ABORDAGEM DE SEGURANÇA NO DESENVOLVIMENTO DE APLICAÇÕES WEB

SECURITY APPROACH TO WEB APPLICATION DEVELOPMENT

Moisés Ferreira dos Santos, Fatec Araraquara, moises.santos17@fatec.sp.gov.br
Paola Aparecida Barbosa, Fatec Araraquara, paola.barbosa@fatec.sp.gov.br
João Emmanuel D Alkmin Neves, Fatec Araraquara, jeneves@gmail.com

Resumo

O presente artigo aborda os conceitos fundamentais da segurança em aplicações web, oferecendo exemplos representativos das ameaças predominantes. Tais sistemas enfrentam uma ampla gama de ameaças, incluindo erros de codificação, deficiências na aplicação e inadequações nas configurações dos servidores, com potencial para infligir danos substanciais às organizações. A abordagem do Desenvolvimento Sustentável Seguro (DSS) emerge como um modelo promissor, visando a reutilização do código-fonte para prevenir a redundância de esforços. Alinhada com os princípios da sustentabilidade, essa estratégia objetiva reduzir o consumo de recursos, em sintonia com a crescente ênfase na sustentabilidade ambiental. Esta pesquisa investiga as melhores práticas no desenvolvimento seguro de aplicações web, abrangendo a implementação, gestão de senhas, configuração de servidores e atualizações de software, entre outras medidas. A relevância deste tema é incontestável na atualidade, à luz da expansão contínua da base global de usuários de aplicações web. Além disso, este esforço é catalisado pela crescente incidência de ataques cibernéticos, particularmente no contexto brasileiro, onde tais incidentes têm uma presença marcante.

Palavras-chave: Segurança, aplicações web, ameaças, desenvolvimento sustentável seguro, ataques cibernéticos.

Abstract

This article addresses the fundamental concepts of security in web applications, offering representative examples of prevalent threats. Such systems face a wide range of threats, including coding errors, application deficiencies, and server configuration inadequacies, with the potential to inflict substantial harm on organizations. The Secure Sustainable Development (SSD) approach emerges as a promising model, aiming at source code reuse to prevent redundancy of efforts. Aligned with sustainability principles, this strategy aims to reduce resource consumption in line with the growing emphasis on environmental sustainability. This research investigates best practices in secure web application development, covering implementation, password management, server configuration, and software updates, among other measures. The relevance of this topic is undeniable in today's context, given the continued expansion of the global user base for web applications. Furthermore, this effort is catalyzed by the increasing incidence of cyberattacks, particularly in the Brazilian context, where such incidents have a significant presence.

Keywords: *Security, web applications, threats, secure sustainable development, cyberattacks.*

1. Introdução

Nos tempos atuais, a segurança no mundo das aplicações Web é um assunto muito falando, principalmente com o crescente número de usuários em todo o mundo. As aplicações web tem diversos fatores que podem trazer ameaças, como falhas referentes a codificação incorreta, problemas na aplicação, servidores sem as devidas configurações. Logo, temos várias ações que podemos seguir para nos proteger contra ameaças e ataques, que muitas vezes causam prejuízos catastróficos em uma organização.

O conceito de Desenvolvimento Sustentável Seguro (DSS) é um modelo que se submete ao desenvolvimento de sistemas digitais onde tem como objetivo e principal fator, o repúdio de código fonte para impedir o retrabalho dos desenvolvedores. O pensamento origina-se das teorias de sustentabilidade onde se tem a mitigação de uso de recursos, sendo assim essa hipótese muito conhecida pelo boom da sustentabilidade ambiental que tem acontecido nos últimos anos.

Esse tema também vai ao encontro ao objetivo 17.6 de Tecnologia do desenvolvimento sustentável da ONU Brasil, que visa fortalecer os meios de implementação e revitalizar a parceria global para o desenvolvimento sustentável. Ele visa melhorar e aumentar o compartilhamento de conhecimentos dos termos que são acordados entre si com a intenção de utilizar mecanismos que facilitam a tecnologia global.

Portanto, este trabalho tem por objetivo estudar quais são as melhores maneiras de se desenvolver aplicações seguras para aplicações da internet, além de visar melhores práticas de ferramentas, entendendo e reduzindo os riscos e ameaças em aplicações web.

Esta pesquisa justifica-se pelo aumento do número de ataques hackers pelo mundo, principalmente no Brasil que é um dos países que mais sofrem com esses ataques.

2. Desenvolvimento seguro para aplicações web: o que é segurança no site?

O processo de desenvolvimento seguro de aplicações web hoje em dia está cada vez mais diversificado. Antes era muito comum apenas se falar sobre a segurança no protocolo HTTPS. Com o avanço da tecnologia e as necessidades de se obter um ambiente cada vez mais seguro, ferramentas começaram a surgir para ajudar nessa construção.

Atualmente, para navegar na internet precisamos tomar os devidos cuidados. A internet

tem sido alvo frequente de ataques criminosos ocasionando a queda dos serviços, senhas, cartões e vários outros dados sendo expostos, trazendo muitos prejuízos para pessoas e empresas.

Segundo Mozilla (2023), “a Internet é um lugar perigoso! Com grande regularidade, ouvimos sobre a indisponibilidade de sites devido a ataques de negação de serviço ou a exibição de informações modificadas (e muitas vezes prejudiciais) em suas páginas iniciais. Em outros casos de alto perfil, milhões de senhas, endereços de Email e detalhes de cartão de crédito foram vazados para o domínio público, expondo os usuários do site a vergonha pessoal e risco financeiro.”

A segurança se torna o principal fator para qualquer tipo desses ataques, protegendo as pessoas contra modificação de dados, indisponibilidade de sites, dados vazados, dentre muitos outros.

“O objetivo da segurança do site é impedir esse (ou qualquer) tipo de ataque. A definição mais formal de segurança do site é o ato ou prática de proteger sites contra acesso, uso, modificação, destruição ou interrupção não autorizados.” (MOZILLA, 2023)

E como fazemos para deixar um site mais seguro? Vamos lá, podemos começar pelo design do site, aplicar regras nos servidores web, configurar políticas de senhas com diferentes caracteres, além de renová-la a cada certo tempo. Temos frameworks que também servem como grandes aliados para habilitar mecanismos potentes de defesas e assim, mitigar os riscos.

Segundo Mozilla (2023), a segurança efetiva do site requer um esforço de design em todo o site: em sua aplicação web, na configuração do servidor da web, em suas políticas para criar e renovar senhas e no código do cliente. Embora tudo isso pareça muito ameaçador, a boa notícia é que, se você estiver usando framework (uma estrutura da web) no servidor, é provável que ele habilitará "por padrão" mecanismos de defesa robustos e bem pensados contra vários ataques mais comuns. Outros ataques podem ser mitigados através da configuração do servidor da web, por exemplo, ativando o HTTPS. Por fim, existem ferramentas de scanner de vulnerabilidades disponíveis publicamente que podem ajudá-lo a descobrir se você cometeu algum erro óbvio.

3. Ameaças à segurança do site

Abaixo, vamos listar as ameaças mais comuns dos sites e as possíveis maneiras de se diminuir.

3.1. *Cross-Site Scripting* (XSS)

O XXS é um ataque onde o criminoso injeta scripts do lado do cliente através de aplicações web em navegadores de outros usuários. Com isso, ele pode enviar cookies deste site ao invasor, e com essa informação o invasor poderá fazer login em qualquer outro site se passando pelo usuário, conseguindo obter informações pessoais, dados de cartões bancários, acesso a senhas.

Segundo Mozilla (2023), XSS é um termo usado para descrever uma classe de ataques que permitem que um invasor injete scripts do lado do cliente através do site nos navegadores de outros usuários. Como o código injetado chega ao navegador a partir do site, o código é confiável e pode fazer coisas como enviar o cookie de autorização do site do usuário ao invasor. Quando o invasor possui o cookie, ele pode fazer login em um site como se fosse o usuário e fazer tudo que o usuário pode, como acessar os detalhes do cartão de crédito, ver detalhes do contato ou alterar senhas.

Acontece que, após o script ser inserido no site ele não é modificado de imediato. Ele será executado após o carregamento da página. Em um exemplo do que esse ataque pode causar, quando o usuário alvo clicar neste link já infectado, o atacante terá acesso a todas as informações fornecidas, podendo fazer uso em vários outros sites se passando pelo usuário.

A vulnerabilidade XSS é chamada de persistente quando após o script ter sido infiltrado no código, outro usuário poderá executar sem querer, pois o site é exibido sem alterações. Um exemplo desse ataque é em uma conversa dentro de um chat entre usuários que contêm HTML. Esses comentários podem armazenar scripts mal-intencionados de invasores e, quando houver a exibição destes comentários, o script será executado e o invasor terá acesso as informações dos usuários.

Mesmo que os dados de solicitações do tipo ‘POST’e ‘GET’ estão entre as fontes mais comuns das vulnerabilidades XSS, qualquer tipo de dado dentro do navegador podem ser vulneráveis.

"A melhor defesa contra as vulnerabilidades do XSS é remover ou desativar qualquer marcação que possa conter instruções para executar o código. Para HTML, isso inclui elementos, como <script>, <object>, <embed> e <link>." (MOZILLA, 2023)

Para que não ocorra essa modificação de dados através da execução de scripts, usamos o processo de limpeza de entrada. Algumas estruturas da web limpam, por padrão, estes dados automaticamente.

3.2. Injeção de *SQL*

A injeção de SQL é a porta de entrada para que os atacantes executem o código SQL ilegal em um banco de dados, e assim, dando acessos indevidos aos atacantes que, que podem modificar e até excluir os dados.

> Os cibercriminosos usam ataques de injeção de *SQL* para entrar no banco de dados do site. Eles podem simplesmente querer causar caos ao excluir dados ou editar o banco de dados, especialmente se sites financeiros forem o alvo. Assim que o *cibercriminoso* consegue o controle do banco de dados, é fácil para ele bagunçar o saldo de conta das pessoas e desviar o dinheiro para a conta dele. (AVAST, 2020)

Temos alguns tipos de injeção de SQL: baseada em erro, baseada em erros booleanos e injeção em tempo.

Este tipo de vulnerabilidade poderá acontecer se a entrada do usuário puder ser alterada no significado de sua instrução. O exemplo de código abaixo mostra todos os usuários fornecidos através de um formulário em HTML:

Figura 1 – Instrução *SQL*

```
statement = "SELECT * FROM usuarios WHERE name = '" + nomeUsuario + "';"
```

Fonte: MOZILLA (2023)

> Se o usuário especificar um nome real, a instrução funcionará como pretendido. No entanto, um usuário mal-intencionado pode alterar completamente o comportamento dessa instrução *SQL* para a nova instrução no exemplo a seguir, simplesmente especificando o texto em negrito para o nomeUsuario. (MOZILLA, 2023)

Figura 2 – Instrução *SQL*

```
SELECT * FROM usuarios WHERE name = 'a';DROP TABLE usuarios; SELECT * FROM
userinfo WHERE 't' = 't';
```

Fonte: MOZILLA (2023)

A instrução SQL modificada acima cria uma outra instrução que exclui uma tabela de usuários e seleciona os dados da tabela “userinfo”, que mostrará todas as informações de todos os usuários, individualmente.

Para evitar com que esse tipo de ataque aconteça, precisamos garantir de que os dados do usuário que serão usados na consulta SQL não poderão modificar a natureza da consulta. Podemos usar o chamado ‘escape’ nos caracteres de entrada de dados do usuário com significados relevantes no SQL.

3.3. *Cross-Site Request Forgery (CSRF)*

Um dos tipos de ataques de CSRF consiste num usuário malicioso faça algumas ações usando as credenciais de outro usuário (vítima) sem o conhecimento ou consentimento desse mesmo.

Com este tipo de ataque conseguimos fazer uma explanação do que é realmente consiste em esta vulnerabilidade, por exemplo. Maicon é um cibercriminoso mal-intencionado que conhece e alguns site específico que permite que usuários conectados enviem quantias para uma conta especificada usando uma solicitação HTTP POST que necessita do nome e da conta o valor ser enviado. Maicon, entretanto, cria um formulário e inseri seus dados bancários e uma quantia qualquer como campos ocultos e dispara os e-mails para outros usuários que estão no site como exemplo (cria o botão Enviar, disfarçado como um link para um site "Você rico rapidamente").

> *CSRF* é um dos ataques mais conhecidos, existe desde a “fundação” da *Web*. Ele ocorre quando uma requisição *HTTP* é feita entre sites na tentativa de se passar por um usuário legítimo. Quem se utiliza desse tipo de ataque normalmente foca em fazê-lo esperando que usuário alvo esteja autenticado no site onde a requisição fraudulenta será realizada, a fim de se ter mais privilégios e acessos a operações. E a razão de todo o problema está em como os navegadores lidam com os *Cookies*. (TEDESCO,2004)

Agora neste exato momento se algum usuário despercebido clicar no botão enviar, será feita uma solicitação HTTP POST onde será enviada ao servidor trazendo maiores detalhes da transação e quaisquer cookies do lado do cliente que o navegador de internet associou ao site (adicionar cookies do site associados a solicitações é um comportamento normal do navegador). O servidor irá verificar os cookies e usá-los para determinar se o usuário estiver ou não conectado e tem permissão para fazer a transação.

Com isso temos o seguinte resultado, é que qualquer usuário desatento ou até mesmo na pressa acaba clicando no botão Enviar que enquanto estiver conectado ao site de negociação fará a transação. O atacante John cada vez mais fica rico.

Uma contramedida para impedir esse tipo de ataque é fazer o servidor exigir que as solicitações de POST adicionem um segredo gerado pelo site específico do usuário. Este tipo de segredo seria fornecido pelo servidor no momento que for enviar o formulário da web usado para fazer transferências. Com este tipo abordagem previne o atacante (Maicon) de criar seu próprio formulário, uma vez que ele precisaria desvendar o segredo que o servidor está fornecendo ao usuário em si. Entretanto se o atacante mesmo assim conseguisse descobrir o segredo e fizesse um formulário para um usuário específico, ele não seria capaz de usar o mesmo formulário para atacar todos os usuários.

Temos frameworks web, que geralmente incluem esses mecanismos de prevenção para CSRF.

Vejamos agora abaixo alguns outros tipos de ataques e vulnerabilidades comuns.

Clickjacking. Com esse ataque, temos um usuário mal-intencionado que rouba cliques destinados a um site de nível superior visível e os manda para uma página oculta abaixo. Esse tipo técnico pode ser aplicado, por exemplo, para expor um site banco verdadeiro, para apanhar as credenciais de login invisível deixando assim controlado pelo invasor. O *clickjacking* também pode ser empregado para fazer com que o usuário seja forçado em clicar em um botão em um site visível, mas, no momento de ele fazer isso, este clique inconscientemente gera um botão completamente diferente. Uma medida defesa para este tipo de ataque, o seu site pode impedir que ele seja incorporado em um frame em outro site, definindo os cabeçalhos HTTP apropriados e seguros.

> O *Clickjacking* é uma classe de vulnerabilidade que afeta a maior parte dos navegadores de internet, como por exemplo: o *Firefox, Chrome*, internet *Explorer*, opera e safari. Através desse tipo de ataque é possível que o invasor controle a *Webcam* e o microfone de suas vítimas. (PMG ACADEMY, 2022)

Negação de Serviço (DoS, em inglês). O DoS geralmente é atingido é enchendo um site de destino com solicitações falsas para que o acesso a um site seja interrompido imediatamente por usuários legítimos. Os requerimentos podem ser simplesmente altamente numerosos, entretanto podem também consumir grandes quantidades de recursos individualmente (por exemplo, leituras lentas ou upload de arquivos grandes). Uma contramedida de DoS normalmente acabam identificando e fazendo bloqueios ao tráfego "ruim" fazendo com que deixam permitir a passagem de mensagens legítimas. Essas defesas frequentemente ficam localizadas antes ou no servidor da web (elas não fazem parte da própria aplicação web).

Directory Traversal (arquivo e divulgação). Vejamos nesse ataque, um atacante mal-intencionado tenta acessar partes do sistema de arquivos do servidor da *web*, porém ele não deve acessar. Esse tipo vulnerabilidade acontece no momento em que o usuário consegue enviar nomes de arquivos que contêm os caracteres de navegação do sistema de arquivos (por exemplo, ../../). A solução para este tipo de ataque, é simplesmente limpar a entrada antes de usá-la.

> Acessando a aplicação pelo browser o atacante pode olhar os links absolutos para arquivos armazenados no Web Server. Manipulando as variáveis que referência os arquivos utilizando "*dot-dot-slash* (../)" utilizando diferentes seqüências e variações podem permitir acesso arbitrário a arquivos e diretórios no file system, incluindo código fonte da aplicação, arquivos críticos e de configuração do sistema operacional entre outros. Utilizando "../" o atacante consegue direcionar a aplicação para diretórios acima do diretório padrão. (ABRAWEB, 2018)

Inclusão de arquivo. Temos também este tipo ataque, que consiste em um usuário que pode especificar um arquivo "não intencional" para exibição ou execução nos dados passados para o servidor. Quando forem carregados, esse arquivo eles podem ser executados no servidor da *web* ou no lado do cliente (levando a um ataque *XSS*). Na maioria das vezes a melhor solução é fazer a limpeza na entrada antes de usá-la.

Segundo *Go-Cache* (2021), usando a inclusão de arquivo remoto (*RFI*), um invasor pode fazer com que o aplicativo da *web* inclua um arquivo remoto. Isso é possível para aplicativos da *web* que incluem dinamicamente arquivos externos ou scripts. As possíveis

consequências para a segurança da *Web* de um ataque *RFI* bem-sucedido vão desde a divulgação de informações confidenciais e *Cross-site Scripting* (*XSS*) até a execução remota de código e, como resultado, o comprometimento total do sistema.

Injeção de comando. Os ataques deste tipo de injeção de comando autorizam que um usuário mal-intencionado execute vários tipos de comandos arbitrários do sistema dentro do sistema operacional host. Uma contramedida eficaz para este ataque deixar a entrada do usuário limpa antes que ela possa ser usada nas chamadas do sistema.

> Esta classe de ataques é o bicho papão de todo programador. Eles são os ataques mais comuns e bem-sucedidos na Internet devido a seus numerosos tipos, grande superfície de ataque e a complexidade às vezes necessária para protegê-los. Todos os aplicativos precisam de dados de algum lugar para funcionar. *Cross-Site Scripting* e *IU Redress* são, em particular, tão comuns que eu dediquei o próximo capítulo a eles e estes são geralmente categorizados separadamente de *Injection Attacks* como sua própria classe, dada a sua importância. (PETTER, 2018)

4. Considerações finais

É muito importante salientar que um desenvolvimento de um código robusto e que funcione não diz respeito sobre a eficácia de sua segurança. Entretanto, podemos se dizer que a notícia é boa e que algumas ferramentas se têm cada vez mais proporcionado a resolver os problemas com o quesito de segurança. E com isso temos implementações em diversos procedimento para obter a proteção contra as principais vulnerabilidades e ataques. Convenhamos em dizer, que fica indispensável e de muita atenção no tocante o uso de boas práticas de desenvolvimento seguro para que a aplicação esteja, se possível, livre de ataques e vulnerabilidades. É de extrema importância que os gestores: desenvolvam uma planejamento claro e objetivo para se garantir segurança no processo de desenvolvimento da aplicação; fornecem um mecanismo para que sua equipe e os fornecedores tenham a capacidade de desenvolver aplicações mais seguras; realizem uma análise e testem a segurança do ambiente de desenvolvimento e das funcionalidades projetadas pela equipe e fornecedores; elaborem um projeto em Gestão de Vulnerabilidades e testes de segurança que tem em vista garantir que todas as aplicações estarão sob nítido controle contra as ataques hackers.

Referências

ALURA. Os 3 pilares do desenvolvimento seguro de aplicações *Web*. 2019. Disponível em: https://www.alura.com.br/artigos/os-3-pilares-do-desenvolvimento-seguro-de-aplicacoes-web. Acesso em: 20 maio 2023.

ANTONIO, Adriano Martins. O que é *Clickjacking* e quais são os seus riscos? 2020. Disponível em: https://www.pmgacademy.com/seguranca-cibernetica/o-que-e-clickjacking/. Acesso em: 21 maio 2023.

BELCIC, Ivan. O que é injeção de *SQL* e como ela funciona? 2020. Disponível em: https://www.avast.com/pt-br/c-sql-injection. Acesso em: 20 maio 2023.

GOCACHE. O que é Inclusão de Arquivo Remoto (*Remote File Inclusion* – RFI)? 2021. Disponível em: https://www.gocache.com.br/seguranca/o-que-e-inclusao-de-arquivo-remoto-remote-file-inclusion-rfi/. Acesso em: 21 maio 2023.

LOPES, Petter. ATAQUES DE INJEÇÃO – *INJECTION ATTACKS*. 2018. Disponível em: https://periciacomputacional.com/ataques-de-injecao-injection-attacks/. Acesso em: 21 maio 2023.

MACHADO, Ivo. Ataques de *Path* Transversal. 2018. Disponível em: https://abraweb.com.br/index.php/artigos/ataques-de-path-transversal. Acesso em: 21 maio 2023.

MOZILLA. Segurança em aplicação *web*. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/First_steps/Website_security. Acesso em: 20 maio 2023.

SCHENDES, William. Ataques cibernéticos no Brasil cresceram 46% no segundo trimestre. 2022. Disponível em: https://olhardigital.com.br/2022/08/09/seguranca/ataques-ciberneticos-brasil-cresce-46/. Acesso em: 20 maio 2023.

TEAM, Communication. Saiba como aumentar a segurança de aplicações *web*. Disponível em: https://blog.convisoappsec.com/saiba-como-aumentar-a-seguranca-de-aplicacoes-web/. Acesso em: 21 maio 2023.

UNIDAS, Nações. Objetivo de Desenvolvimento Sustentável 17 Parcerias e meios de implementação. 2019. Disponível em: https://brasil.un.org/pt-br/sdgs/17. Acesso em: 20 maio 2023.