TECNOLOGIA DA INFORMAÇÃO

OWASP Top 10

Livro Eletrônico

GRAN CURSOS ONLINE

SUMÁRIO

APRESENTAÇÃO

Olá, querido (a) amigo(a), tudo bem?

Na aula de hoje são destacadas **melhores práticas de codificação segura com foco em OWASP TOP 10**.

E lembre-se de que perseguir, sem cessar, uma meta: esse é o caminho para o sucesso! Quem pensa pequeno alcança pequeno!

Em caso de dúvidas, acesse o fórum do curso ou entre em contato.

Que Deus o(a) abençoe e sucesso nos estudos!

OWASP TOP 10

O QUE É OWASP?

Fundada em 1º de dezembro de 2001, **OWASP** (***Open Web Application Security Project*** - **Projeto Aberto de Segurança em Aplicações Web**) é um organismo internacional que atua na Segurança de Software (OWASP, 2021).

Trata-se de uma **entidade sem fins lucrativos**, que produz uma variedade de materiais de forma **colaborativa**, **transparente** e **aberta**, para empresas e profissionais da área manterem e implementarem aplicações confiáveis, além de fornecer um dos principais projetos de pesquisa utilizados para testes de intrusão e desenvolvimento seguro, o **OWASP TOP 10** (OWASP, 2017).

O QUE É OWASP TOP 10?

De acordo com a própria OWASP (OWASP, 2021), o **projeto Top 10** é um documento padrão de conscientização de segurança de aplicativos da web para desenvolvedores. Ele representa um consenso sobre os **riscos de segurança da informação mais críticos para aplicativos web**.

Obs.: **Risco** é a **combinação de fatores que ameaçam o sucesso do negócio**. Isto pode ser descrito conceitualmente da seguinte forma: um **agente** de ameaça interage com um sistema, no qual pode haver uma **vulnerabilidade** latente que quando explorada causa um **impacto**.
Vamos contextualizar de outra forma: um ladrão de carros (**agente** de ameaça) passa por um estacionamento verificando nos carros presentes (o **sistema**) a existência de portas destrancadas (a **vulnerabilidade**) e quando a encontra, abre a porta (a **exploração**) e leva para si o que está dentro do carro (o **impacto**). Todos esses fatores desempenham um papel no desenvolvimento de **software seguro** (OWASP, 2012).

É considerado pelos desenvolvedores como o **primeiro passo para uma codificação mais segura**. Em seu lançamento de 2021 (OWASP, 2021), a fundação busca estimular as empresas para que adotem este documento, a fim de garantir a minimização dos riscos abordados.

Obs.: Em **2021** houve a atualização das categorias da **OWASP TOP 10**, com relação à versão de **2017.**

OWASP busca mitigar **vulnerabilidades de segurança na web**, e, por isso, a iniciativa se organiza em diferentes capítulos locais para elaborar ações de conscientização.

OWASP TOP 10 2021

O OWASP mantém **uma lista com as 10 falhas de segurança de aplicativos da Web mais perigosas**, juntamente com os métodos mais eficazes para lidar com elas.

A seguir, destacamos o **top 10 OWASP (**conforme **análise de 2021)** por ordem de **maior risco para o menor.**

Figura. Comparativo OWASP TOP 10 do ano 2017 a 2021
Fonte: OWASP (2021)

Ao analisar a figura anterior, é possível perceber que todas as categorias foram mantidas de 2017 para 2021, porém com transformações em suas classificações (ORTEGA E CÂMARA, 2021).

Houve a consolidação de mais de uma categoria da última edição, sendo agrupada em apenas uma. Houve também a inclusão de três novas categorias (ORTEGA E CÂMARA, 2021), que são:

- **A04:2021-*Insecure Design***, focada nos riscos relacionados a falhas de design e arquitetura de aplicações;
- **A08:202- *Software and Data Integrity Failures*** com o intuito de realizar suposições relacionadas a atualizações de software; e
- **A10:2021- *Server-Side Request Forgery* (SSRF)**, que foi a primeira das duas categorias adicionadas pelo processo de pesquisa da comunidade da OWASP (OWASP, 2021).

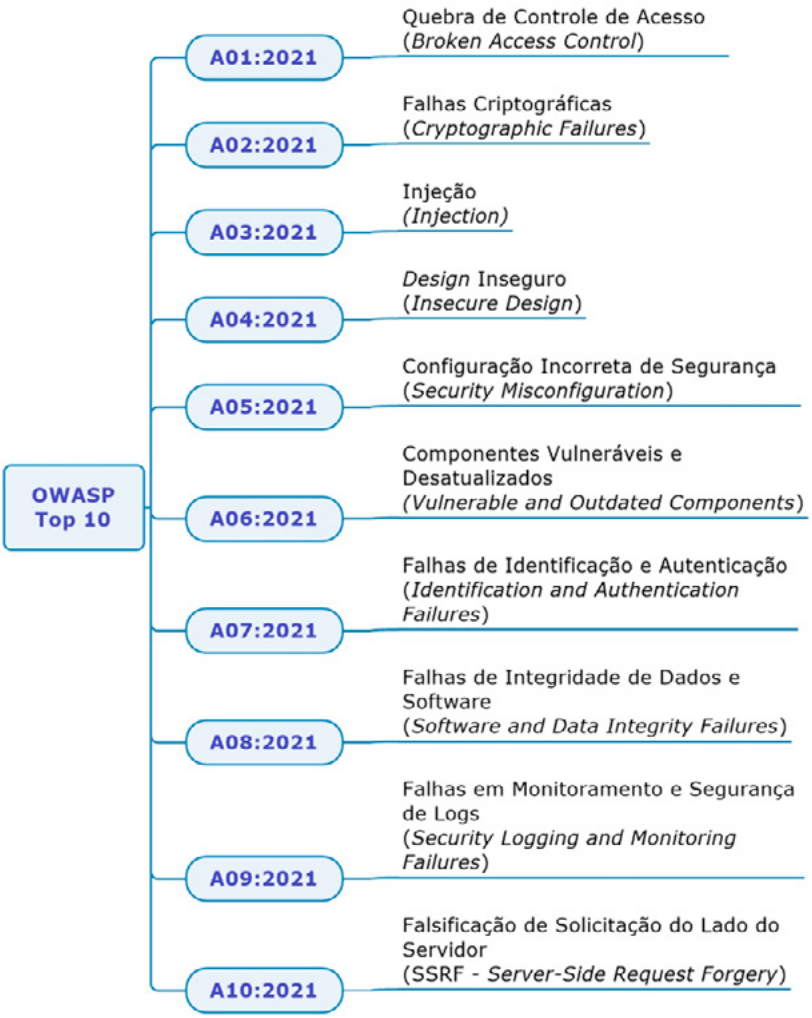

Figura. OWASP Top 10. Fonte: (QUINTÃO, 2022) e OWASP (2021)

Direto do Concurso

001. (CEBRASPE (CESPE)/ANALISTA DE CONTROLE EXTERNO (TCE-RJ)/ ORGANIZACIONAL/TECNOLOGIA DA INFORMAÇÃO/2022) Julgue o item a seguir, quanto às ferramentas e técnicas de exploração de vulnerabilidades em aplicativos web, à Open Web Application Security Project (OWASP) e às ameaças e vulnerabilidades em aplicações web.
Classificação de Risco para o Top 10 é uma metodologia baseada na OWASP Risk Rating Methodology e consiste em estimar, para cada categoria do Top 10, o risco peculiar que cada falha introduz em uma aplicação web típica e, posteriormente, ordenar o Top 10 de acordo com as falhas que tipicamente introduzem o risco mais significativo para uma aplicação.

OWASP mantém **uma lista com as 10 falhas de segurança de aplicativos da Web mais perigosas**, juntamente com os métodos mais eficazes para lidar com elas.
Certo.

Vamos então ao **top 10 OWASP,** conforme reportado em 2021:

A01:2021-Broken Access Control (Quebra de Controle de Acesso)

Subiu da quinta posição (em 2017) para a primeira (em 2021). 94% dos aplicativos que foram testados na pesquisa sofreram alguma **violação** no controle de acesso (GODOY, 2021).

Ortega e Câmara (2021) destacam que o controle de acesso impõe a política de modo que os usuários **não** possam agir fora de suas permissões pretendidas.

Muitas organizações já foram vítimas deste ataque, devido à falta de implementação de um mecanismo de controle de acesso adequado, levando a uma autenticação fraca, falta de controle de acesso de nível funcional e gerenciamento de sessão (SIEMBA, 2021).

A quebra do controle de acesso, normalmente leva à **divulgação, modificação ou destruição de dados, informações ou ao desempenho de uma funcionalidade comercial do sistema**, **que se encontra fora dos limites do usuário** (ORTEGA E CÂMARA, 2021).

Algumas **vulnerabilidades** **comuns de controle de acesso OWASP** (2021):

- violação dos princípios do mínimo privilégio e acessos anônimos;
- omissão de verificações de controle de acesso, a partir de modificações diretas na URL (como adulteração de parâmetros), no estado interno da aplicação, na página HTML ou utilizando uma ferramenta de ataque que modifica as solicitações que vão para a API;
- permissão da visualização ou edição da conta de outro usuário, fornecendo seu identificador exclusivo;
- acesso à API sem controles de acesso de métodos de solicitação HTTP POST, PUT e DELETE;
- falhas na implementação *JSON Web Token* (JWT) - um padrão aberto (RFC 7519) que define uma maneira compacta e independente para transmitir informações com segurança entre partes- e *cookies* com informações que podem ser manipuladas.

Como **forma de prevenção**, a fundação OWASP (2021) sugere que **os tratamentos de controles de acessos devem ser aplicados do lado do servidor confiável**, de forma que o invasor não consiga realizar modificações na verificação de controle de acesso.

- OWASP destaca os seguintes pontos como forma de mitigação:
- Exceto para recursos públicos, negar por padrão.
- Implementação e reutilização de mecanismos de controle de acesso em todo o aplicativo, incluindo a minimização do uso de *Cross-Origin Resource Sharing* (**CORS**).

Obs.: Segundo Mozilla (2021), *Cross-Origin Resource Sharing* (**CORS**) é um mecanismo baseado em cabeçalho HTTP que permite a um servidor indicar qualquer origem (domínio, esquema ou porta) diferente da sua própria, a partir da qual um navegador deve permitir o carregamento de recursos.

- Os controles de acesso ao modelo devem impor a propriedade do registro, em vez de aceitar que o usuário possa criar, ler, atualizar ou excluir qualquer registro.
- Os requisitos de limite de negócios de aplicativos exclusivos devem ser impostos por modelos de domínio.
- Desativar a lista de diretórios do servidor da web e certificar de que os metadados do arquivo - por exemplo o arquivo.git – arquivo do sistema GIT de controle de versão de arquivos do projeto - e os arquivos de *backup* não estejam presentes nas raízes da web (web roots).
- Registro de falhas de controle de acesso e notificação aos administradores da aplicação, quando apropriado (por exemplo, falhas repetidas).
- Limite de taxa ao acesso da API e do controlador, afim de minimizar os danos de um possível conjunto de ferramentas de ataque automatizado.
- Os identificadores de sessão com estado devem ser invalidados no servidor após o *logout*. Os tokens JWT sem estado devem ter vida curta para que a janela de oportunidade para um invasor seja minimizada. Para JWTs de longa duração, é altamente recomendável seguir os padrões OAUTH para revogar o acesso.

A02:2021 - Cryptographic Failures (Falhas Criptográficas)

Subiu da terceira posição (em 2017) para a segunda (em 2021). Anteriormente conhecida como Exposição de Dados Sensíveis (*Sensitive Data Exposure*), o qual era tratado como um sintoma, e não uma causa raiz.

A renovação está nas falhas relacionadas à criptografia (ou falta dela), que geralmente levam à exposição de dados confidenciais ou comprometimento do sistema (GODOY, 2021).

A **mitigação de falhas criptográficas** pode ser realizada de inúmeras maneiras, dependendo de seu escopo e da análise realizada sobre os dados do sistema (CIPHERSTASH (2021):

- Classificação dos dados processados, armazenados ou transmitidos por um aplicativo, com o objetivo de entender quais dados são confidenciais de acordo com as leis de privacidade, requisitos regulamentares ou necessidades de negócios.
- **Não armazene dados confidenciais desnecessariamente. Descarte-o o mais rápido possível ou use tokenização compatível com PCI DSS ou mesmo truncamento.** Os dados não retidos não podem ser roubados.
- Certifique-se de **criptografar todos os dados confidenciais armazenados**.
- Certifique-se de que algoritmos, protocolos e senhas de padrão forte e atualizados estejam em vigor; use o gerenciamento de senhas adequado.
- Criptografe todos os dados em trânsito com protocolos seguros, como TLS com cifras de sigilo de encaminhamento (FS), priorização de cifras pelo servidor e parâmetros seguros. Aplique a criptografia usando diretivas como HTTP Strict Transport Security (HSTS).
- Desative o armazenamento em cache para respostas que contenham dados confidenciais.

A03: 2021 - Injection (Injeção)

Caiu para a terceira posição em 2021. 94% dos aplicativos que foram testados na pesquisa sofreram alguma forma de **injeção**, tendo o segundo maior número de ocorrências em aplicativos (GODOY, 2011).

PINTO e col. (2019) dizem que a injeção SQL é um tipo ataque que ocorre quando usuários com intenções maliciosas, injetam um código malicioso a partir de formulários de entrada de dados, sem validação.

Cross-site Scripting (XSS) foi incluso na categoria de Injeção (*Injection*), o que tecnicamente, é uma injeção de *script* direcionada a outros usuários que acessaram o site afetado. Este é um ataque diferente dos métodos de injeções que acontecem do lado do servidor, como SQL, comandos do sistema operacional e injeções LDAP (HKCERT, 2021).

OWASP (2021) cita que uma aplicação é vulnerável a ataques de injeção quando:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo;
- Consultas dinâmicas ou chamadas não parametrizadas sem escape ciente do contexto são usadas diretamente no interpretador.
- Dados hostis são usados nos parâmetros de pesquisa de mapeamento relacional de objeto (ORM - Object Relational Mapping Tools) para extrair registros confidenciais adicionais.
- Dados hostis são usados diretamente ou concatenados. O SQL ou comando contém a estrutura e os dados maliciosos em consultas dinâmicas, comandos ou procedimentos armazenados.

OWASP (2021) destaca que para prevenir a injeção, deve-se manter os dados separados dos comandos e consultas. Formas de mitigar o problema:

- A utilização de uma API segura, evitando o uso do interpretador SQL inteiramente, fornecendo uma interface parametrizada com *Object Relational Mapping Tools* (ORMs).
- Use validação de entrada positiva ou *safelist* do lado do servidor. Esta não é uma defesa completa, pois muitos aplicativos requerem caracteres especiais, como áreas de texto ou APIs para aplicativos móveis.
- Poderá ser necessário usar escape para caracteres especiais usando a sintaxe para o interpretador.
- Uso do LIMIT e outros controles SQL em consultas para evitar a divulgação em massa de registros no caso de injeção de SQL. GALLAGHER (2020) diz que o LIMIT é uma clausula SQL, que restringe quantos registros serão retornadas a partir de uma consulta SQL.

Direto do Concurso

002. (CEBRASPE(CESPE)/ANALISTADECONTROLEEXTERNO(TCE-RJ)/ORGANIZACIONAL/TECNOLOGIA DA INFORMAÇÃO/2022) Julgue o item a seguir, quanto às ferramentas e técnicas de exploração de vulnerabilidades em aplicativos web, à Open Web Application Security Project (OWASP) e às ameaças e vulnerabilidades em aplicações web.

Caso uma aplicação permita que comandos SQL sejam digitados nos inputs de seus formulários e concatenados diretamente nos comandos SQL da própria aplicação, sem que seja realizada uma validação ou tratamento antecedente, certamente essa aplicação estará vulnerável ao ataque conhecido como SQL Injection.

PINTO e col. (2019) dizem que a injeção SQL é um tipo ataque que ocorre quando usuários com intenções maliciosas, injetam um código malicioso a partir de formulários de entrada de dados, sem validação. Conforme PINTO e col. (2019), pode-se citar alguns exemplos de entradas de dados de usuários, como formulários de cadastro como nome, data de nascimento, endereço, também formulários de login ou campos de pesquisa presentes em websites.

WALLARM (2017) complementa que estas entradas maliciosas vindas do usuário levam a execução remota do código, que pode ocasionar a perda de dados, ações não autorizadas, corrupção de dado, dentre outros.

Certo.

A04: 2021 - Insecure Design (Design Inseguro)

Categoria nova para 2021, com foco **nos riscos relacionados a falhas de *design*** e arquitetura de aplicações (GODOY, 2021). Busca alertar os desenvolvedores para que façam mais uso de modelagem de ameaças, padrões de *design* seguros e arquiteturas de referência, a fim de antecipar ataques e aplicar medidas preventivas para proteger adequadamente as aplicações Web.

Obs.: Um ***design* seguro** pode ter defeitos de implementação que levam a vulnerabilidades que podem ser exploradas. Já um ***design* inseguro** não pode ser corrigido nem mesmo por uma implementação perfeita, pois, por definição, na arquitetação base do projeto, os controles de segurança necessários **não** foram criados para a defesa contra ataques específicos (OWASP, 2021).

Vejamos algumas etapas que podem ser de grande valia na projeção de uma aplicação (IMMUNIWEB, 2021):

- implementação do **SDLC (*Software Development Life Cycle* - ciclo de vida de desenvolvimento seguro)** que busca produzir um software de alta qualidade que atenda às expectativas do cliente para ajudar a avaliar e projetar controles relacionados à segurança e à privacidade;
- implementação de uma biblioteca de padrões de *design* seguros para utilização na aplicação;
- **avaliação do processo de autenticação de aplicativos, controle de acesso, lógica de negócios** e a **utilização da modelagem de ameaças para identificação de possíveis vetores de ataque**;
- criação de testes de integração e verificações para validar se todos os fluxos críticos são resistentes ao modelo de ameaça projetado;
- atenção aos recursos computacionais e ao limite do consumo pelo usuário do serviço.

A05: 2021 - Security Misconfiguration (Configuração Incorreta de Segurança)

Obs.: Configurações incorretas de segurança ou *Security Misconfiguration* estão relacionadas a **controles de segurança que são configurados incorretamente ou de forma insegura**, **colocando sistemas e dados em risco**, como alterações de configuração mal documentadas ou a utilização de configurações padrão.

Essa categoria subiu da 6ª posição na edição anterior para a quinta (em 2021). Observe que 90% dos aplicativos que foram testados na pesquisa possuíam algum tipo de configuração incorreta (GODOY, 2021). Com mais mudanças em software altamente configurável, não é surpreendente ver essa categoria subir.

A antiga categoria de XML *External Entities* (XXE) agora faz parte desta categoria. De acordo com Vicente (2019), a essência dos riscos de injeção de XXE é uma configuração incorreta nos servidores que aceitam XML como entrada de aplicativos *clients* não confiáveis, na qual, invasores abusam desse aspecto para injetar cargas úteis de XMLs personalizados, para extrair dados, falsificar solicitações do lado do servidor (SSRF), ou conduzir tentativas de negação de serviço (DoS).

Dentre as formas de prevenção: siga boas práticas de *hardening (tenha* atenção especial com todos os aspectos das configurações dos ativos), avalie criteriosamente a necessidade de funcionalidades e serviços em execução, principalmente em máquinas virtuais, use critério semelhante ao mínimo privilégio e tenha em execução apenas os serviços que são essenciais para que o negócio funcione adequadamente. Além disso, procure configurar os serviços de forma a dificultar ao máximo o reconhecimento de tecnologias em uso pelo atacante (HAGI, 2021).

A06:2021 - Vulnerable and Outdated Components (Componentes Vulneráveis e Desatualizados)

Anteriormente intitulada como Uso de Componentes com vulnerabilidades conhecidas (*Using Components with Known Vulnerabilities)*, essa categoria passou da 9ª posição em 2017 para a 6ª posição em 2021 e é um problema conhecido em que muitos não avaliam os riscos que pode gerar (GODOY, 2021).

OWASP (2021) cita algumas **formas de mitigação** para este risco, como:

- remoção de dependências não utilizadas, recursos, componentes, arquivos e documentações desnecessárias;
- levantamento contínuo das versões dos componentes do lado do cliente e do lado do servidor (por exemplo: estruturas, bibliotecas) e suas dependências, usando ferramentas como OWASP Dependency-Check, etc.;
- monitoramento de bibliotecas e componentes sem manutenção ou que não possuem lançamentos de *patches* de segurança para versões anteriores;
- coletar componentes e dependências apenas de fontes oficiais, por meio de *links* seguros.

A07: 2021 - Identification and Authentication Failures (Falhas de Identificação e Autenticação)

Esta categoria trata das vulnerabilidades intrínsecas à identificação e à autenticação.

- **Identificação** se resume à capacidade de constatar exclusivamente um usuário de um sistema ou um aplicativo em execução no sistema.
- **Autenticação** consiste na capacidade de provar que um usuário ou aplicativo é genuinamente quem essa pessoa ou aplicativo alega ser.

No ranking 2017 essa categoria era intitulada como *Broken Authentication* ou Autenticação Quebrada, e permanecia na 2ª posição. Na análise de 2021, foi para a 7ª posição no *ranking* e agora inclui Enumerações de Fraquezas Comuns (CWEs) que estão mais relacionados a falhas de identificação (GODOY, 2021).

Veja algumas formas de prevenção sugeridas pela OWASP (2021):

- implementação de autenticação multifatores;
- não utilização de credenciais padrão, especialmente para usuários administradores;
- implementação de verificações de senha fracas, como também validar senhas baseadas em históricos do usuário, ou em relação a lista de senhas inseguras;
- mapeamento de complexidade e políticas de rotação de senha com as diretrizes do Instituto Nacional de Padrões e Tecnologia (NIST);

- garantir que os PATHs das páginas de registro, recuperação de credencial e API sejam protegidos contra ataques de enumeração de contas usando as mesmas mensagens para todos os resultados;
- limitar e atrasar cada vez mais as tentativas de login malsucedidas;
- registro de todas as falhas e alerta aos administradores quando o enchimento de credenciais, força bruta ou outros ataques forem detectados;
- uso de um gerenciador de sessão integrado e seguro do lado do servidor que gera um novo token de sessão aleatório com alta entropia após o *login*. O identificador de sessão não deve estar no URL, e ser armazenado com segurança e invalidado após o *logout*.

A08:2021 - Software and Data Integrity Failures (Falhas de Integridade de Dados e Software)

É uma nova categoria para 2021, com foco em fazer suposições relacionadas a atualizações de software, dados críticos e pipelines de CI / CD sem verificar a integridade. *Insecure Deserialization* de 2017 agora faz parte dessa categoria (GODOY, 2021).

Obs.: O termo CI pode ser definido como **Continuous Integration**, e CD como *Continuous Delivery*. Uma pipeline de CI / CD consiste numa série de etapas que devem ser executadas para fornecer uma nova versão de software (REDHAT, 2019).

HKCERT (2021), este risco relaciona-se **ao código ou infraestrutura não protegida contra violações de integridade.**

Formas de mitigação para este risco (OWASP, 2021):

- Uso de assinaturas digitais ou mecanismos semelhantes para verificar se o software ou os dados são da fonte esperada e não foram alterados;
- certificar-se das bibliotecas e dependências;
- uso de uma ferramenta de segurança da cadeia de suprimentos de software, como OWASP Dependency Check ou OWASP CycloneDX, para verificar se os componentes não contêm vulnerabilidades conhecidas;
- revisão do código-fonte da aplicação, para evitar alterações de configuração indesejadas, ou introdução de código malicioso no software;
- certificar-se que a pipeline de CI/CD tenha segregação, configuração e controle de acesso adequados, a fim de garantir a integridade do código que flui pelos processos de construção e implantação;
- certificar-se que nenhum dado serializado ou não criptografado não seja enviado a clientes não confiáveis sem alguma forma de verificação de integridade ou assinatura digital para detectar adulteração ou repetição dos dados serializados.

A09:2021 - Security Logging and Monitoring Failures (Falhas em Monitoramento e Segurança de Logs)

OWASP (2021) cita que essa categoria foi expandida para incluir mais tipos de falhas, incluindo aquelas que podem afetar diretamente a visibilidade, o alerta de incidentes e a perícia.

Algumas formas de prevenção:

- todas as falhas de *login*, controle de acesso e validações de entrada do lado do servidor, devem ser registradas para identificação de contas suspeitas ou maliciosas e retidas por tempo suficiente para permitir análise posterior;
- fazer a geração de *logs* em um formato flexível, para que as soluções de gerenciamento dos *logs* possam ser acessadas e consumidas facilmente;
- dados de registro devem ser codificados corretamente para evitar injeções ou ataques nos sistemas de registro ou monitoramento;
- transações de alto valor devem possuir uma trilha de auditoria com controles de integridade, para evitar adulteração ou exclusão;
- estabelecimento de monitoramento e alertas eficazes para que atividades suspeitas sejam detectadas e respondidas rapidamente;
- estabelecimento e adoção de planos de resposta e recuperação de incidentes, etc.

A10: 2021 - Server-Side Request Forgery ou SSRF (Falsificação de Solicitação do Lado do Servidor)

Os dados mostram que a taxa de incidência da ou Falsificação de solicitação do lado do servidor é relativamente baixa (GODOY, 2021) e ela está na última posição do ranking TOP 10.

No entanto, Immuniweb (2021) destaca que as falsificações de solicitações a partir do servidor vieram a se tornar uma das vulnerabilidades mais discutidas em 2021, devido a grandes danos causados por agentes APT (*Advanced Persistent Threats* ou ataque persistente avançado) e por *ransomwares*.

Vejamos algumas **formas de prevenção de SSRF** conforme OWASP (2021):

Da camada de rede:

1. Segmentação de funcionalidades de acesso a recursos remotos em redes separadas para reduzir o impacto de SSRF;

2. implementação de políticas de *firewall*, como “negar por padrão” ou regras de controle de acesso à rede para bloquear todo o tráfego da intranet, exceto o essencial;

3. Estabelecimento de uma propriedade e um ciclo de vida para regras de firewall, baseadas em aplicativos.

4. Registrar todos os fluxos de rede aceitos e bloqueados em firewalls.

Da camada de aplicativo:

1. Limpeza e validação de todos os dados de entrada fornecidos pelo cliente.
2. Implementação do esquema de URL, porta e destino, com uma lista de permissões positiva.
3. Não enviar respostas brutas aos clientes, sempre realizar o refinamento de seu conteúdo.
4. Desativar redirecionamentos HTTP.

Lista de Verificação de Práticas de Programação Segura

O OWASP (*Open Web Application Security Project*) criou um conjunto de listas de verificação de práticas de programação segura para as diversas etapas ou tarefas do desenvolvimento de software.

Veja os pontos principais dessas listas de verificação a seguir (OWASP, 2010).

1. Validação dos Dados de Entrada:

Vamos então aos itens de verificação para a **validação dos dados de entrada (OWASP, 2010)**:

- Efetuar toda a **validação dos dados em um sistema confiável**. Ex.: centralizar todo o processo no servidor.
- Identificar todas as fontes de dados e **classificá-las como sendo ou não de confiança**. Em seguida, validar os dados provenientes de fontes nas quais não exista confiança.
- A **rotina de validação de dados de entrada** deve ser **centralizada na aplicação**.
- Especificar o **conjunto de caracteres apropriados**, como UTF-8, para todas as fontes de entrada de dados.
- **Codificar os dados** para um conjunto de caracteres comuns antes da validação.
- Quando há **falha de validação** a aplicação deve **rejeitar os dados fornecidos**.
- **Determinar se o sistema suporta conjuntos de caracteres estendidos** UTF-8 e em caso afirmativo, validar após efetuar a descodificação UTF-8.
- **Validar todos os dados provenientes dos clientes antes do processamento**, incluindo todos os parâmetros, campos de formulário, conteúdos das URLs e cabeçalhos HTTP.
- **Verificar se os valores de cabeçalho**, tanto dos pedidos, como das respostas, contêm apenas caracteres ASCII.
- **Validar dados provenientes de redirecionamentos**.
- Validar **tipos de dados esperados**.
- Validar intervalo de dados.
- Validar o **tamanho dos dados**.
- **Validar, sempre que possível, todos os dados de entrada através de um método baseado em "listas brancas"** (utilizam uma lista de caracteres ou expressões regulares para definirem os caracteres permitidos).

- Se qualquer caractere **potencialmente perigoso** precisa ser permitido na entrada de dados da aplicação, certifique-se que foram implementados mecanismos adicionais como codificação dos dados de saída, APIs especificas que fornecem tarefas seguras e caminhos de auditoria no uso dos dados pela aplicação.
- Se a rotina de validação padrão **não** abordar as seguintes entradas, então elas devem ser verificadas:

a) Verificar bytes nulos (%00)
b) Verificar se há caracteres de nova linha (%0d, %0a, \r, \n)
c) Verificar se há caracteres "ponto-ponto barra" (../ ou..\) que alteram caminhos.

2. Codificação de Dados de Saída

A lista de verificação é a seguinte (OWASP, 2010):

- **Efetuar toda a codificação dos dados em um sistema confiável**. Ex.: centralizar todo o processo no servidor.
- Utilizar uma **rotina padrão**, testada, para cada tipo de codificação de saída.
- Realizar a **codificação, baseada em contexto, de todos os dados enviados para o cliente que têm origem num ambiente fora dos limites de confiança da aplicação**.
- **Codificar todos os caracteres**, a menos que sejam conhecidos por serem seguros para o interpretador de destino.
- Realizar o **tratamento (limpeza), baseado em contexto**, de todos os dados provenientes de fontes que não sejam de confiança usados para construir consultas SQL, XML, e LDAP.
- Tratar todos os dados provenientes de **fontes que não sejam de confiança** que gerem comandos para o sistema operacional.

3. Autenticação e Gestão de Credenciais

Vamos então aos itens de verificação para a **autenticação e gestão de credenciais (OWASP, 2010)**:

- Requerer **autenticação para todas as páginas e recursos**, **exceto para aqueles que são intencionalmente públicos**.
- Os **mecanismos de autenticação** devem ser executados **num sistema confiável** (Ex.: centralizar todo o processo no servidor).
- Sempre que possível, estabelecer e utilizar **serviços de autenticação padronizados e testados**.
- Utilizar uma **implementação centralizada** para realizar os procedimentos de autenticação, disponibilizando bibliotecas que invocam os serviços de autenticação externos.
- **Separar a lógica de autenticação do recurso que está a ser requisitado** e **usar redirecionadores dos controladores de autenticação centralizados**

- Quando ocorrerem situações excecionais nos controladores de autenticação, **executar procedimentos em caso de falha** com o propósito de manter o sistema seguro.
- Todas as **funções administrativas e de gestão de contas** devem ser tão seguras quanto o mecanismo de autenticação principal.
- Se a aplicação gerir um **repositório de credenciais**, esta deverá garantir que as **senhas são armazenadas na base de dados somente sob a forma de resumo/hash da senha** na forma de one-way salted hashes, e que a tabela/arquivo que armazena as senhas e as próprias chaves são manipuladas apenas pela aplicação. **Não** utilizar o algoritmo de hash MD5, sempre que esse puder ser evitado.
- A **geração dos resumos (hash) das senhas deve ser executada em um sistema confiável**, por exemplo, centralizar o controle no servidor.
- **Validar os dados de autenticação somente no final de todas as entradas de dados**, especialmente para as implementações de autenticação sequencial.
- As **mensagens de falha na autenticação não devem indicar qual parte dos dados de autenticação está incorreta**. Por exemplo, em vez de exibir mensagens como “Nome de usuário incorreto” ou “Senha incorreta”, utilize apenas mensagens como: “Usuário e/ou senha inválidos”, para ambos os casos de erro. As respostas de erro devem ser idênticas nos dois casos.
- Utilizar **autenticação para ligação a sistemas externos** que envolvam tráfego de informação sensível ou acesso a funções.
- As **credenciais de autenticação para acessar serviços externos à aplicação devem ser cifradas** e armazenadas num local protegido de um sistema confiável, por exemplo, no servidor da aplicação. Obs.: **o código-fonte não é considerado um local seguro.**
- Utilizar **apenas pedidos POST para transmitir credenciais de autenticação.**
- **Somente trocar senhas (não temporárias) através de uma ligação protegida (SSL/TLS) ou como dado cifrado**, como no caso de envio de e-mail cifrado. Senhas temporárias enviadas por e-mail podem ser um caso de exceção aceitável.
- Exigir que os **requisitos de complexidade de senha** estabelecidos pela política ou regulamento sejam cumpridos.
- Exigir que os **requisitos de comprimento de senha** estabelecidos pela política ou regulamento sejam cumpridos. O uso de oito caracteres é o mais comum, porém usar 16 é mais recomendado. Considere, ainda, o uso de senhas que contenham várias palavras (uma frase).
- A **entrada da senha deve ser ocultada no dispositivo de apresentação do usuário**.
- **Desativar a conta após um número predefinido de tentativas inválidas de autenticação** (ex: cinco tentativas é o mais comum).
- Os **processos de redefinição de senhas e operações de mudanças** devem exigir os mesmos níveis de controle previstos para a criação de contas e autenticação.

- **Esquemas de pergunta/resposta (predefinidas)** usadas para a redefinição da senha devem evitar ataques que lançam respostas aleatórias.
- Se optar por usar **redefinição de senha baseada em e-mail,** envie um e-mail somente para o endereço predefinido contendo um *link* ou senha de acesso temporário que permitam ao usuário redefinir a senha.
- O **tempo de validade das senhas e dos links temporários deve ser curto**.
- Exigir a mudança de senhas temporárias na próxima vez que o usuário realizar a autenticação no sistema.
- **Notificar o usuário quando a sua senha for reiniciada** (reset).
- **Prevenir a reutilização de senhas**.
- As senhas devem ter, pelo menos, um dia de duração antes de poderem ser alteradas, a fim de evitar ataques de reutilização de senhas.
- Garantir que a troca de senhas está em conformidade com os requisitos estabelecidos na política ou regulamento. O tempo entre as trocas de senhas deve ser controlado administrativamente.
- Desativar a **funcionalidade de lembrar a senha** nos campos de senha do navegador.
- A **data/hora da última utilização** (bem ou mal sucedida) de uma conta de usuário deve ser comunicada na próxima entrada no sistema.
- Realizar monitorização para identificar ataques contra várias contas de utilizadores, utilizando a mesma senha. Esse padrão de ataque é utilizado para explorar o uso de senhas padrão
- **Modificar todas as senhas que, por padrão, são definidas pelos fornecedores**, bem como os identificadores de usuários (IDs) ou desativar as contas associadas.
- Exigir **nova autenticação** dos usuários **antes da realização de operações críticas**.
- Utilizar **autenticação de múltiplos fatores** (utilizando simultaneamente token, senha, biometria etc.) **para contas altamente sensíveis ou de elevado valor transacional.**
- Caso **utilize código de terceiros para realizar a autenticação**, inspecione-o cuidadosamente para garantir que o mesmo não é afetado por qualquer código malicioso.

4. Gestão de Sessões

Vamos então aos itens de verificação para a **gestão de sessões (OWASP, 2010)**:

- Utilizar **controles de gestão de sessão baseados no servidor ou em *framework***. A aplicação deve reconhecer apenas esses identificadores de sessão como válidos.
- A **criação dos identificadores** de sessão deve ser **sempre realizada num sistema de confiança**, por exemplo, centralizado no servidor.
- O controle de gestão de sessão deve usar **algoritmos conhecidos, padronizados e bem testados** que garantam a aleatoriedade dos identificadores de sessão.

- Definir **o domínio e o caminho para os *cookies*** que contêm identificadores de sessão autenticados, para um valor devidamente restrito para o site.
- A funcionalidade de saída (***logout***) **deve encerrar completamente** a sessão ou ligação associada.
- A funcionalidade de saída (***logout***) **deve estar disponível em todas as páginas que requerem autenticação**.
- Estabelecer um tempo de **expiração** da sessão que **seja o mais curto possível**, baseado no balanceamento dos riscos e requisitos funcionais do negócio. Na maioria dos casos não deve ser mais do que algumas horas.
- **Não permitir *logins* persistentes** (sem prazo de expiração) e **realizar o encerramento da sessão periodicamente**, mesmo quando ela estiver ativa. Isso deve ser feito, especialmente, em aplicações que suportam várias ligações de rede ou que se ligam a sistemas críticos. O tempo de encerramento deve estar de acordo com os requisitos do negócio e o usuário deve receber notificações suficientes para atenuar os impactos negativos dessa medida.
- Se uma **sessão** estava **estabelecida** antes do *login* ela deve ser **encerrada** para que uma nova seja estabelecida após o login.
- Gerar um **novo identificador de sessão quando houver uma nova autenticação**.
- **Não permitir conexões simultâneas com o mesmo identificador** de usuário.
- **Não expor os identificadores de sessão em URLs, mensagens de erro ou *logs*.** Os identificadores de sessão devem apenas ser encontrados no cabeçalho do cookie HTTP. Por exemplo, não enviar os identificadores de sessão sob a forma de parâmetros GET.
- **Proteger os dados de sessão do lado servidor** contra acessos não autorizados por outros usuários do servidor, através da implementação de controles de acesso apropriados no servidor.
- Gerar um **novo identificador de sessão e desativar o antigo periodicamente**. Isso pode mitigar certos cenários de ataques de sequestro de sessão (*session hijacking*), quando o identificador de sessão original for comprometido.
- Gerar um **novo identificador de sessão caso a segurança da ligação mude de HTTP para HTTPS**, como pode ocorrer durante a autenticação. Internamente à aplicação, é recomendável utilizar HTTPS de forma constante em vez de alternar entre HTTP e HTTPS.
- Utilizar **mecanismos complementares ao mecanismo padrão de gestão de sessões para operações sensíveis do lado do servidor** – como no caso de operações de gestão de contas – através da utilização de tokens aleatórios ou parâmetros associados à sessão. Este método pode ser usado para prevenir ataques do tipo ***Cross Site Request Forgery*** (CSRF).

- Utilizar **mecanismos complementares à gestão de sessões para operações altamente sensíveis ou críticas**, utilizando tokens aleatórios ou parâmetros em cada requisição em vez de basear-se apenas na sessão.
- Configurar o **atributo "secure" para *cookies*** transmitidos através de uma ligação TLS.
- Configurar os ***cookies* com o atributo "HttpOnly"**, a menos que seja explicitamente necessário ler ou definir os valores dos mesmos através de scripts do lado cliente da aplicação.

5. Controle de Acessos

Vamos aos itens de verificação para o **controle de acessos (OWASP, 2010)**:

- Utilizar **apenas objetos do sistema que sejam de confiança**, como ocorre com os objetos de sessão do servidor, para realizar a tomada de decisão sobre a autorização de acesso.
- Utilizar um **único componente em toda a aplicação Web para realizar o processo de verificação de autorização de acesso**. Isto inclui bibliotecas que invocam os serviços externos de autorização.
- Quando ocorrer alguma **falha** no controle de acesso, ela deve ocorrer de modo seguro.
- **Negar todos os acessos**, caso a aplicação não consiga ter acesso às informações contidas na configuração de segurança.
- Garantir o controle de autorização em todos os pedidos, inclusive em scripts do lado do servidor, "includes" e pedidos provenientes de tecnologias do lado cliente, como AJAX e Flash.
- **Isolar do código da aplicação os trechos de código que contêm lógica privilegiada**.
- **Restringir o acesso aos arquivos e outros recursos**, incluindo aqueles que estão fora do controle direto da aplicação, somente aos usuários autorizados.
- **Restringir o acesso às URLs protegidas** somente aos usuários autorizados.
- **Restringir o acesso às funções protegidas** somente aos usuários autorizados.
- **Restringir o acesso às referências diretas aos objetos** somente aos usuários autorizados.
- **Restringir o acesso aos serviços** somente aos usuários autorizados.
- **Restringir o acesso aos dados da aplicação** somente aos usuários autorizados.
- **Restringir o acesso aos atributos e dados dos usuários**, bem como informações das políticas usadas pelos mecanismos de controle de acesso.
- **Restringir o acesso às configurações de segurança** relevantes apenas aos usuários autorizados.
- As regras de controle de acesso representadas pela camada de apresentação devem coincidir com as regras presentes no lado servidor.

- Se o estado dos dados deve ser armazenado no lado do cliente, utilizar mecanismos de criptografia e verificação de integridade no lado servidor para detectar possíveis adulterações.
- Garantir que os fluxos lógicos da aplicação obedecem as regras de negócio.
- Limitar o **número de transações que um único usuário ou dispositivo** pode executar em determinado período de tempo.
- Utilizar o campo "referer" do cabeçalho somente como forma de verificação suplementar. O mesmo não deve ser usado sozinho como forma de validação de autorização porque ele pode ter o valor adulterado.
- **Se for permitida a existência de sessões autenticadas por longos períodos de tempo**, fazer a **revalidação periódica da autorização** do usuário para garantir que os privilégios não foram modificados e, caso tenham sido, realizar o registo em log do utilizador e exigir nova autenticação.
- Implementar a **auditoria das contas de usuário** e assegurar a desativação de contas não utilizadas.
- A aplicação deve dar suporte à desativação de contas e ao encerramento das sessões quando terminar a autorização do usuário.
- As contas de serviço ou contas de suporte a ligações provenientes ou destinadas a serviços externos devem possuir o menor privilégio possível.
- Criar uma Política de Controle de Acesso para documentar as regras de negócio da aplicação, tipos de dados e critérios ou processos de autorização para que os acessos possam ser devidamente concedidos e controlados. Isto inclui identificar requisitos de acessos, tanto para os dados, como para os recursos do sistema.

6. Práticas de Criptografia

Lista de verificação:

- Todas as funções de criptografia utilizadas para proteger dados sensíveis dos usuários da aplicação, devem ser implantadas em um sistema confiável (neste caso o servidor).
- A senha mestre deve ser protegida contra acessos não autorizados.
- Quando ocorrer alguma falha nos módulos de criptografia, permitir que as mesmas ocorram de modo seguro.
- Todos os números, nomes de arquivos, GUIDs e strings aleatórias devem ser gerados usando um módulo criptográfico com gerador de números aleatórios, somente se os valores aleatórios gerados forem impossíveis de serem deduzidos.
- Estabelecer e utilizar uma política e processo que defina como é realizada a gestão das chaves criptográficas.

7. Tratamento de Erros e Log

Lista de verificação:

- **Não expor informações sensíveis nas repostas de erros**, inclusive detalhes de sistema, identificadores de sessão ou informação da conta do usuário.
- **Usar mecanismos de tratamento de erros** que não mostrem informações de depuração (debug) ou informações da pilha de exceção.
- Usar mensagens de erro genéricas e páginas de erro personalizadas.
- A aplicação deve tratar os erros sem se basear nas configurações do servidor.
- A memória reservada (alocada) deve ser liberada de modo apropriado quando ocorrerem condições de **erro**.
- O tratamento de erros lógicos associados com os controles de segurança deve, por padrão, negar o acesso.
- Todos os controles de log devem ser implementados num sistema de confiança, por exemplo, centralizar todo o processo no servidor.
- Os controles de log devem dar suporte tanto para os casos de sucesso como os de falha relacionados com os eventos de segurança.
- Garantir que os logs armazenam eventos importantes.
- Garantir que as entradas de log que incluam dados nos quais não se confia não sejam executadas como código-fonte na interface de visualização de logs.
- Restringir o acesso aos logs apenas para pessoal autorizado.
- Utilizar uma rotina centralizada para realizar todas as operações de log.
- **Não armazenar informações sensíveis nos registos de logs**, como detalhes desnecessários do sistema, identificadores de sessão e senhas.
- Garantir o uso de algum mecanismo que conduza (ou facilite) o processo de análise de logs.
- Registar em log todas as falhas de validação de entrada de dados.
- Registar em log todas as tentativas de autenticação, especialmente as que falharam por algum motivo.
- Registar em log todas as falhas de controle de acesso.
- Registar em log todos os eventos suspeitos de adulteração, inclusive alterações inesperadas no estado dos dados.
- Registar em log as tentativas de ligação com tokens de sessão inválidos ou expirados.
- Registar em log todas as exceções lançadas pelo sistema.
- Registar em log todas as funções administrativas, inclusive as mudanças realizadas nas configurações de segurança.
- Registar em log todas as falhas de ligação TLS com o backend.
- Registar em log todas as falhas que ocorreram nos módulos de criptografia.
- Utilizar uma função de hash criptográfica para validar a integridade dos registros de log.

8. Proteção de Dados

Lista de verificação:

- Implementar uma política de privilégio mínimo, restringindo os usuários apenas às funcionalidades, dados e informações do sistema que são necessárias para executarem as suas tarefas.
- **Proteger contra acesso não autorizado todas as cópias temporárias ou registadas em cache que contenham dados sensíveis e estejam armazenadas no servidor** e excluir esses arquivos logo que não forem mais necessários.
- Cifrar informações altamente sensíveis quando armazenadas – como dados de verificação de autenticação – mesmo que estejam no lado servidor, usando sempre algoritmos conhecidos, padronizados e bem testados.
- Proteger o código-fonte presente no servidor para que não seja descarregado por algum usuário.
- Não armazenar senhas, *strings* de ligação ou outras informações confidenciais em texto claro/legível ou em qualquer forma **criptograficamente insegura** no lado cliente.
- Remover comentários do código de produção que podem ser acessados pelos usuários e podem revelar detalhes internos do sistema ou outras informações sensíveis.
- Remover aplicações desnecessárias e documentação do sistema que possam revelar informações importantes para os atacantes.
- Não incluir informações sensíveis nos parâmetros de pedidos HTTP GET.
- Desativar a funcionalidade de auto completar nos formulários que contenham informações sensíveis, inclusive no formulário de autenticação.
- Desativar a cache realizada no lado do cliente das páginas que contenham informações sensíveis.
- A aplicação deve dar suporte à remoção de dados sensíveis quando estes não forem mais necessários. Por exemplo: informação pessoal ou dados financeiros.
- Implementar mecanismos de controle de acesso apropriados para dados sensíveis armazenados no servidor. Isto inclui dados em cache, arquivos temporários e dados que devem ser acessíveis somente.

9. Segurança nas Comunicações

Lista de verificação:

- Utilizar **criptografia na transmissão de todas as informações sensíveis**. Isto deve incluir **TLS** para proteger a ligação e deve ser complementado com criptografia de ficheiros que contém dados sensíveis ou ligações que não usam o protocolo HTTP.
- Os **certificados TLS devem ser válidos**, possuírem o nome de domínio correto, **não** estarem expirados e serem instalados com certificados intermédios, quando necessário.

- Quando ocorrer alguma falha nas ligações TLS, **o sistema não deve fornecer uma ligação insegura**.
- Utilizar ligações **TLS** para todo o conteúdo que requer acesso autenticado ou que contenha informação sensível.
- Utilizar **TLS** para ligações com sistemas externos que envolvam funções ou informações sensíveis.
- Utilizar um padrão único de implementação **TLS** configurado de modo apropriado.
- Especificar a **codificação dos caracteres** para todas as conexões.
- **Filtrar os parâmetros que contenham informações sensíveis**, provenientes do "HTTP referer", **nos links para sites externos**.

10. Configuração do Sistema

Lista de verificação:

- Garantir que os **servidores, *frameworks* e componentes do sistema estão a executar a última versão** aprovada.
- Garantir que os servidores, frameworks e componentes do sistema possuem as atualizações mais recentes aplicadas para a versão em uso.
- **Desativar a listagem de diretórios**.
- Restringir, para o mínimo possível, os privilégios do servidor Web, dos processos e das contas de serviços.
- Quando ocorrerem exceções no sistema, garantir que as falhas ocorrem de modo seguro.
- Remover todas as funcionalidades e arquivos desnecessários.
- Remover código de teste ou qualquer funcionalidade desnecessária para o ambiente de produção, antes de disponibilizar o sistema no servidor de produção.
- Prevenir a divulgação da estrutura de diretórios **impedindo que robôs de busca façam a indexação de arquivos sensíveis**, através da configuração do arquivo "robots.txt". Os diretórios que não devem ser acessados por estes indexadores devem ser colocados num diretório isolado. Assim, apenas é necessário negar o acesso ao diretório pai definido no arquivo "robots.txt", evitando ter que negar o acesso a cada diretório individualmente.
- **Definir quais métodos HTTP, GET ou POST, a aplicação irá suportar** e se serão tratados de modo diferenciado nas diversas páginas da aplicação.
- **Desativar as extensões HTTP desnecessárias** como, por exemplo, o WebDAV. Caso seja necessário o uso de alguma extensão HTTP com o propósito de suportar a manipulação de arquivos, utilize um mecanismo de autenticação conhecido, padronizado e bem testado.
- Se o servidor processa tanto pedidos HTTP 1.0 como HTTP 1.1, certificar-se de que ambos são configurados de modo semelhante ou assegure que qualquer diferença que possa existir seja compreendida, como, por exemplo, o manuseamento de métodos HTTP estendidos.

- Remover **informações desnecessárias** presentes nos cabeçalhos de resposta HTTP e que podem estar relacionadas com o sistema operacional, versão do servidor web e *frameworks* de aplicação.
- O armazenamento da configuração de segurança para a aplicação deve ser capaz de ser produzida de forma legível para dar suporte à auditoria.
- Implementar um **sistema de gestão de ativos** para manter o registo dos componentes e programas.
- **Isolar o ambiente de desenvolvimento da rede de produção** e conceder acesso somente para grupos de desenvolvimento e testes. É comum os ambientes de desenvolvimento serem configurados de modo menos seguro do que os ambientes de produção. Deste modo, os atacantes podem usar essa diferença para descobrir vulnerabilidades comuns ou encontrar formas de exploração das mesmas.
- **Implementar um sistema de controle de mudanças** para gerir e registar as alterações no código, tanto de desenvolvimento, como dos sistemas em produção

11. Segurança em Banco de Dados

Lista de verificação:

- **Usar consultas parametrizadas** fortemente **tipificadas**.
- Utilizar **validação de entrada e codificação de saída** e assegurar a abordagem de meta caracteres. Se houver falha, o comando não deverá ser executado na base de dados.
- Certificar-se de que as variáveis são fortemente tipificadas.
- Realizar a **codificação (*escaping*) de meta caracteres** em instruções SQL 8.
- A aplicação **deve usar o menor nível possível de privilégios ao acessar a base de dados**.
- Usar credenciais seguras para acessar a base de dados.
- **Não** incluir strings de conexão na aplicação. As strings de conexão devem ser armazenadas num arquivo de configuração separado, armazenado num sistema de confiança e as informações devem ser cifradas.
- Usar procedimentos armazenados (***stored procedures***) para abstrair o acesso aos dados e permitir a remoção de permissões das tabelas no banco de dados.
- Encerrar a conexão assim que possível.
- **Remover ou modificar senhas padrão de contas administrativas**. Utilizar **senhas robustas** (pouco comuns ou difíceis de deduzir) ou **implementar autenticação de múltiplos fatores**. Desativar qualquer funcionalidade desnecessária na base de dados, como “stored procedures” ou serviços não utilizados. Instalar o conjunto mínimo de componentes ou de opções necessárias (redução da superfície de ataque).
- **Eliminar o conteúdo desnecessário** incluído por padrão pelo fornecedor como esquemas e bases de dados de exemplo.

- **Desativar** todas as **contas** criadas **por padrão** e que não sejam necessárias para suportar os requisitos de negócio.
- A aplicação deve conectar-se à base de dados com **diferentes credenciais** de segurança para cada tipo de necessidade como, por exemplo, usuário, somente leitura, convidado ou administrador.

12. Gestão de Arquivos

Lista de verificação:

- **Não repassar dados fornecidos pelos usuários** diretamente a uma função de inclusão dinâmica.
- **Solicitar autenticação** antes de permitir que seja feito o carregamento de arquivos.
- Limitar os tipos de arquivos que podem ser enviados para aceitar somente os necessários ao propósito do negócio.
- Validar se os arquivos enviados são do tipo esperado, através da validação dos cabeçalhos, pois, realizar a verificação apenas pela extensão não é suficiente.
- Não gravar arquivos no mesmo diretório de contexto da aplicação Web. Os arquivos devem ser armazenados no servidor de conteúdos ou na base de dados.
- Prevenir ou restringir o carregamento de qualquer arquivo que possa ser interpretado ou executado pelo servidor Web.
- Desativar privilégios de execução nos diretórios de armazenamento de arquivos.
- Implantar o carregamento seguro nos ambientes UNIX por meio da montagem do diretório de destino como uma unidade lógica, usando o caminho associado ou o ambiente de “chroot”.
- Ao referenciar arquivos, usar uma lista branca (whitelist) de nomes e de tipos de arquivos permitidos. Realizar a validação do valor do parâmetro passado e caso ele não corresponda ao que é esperado, rejeitar a entrada ou utilizar um valor padrão.
- Não transmitir, sem nenhum tipo de tratamento, os dados informados pelo usuário a redirecionamentos dinâmicos. Se isso for necessário, o redirecionamento deverá aceitar apenas URLs relativas e validadas.
- Não passar caminhos de diretórios ou de arquivos em pedidos. Usar algum mecanismo de mapeamento desses recursos para índices definidos numa lista predefinida de caminhos.
- Nunca enviar o caminho absoluto do arquivo para o cliente.
- Certificar-se de que os arquivos da aplicação e os recursos estão definidos somente com o atributo de leitura.
- Verificar os arquivos que os usuários submeterem através do mecanismo de carregamento em busca de vírus e malwares.

13. Gestão de Memória

Lista de verificação:

- Utilizar **controle de entrada/saída de dados** que não sejam de confiança.
- Verificar se o **buffer é tão grande** quanto o especificado.
- Ao usar funções que aceitem determinado número de bytes para realizar cópias, como strncpy(), esteja ciente de que se o tamanho do buffer de destino for igual ao tamanho do buffer de origem, ele não pode encerrar a sequência de caracteres com valor nulo (null).
- **Verificar os limites do buffer** caso as chamadas à função sejam realizadas em ciclos e verificar se não há nenhum risco de ocorrer gravação de dados além do espaço reservado.
- **Truncar todas as strings de entrada** para um tamanho razoável antes de passá-las para as funções de cópia e concatenação.
- Na **liberação de recursos alocados** para objetos de conexão, identificadores de arquivos etc., não contar com o "garbage collector" e realizar a tarefa explicitamente.
- Usar pilhas não executáveis, quando disponíveis.
- Evitar o uso de **funções reconhecidamente vulneráveis** como printf(), strcat(), strcpy() etc.
- Liberar a memória alocada (reservada) de modo apropriado após concluir a sub-rotina (função/método) e em todos pontos de saída.

14.Práticas Gerais de Programação

Lista de verificação:

- Para tarefas comuns, utilizar sempre código testado, gerido e aprovado ao invés de criar código novo e não gerido.
- Utilizar **APIs** que executem tarefas específicas para realizar operações do **sistema operacional**. **Não** permitir que a aplicação execute comandos diretamente no sistema operacional, especialmente através da utilização de "shells" de comando iniciadas pela aplicação.
- Utilizar **mecanismos de verificação de integridade** por "checksum" ou "hash" para verificar a integridade do código interpretado, bibliotecas, arquivos executáveis e arquivos de configuração.
- Utilizar mecanismos de bloqueio para evitar pedidos simultâneos para a aplicação ou utilizar um mecanismo de sincronização para evitar condições de concorrência (*race conditions*).
- Proteger as variáveis compartilhadas e os recursos contra acessos concorrentes inapropriados.
- Instanciar explicitamente todas as variáveis e dados persistentes durante a declaração, ou antes da primeira utilização.

- Quando a aplicação tiver que ser executada com privilégios elevados, aumentar os privilégios o mais tarde possível e revogá-los logo que seja possível.
- **Evitar erros de cálculo** decorrentes da falta de entendimento da representação interna da linguagem de programação usada e de como é realizada a interação com os aspetos de cálculo numérico.
- **Não transferir, diretamente, dados fornecidos pelo usuário** para qualquer função de execução dinâmica sem realizar o tratamento dos dados de modo adequado.
- Restringir a **geração e a alteração de código** por parte dos usuários.
- Rever todas as **aplicações secundárias, códigos e bibliotecas de terceiros** para determinar a necessidade do negócio e validar as funcionalidades de segurança, uma vez que estas podem introduzir novas **vulnerabilidades**.
- Implementar **atualizações de modo seguro**. Se a aplicação precisar realizar atualizações automáticas, utilizar mecanismos de assinatura digital para garantir a integridade do código e garantir que os clientes façam a verificação da assinatura após descarregarem as atualizações. Usar canais cifrados para transferir o código a partir do host do servidor.

RESUMO

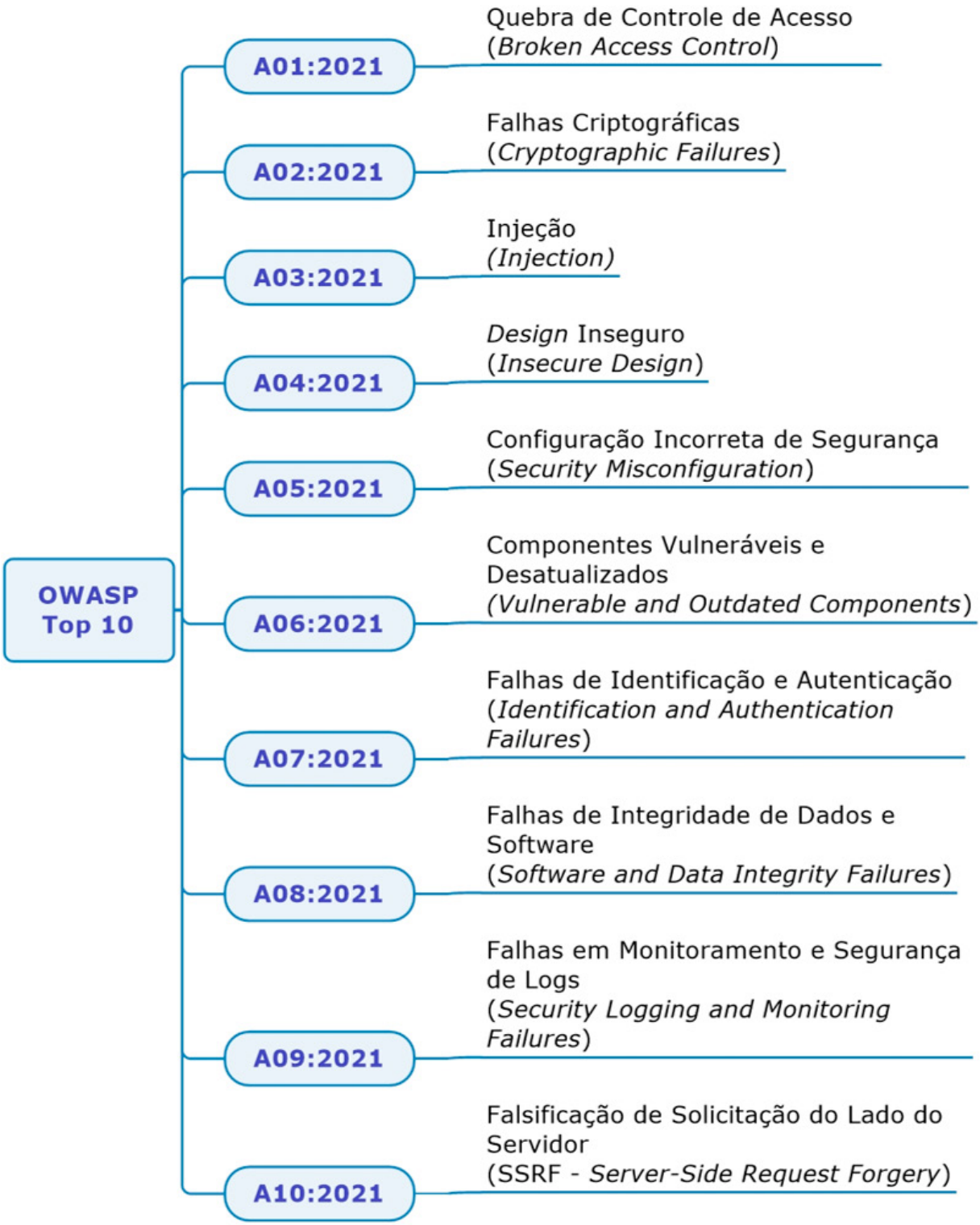

Figura. OWASP Top 10 - 2021 (Fonte: Quintão, 2022)

QUESTÕES COMENTADAS NA AULA

001. (CEBRASPE (CESPE)/ANALISTA DE CONTROLE EXTERNO (TCE-RJ)/ ORGANIZACIONAL/TECNOLOGIA DA INFORMAÇÃO/2022) Julgue o item a seguir, quanto às ferramentas e técnicas de exploração de vulnerabilidades em aplicativos web, à Open Web Application Security Project (OWASP) e às ameaças e vulnerabilidades em aplicações web.
Classificação de Risco para o Top 10 é uma metodologia baseada na OWASP Risk Rating Methodology e consiste em estimar, para cada categoria do Top 10, o risco peculiar que cada falha introduz em uma aplicação web típica e, posteriormente, ordenar o Top 10 de acordo com as falhas que tipicamente introduzem o risco mais significativo para uma aplicação.

002. (CEBRASPE (CESPE)/ANALISTA DE CONTROLE EXTERNO (TCE-RJ)/ ORGANIZACIONAL/TECNOLOGIA DA INFORMAÇÃO/2022) Julgue o item a seguir, quanto às ferramentas e técnicas de exploração de vulnerabilidades em aplicativos web, à Open Web Application Security Project (OWASP) e às ameaças e vulnerabilidades em aplicações web.
Caso uma aplicação permita que comandos SQL sejam digitados nos inputs de seus formulários e concatenados diretamente nos comandos SQL da própria aplicação, sem que seja realizada uma validação ou tratamento antecedente, certamente essa aplicação estará vulnerável ao ataque conhecido como SQL Injection.

QUESTÕES DE CONCURSO

003. (CEBRASPE (CESPE)/ANALISTA JUDICIÁRIO (TJ-RJ)/TECNOLOGIA DA INFORMAÇÃO/ANALISTA DE SEGURANÇA DA INFORMAÇÃO/2021) O OWASP (Open Web Application Security Project) criou um conjunto de listas de verificação de práticas de programação segura para as diversas etapas ou tarefas do desenvolvimento de software, como a gestão de sessões, com os seguintes itens de verificação:

I – não permitir logins persistentes;

II – não permitir conexões simultâneas com o mesmo identificador de usuário;

III – utilizar apenas pedidos POST para transmitir credenciais de autenticação.

Considerando a gestão de sessões, no desenvolvimento seguro de software, assinale a opção correta.

a) Apenas o item I está certo.

b) Apenas o item III está certo.

c) Apenas os itens I e II estão certos.

d) Apenas os itens II e III estão certos.

e) Todos os itens estão certos.

004. (CEBRASPE (CESPE)/ANALISTA DE CONTROLE EXTERNO (TCE-RJ)/ORGANIZACIONAL/TECNOLOGIA DA INFORMAÇÃO/2022) Julgue o item a seguir, quanto às ferramentas e técnicas de exploração de vulnerabilidades em aplicativos web, à Open Web Application Security Project (OWASP) e às ameaças e vulnerabilidades em aplicações web.

As ferramentas de spidering são usadas na exploração de falhas e têm como finalidade catalogar solicitações HTTP/S enviadas a partir do navegador e respostas geradas pela aplicação web.

005. (FGV/ANALISTA JUDICIÁRIO (TJDFT)/APOIO ESPECIALIZADO/ANÁLISE DE SISTEMAS/2022) PedidosSemEstresse é uma aplicação Web destinada a digitalizar o processo de pedidos de serviços de um órgão da administração pública. A interface de PedidosSemEstresse utilizada pelos usuários faz chamadas a uma API RESTful e não utiliza facilidades de login único (single sign-on – SSO). Recentemente, o usuário interno João utilizou suas próprias credenciais com privilégios somente de execução de métodos GET para explorar vulnerabilidades e teve acesso direto a API RESTful. Assim, João fez chamadas a métodos POST com sucesso.

Com base no OWASP Top Ten, a vulnerabilidade explorada por João é da categoria:

a) Injection;

b) Broken Access Control;

c) Software and Data Integrity Failures;

d)Vulnerable and Outdated Components;

e) Identification and Authentication Failures.

006. (CEBRASPE(CESPE)/SEFAZ-CE/AUDITOR FISCAL DE TECNOLOGIA DA INFORMAÇÃO DA RECEITA ESTADUAL/2021) A respeito de autenticação de dois fatores e OWASP Top 10, julgue o item a seguir.

A inadequada configuração de segurança, um dos riscos da OWASP Top 10, pode ocorrer em qualquer nível de serviço de uma aplicação; em razão disso, o uso de scanners e testes automatizados é ineficaz na tarefa de detectar falhas de configuração.

007. (CEBRASPE(CESPE)/SEFAZ-AL/AUDITOR FISCAL DE FINANÇAS E CONTROLE DE ARRECADAÇÃO DA FAZENDA ESTADUAL/2021) A respeito de MPS/BR, OWASP e criptografia, julgue o próximo item.

De acordo com a OWASP TOP 10 2021, para o risco *design* inseguro, são medidas de prevenção para o desenvolvimento seguro o uso da modelagem de ameaças para autenticações críticas, controle de acesso e lógica de negócios.

008. (CEBRASPE(CESPE)/SERPRO/ANALISTA/ESPECIALIZAÇÃO: DESENVOLVIMENTO DE SISTEMAS/2021) No que se refere a autenticação e riscos de segurança, julgue o item subsequente.

Quanto aos riscos de segurança derivados da exposição de dados sensíveis contidos na lista OWASP Top 10, é recomendável que o tráfego de dados confidenciais seja criptografado e que o seu armazenamento interno seja feito sem criptografia, de modo a viabilizar as funções de auditoria dos sistemas.

009. (CEBRASPE (CESPE)/PROFISSIONAL DE TECNOLOGIA DA INFORMAÇÃO (ME)/ ATIVIDADES TÉCNICAS DE COMPLEXIDADE GERENCIAL, DE TECNOLOGIA DA INFORMAÇÃO E DE ENGENHARIA SÊNIOR/SEGURANÇA DA INFORMAÇÃO E PROTEÇÃO DE DADOS/2020) Julgue o seguinte item, a respeito de testes de invasão (pentest) em aplicações web, banco de dados, sistemas operacionais e dispositivos de redes.

O guia de testes do OWASP enumera verificações para cerca de setenta vulnerabilidades, agrupadas em classes, como a de gerenciamento de sessões, que trata de erros na implementação das regras de negócio.

010. (FGV/ALERJ/ESPECIALISTA LEGISLATIVO/TECNOLOGIA DA INFORMAÇÃO/2017) A implementação de mecanismos de segurança é necessária para manter a confidencialidade, a integridade e a disponibilidade dos recursos de informação em sistemas de software. Sobre mecanismos de segurança para mitigar as ocorrências de vulnerabilidades em aplicações web, analise as afirmativas a seguir:

I – As rotinas de validação de dados de entrada devem ser centralizadas nos componentes que rodam no navegador por meio do uso intensivo de JavaScript.

II – Utilizar apenas pedidos POST para transmitir credenciais de autenticação.

III – Ativar o cache do navegador para as páginas que contenham informações sensíveis.

Está correto o que se afirma em:

a) somente I;

b) somente II;

c) somente III;

d) somente I e II;

e) I, II e III.

GABARITO

1. C.
2. C.
3. c
4. E
5. b
6. e
7. C
8. E
9. E
10. b

GABARITO COMENTADO

003. (CEBRASPE (CESPE)/ANALISTA JUDICIÁRIO (TJ-RJ)/TECNOLOGIA DA INFORMAÇÃO/ANALISTA DE SEGURANÇA DA INFORMAÇÃO/2021) O OWASP (Open Web Application Security Project) criou um conjunto de listas de verificação de práticas de programação segura para as diversas etapas ou tarefas do desenvolvimento de software, como a gestão de sessões, com os seguintes itens de verificação:

I – não permitir logins persistentes;

II – não permitir conexões simultâneas com o mesmo identificador de usuário;

III – utilizar apenas pedidos POST para transmitir credenciais de autenticação.

Considerando a gestão de sessões, no desenvolvimento seguro de software, assinale a opção correta.

a) Apenas o item I está certo.

b) Apenas o item III está certo.

c) Apenas os itens I e II estão certos.

d) Apenas os itens II e III estão certos.

e) Todos os itens estão certos.

A questão relaciona alguns dos itens de verificação para a **gestão de sessões**.

I – Certa**. Não permitir *logins* persistentes** (**sem prazo de expiração**) e **realizar o encerramento da sessão periodicamente**, mesmo quando ela estiver ativa. Conforme OWASP (2010) isso deve ser feito, especialmente, em aplicações que suportam várias ligações de rede ou que se ligam a sistemas críticos. O tempo de encerramento deve estar de acordo com os requisitos do negócio e o usuário deve receber notificações suficientes para atenuar os impactos negativos dessa medida.

II – Certa. Não se deve permitir conexões simultâneas com o mesmo identificador de usuário.

III – Errada. Utilizar apenas pedidos POST para transmitir credenciais de autenticação é um item de verificação relacionado à autenticação e gestão de credenciais, não sendo ligado à gestão de sessão.

Veja mais: <https://owasp.org/www-pdf-archive/OWASP_SCP_v1.3_pt-BR.pdf>.

Letra c.

004. (CEBRASPE (CESPE)/ANALISTA DE CONTROLE EXTERNO (TCE-RJ)/ORGANIZACIONAL/TECNOLOGIA DA INFORMAÇÃO/2022) Julgue o item a seguir, quanto às ferramentas e técnicas de exploração de vulnerabilidades em aplicativos web, à Open Web Application Security Project (OWASP) e às ameaças e vulnerabilidades em aplicações web.

As ferramentas de spidering são usadas na exploração de falhas e têm como finalidade catalogar solicitações HTTP/S enviadas a partir do navegador e respostas geradas pela aplicação web.

Crawler (robô ou *spider*) é um software que faz uma varredura na internet e cadastra os *links* encontrados, possibilitando encontrar novas páginas. O processo que um Web crawler executa é chamado de *Web crawling* ou *spidering*.
Referências: 1)https://www.oficinadanet.com.br/artigo/otimizacao__seo/qual-a-diferenca-entre-robo-spider-e-crawler
2 https://neilpatel.com/br/blog/web-crawler/
Errado.

005. (FGV/ANALISTA JUDICIÁRIO (TJDFT)/APOIO ESPECIALIZADO/ANÁLISE DE SISTEMAS/2022) PedidosSemEstresse é uma aplicação Web destinada a digitalizar o processo de pedidos de serviços de um órgão da administração pública. A interface de PedidosSemEstresse utilizada pelos usuários faz chamadas a uma API RESTful e não utiliza facilidades de login único (single sign-on – SSO). Recentemente, o usuário interno João utilizou suas próprias credenciais com privilégios somente de execução de métodos GET para explorar vulnerabilidades e teve acesso direto a API RESTful. Assim, João fez chamadas a métodos POST com sucesso.
Com base no OWASP Top Ten, a vulnerabilidade explorada por João é da categoria:
a) Injection;
b) Broken Access Control;
c) Software and Data Integrity Failures;
d)Vulnerable and Outdated Components;
e) Identification and Authentication Failures.

O **controle de acesso** impõe a política de modo que os usuários não possam agir fora de suas permissões pretendidas. Dessa forma, **a quebra do controle de acesso (*Broken Access Control*)** normalmente leva à divulgação, modificação ou destruição de dados, informações ou ao desempenho de uma funcionalidade comercial do sistema, que se encontra fora dos limites do usuário (OWASP, 2021)
Letra b.

006. (CEBRASPE(CESPE)/SEFAZ-CE/AUDITOR FISCAL DE TECNOLOGIA DA INFORMAÇÃO DA RECEITA ESTADUAL/2021) A respeito de autenticação de dois fatores e OWASP Top 10, julgue o item a seguir.
A inadequada configuração de segurança, um dos riscos da OWASP Top 10, pode ocorrer em qualquer nível de serviço de uma aplicação; em razão disso, o uso de scanners e testes automatizados é ineficaz na tarefa de detectar falhas de configuração.

Configurações incorretas de segurança ou *Security Misconfiguration* estão relacionadas a **controles de segurança que são configurados incorretamente ou de forma insegura**, **colocando sistemas e dados em risco**, como alterações de configuração mal documentadas ou a utilização de configurações padrão.
Nesse contexto, o uso de *scanners* e ferramentas de testes automatizadas é de grande valia pois permite relatar e relacionar as fragilidades encontradas.
Errado.

007. (CEBRASPE(CESPE)/SEFAZ-AL/AUDITOR FISCAL DE FINANÇAS E CONTROLE DE ARRECADAÇÃO DA FAZENDA ESTADUAL/2021) A respeito de MPS/BR, OWASP e criptografia, julgue o próximo item.
De acordo com a OWASP TOP 10 2021, para o risco *design* inseguro, são medidas de prevenção para o desenvolvimento seguro o uso da modelagem de ameaças para autenticações críticas, controle de acesso e lógica de negócios.

De acordo com a OWASP TOP 10 2021, para o risco *design* inseguro, são medidas de prevenção para o desenvolvimento seguro:
implementação do **SDLC (*Software Development Life Cycle* - ciclo de vida de desenvolvimento seguro);**
implementação de uma biblioteca de padrões de *design* seguros para utilização na aplicação;
avaliação do processo de autenticação de aplicativos, controle de acesso, lógica de negócios e a **utilização da modelagem de ameaças para identificação de possíveis vetores de ataque**;
criação de testes de integração e verificações para validar se todos os fluxos críticos são resistentes ao modelo de ameaça projetado;
atenção aos recursos computacionais e ao limite do consumo pelo usuário do serviço.
Certo.

008. (CEBRASPE(CESPE)/SERPRO/ANALISTA/ESPECIALIZAÇÃO: DESENVOLVIMENTO DE SISTEMAS/2021) No que se refere a autenticação e riscos de segurança, julgue o item subsequente.
Quanto aos riscos de segurança derivados da exposição de dados sensíveis contidos na lista OWASP Top 10, é recomendável que o tráfego de dados confidenciais seja criptografado e que o seu armazenamento interno seja feito sem criptografia, de modo a viabilizar as funções de auditoria dos sistemas.

A recomendação é que não se armazene dados confidenciais desnecessariamente. Descarte-o o mais rápido possível ou use tokenização compatível com PCI DSS ou mesmo truncamento. Os dados não retidos não podem ser roubados.

Errado.

009. (CEBRASPE (CESPE)/PROFISSIONAL DE TECNOLOGIA DA INFORMAÇÃO (ME)/ ATIVIDADES TÉCNICAS DE COMPLEXIDADE GERENCIAL, DE TECNOLOGIA DA INFORMAÇÃO E DE ENGENHARIA SÊNIOR/SEGURANÇA DA INFORMAÇÃO E PROTEÇÃO DE DADOS/2020) Julgue o seguinte item, a respeito de testes de invasão (pentest) em aplicações web, banco de dados, sistemas operacionais e dispositivos de redes.

O guia de testes do OWASP enumera verificações para cerca de setenta vulnerabilidades, agrupadas em classes, como a de gerenciamento de sessões, que trata de erros na implementação das regras de negócio.

O OWASP mantém **uma lista com as 10 falhas de segurança de aplicativos da Web mais perigosas**, juntamente com os métodos mais eficazes para lidar com elas. São elas:

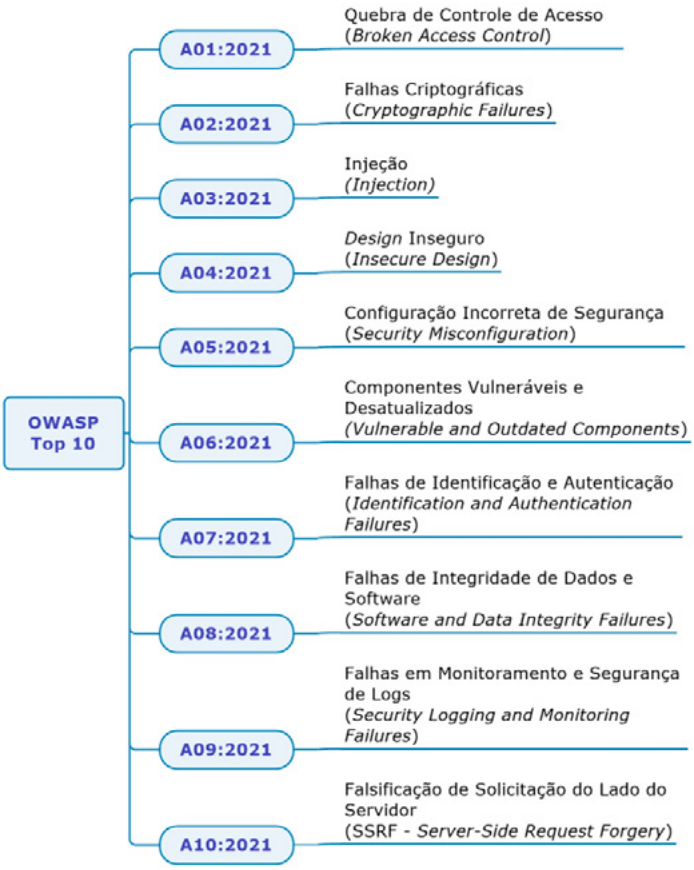

Figura. OWASP Top 10 - 2021 (Fonte: Quintão, 2022)

O documento de **Melhores Práticas de Codificação Segura OWASP – Guia de Referência Básica** destaca recomendações em formas de **listas de verificações**, que podem ser integradas no ciclo de desenvolvimento de aplicações, permitindo reduzir as vulnerabilidades mais comuns em aplicações web. As recomendações da Gestão (ou Gerenciamento) de Sessões baseiam-se no tratamento inseguro dos identificadores de sessão.

Errado.

---

010. (FGV/ALERJ/ESPECIALISTA LEGISLATIVO/TECNOLOGIA DA INFORMAÇÃO/2017) A implementação de mecanismos de segurança é necessária para manter a confidencialidade, a integridade e a disponibilidade dos recursos de informação em sistemas de software. Sobre mecanismos de segurança para mitigar as ocorrências de vulnerabilidades em aplicações web, analise as afirmativas a seguir:

I – As rotinas de validação de dados de entrada devem ser centralizadas nos componentes que rodam no navegador por meio do uso intensivo de JavaScript.

II – Utilizar apenas pedidos POST para transmitir credenciais de autenticação.

III – Ativar o cache do navegador para as páginas que contenham informações sensíveis. Está correto o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente I e II;
e) I, II e III.

I – Errada**.** Deve-se efetuar toda a **validação dos dados de entrada em um sistema confiável**. Ex.: centralizar todo o processo no servidor (OWASP, 2010).

II – Certa. Deve-se utilizar **apenas pedidos POST para transmitir credenciais de autenticação.**

III – Errada. Deve-se proteger contra acesso não autorizado todas as cópias temporárias ou registadas em cache que contenham dados sensíveis e estejam armazenadas no servidor e excluir esses arquivos logo que não forem mais necessários.

Conforme visto, somente o item II está correto. A letra B é a resposta!

Letra b.

REFERÊNCIAS

GODOY, A. de. **OWASP: O que é e qual a sua importância?** 2021. Disponível em: <https://www.integrasul.com.br/blog/owasp-o-que-e-entenda-agora-como-funciona>. Acesso em: 20 de jul. 2022.

HAGI, R. **A05:2021 – Security Misconfiguration**. Disponível em: <https://www.linkedin.com/pulse/a052021-security-misconfiguration-raphael-hagi/>. Acesso em: ago. 2021.

HKCERT. **OWASP Top 10-2021 is Now Released.** 2021. Disponível em: <https://www.hkcert.org/blog/owasp-top-10-2021-is-now-released>. Acesso em: 11 de dez. 2021.

IMMUNIWEB. **OWASP Top 10 in 2021: Cryptographic Failures Practical Overview**. 2021. Disponível em: <https://www.immuniweb.com/blog/OWASP-cryptographic-failures.html/>. Acesso em: 01 de jan. 2022.

_____. **OWASP Top 10 in 2021: Identification and Authentication Failures Practical Overview**. 2021. Disponível em: < https://www.immuniweb.com/blog/OWASP-identification-and-authentication-failures.html/>. Acesso em: 09 de dez. 2021.

ORTEGA, F. D.; CÂMARA, C. E. **Um estudo aplicado a segurança de aplicações web a study applied to web application security.** Revista Ubiquidade, ISSN 2236-9031 – v.4, n.2 – jul. a dez. de 2021, p. 85. Disponível em: <https://revistas.anchieta.br/index.php/RevistaUbiquidade/article/download/1860/1630>. Acesso em: 08 ago. 2022.

OWASP. **Melhores Práticas de Codificação Segura OWASP Guia de Referência Rápida**. 2010. sponível em: <https://owasp.org/www-pdf-archive/OWASP_SCP_v1.3_pt-PT.pdf>. Acesso em: 02 dez. 2022.

_____. **A01:2021 – Broken Access Control**. 2021. Disponível em: < https://owasp.org/Top10/A01_2021- Broken_Access_Control/> Acesso em: 27 de dez. 2021.

_____. **A02:2021 – Cryptographic Failures**. 2021. Disponível em: < https://owasp.org/Top10/A02_2021- Cryptographic_Failures//> Acesso em: 01 de dez. 2021.

_____. **A03:2021 – Injeção**. 2021. Disponível em: < https://owasp.org/Top10/pt_BR/A03_2021-Injection/>. Acesso em: 04 de dez. 2021.

_____. **A04:2021 – Insecure Design**. 2021. Disponível em: < https://owasp.org/Top10/A04_2021- Insecure_Design/>. Acesso em: 05 de nov. 2021.

_____. **A05:2021 – Security Misconfiguration**. 2021. Disponível em: < https://owasp.org/Top10/A05_2021-Security_Misconfiguration/>. Acesso em: 06 de dez. 2021.

_____. **A06:2021 – Vulnerable and Outdated Components**. 2021. Disponível em: < https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/>. Acesso em: 08 de dez. 2021.

_____. **A07:2021 – Identification and Authentication Failures**. 2021. Disponível em: < https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/>. Acesso em: 09 de dez. 2021.

_____. **A08:2021 – Software and Data Integrity Failures**. 2021. Disponível em: < https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures//>. Acesso em: 13 de dez. 2021.

_____. **A09:2021 – Security Logging and Monitoring Failures**. 2021. Disponível em: < https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/>. Acesso em: 15 de dez. 2021.

_____. **A10:2021 – Server-Side Request Forgery (SSRF)**. 2021. Disponível em: <https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/>. Acesso em: 16 de dez. 2021.

_____. **OWASP Top 10:2021**. 2021. Disponível em: < https://owasp.org/www-project-top-ten/#>. Acesso em: 01 ago. 2022.

_____. **OWASP Top 10:2021.** 2021. Disponível em: <https://owasp.org/Top10/>. Acesso em: 01 ago. 2022.

_____. **OWASP Cheat Sheet Series**. Disponível em: <https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html#authorization-cheat-sheet>. Acesso em: 08 ago. 2022.

QUINTÃO, PATRÍCIA LIMA**. Notas de aula**. 2022.

REDHAT. **What is a CI/CD pipeline?** 2019. Disponível em: <https://www.redhat.com/en/topics/devops/what-cicd-pipeline>. Acesso em: 26 de dez. 2021.

SEGOVIA, Antonio Jose. **How to use penetration testing for ISO 27001 A.12.6.1.** Advisera. 2016. Disponível em: <https://advisera.com/27001academy/blog/2016/01/18/how-to-use-penetration-testing-for-iso-27001-a-12-6-1/>. Acesso em: 17 de out. 2021.

SIEMBA. **OWASP Top 10: Broken Access Control**. 2021. Disponível em: < https://www.siemba.io/post/owasp-top-10-broken-access-control>. Acesso em: 27 de ago. 2021.

Patrícia Quintão

Mestre em Engenharia de Sistemas e computação pela COPPE/UFRJ, Especialista em Gerência de Informática e Bacharel em Informática pela UFV. Atualmente é professora no Gran Cursos Online; Analista Legislativo (Área de Governança de TI), na Assembleia Legislativa de MG; Escritora e Personal & Professional Coach.

Atua como professora de Cursinhos e Faculdades, na área de Tecnologia da Informação, desde 2008. É membro: da Sociedade Brasileira de Coaching, do PMI, da ISACA, da Comissão de Estudo de Técnicas de Segurança (CE-21:027.00) da ABNT, responsável pela elaboração das normas brasileiras sobre gestão da Segurança da Informação.

Autora dos livros: Informática FCC - Questões comentadas e organizadas por assunto, 3ª. edição e 1001 questões comentadas de informática (Cespe/UnB), 2ª. edição, pela Editora Gen/Método.

Foi aprovada nos seguintes concursos: Analista Legislativo, na especialidade de Administração de Rede, na Assembleia Legislativa do Estado de MG; Professora titular do Departamento de Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia; Professora substituta do DCC da UFJF; Analista de TI/Suporte, PRODABEL; Analista do Ministério Público MG; Analista de Sistemas, DATAPREV, Segurança da Informação; Analista de Sistemas, INFRAERO; Analista - TIC, PRODEMGE; Analista de Sistemas, Prefeitura de Juiz de Fora; Analista de Sistemas, SERPRO; Analista Judiciário (Informática), TRF 2ª Região RJ/ES, etc.

@coachpatriciaquintao **/profapatriciaquintao**

@plquintao **t.me/coachpatriciaquintao**