# Sumário

| **Histórico** | | | | | |
|---|---|---|---|---|---|
| Versão | Elaborado por | Data | Revisado por | Data | Principais Alterações |
| 1.0 | Edenilton Soares - ejsoares | 19.04.2021 | Maria das Graças - mgsmfernandes | 08.06.2021 | |
| | | | | | |
| | | | | | |

## Objetivo

Prover aos membros do CDS um framework para avaliar o nível de requerimentos de segurança que deve ser utilizado e implementado, garantindo assim uma maior segurança.

## Introdução

A implementação de uma metodologia que possa auxiliar no desenvolvimento seguro é um dos principais pontos dentro de um processo de Desenvolvimento Seguro. Os requisitos apresentados neste documento foram definidos com base nas especificações propostas pelo Open Web Application Security Project® (OWASP)[1], mais precisamente, pelo projeto da OWASP Application Security Verification Standard (ASVS) desenvolvido para assegurar parâmetros para cuidar da qualidade e da segurança da aplicação durante seu desenvolvimento.

A utilização do ASVS proporciona ao desenvolvedor a possibilidade de criar uma estrutura de avaliação de controles de segurança de aplicações web para, desta forma, atender aos requisitos necessários para proteção das informações que serão tratadas pela aplicação.

## Níveis de Implementação

O ASVS apresenta três níveis de verificação que devem ser observados durante o processo de desenvolvimento de aplicações web. Cada um dos níveis aqui apresentados tem sua particularidade e profundidade.

| Nível 1 - Segurança Mínima Necessária | Nível 2 - Padrão para web | Nível 3 - Alto Valor, Garantia, Segurança e Resiliência |
|---|---|---|
| Para níveis baixos de garantia de segurança. Apresenta o nível mínimo de segurança esperado para qualquer aplicação web. | Para aplicações que contêm dados sensíveis. É o nível recomendado para a maioria das aplicações. | Para as aplicações mais críticas, que requerem o mais alto nível de confiança. |

[1] OWASP é uma fundação sem fins lucrativos que trabalha para melhorar a segurança do software com o suporte da comunidade formada por desenvolvedores, profissionais de segurança e corporações interessadas em promover um ambiente mais seguro para todos na web.

## Nível 1 - Segurança mínima

Para se adequar a este nível, a aplicação não deve possuir vulnerabilidades de fácil descoberta e exploração, bem como as listadas no TOP 10 da OWASP ou mesmo em outras listas de vulnerabilidades similares classificadas como críticos e altos.

O nível 1 é considerado o mínimo de segurança para qualquer aplicação, no entanto, se os dados que devem ser protegidos por sua aplicação são críticos, raramente somente o Nível 1 trará o nível de proteção recomendado.

## Nível 2 – Segurança Recomendada

Uma aplicação atinge o nível 2 do ASVS se tiver resiliência suficiente contra a maioria dos riscos associados a ataques através de recursos externos. Portanto, deve ser utilizado como o nível padrão de segurança para as aplicações web. Este nível, é apropriado para aplicações que tratam dados sensíveis, garantindo que os controles de segurança funcionam e são utilizados dentro da aplicação.

Ameaças a aplicações neste nível serão tipicamente feitas por atacantes habilidosos e motivados, focando em alvos específicos, utilizando ferramentas e técnicas efetivas para o descobrimento e exploração de fraquezas dentro de aplicações.

| Características da Aplicação |
|---|
| Armazenar, transmitir ou processar dados pessoais, confidenciais ou sensíveis de funcionários, parceiros de negócio, terceiros ou prestadores de serviço |
| Aplicações cuja integridade é crítica. |

## Nível 3 – Segurança Alto valor, garantia e resiliência

Este nível é o mais alto utilizado para avaliação utilizando o ASVS e normalmente reservado para aplicações que requerem níveis significativos de verificações de segurança, como em situações de infraestrutura crítica, quando realizam processos críticos para o negócio ou funções sensíveis, quando é necessária a garantia da sua integridade para o funcionamento do negócio. ou seja, nível de segurança 3 é exigido em cenários onde uma falha pode provocar um impacto significativo nas operações da organização.

| Características da Aplicação |
|---|
| Aplicações que executam funções críticas, onde a falha pode afetar significativamente as operações. Ex: uma aplicação interna que seja utilizada por várias outras, Sefaz Identity, Portal de Assinaturas, etc<br><br>Aplicações que manipulam informações sigilosas ou que realizam processos críticos para o negócio ou funções sensíveis. Ex.: manipulação de informações sujeitas ao sigilo fiscal. |

## Seleção dos Requisitos de Segurança

Para o processo de seleção dos requisitos de segurança ser eficiente, deve-se aplicar uma avaliação mista, levando em consideração os ativos que devem ser protegidos e os riscos mapeados.

O anexo 1 apresenta uma lista de requisitos de segurança propostos, divididos nos três níveis de segurança. Além disso, os requisitos propostos nos anexos, não deverão ser tratados como únicos. Caso seja notada a necessidade de adição de um requisito, o mesmo pode e deve ser adicionado à sua respectiva tabela dentro do respectivo anexo.

A partir destes anexos, o PO irá gerar a Lista de Requisitos de Segurança, que será passada para a fase de sprint de desenvolvimento. Para gerar esta Lista de Requisitos de Segurança do Projeto, considere utilizar ferramentas, Kanban boards, Scrum ou tabelas que facilitarão na seleção e acompanhamento da implementação dos requisitos.

## Anexo 1 – Requisitos de Segurança para Aplicações Web

Este anexo apresenta os controles que devem ser avaliados nos cenários de aplicações web, e caso atendam às necessidades de segurança da solução, devem ser aplicados.

Esta lista de requisitos foi baseada primordialmente no ASVS e relacionado com o Guia de Requisitos de Segurança OWASP Top 10 **Proactive Controls**, que apresenta uma lista das técnicas de segurança que deveriam ser incluídas em todo projeto de desenvolvimento de software, organizadas por grau de relevância, escrita por desenvolvedores, para desenvolvedores.

# 1. Requisitos de Verificação de Gerenciamento de Sessão

## 1.1. Gerenciamento de Sessão

| # | Descrição | 1 | 2 | 3 | CWE |
|---|---|---|---|---|---|
| 1.1.1. | Recomenda-se a implementação de Criptografia na Comunicação<br><br>Utilizar o protocolo de TLS (v2 e v3) para garantia da criptografia da comunicação durante a sessão. Por esse motivo é necessário que todas as aplicações hospedadas em servidores web estejam compiladas com framework no mínimo acima do 4.6.2 e que sejam desabilitados os protocolos e cifras considerados obsoletos conforme orientações do item 1.8 do Padrão – Requisito de configuração IIS.<br><br>https://docs.microsoft.com/pt-br/mem/configmgr/core/plan-design/security/enable-tls-1-2<br><br>https://vulncat.fortify.com/en/weakness?q=tls | ✓ | | | |
| 1.1.2. | Recomenda-se a implementação da criptografia no cookie através do uso do “SECURE ATRIBUTO COOKIE”**,** responsável por instruir aos navegadores da web no envio dos cookies apenas por canal seguro criptografado.<br><br>Mecanismo de proteção de sessão para impedir a divulgação do ID de sessão através dos ataques de MITM (Man In The Middle), garantindo que o invasor não capture o trafego do navegador da web.<br><br>No caso do uso de autenticação de formulários, forçaremos conexões seguras para isso também, caso contrário, isso poderá negar o efeito de fazer isso para os cookies, pois os dados precisariam estar disponíveis de forma clara para coincidir com a configuração de autenticação de formulários.<br><br>Se estiver criando cookies manualmente, também poderá marcá-los como seguros em C #:<br><br>https://vulncat.fortify.com/en/weakness?q=Secure | ✓ | | | 614 |
| 1.1.3. | Recomenda-se a utilização do atributo HTTPOnly que auxilia na instrução aos navegadores da web a não permitire que scripts do tipo (Java Script e VBScript) possam acessar os cookies por meio de objeto DOM (Document.cookie). Contramedida necessária para impedir roubo de ID de sessão através de ataques do tipo XSS (Cross Site Scripting).<br><br>O valor do **httpOnlyCookies** é **true** neste caso, como no exemplo anterior, HttpOnly também pode ser definido no código C #:<br><br>https://vulncat.fortify.com/en/weakness?q=HTTPOnly | ✓ | | | 1004 |

| | | | | | |
|---|---|---|---|---|---|
| 1.1.4. | Recomenda-se utilizar lista branca para atribuir permissões de **METHOD HTTP**. Por padrão o **.net framework** permite todos os verbos, sendo assim, mesmo que a configuração negue chamadas GET e POST para todos os usuários, no caso de solicitação HEAD seria permitido.<br><br>O controle de segurança não consegue bloquear verbos que não estão listados.<br><br>O aplicativo atualiza seu estado com base em solicitações GET ou outros verbos HTTP arbitrários.<br><br>Use uma lista branca especificando uma política de autorização que liste apenas os verbos que devem ser permitidos, negando todos os outros. Como o .NET Framework permite caracteres especiais, isso pode ser feito com o uso do caractere *. | ✓ | | | 650 |
| 1.1.5. | O mecanismo de gerenciamento de sessão pode fazer uso de dois tipos de cookies: Persistentes e não persistentes. Portanto é altamente recomendado o uso de cookies não persistentes para fins de gerenciamento de sessão, para que o ID da sessão não permaneça em cache no cliente da web por longos períodos evitando que seja reaproveitado.<br><br>O objetivo é que após o logout seja invalidado o token da sessão, de forma que mesmo com tentativas de usar a opção de voltar para páginas anteriores, o recurso não retorne a uma sessão autenticada.<br><br>Normalmente são de 2 a 5 minutos para aplicativos de alto valor e 15 a 30 minutos para aplicativos de baixo risco, quando uma sessão expirar o aplicativo deverá invalidar a sessão nos dois lados, cliente e servidor. | ✓ | | | 613 |
| 1.1.6. | Recomenda-se a implementação do Content Security Policy (CSP).<br><br>Camada extra de segurança que ajuda a detectar e mitigar ataques de Cross Site Scripting (XSS) e ataques de injeção de dados. Esses ataques tem como objetivo o roubo de dados, defacement da aplicação ou distribuição de malware através de redirecionamentos na aplicação.<br><br>A alteração deve ser realizada no web.config no parâmetro httpProtocol**.**<br><br>https://www.syncfusion.com/blogs/post/shield-your-asp-net-mvc-web-applications-with-content-security-policy-csp.aspx<br><br>https://vulncat.fortify.com/en/weakness?q=Content-Security-Policy | ✓ | | | 79 |

| | | | | | |
|---|---|---|---|---|---|
| 1.1.7. | Recomenda-se a implementação do HSTS (HTTP Strict Transport Security) para garantia de conexões seguras.<br><br>O cabeçalho HTTP Strict Transport Security evita que uma requisição não criptografada (HTTP) seja enviada a partir do navegador para um site. Caso o usuário tente acessar a página via HTTP, o navegador irá fazer um redirecionamento interno para a versão HTTPS da página. Isso evita que um ataque do tipo man-in-the-middle possa interceptar uma requisição ao site e comprometê-la mesmo quando utilizando certificado TLS.<br><br>HSTS requer pelo menos uma solicitação HTTPS bem-sucedida para estabelecer a política HSTS. O aplicativo deve verificar cada solicitação HTTP e redirecionar ou rejeitar a solicitação HTTP.<br><br>**Exemplo:**<br><br>GET https://portal.fazenda.sp.gov.br/<br><br>**Cabeçalhos da resposta:**<br><br>Strict-Transport-Security: max-age=31536000<br><br>A diretiva max-age indica o tempo, em segundos, que o navegador deve guardar a informação de que o site só deve ser acessado via HTTPS. É importante notar que essa configuração só deve ser feita caso não exista uma versão HTTP que deva por algum motivo ser acessada pelo usuário. Caso o cabeçalho HTTP Strict Transport Security seja acrescentado a resposta, a versão HTTP do site ficará inacessível pelo tempo colocado em max-age.<br><br>Para fins de teste, recomenda-se aumentar o valor de max-age gradualmente. Sugestões de valores a serem testados são 3600, 86400, 2592000 e 31536000 (1 hora, 1 dia, 1 mês e 1 ano, respectivamente). Recomenda-se manter o valor 31536000 (1 ano) como valor final.<br><br>Para ambientes de produção que estão implementando HTTPS pela primeira vez, defina o HstsOptions. Defina o valor de horas para não mais do que um único dia, caso precise reverter a infraestrutura HTTPS para HTTP.<br><br>**NÃO DEVE SER USADA A DIRETIVA includeSubDomains EM HIPÓTESE ALGUMA!**<br><br>https://docs.microsoft.com/pt-br/aspnet/core/security/enforcing-ssl?view=aspnetcore-5.0&tabs=visual-studio<br><br>https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/Strict-Transport-Security | ✓ | | | |

| | | | | | |
|---|---|---|---|---|---|
| | https://vulncat.fortify.com/en/weakness?q=HSTS | | | | |
| 1.1.8. | Recomenda-se o uso do X-Frame-Options.<br><br>O cabeçalho de resposta pode ser usado para indicar ou não se um navegador deve ser autorizado a processar uma página. Os sites usarão o X-Frame-Options para evitar ataques de clickjacking (Roubo de click), garantindo que seu conteúdo não seja incorporado a outros sites.<br><br>X-FRAME-OPTIONS é um cabeçalho da Web que pode ser usado para permitir ou negar que uma página seja iframed. Habilitando X-Frame-Options Os cabeçalhos de resposta HTTP defendem contra XFS (Cross-Frame Scripting), clickjacking e outras formas de ataque.<br><br>Opções de definição do cabeçalho:<br><br>• X-FRAME-OPTIONS:DENY<br>A página não pode ser colocada em um quadro, não importa quem seja (incluindo o próprio enquadramento do site). Se você não usa frames em seu próprio site, isso é uma boa solução.<br><br>• X-FRAME-OPTIONS: SAMEORIGIN<br>A página pode ser emoldurada desde que o domínio seja igual. Isso é bom se você estiver usando quadros você mesmo.<br><br>• X-FRAME-OPTIONS: ALLOW-FROM https://myotherdomain.com<br>A página pode ser enquadrada pelos domínios especificados.<br><br>Erros comuns:<br><br>Colocar X-Frame-Options dentro de um elemento <meta> é inútil! Por enquanto, <meta http-equiv="X-Frame-Options" content="deny"> não tem nenhum efeito.<br><br>X-Frame-Options funciona somente colocando a configuração através do cabeçalho HTTP, como nos exemplos abaixo.<br><br>https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/X-Frame-Options<br><br>https://vulncat.fortify.com/en/weakness?q=X-FRAME-OPTIONS | ✓ | | | 1021 |

| | | | | | |
|---|---|---|---|---|---|
| 1.1.9. | Recomenda-se o uso do X-Content-Type.<br><br>O header de resposta HTTP X-Content-Type-Options é um marcador usado pelo servidor para indicar que os MIME types enviados pelos headers Content-Type não devem ser alterados e seguidos. Isto permite que o usuário opte por não participar do chamado MIME Type.<br><br>Para atenuar essa constatação, o programador também pode: (1) defini-lo globalmente para todas as páginas do aplicativo no arquivo web.config ou (2) definir o cabeçalho necessário página por página apenas para as páginas que possam incluir conteúdo controlável pelo usuário.<br><br>Para defini-lo globalmente, adicione o cabeçalho no arquivo web.config para o aplicativo hospedado pelo IIS (Serviços de Informações da Internet):<br><br>https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/X-Content-Type-Options<br><br>https://vulncat.fortify.com/en/weakness?q=X-Content-Type | ✓ | | | |
| 1.1.10. | Recomenda-se a utilização do método Post.<br><br>Garantir que os dados sensíveis de sessão principalmente formulários que enviam senhas devem usar o método POST.<br><br>Os aplicativos devem especificar o atributo de método da tag FORM como method=“POST". | ✓ | | | 598 |
| 1.1.11. | Recomenda-se a implementação do SameSite.<br><br>Permite que um servidor defina um atributo de cookie, impossibilitando o navegador de enviar esse cookie junto com solicitações entre sites. O objetivo principal é reduzir o risco de vazamento de informações entre origens e fornecer alguma proteção contra ataques de falsificação de solicitações entre sites.<br><br>https://docs.microsoft.com/pt-br/aspnet/samesite/csharpwebforms<br><br>https://vulncat.fortify.com/en/weakness?q=Samesite | ✓ | | | 1275 |

| | | | | | |
|---|---|---|---|---|---|
| 1.1.12. | Recomenda-se o uso do Session Fixation.<br><br>Autenticar um usuário ou estabelecer uma nova sessão de usuário, sem invalidar nenhum identificador de sessão existente, dá a um invasor a oportunidade de roubar sessões autenticadas.<br><br>Os aplicativos da Web devem ignorar qualquer ID de sessão fornecido pelo navegador do usuário no login e devem sempre gerar uma nova sessão na qual o usuário fará login caso seja autenticado com êxito.<br><br>Impedir que o invasor obtenha um ID de sessão válido, um aplicativo da web em um sistema estrito só deve emitir IDs de sessões recém-geradas para os usuários após eles terem se autenticado com sucesso (ao invés de emiti-los junto com o formulário de login). Isso significa que um invasor que não seja um usuário legítimo do sistema não será capaz de obter uma ID de sessão válida e, portanto, não poderá realizar um ataque de fixação de sessão.<br><br>Destruição de Sessão No Logout/Timeout ou quando o usuário fechar a página, a sessão deve ser 'interrompida'. Isso precisa ser feito no servidor e não apenas no navegador, excluindo o cookie de sessão:<br><br>https://dopeydev.com/session-fixation-attack-and-prevention-in-asp-net/<br><br>https://owasp.org/www-community/controls/Session_Fixation_Protection<br><br>https://vulncat.fortify.com/en/weakness?q=Session%20Fixation | ✓ | | | |