito de vulnerabilidades pode ser vista como qualquer fator que possa contribuir para gerar invasões, roubos de dados ou acessos não autorizados a recursos, ou melhor, pontos da infraestrutura de TI que tornam mais fácil o acesso não autorizado aos recursos do negócio.

Eles são a interseção de três elementos: uma suscetibilidade ou falha do sistema, acesso do atacante à falha e a capacidade do atacante de explorar a falha, podendo ser softwares mal configurados, aparelhos com sistemas desatualizados e arquivos internos expostos publicamente.

# Resolução do item 2)

O OWASP (*Open Web Application Security Project*), ou Projeto Aberto de Segurança em Aplicações Web, é uma entidade sem fins lucrativos e com reconhecimento internacional, atuando com foco na colaboração para o fortalecimento da segurança de softwares em todo o mundo, fornecendo de forma gratuita artigos, metodologias, documentação, ferramentas e tecnologias no campo da segurança de aplicações web. O grupo que sustenta o projeto é composto por diversos especialistas em segurança na web espalhados por todo o mundo, em que eles compartilham seus conhecimentos e experiências sobre as vulnerabilidades, ameaças, ataques e contramedidas existentes.

A ideia do OWASP é reunir as informações mais importantes que permitam a avaliação

dos riscos de segurança e as formas de combatê-las eficientemente. O projeto é importante para a comunidade pois ele fornece, de forma gratuita e aberta, uma riqueza absoluta de conhecimentos e ferramentas para ajudar qualquer pessoa envolvida nos processos de criação, desenvolvimento, testes, implementação e suporte de uma aplicação web para garantir que a segurança seja constituída desde o início e que o produto final seja tão seguro quanto possível.

# Resolução do item 3)

O OWASP categoriza seus grandes projetos em três níveis deferentes, chamados de *Flagship Projects*, *Lab Projects* e *Incubator Projects*.

*Incubator Projects* ou Projetos da Incubadora representam o playground experimental onde os projetos ainda estão sendo desenvolvidos, as ideias ainda estão sendo comprovadas e o desenvolvimento ainda está em andamento. Evoluindo um pouco, o proximo nível é o *Lab Projects*, ou Projetos de laboratórios, que são os projetos que demonstram algum valor e já foram revisadas pelo OWASP.

Por fim, os *Flagship Projects* são os projetos principais do OWASP, já sendo consolidados e que demonstraram valor estratégico para o OWASP e a segurança de aplicativos como um todo. Atualmente o OWASP mantém 18 grandes projetos nesse nível, sendo eles:

- **Amass -** Esse projeto é uma ferramenta para ajudar os profissionais de segurança da informação a realizar o mapeamento de redes de superfícies de ataque e realizar a descoberta de ativos externos usando coleta de informações de código aberto e técnicas de reconhecimento ativo;
- **Application Security Verification Standard -** Ele é constituído por uma base para testar os controles técnicos de segurança de aplicativos da Web, além de fornecer aos desenvolvedores uma lista de requisitos para desenvolvimento seguro;
- **Cheat Sheet Series -** É um conjunto de dicas para fornecer um conjunto de guias simples de boas práticas para desenvolvedores e defensores de aplicativos seguirem;

- **CSRFGuard -** É uma biblioteca que implementa uma variante do padrão de token do sincronizador para mitigar o risco de ataques *Cross-Site Request Forgery* (CSRF), sendo uma das ferramentas de segurança gratuitas mais populares do mundo, sendo mantida ativamente por um grupo de voluntários internacionais;

- **CycloneDX -** Esse projeto é um padrão leve de lista de materiais (*Bill of Materials* - BOM) projetado para uso em contextos de segurança de aplicativos e análise de componentes da cadeia de suprimentos;

- **Defectdojo -** É uma ferramenta de gerenciamento de vulnerabilidades de código aberto que simplifica o processo de teste oferecendo modelos, geração de relatórios, métricas e ferramentas de autoatendimento de linha de base;

- **Dependency-Check -** É uma ferramenta de Análise de Composição de Software (*Software Composition Analysis - SCA*) que tenta detectar vulnerabilidades divulgadas publicamente contidas nas dependências de um projeto;

- **Dependency-Track -** Esse projeto é uma plataforma inteligente de Análise de Componentes que permite às organizações identificar e reduzir riscos na cadeia de fornecimento de software, aproveitando recursos do Software Bill of Materials (SBOM);

- **Juice Shop -** O Juice Shop é provavelmente o aplicativo web inseguro mais moderno e sofisticado (de acordo dom o site do OWASP). Ele pode ser usado em treinamentos de segurança, demonstrações de conscientização, CTFs e como cobaia para ferramentas de segurança. Vale lembrar que ele foi a aplicação web usada na competição em sala do TP8;

- **Mobile Security Testing Guide -** Esse projeto fornece um padrão de segurança para aplicativos móveis (OWASP MASVS) e um guia de teste abrangente (OWASP MSTG) que abrange os processos, técnicas e ferramentas usadas durante um teste de segurança de aplicativo móvel, bem como um conjunto exaustivo de casos de teste que permite que os testadores forneçam resultados consistentes e completos;

- **ModSecurity Core Rule Set -** É um conjunto de regras genéricas de detecção de ataque para uso com o *ModSecurity* ou *firewalls* de aplicativos da web compatíveis, visando proteger aplicativos da web de uma ampla gama de ataques;

- **OWTF -** É um projeto que visa tornar as avaliações de segurança o mais eficientes possível, automatizando a parte manual e não criativa do *pentesting*. Ele fornece suporte pronto para uso para o *OWASP Testing Guide*, os padrões NIST e PTES;

- **SAMM -** O SAMM tem como missão fornecer uma maneira eficaz e mensurável para o usuário analisar e melhorar seu ciclo de vida de desenvolvimento seguro. O SAMM suporta todo o ciclo de vida do software e é independente de tecnologia e processo;

- **Security Knowledge Framework -** É um aplicativo da Web de código aberto que explica os princípios de codificação segura em várias linguagens de programação, ajudando o usuário a aprender e integrar a segurança desde o projeto em seu desenvolvimento de software e criar aplicativos que sejam seguros desde o projeto;

- **Security Shepherd -** O projeto é uma plataforma de treinamento de segurança de aplicativos móveis e da Web. O objetivo dele é levar iniciantes em *AppSec* ou engenheiros experientes e aprimorar seu conjunto de habilidades de teste de penetração para o status de especialista em segurança;

- **Top Ten -** É um documento de conscientização padrão para desenvolvedores e segurança de aplicativos da web. Ele representa um amplo consenso sobre os riscos de segurança mais críticos para aplicativos da web. Ele será mais explicado no próximo exercício;

- **Web Security Testing Guide -** É um guia abrangente para testar a segurança de aplicativos e serviços da Web, fornecendo uma estrutura de práticas recomendadas usadas por testadores de penetração e organizações em todo o mundo;

- **ZAP -** Esse projeto é um *scanner* de segurança de aplicativos da web de código aberto. Ele deve ser usado tanto por aqueles que são novos em segurança de aplicativos quanto por testadores de penetração profissionais.

Ainda, analisando uma parte diferente do site do OWASP, alguns outros projetos pertencem a esse nível de projetos, mas não está escrito no site principal (está no *dev.owasp.org/projects* mas não está no *owasp.org/projects*), eles são:

- **AppSensor -** É um projeto que define uma estrutura conceitual e uma metodologia que oferece orientação prescritiva para implementar a detecção de intrusão da camada de aplicativo e a resposta automatizada. Além disso, há uma implementação de referência que fornece um kit de ferramentas para criar aplicativos de autodefesa por meio de detecção de eventos em tempo real;

- **Cloud-Native Application Security Top 10 -** É um guia que fornece assistência e educação para organizações que desejam adotar aplicativos *Cloud-Native* com segurança. O guia fornece informações sobre quais são os riscos de segurança mais importantes para aplicativos nativos da nuvem, os desafios envolvidos e como superá-los;
- **Top 10 Low-Code/No-Code Security Risks -** Com o objetivo semelhante no anterior, esse projeto é um guia que é fornece assistência e educação para organizações que desejam adotar e desenvolver aplicativos *Low-Code/No-Code*, da mesma forma que o *Cloud-Native Application Security Top 10*.

# Resolução do item 4)

Como já explicado no exercício passado, o projeto *Top Ten* é um documento de conscientização padrão para desenvolvedores e segurança de aplicativos da web. Ele representa um amplo consenso sobre os riscos de segurança mais críticos para aplicativos da web. Nele está listado, por ordem de prioridade, os principais riscos de segurança de aplicativos da Web, e ele foi reajustado em 2021, que são, da maior para a menor:

1. **Controle de acesso quebrado -** O controle de acesso impõe a política de forma que os usuários não possam agir fora de suas permissões pretendidas. As falhas geralmente levam à divulgação, modificação ou destruição de informações não autorizadas de todos os dados ou à execução de uma função comercial fora dos limites do usuário;
2. **Falhas criptográficas -** A primeira coisa é determinar as necessidades de proteção dos dados em trânsito e em repouso. Por exemplo, senhas, números de cartão de crédito, registros de saúde, informações pessoais e segredos comerciais exigem proteção extra, principalmente se esses dados estiverem sob leis de privacidade ou regulamentos;
3. **Injeção -** Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo, consultas dinâmicas ou chamadas não parametrizadas sem escape sensível ao contexto são usadas diretamente no interpretador, dados hostis são usados em parâmetros de pesquisa de mapeamento relacional de objeto para extrair registros confidenciais adicionais ou usados diretamente ou concatenados são caracterizados ataques de Injeção;
4. **Design inseguro -** O design inseguro é uma categoria ampla que representa diferentes pontos fracos, expressos como "design de controle ausente ou ineficaz";

5. **Configuração incorreta de segurança -** Falta de proteção de segurança apropriada em qualquer parte da pilha de aplicativos ou permissões configuradas incorretamente em serviços de nuvem, recursos desnecessários são ativados ou instalados, software está desatualizado ou vulnerável entre outras são características de configuração incorreta de segurança;

6. **Componentes vulneráveis e desatualizados -** Se o usuário não conhece as versões de todos os componentes que usa, se o software for vulnerável, sem suporte ou desatualizado, se os desenvolvedores de software não testarem a compatibilidade de bibliotecas atualizadas, atualizadas ou corrigidas entre outras são características desse risco;

7. **Falhas de identificação e autenticação -** A confirmação da identidade do usuário, autenticação e gerenciamento de sessão é fundamental para proteger contra ataques relacionados à autenticação;

8. **Falhas de integridade de software e dados -** As falhas de integridade de software e dados estão relacionadas a código e infraestrutura que não protegem contra violações de integridade. Um exemplo disso é quando um aplicativo depende de *plugins*, bibliotecas ou módulos de fontes não confiáveis, repositórios e redes de entrega de conteúdo;

9. **Falhas de registro e monitoramento de segurança -** Esta categoria é para ajudar a detectar, escalar e responder a violações ativas. Sem registro e monitoramento, as violações não podem ser detectadas;

10. **Falsificação de solicitação do lado do servidor -** Essas falhas ocorrem sempre que um aplicativo da Web está buscando um recurso remoto sem validar a URL fornecida pelo usuário. Ele permite que um invasor force o aplicativo a enviar uma solicitação criada para um destino inesperado, mesmo quando protegido por um *firewall*, *VPN* ou outro tipo de lista de controle de acesso à rede.

# Resolução do item 5)

A sigla CVE significa Exposição a Vulnerabilidades Comuns (*Common Vulnerabilities Exposure*) e é um dicionário de vulnerabilidades e exposições de segurança conhecidas publicamente. Ele tem como benefício o compartilhamento de dados de vulnerabilidade em diferentes bancos de dados e ferramentas, permitindo que essas ferramentas de segurança podem "conversar" entre si usando uma linguagem comum, fornecendo uma linha de base

para avaliar a cobertura das ferramentas de segurança de uma organização.

Já a CWE que é a Enumeração de Fraquezas Comuns (*Common Weakness Enumeration*) é também um dicionário desenvolvido pela comunidade de tipos de fraquezas de software. Seu beneficio é oferecer uma "régua"de medição padrão para segurança de software, gerando uma linha de base comum para os esforços de identificação, mitigação e prevenção de pontos fracos.

Por fim, o CVSS ou Sistema de Pontuação de Vulnerabilidades Comuns (*Common Vulnerability Scoring System*) é um padrão aberto da indústria, independente de fornecedor, projetado para transmitir a gravidade da vulnerabilidade. Ele ajuda a determinar a urgência e a prioridade de resposta quando as vulnerabilidades são detectadas, resolvendo o problema de vários sistemas de pontuação incompatíveis.

# Resolução do item 6)

Esse exercício já foi realizado no item 4, portanto as 5 maiores vulnerabilidades serão mais aprofundadas.

## Controle de acesso quebrado

Controle de acesso quebrado ou *Broken Access Control* são projetados para impedir que os usuários atuem fora de suas permissões pretendidas. Os usuários podem realizar ações além do escopo de suas permissões autorizadas se houver vulnerabilidades nesses controles ou se elas não existirem. Isso pode permitir que invasores roubem informações de outros usuários, modifiquem dados e executem ações como outros usuários.

Dois nomes comuns para dividir as vulnerabilidades de controle de acesso em categorias são escalonamento horizontal de privilégios (um usuário pode realizar uma ação ou acessar dados de outro usuário com o mesmo nível de permissões) e escalonamento vertical de privilégios (um usuário pode executar uma ação ou acessar dados que requerem um nível de acesso além

de sua função).

Os controles de acesso quebrados podem colocar os aplicativos em risco de violação de dados, geralmente resultando na perda de confidencialidade e integridade. Um adversário pode roubar informações acessadas pelos usuários do aplicativo, manipular dados executando ações que várias funções de usuário podem realizar dentro do aplicativo e, em determinadas circunstâncias, comprometer o servidor web. A manipulação de dados pode permitir o sequestro de contas, roubo se o aplicativo lidar com moeda ou bens tangíveis e controle de sistemas/serviços monitorados pelo aplicativo. Outros ataques contra o servidor web e a infraestrutura podem ser possíveis, dada a natureza do aplicativo.

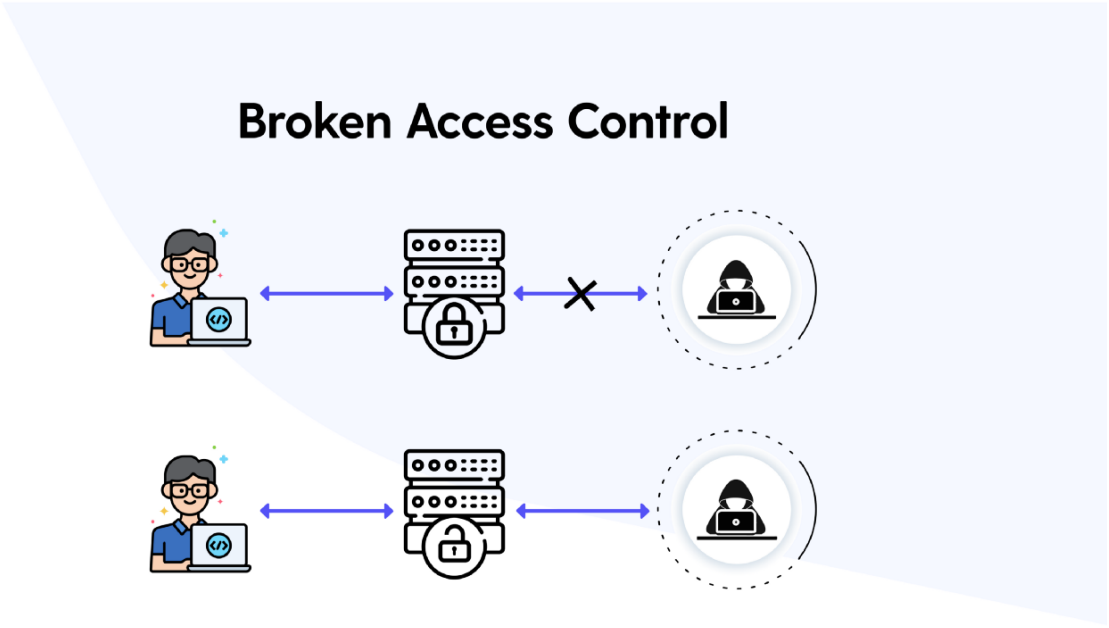


*Figura 1: Vulnerabilidades geradas pelo controle de acesso*

## Falha criptográfica

Falha criptográfica ou *Cryptographic Failures* é uma falha de segurança que ocorre quando uma entidade de terceiros (aplicativos, páginas da Web, sites diferentes) expõe dados confidenciais. Para ser exato, é quando essa entidade o faz sem uma intenção específica por trás disso. Seja negligência, incompetência ou lapso de julgamento, uma falha criptográfica pode ter consequências catastróficas, tanto pessoais quanto comerciais.

Originalmente chamado de exposição de dados confidenciais, uma falha criptográfica ocorre quando um sistema torna os dados confidenciais acessíveis a bisbilhoteiros potencialmente maliciosos. Também ocorre quando você tem um incidente de segurança que permite o apagamento acidental ou ilegal, destruição, alteração ou divulgação injustificada de informações confidenciais.

Em geral, as falhas criptográficas se enquadram em três categorias:

- **Quebra de confidencialidade:** É o que acontece quando um terceiro consegue acessar dados confidenciais ou quando uma organização divulga esses dados por acidente;
- **Violação de integridade:** Este descreve uma situação em que dados sensíveis são alterados, mais uma vez, sem autorização e/ou intenção por trás disso.
- **Violação de disponibilidade:** O que pertence a essa categoria são os cenários em que dados confidenciais são destruídos ou quando você perde o acesso a eles. A categoria abrange a perda de dados permanente e temporária.

## Injeção

Injeção ou *Injection* é um ataque em que o invasor simplesmente envia dados maliciosos, a fim de fazer com que o aplicativo os processe e faça algo que não deveria fazer. As falhas de injeção são muito prevalentes, principalmente no código legado. O motivo principal é: os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

Alguns outros motivos potenciais incluem o uso de consultas dinâmicas diretamente no interpretador, chamadas não parametrizadas sem implementar o escape com reconhecimento de contexto diretamente no interpretador, dados hostis sendo usados em parâmetros de pesquisa de mapeamento relacional de objeto para extrair registros confidenciais adicionais ou ainda dados hostis sendo usados ou concatenados diretamente, de modo que o SQL ou comando contenha tanto estrutura quanto dados hostis em consultas dinâmicas, comandos ou

procedimentos armazenados.


*Figura 2: Exemplo de ataques utilizando injeção*

## Design inseguro

Design inseguro ou também *Insecure Design*, significa os riscos relacionados a falhas de design e arquitetura que são incorporadas desde o início do desenvolvimento de software, se as mitigações de segurança apropriadas não forem tomadas.

Um design seguro ainda pode levar a defeitos se for implementado incorretamente, resultando em vulnerabilidades que podem ser exploradas. Da mesma forma, o design inseguro não pode ser corrigido por uma implementação perfeita, pois, por definição, os controles de segurança necessários nunca foram criados para se defender contra ataques específicos. É por isso que é tão importante que o design seja tratado como sua própria categoria - alguns pontos fracos só podem ser identificados antes do início da implementação.

O design seguro é tanto sobre cultura quanto sobre metodologia. Trata-se de mudar a mentalidade em torno de qual estágio a segurança precisa para entrar no processo de desenvolvimento de aplicativos e é a crença fundamental que o verdadeiro desenvolvedor web só pode ser alcançado se a segurança for considerada desde o início.

### Configuração incorreta de segurança

Por fim temos a Configuração incorreta de segurança, ou *Security Misconfiguration* que é uma série de erros como cabeçalhos HTTP mal configurados, cabeçalhos desnecessários, mensagens de erro verbosas, bancos de dados abertos à internet entre inúmeros outros exemplos. Atacantes exploram essa vulnerabilidade buscando conhecer mais profundamente a API em busca de informações que direcionam melhor seu ataque e brechas que sirvam de porta de entrada.

Esse problema pode ser particularmente delicado em operações grandes e complexas. Por mais que haja validações manuais e automáticas para identificá-lo em diferentes instâncias, esses processos abrangem apenas erros de configuração conhecidos e divulgados. A quantidade de partes móveis também aumenta a probabilidade de alguma área estar sujeita a erros. Por fim, as falhas de configuração podem estar escondidas nas interações entre essas partes móveis, ocultando o problema.