FACULDADE DE TECNOLOGIA DE AMERICANA - RALPH BIASI

Curso Superior de Tecnologia em Segurança da Informação

Henrique Grana

SEGURANÇA EM APLICAÇÕES WEB

Cross-Site Scripting (XSS)

Americana, SP

2019

FACULDADE DE TECNOLOGIA DE AMERICANA - RALPH BIASI

Curso Superior de Tecnologia em Segurança da Informação

Henrique Grana

SEGURANÇA EM APLICAÇÕES WEB

Cross-Site Scripting (XSS)

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof. Esp. Marcus Vinícius Lahr Giraldi

Área de concentração: Estudo de casos.

Americana, SP

2019

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

G774s GRANA, Henrique

Segurança em aplicações web: cross-site scripting (XSS). / Henrique Grana. – Americana, 2019.

35f.

Monografia (Curso Superior de Tecnologia em Segurança da Informação) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Esp. Marcus Vinícius Lahr Giraldi

1. Segurança em sistemas de informação I. GIRALDI, Marcus Vinícius Lahr II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.518.5

Fatec Americana
Ministro Ralph Biasi

Centro Paula Souza 50 anos 1969 • 2019

SÃO PAULO
GOVERNO DO ESTADO

Faculdade de Tecnologia de Americana

Henrique Grana

SEGURANÇA EM APLICAÇÕES WEB
Cross-site Scripting (XSS)

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana.
Área de concentração: Estudo de casos

Americana, 15 de junho de 2019.

Banca Examinadora:

Marcus Vinícius Lahr Giraldi (Presidente)
Especialista
FATEC Americana

Alberto Martins Júnior (Membro)
Mestre
FATEC Americana

Thaís Godoy Vazquez Macetti (Membro)
Doutora
FATEC Americana

AGRADECIMENTOS

Em primeiro lugar agradeço aos familiares e amigos pelo incentivo que me foi dado para a realização deste trabalho. Agradeço ao Prof. Marcus Vinícius Lahr Giraldi que se propôs a me auxiliar na elaboração deste trabalho.

DEDICATÓRIA

Dedico este trabalho aos meus pais que sempre estiveram me apoiando, desejando o meu melhor e aos meus amigos, que me incentivaram na tarefa de concluir este trabalho.

RESUMO

As aplicações *Web* tem ficado cada vez mais populares no passar dos anos, assim como os seus usuários e atividades diárias tem aumentado. Esse aumento traz grandes vantagens para as empresas, mas também chama atenção para potenciais ataques. Este trabalho tem a intenção de abordar os ataques de *Cross-site scripting* (XSS), fazendo uma análise nos estudos de casos e seus impactos nas organizações e usuários. Esses ataques ocorrem por meio de injeção de código malicioso, podendo afetar um sistema ou abrir portas para ataques maiores. Este trabalho ira abordar o que é uma aplicação *Web*, o que é o ataque XSS e seus diferentes tipos, como o ataque é efetuado, assim como seus impactos, formas de prevenção e mitigação para ataques futuros e análise de estudos de casos em que XSS foi utilizado como método de invasão ou ponte para outros ataques. Todo o conteúdo neste trabalho pode auxiliar os desenvolvedores a mitigar e prevenir futuros ataques a uma aplicação *Web*.

Palavras Chave: *Cross-site scripting*; ataques XSS; injeção de código malicioso.

ABSTRACT

Web applications have become increasingly popular over the years as well as their users and daily activities. This increase brings great benefits to companies, but also draws attention to potential attacks. This work is intended to address cross-site scripting (XSS) attacks, making an analysis of case studies and their impacts on organizations and users. These attacks occur through the injection of malicious code, which can affect a system or open doors to larger attacks. This work will address what is a Web application, what is XSS attacks and it's different types, how the attack is carried out, as well as its impacts, forms of prevention and mitigation for future attacks and analysis of case studies in which XSS was used as a method of invasion or bridge to other attacks. All content in this work can help developers mitigate and prevent attacks on a Web application.

Keywords: *Cross-site scripting; xss attacks; code injection.*

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ASCII	American Standard Code for Information Interchange
DOM	Document Object Model
HTML	HyperText Markup Language
IBGE	Instituto Brasileiro de Geografia e Estatística
URL	Uniform Resource Locator
XSS	Cross-site Scripting

LISTA DE FIGURAS

1 INTRODUÇÃO

Com o avanço da Internet, a diversidade de aplicações *Web* tem aumentado exponencialmente, com isso, os ataques as mesmas aumentaram drasticamente. A segurança em aplicações *Web* tem se aperfeiçoado para evitar esses tipos de ataques e proteger os usuários finais de invasões e perda de dados sensíveis.

Este projeto tem como propósito mostrar os riscos que uma aplicação *Web* pode sofrer se a mesma não estiver bem configurada. Fazendo uma análise sobre uma das principais ameaças à aplicações *Web*, que é o *Cross-Site Scripting* (XSS). Esse tipo de ataque específico tem como objetivo extrair informações sensíveis quando um sistema tem alguma parte comprometida, devido à má configuração ou vulnerabilidades conhecidas em componentes utilizados. Diversos casos foram analisados para obter informações sobre como o ataque ocorreu e o que pode ser utilizado para se evitar novos ataques.

O Objetivo geral é avaliar os riscos que uma aplicação *Web* mal configurada pode sofrer se for alvo de ataques específicos. O Objetivo específico é analisar os riscos de uma invasão via Injeção de Código Malicioso e como tomar as precauções cabíveis.

A realização deste trabalho possibilita a novos desenvolvedores conhecimento para prevenção de ataques já há muito tempo conhecidos na comunidade, mas que por falta de experiência na área, continuam aparecendo em aplicações. A importância de se proteger das vulnerabilidades já conhecidas é essencial, tanto para a comunidade desenvolvedora, tanto para os usuários finais das aplicações.

O método científico utilizado foi de revisão bibliográfica, tendo como base estudos críticos em relação a aplicações *Web* que já sofreram ou que poderão sofrer com ataques desse cunho.

2 REVISÃO DE LITERATURA

Este capítulo tem como objetivo sintetizar diversas abordagens sobre aplicações *Web* e suas utilidades na Tecnologia da Informação, bem como mostrar estatísticas sobre ataques e possíveis vulnerabilidades.

2.1 APLICAÇÕES WEB

Segundo a PC Magazine (2019), uma aplicação é um *software* que processa dados para o usuário, com exceção "*software* de sistemas", que provém a infraestrutura necessária do computador, todos os *softwares* são aplicações. Portanto, uma aplicação *Web* é todo *software* que processa dados e está funcionando sobre a *Internet*.

Nessa lógica, observa-se que há várias vantagens em se ter uma aplicação rodando na *Internet*, disponibilidade, fácil manutenção, compatibilidade com diversos sistemas, etc.

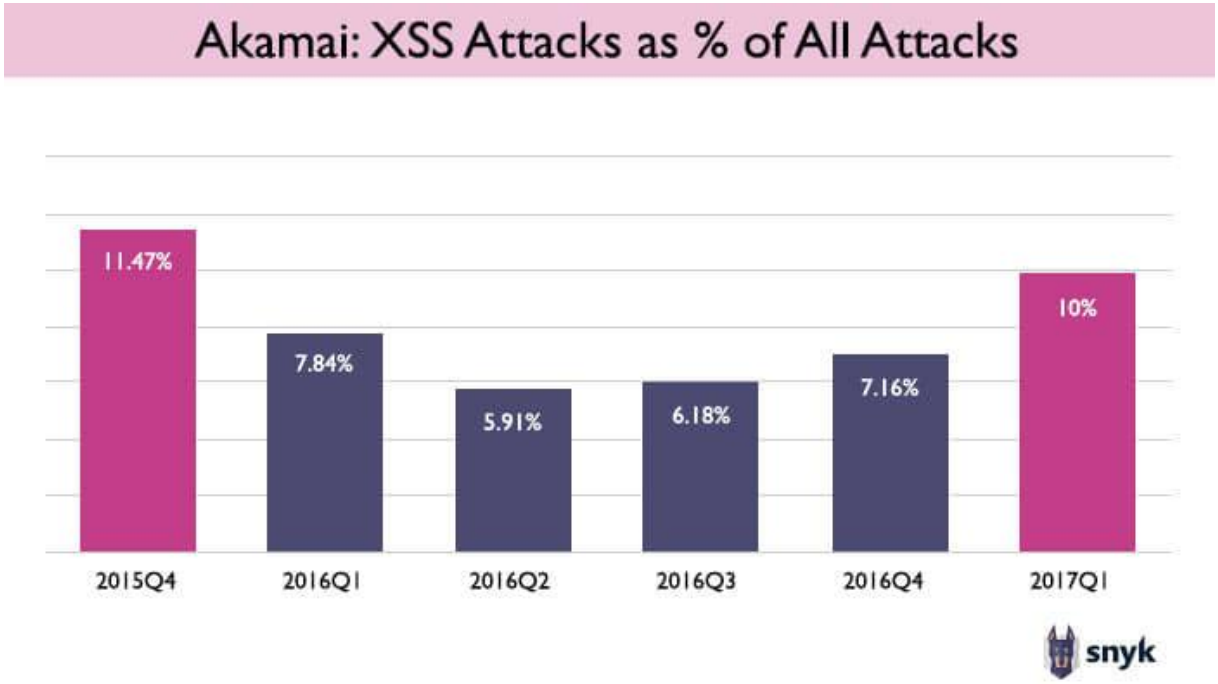
As aplicações *Web* tem suas desvantagens, se uma aplicação pode ser acessada publicamente pela *Internet*, a mesma deverá ser extensivamente testada para conter possíveis ameaças externas, como ataques e negações de serviço.

2.2 ESTATÍSTICAS DE ATAQUES

A *Internet* é hoje uma realidade na vida de maioria dos brasileiros, segundo o IBGE (2016), 64,7% de toda a população tem acesso à Internet, são 116 milhões de pessoas que a utilizam. Seguindo os benefícios de se utilizar uma aplicação *Web*, pode-se associar o aumento de pessoas que usam a *Internet*, com o aumento de aplicações *Web* disponíveis, e consequentemente suas ameaças e vulnerabilidades.

O aumento da dependência da *Internet* nas empresas têm elevado as chances de ataques por invasores. Segundo a Akamai (2017), o artigo aponta uma crescente nos ataques XSS no ano de 2017, anteriormente em queda no começo do ano de 2016 o aumento foi de aproximadamente 39% em relação ao quadrimestre anterior, como pode ser observado na figura 1.

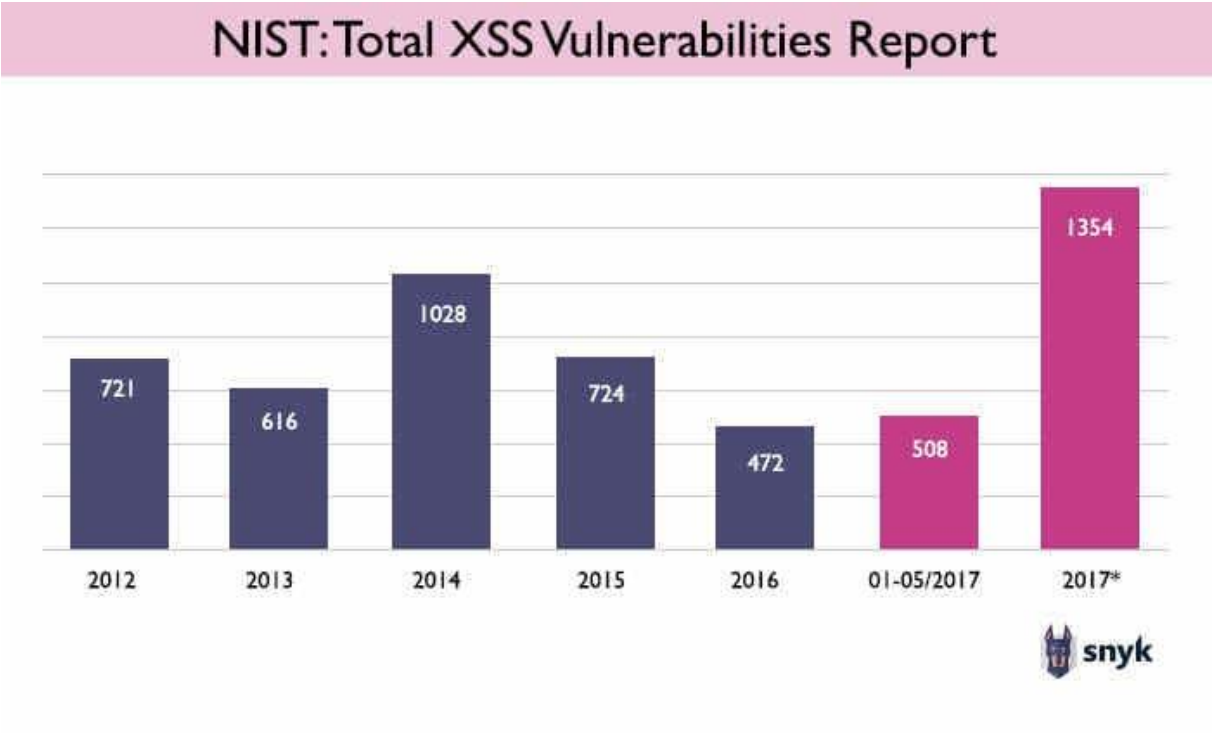
Figura 1: Porcentagem de ataques XSS de todos os ataques a aplicações Web.



Fonte: AKAMAI (2017).

A figura 2 representa o número absoluto de ataques XSS em aplicações *Web*. Vale ressaltar que esses dados não contabilizam todos os tipos de ataques XSS, apenas os que acontecem e se propagam no lado do servidor da aplicação, que são possíveis de detecção. Ataques como XSS tipo 0 não são detectáveis, por isso são muito comuns em aplicações *Web* chamadas de aplicações de página únicas, onde estão cada vez mais ricas de informações e funcionalidades no lado do usuário.

Figura 2:Número total de vulnerabilidades XSS.



Fonte: National Vulnerabilities Database (2017).

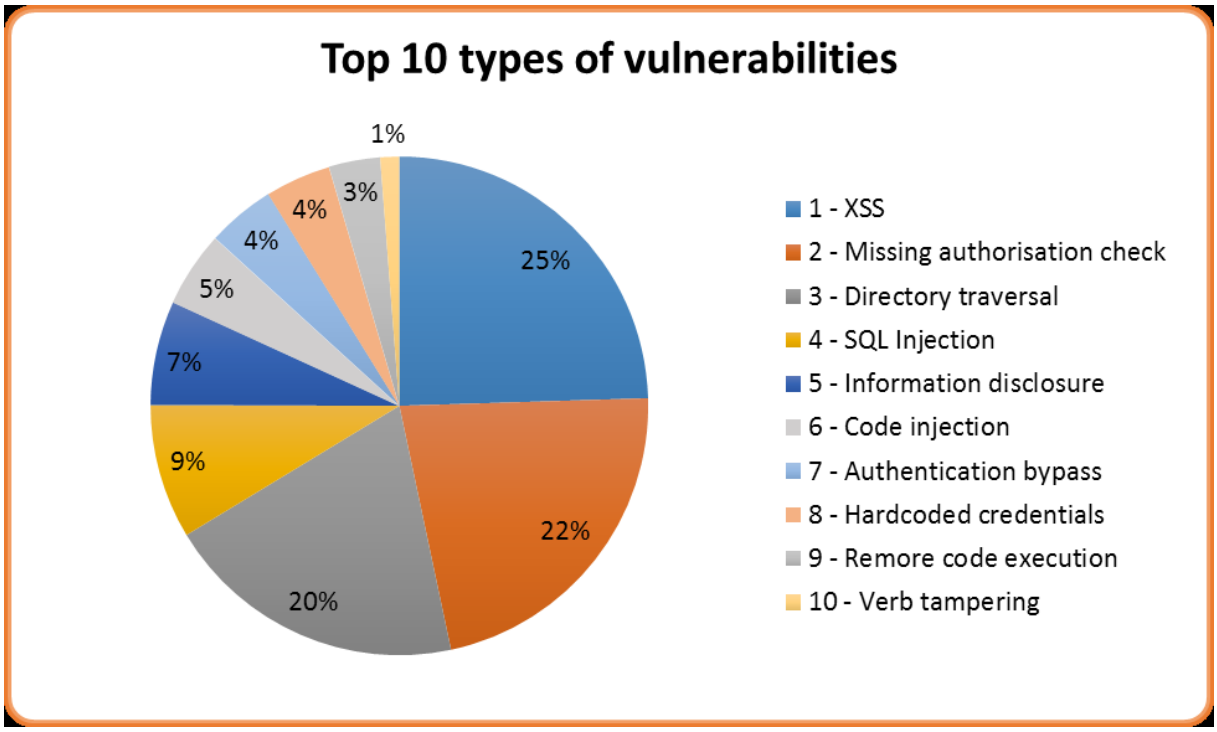
O artigo da ERPScan, *SAP Securtiy in figures* (2013) analisa como foram os ataques e as porcentagens de cada um. De acordo com o artigo, até 2013 foram escritos mais de 2,700 notas de segurança.

Com base no artigo da ERPScan (2013), os fatores primários que influenciaram os ataques desse cunho foram o aumento do número de aplicações *Web* disponíveis, portanto aumentando o número de vulnerabilidades e ameaças. O melhoramento de análise de código estático que mostra as vulnerabilidades que são fáceis de encontrar diminuíram, por outro lado, os números de vulnerabilidades e ameaças complexas e que requerem uma análise mais profunda tem aumentado.

Segundo o artigo da ERPScan (2013), o aumento de ataques XSS era previsível sendo que o aumento de aplicações *Web* tem crescido exponencialmente. O aumento de XSS deve-se ao fato de seu alto nível de impacto e por que qualquer vulnerabilidade de injeção será mais fácil de detectar com analisadores de códigos aprimorados.

Observando a figura 3, podemos analisar que aproximadamente 20% dos ataques foram específicos do *ERP* (Sistema Integrado de Gestão Empresarial) *SAP*, porém, pode-se notar que o líder de notas publicadas é o *Cross-Site Scripting* (XSS), segundo o artigo da ERPScan (2013).

Figura 3:Vulnerabilidades SAP separado por tipos.



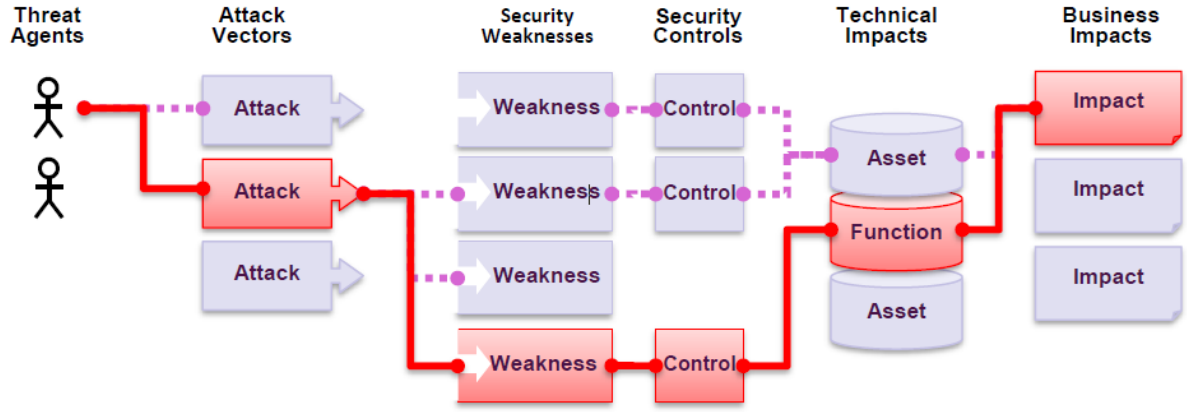
Fonte: ERPScan (2013)

2.3 RISCOS EM APLICAÇÕES WEB

Um atacante em potencial pode escolher diversos métodos para atacar uma aplicação *Web*. Dependendo do método utilizado pelo atacante, pode ser que seja necessário gerar atenção para tal, segundo a OWASP (2017).

A figura 4 representa um possível método para um atacante. O atacante usa vetores de ataques conhecidos, se aproveitando de vulnerabilidades do sistema consegue acesso indevido a aplicação e de modo geral causa impactos no sistema ou na organização.

Figura 4: Passos para um possível atacante.



Fonte: OWASP (2017)

Algumas vezes os métodos utilizados para uma vulnerabilidade são fáceis de se encontrar em uma aplicação, porém, o contrário pode ser válido. O mesmo raciocínio deve ser aplicado às consequências dessa invasão, uma vulnerabilidade no sistema pode colocar toda a empresa fora do ar e fora do mercado.

Pode-se calcular o risco em sua organização levando em conta todos os vetores de ataques possíveis, suas fraquezas na aplicação e os agentes alvo, combinando tudo com uma estimativa de impacto técnico e organizacional. Todos esses itens determinam seu risco total, segundo a OWASP (2017).

3 ATAQUES A APLICAÇÕES *WEB*

Os ataques a aplicações *Web* podem vir em diferentes formatos e estratégias, segundo a fundação OWASP (2018). Um ataque são técnicas que invasores usam para explorar vulnerabilidades em uma aplicação. Um ataque não deve ser confundido com uma vulnerabilidade, pois a vulnerabilidade é uma fraqueza de uma aplicação e não algo que um atacante faria.

3.1 *CROSS-SITE SCRIPTING* (XSS)

Segundo a OWASP (2018), *cross-site scripting* (XSS) é um tipo de ataque de injeção, que no caso códigos maliciosos são injetados em outrora sites confiáveis. Vulnerabilidades que permitem esses ataques acontecerem são populares e ocorrem sempre que uma aplicação não valida e/ou criptografa um *input* de dados do usuário para o sistema.

Um atacante pode usar XSS para enviar códigos maliciosos para um usuário desavisado. O navegador do usuário final não tem como saber se o código é confiável e o executa. Dessa maneira o código malicioso poderá ter acesso a quaisquer *cookies, tokens* ou dados sensíveis do usuário que são armazenados pelo navegador afetado. Em alguns casos, pode ser que o usuário tenha o *Document Object Model* (*DOM*) alterado, comprometendo a usabilidade do site e aumentando as chances de invasão.

3.1.1 XSS BASEADO EM DOM (TIPO 0)

XSS baseado em DOM ocorre, segundo a OWASP (2018), quando todo o ataque é feito pela parte de interação do usuário com a aplicação *Web*, ou seja, a fonte do código malicioso está no DOM, os gatilhos para o ataque acontecer também

estão no DOM e o fluxo de dados nunca sai do navegador. Com isso, nenhuma ferramenta de detecção será ativada e o usuário será comprometido.

3.1.2 XSS ARMAZENADO OU PERSISTENTE (TIPO 1)

XSS Armazenado ocorre quando o código malicioso está armazenado no servidor do *site*, nos bancos de dados, *logs*, mensagens, campo de comentários, etc. O atacante consegue obter as informações antes que o site faça esses campos seguros. Com novas técnicas de *Hyper Text Markup Language* (HTML) as informações podem ser armazenadas no navegador do usuário, sem a necessidade de ser enviadas para o servidor, o que pode ocasionar em invasões e roubo de dados, segundo OWASP (2018).

3.1.3 XSS REFLETIDO OU NÃO-PERSISTENTE (TIPO 2)

De acordo com OWASP (2018), XSS Refletido ocorre quando o código malicioso é armazenado fora do servidor, como em uma mensagem de erro, um resultado de busca ou qualquer resposta que inclui algum tipo de dado que foi enviado para o servidor. Ataques refletidos chegam a vítima de outras formas, como um e-mail ou outro site. O usuário pode clicar acidentalmente ou ser induzido a clicar em um link malicioso, preencher um formulário suspeito, etc. Assim, o código malicioso atinge o site alvo.

3.1.4 TÉCNICA DE CODIFICAÇÃO DE CARACTERES

Os ataques XSS de tipo 1 e 2 são decorrentes de uma falha na técnica (ou abstenção da mesma) de escape, ou codificação de caracteres. Essa falha permite

que códigos mal intencionados quebrem as *tags* onde elas residem e execute sem permissão.

A técnica consiste em substituir uma gama de caracteres por um código relacionado à *American Standard Code for Information Interchange (ASCII),* após trocados, esses caracteres serão interpretados pelo navegador, impedindo que o código malicioso possa quebrar a página. Nota-se que o navegador deve saber quais codificações serão usadas na página HTML, essa informação é passada através de uma *tag* HTML de tipo *meta.* Um exemplo de *tag* pode ser observado na figura 5.

Figura 5: Tag HTML para declaração do tipo de codificação.

Fonte: OWASP (2019).

4 OS REAIS IMPACTOS DO *CROSS-SITE SCRIPTING*

Cross-site scripting é provavelmente um dos riscos mais prevalentes nas aplicações *Web* e mesmo assim é um dos mais negligenciados. Os principais riscos estão associados ao fato que o usuário é diretamente afetado pela invasão, sendo ele o principal vetor de ataque, segundo a Dionach (2016).

A maioria dos casos que ocorrem XSS são para tentativa de algo maior, como sequestro de sessão, roubo de credenciais e acesso a dados sensíveis privados. Na sequência do capítulo, será abordado alguns exemplos de como isso é possível.

4.1 SEQUESTRO DE CONTAS

Um dos principais vetores de ataque são os sequestros de contas legítimas, normalmente o atacante consegue acesso roubando o *cookie* de sessão do usuário. Esse tipo de ataque faz com que o atacante tenha total acesso a tudo o que esse usuário tem: dados privados e funcionalidades passíveis apenas para o usuário. Esse tipo de ataque pode ser conduzido colocando *scripts* em campos vulneráveis da aplicação *Web* (XSS tipo 1), como pode ser observado na figura 6.

Figura 6: Exemplo de script à ser colocado em um campo vulnerável.

```
<script>
  image = new Image();
  image.src='http://[Attacker IP]:8080/?'+document.cookie;
</script>
```

Fonte: Dionach (2016).

Quando a vítima acessar a página infectada pelo código malicioso, o navegador irá fazer uma requisição contendo o identificador de sessão para o servidor do atacante. No servidor essa informação é recuperada e guardada de acordo com as

intenções do atacante. Dessa maneira, a sessão fica sequestrada e a vítima não irá perceber, segundo a Dionach (2016).

Os riscos desse vetor de ataque são absurdos. Um atacante que tenha acesso a uma conta de administrador pode roubar dados sensíveis e causar danos à aplicação *Web*.

4.2 ROUBO DE CREDENCIAIS

Outro vetor de ataque comum é usar HTML e *JavaScript* para roubar as informações de acesso do usuário a aplicação *Web*. Esse ataque pode ser efetuado realizando uma parte de engenharia social com uma cópia falsa da página de *login* da aplicação *Web* juntamente com uma vulnerabilidade XSS para ser enviado as vítimas. Assim como no caso do sequestro de contas, a vítima entra em uma página onde o código malicioso é executado mostrando uma falsa tela de *login*. Se o usuário tentar fazer o *login*, as informações de acesso serão enviadas ao servidor do atacante, onde ele poderá usar essas informações para ter acesso a conta da vítima, segundo Dionach (2016).

Essa tática é a mais benéfica para o atacante, sendo possível obter dados de acesso em texto puro ao invés de um *cookie* de sessão. Um vetor de ataque extra inclui utilizar engenharia social para persuadir uma vítima a colocar dados de acesso em formulários falsos, que geralmente rouba as credenciais da vítima.

4.3 IMPACTO NAS ORGANIZAÇÕES

Segundo a Acunetix (2014), os danos causados por XSS nas organizações podem ser diversos, fora os danos causados pelo ataque em si e por outros que usam o XSS como base para ataques ainda mais devastadores. Um ataque bem sucedido pode impactar os usuários e os funcionários de uma empresa, assim como a sua

capacidade de gerar renda ou produção. Mesmo se uma empresa nunca sofreu com ataques XSS, se as vulnerabilidades estão presentes na aplicação *Web*, isso já pode ser o suficiente para impactar as relações públicas da empresa, ainda mais se essas vulnerabilidades são publicadas em portais de segurança da informação ou em canais de notícias.

4.3.1 OS CUSTOS PARA RESPONDER A UM ATAQUE

Segundo o artigo da Ponemon Institute (2014), o tempo médio entre detecção e resolução de um ataque XSS está em torno de 32 dias, com um custo médio de 32.469 dólares. Levando-se em conta todas as empresas analisadas, o preço tende a ser maior conforme o tamanho da empresa, porém, empresas pequenas não se livram dos custos de um ataque.

As empresas onde o seu faturamento advém da *Internet* tendem a ser as mais afetadas. Um ataque XSS bem-sucedido em um funcionário da empresa pode causar enormes danos, como: impossibilitar o acesso, infectar as estações de trabalho dos empregados e no pior dos casos, roubar dados sigilosos. As empresas que sofrem com ataques XSS tem que resolver outro problema, tempo de inatividade da aplicação *Web*. O tempo de inatividade pode vir de ter que consertar as brechas para ataques, negação de serviço ou a soma dos dois fatores. Se a aplicação tem uma vulnerabilidade, além dos procedimentos de recuperação de controle, alguns passos terão que ser analisados:

- Investigar o código responsável por tal brecha;
- Consertar o código e fazer um *patch* de segurança;
- Testar e implementar o *patch* para a versão de produção.

Em conclusão, um ataque bem-sucedido pode causar um impacto devastador em uma empresa, visto que do momento de sua detecção até a implantação de um *patch* para resolver a brecha, a empresa está no prejuízo.

4.3.2 RESPOSTA A INCIDENTES E CONTINUIDADE DE NEGÓCIOS

Para entender como as respostas a incidentes nas organizações são realizadas, é necessário entender como o ciclo de resposta a incidentes ocorre e como as empresas podem sofrer se a resposta for muito tardia. Premeditar os riscos que uma empresa pode sofrer requer conhecimento do modelo da empresa, processos operacionais e vulnerabilidades, segundo o artigo feito pela Deloitte (2016).

A triagem do incidente é uma das partes mais cruciais da resposta ao ataque. Essa fase incluí descobrir onde a falha XSS acontece, como acontece e o que fazer para impedir que novos ataques aconteçam. Uma revisão do que foi afetado é necessário, pois se o ataque afetou os funcionários do sistema, uma análise profunda será obrigatória, segundo Deloitte (2016).

O gerenciamento dos impactos é a etapa onde se desenvolve os esforços para conter possíveis novos ataques. Se uma parte do sistema foi atacada e ainda não se resolveu o problema, essas partes devem ser trocadas por partes provisórias. A remoção de partes do sistema tende a ser complicadas para ataques XSS, pois a parte afetada é diretamente ligada a interface de usuário e possíveis interações.

A recuperação do negócio pode demorar anos, a relação pública da empresa pode ser afetada, assim como a confiança de novos clientes. Atualizar a aplicação *Web* para impedir novos ataques pode demorar meses. Entre as fases de desenvolvimento, testes e implantação pode acontecer de haver mudanças na estrutura do projeto atrasando a liberação de atualizações. Todas essas etapas são necessárias para que a empresa possa se recuperar do ataque e continuar a operar, segundo Deloitte (2016).

5 ESTUDOS DE CASO

Crimes cibernéticos são todas as atividades criminosas em que um computador ou rede de computadores é utilizado como meio para se obter dados ou informações sobre a vítima, sendo a vítima pessoa física ou jurídica.

Nesse capítulo, serão abordados diversos casos onde os atacantes usaram XSS para fazer a invasão ou roubar dados da aplicação.

5.1 “SAMY”, O VÍRUS MAIS RÁPIDO DA HISTÓRIA

Em 2005, um rapaz de 19 anos de idade lançou o vírus de computador conhecido como “Samy”. Talvez o vírus de computador mais rápido da história que mudou a segurança na *Internet* para sempre. (MOTHERBOARD TECH BY VICE, 2015).

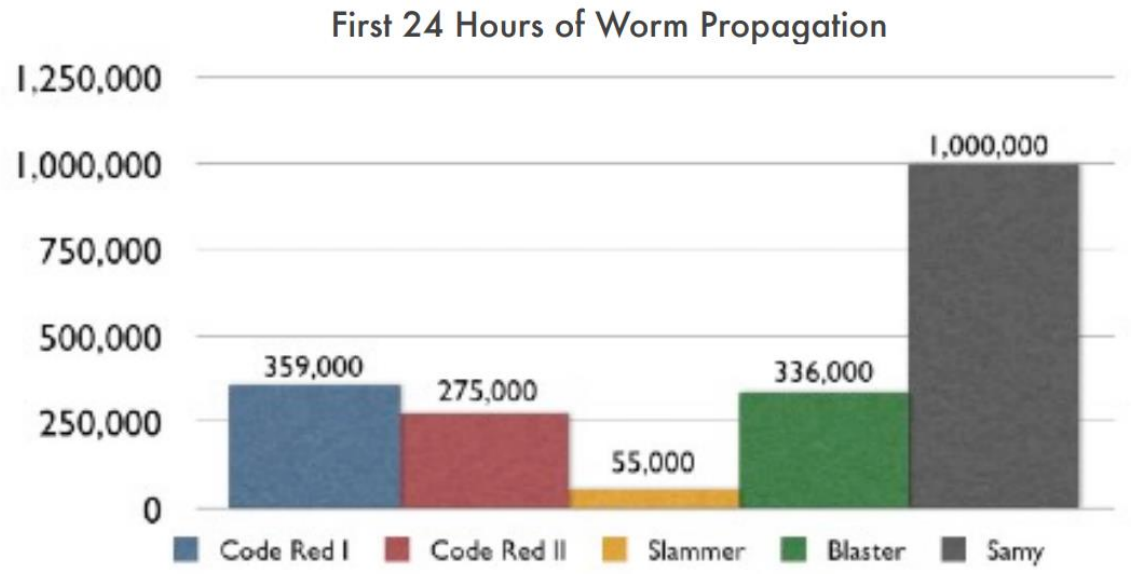
Na época, o MySpace permitia a utilização de código HTML para customização de perfil do usuário, isso ocasionou uma falha grave na segurança do sistema permitindo que um usuário final, com algumas linhas de código *JavaScript*, fizesse mudanças não esperadas na plataforma. Apesar do vírus se espalhar rapidamente, ele era inofensivo. O maior impacto do vírus foi mostrar como os sites na época estavam mal configurados e expostos a vulnerabilidades que as próprias empresas não tinham conhecimento.

Na época, aproximadamente 80-90% dos sites poderiam sofrer com algum tipo de ataque XSS de acordo com a OWASP (2015). Foram criados protocolos que poderiam ser seguidos para mitigar tais riscos e minimizar tais vulnerabilidades. Após 10 anos, aproximadamente 47% dos sites podem sofrer com ataques XSS, de acordo com o WhiteHat's Security (2015).

Anos antes, outros vírus foram escritos utilizando falhas de outros serviços, como “*Blaster*” em 2003, que constituía em infectar o sistema operacional *Windows*

da Microsoft. Comparando sua eficácia com outros vírus nas primeiras 24 horas, podemos analisar na figura 7 que um vírus que se espalha pela internet é muito mais efetivo e rápido do que um vírus tradicional e operacional.

Figura 7: Primeiras 24 horas de propagação de um vírus.



Fonte: WhiteHat Security (2011).

5.2 ANÁLISE DE VULNERABILIDADES XSS, AKAMAI (2016)

Diversos ataques foram analisados pela empresa Akamai, em seu artigo *An Analysis of* XSS *Exploitation through Remote Resource Injection* (2016). Nesse documento, foi analisado que apenas 2% dos códigos injetados são maliciosos ou ilegítimos sendo os 3 maiores casos como injeção de *advertising* ilegítima, *frameworks* de XSS para vulnerabilidades e mineração de cripto-moedas.

Comumente são usados encurtadores de *Uniform Resource Locator* (URL), que fazem com que seja difícil a localização desses arquivos maliciosos. Dessa maneira, os atacantes usam encurtadores confiáveis no caminho de seus arquivos maliciosos, assim dificultando que um buscador ou indexador coloque todo o domínio em uma lista negra, segundo o artigo da Akamai (2016).

Continuando a análise do artigo, foi relatado que normalmente os *scripts* chamam outros *scripts* após sua execução, dificultando mais ainda que um servidor detecte que está infectado. Em todos os casos, os códigos estavam camuflados devido a uma técnica de escape hexadecimal do *JavaScript*, essa técnica permite que se escreva trechos de códigos complexos e execute depois que houve o carregamento da página, tendo assim eficácia contra firewalls dos navegadores.

A conclusão desse artigo revela algo que se sabe há muito tempo, ataques XSS ocorrem há vários anos e eles só vem aumentando. Há várias maneiras de mitigar riscos, mas devido a inexperiência ou a prazos curtos os desenvolvedores lançam aplicativos inacabados ou não testados contra falhas e invasões.

5.3 ATAQUES AO EBAY USANDO XSS PERSISTENTE (TIPO 1)

Em 2017, diversos casos de invasões foram registrados pelo *site* Ebay*,* esses ataques foram XSS persistente (tipo 1), que é quando o código malicioso se encontra alojado no *site* que se navega. Todos os ataques foram possíveis pois na época o Ebay permitia o uso de *tags* HTML para personalização de descrições nos anúncios, do mesmo jeito que o *worm* “Sammy” funcionava a vítima entra na página de anúncio infectada e o código malicioso é executado. Ainda nesse esquema, os atacantes se espalharam em contas com reputações boas e infectaram anúncios com o código, segundo o artigo da Netcraft (2017).

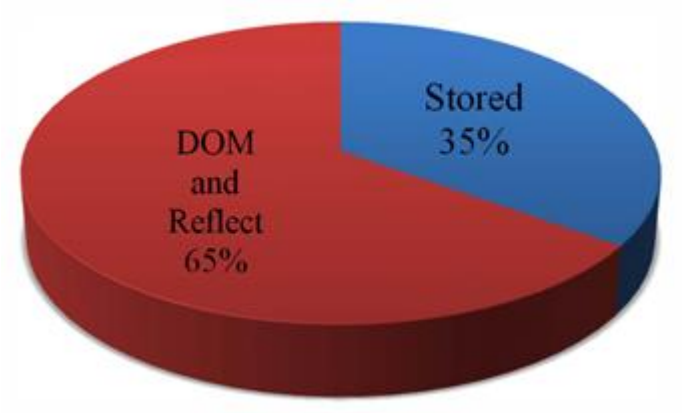
Os ataques aconteciam de maneira metódica, um anúncio com código malicioso injetado era colocado no site e depois de um curto período de tempo, vários usuários já tinham acessado tal produto. Com o acesso ao anúncio, o usuário era redirecionado a uma página falsa de *login* do *site* Ebay e com isso passava as informações de acesso ao atacante. O atacante com os dados de acesso da vítima, conseguia propagar ainda mais o código malicioso se o usuário fosse alguém com uma boa reputação no *site*. Desse modo, modificaria os anúncios que já existiam e mais pessoas poderiam entrar nas páginas e executar o código malicioso, segundo o artigo da Netcraft *(2017)*.

5.4 AVALIAÇÃO DE VULNERABILIDADES DE APLICAÇÕES *WEB* DE BANGLADESH, ESTUDO DE CASO DE XSS

Em ataques coordenados ao Banco de Bangladesh, atacantes conseguiram inserir remotamente, códigos maliciosos nos servidores do banco. Os invasores usaram técnicas já conhecidas de invasão para ganharem acesso e alojarem os códigos. Previamente, a atenção que o governo e os desenvolvedores davam as vulnerabilidades eram mínimas antes dos ataques ocorrerem, segundo o artigo.

De acordo com o estudo, quase 30% das aplicações *Web* são vulneráveis a ataques XSS. Por esses testes demorarem para serem executados, foram analisados um grupo de 500 aplicações, sendo que em 335 delas foram encontradas vulnerabilidades. Nenhuma aplicação *Web* é afetada pelos 3 tipos de ataques XSS, sendo que 35% das vulnerabilidades encontradas são do tipo 1 e 65% são do tipo 0 e 2, segundo a figura 8.

Figura 8: Porcentagem dos tipos de vulnerabilidades XSS



Fonte: Farah (2016).

Em algumas das 335 aplicações *Web*, foram necessárias múltiplas formas de ataques, sendo que algumas precisaram de ataques CSRF *(Cross-Site Request Forgery)* e após o sucesso desse primeiro ataque o ataque XSS pôde ser executado.

Nenhuma das 335 aplicações tinham *firewalls* contra esses tipos de ataques, segundo o artigo.

A conclusão do artigo é clara, os ataques recentes indicam que não ocorreram apenas uma vez, mas sim que talvez tenha sido parte de uma sequência de ataques que apareceram em destaque. Se as autoridades e os desenvolvedores tivessem levado em consideração as instruções de *sites* como OWASP*,* talvez esse ataque milionário tivesse sido evitado. Os níveis de segurança nas aplicações *Web* testadas foram consideradas medíocres, inexistentes ou obsoletas, nenhuma aplicação continha *firewalls* para proteção contra ataques XSS e CSRF. Essas condições indicam uma tendência das empresas e dos governos a esconderem informações de ataques e os desenvolvedores e usuários deveriam ser avisados de que as aplicações contêm vulnerabilidades e com isso, ajudar na prevenção e mitigação de ataques, segundo o artigo.

6 MITIGAR RISCOS E PREVENÇÃO

Todos os casos analisados sofreram danos que poderiam ter sido evitados se a gerência e os desenvolvedores tivessem seguido os modelos de prevenção a ataques XSS. A OWASP provém inúmeras listas de prevenção a aplicações *Web,* uma delas sendo contra ataques XSS e a todas as suas variações.

Problemas com orçamento podem ser prejudiciais contra a prevenção de ataques cibernéticos, principalmente em empresas de pequeno e médio porte. Segundo a pesquisa feita pela Ernest & Young (2016) que conta com mais de 1.735 executivos da área de Tecnologia da Informação de todo o mundo, diz que 89% das empresas não estão preparadas para receber um ataque e suas funções de segurança da informação não são totalmente competentes.

Segundo a pesquisa, é um pedido comum dos executivos de TI um aumento no orçamento anual, porém, 39% das empresas entrevistadas afirmam que só iriam aumentar o orçamento anual se houvesse um ataque significativo ou se dados fossem roubados das empresas e que precisariam de um mínimo de 25% de aumento nos orçamentos.

6.1 GUIA DE PROTEÇÃO CONTRA ATAQUES XSS TIPO 1 E 2 *(OWASP PREVENTION SHEET)*

O artigo da *OWASP (2019)* traz diversas abordagens de como proteger sua aplicação *Web* contra dados não confiáveis, desse modo, todo e qualquer tipo de dado que um usuário final coloca no sistema são dados não confiáveis.

Tratando a página *Web* como um modelo, há diversos campos onde serão inseridos os dados não confiáveis, assim, é necessário ter diversas precauções para que o dado que for inserido não quebre para outras *tags* e ser executado (no caso de um ataque XSS). Nesse modo, as páginas HTML são tratadas como um arquivo

parametrizado, onde todos os dados são mantidos em locais específicos e isolados através da técnica de escape.

O artigo propõe algumas regras a se seguir, a regra 0 é de que nunca se deve colocar dados não confiáveis na aplicação, exceto em locais permitidos. Não colocar dados inconfiáveis na página *Web* ao menos que seja nos campos definidos nas regras de 1 a 5. O artigo aponta que colocar dados inconfiáveis em qualquer lugar se torna insustentável de manter uma lista de escape e consequentemente os riscos se tornam absurdos. Portanto deve-se pensar que, mesmo que o desenvolvedor opte em colocar dados não confiáveis em locais que não sejam das regras 1 a 5, é necessário fazer diversos *cross-browser tests* e mesmo assim não estaria garantido. Sendo assim, a regra 0 aponta esses campos como impróprios para uso:

Figura 9: Tag de script

```
<script>...NEVER PUT UNTRUSTED DATA HERE...</script>
```

Fonte: OWASP (2019)

A figura 9 representa uma *tag* de *script, ela é* destinada para execução de *JavaScript* no navegador. Se um trecho de código mal-intencionado é colocado nessa *tag* sem a codificação necessária, o código vai ser executado e sua aplicação estará exposta.

Figura 10: Chave de comentário para HTML

```
<!--...NEVER PUT UNTRUSTED DATA HERE...-->
```

Fonte: OWASP (2019)

Um atacante pode avaliar como ele vai escrever o código mal-intencionado. Se o atacante souber que dados são escritos em campos de comentários, como pode ser exemplificado na figura 10, ele pode escrever o código dele para quebrar essa *tag* e dessa maneira, executar o *script*.

Figura 11: Como um atributo para uma tag

```
<div ...NEVER PUT UNTRUSTED DATA HERE...=test />
```

Fonte: OWASP (2019)

Colocar atributos personalizados em uma *tag* HTML é comum em desenvolvimento *Web*. A figura 11 exemplifica uma customização de atributos, porém é necessário atenção se esse atributo advém de dados não confiáveis. Um atacante pode inserir uma quebra de *tag* e colocar uma ação para redirecionar o usuário final a uma outra página, por exemplo.

Figura 12: Exemplo de uma tag customizada

```
<NEVER PUT UNTRUSTED DATA HERE... href="/test" />
```

Fonte: OWASP (2019)

Assim como o exemplo anterior, a figura 12 exemplifica que é necessário atenção de como a *tag* é gerada e como se está fazendo a inserção dos dados não confiáveis. Uma inserção livre de *tags* pode ser catastrófica.

Figura 13: Tag de estilo para HTML

```
<style>
...NEVER PUT UNTRUSTED DATA HERE...
</style>
```

Fonte: OWASP (2019)

Muito comum em aplicações *Web* voltada a comércio eletrônico e redes sociais, a figura 13 mostra a customização da interface de usuário pelo próprio usuário , que é algo que se deve levar em consideração. Assim como já foi falado nos estudos de caso, o *worm* “Sammy” foi propagado dessa maneira, um ataque XSS persistente que tinha como objetivo se espalhar o mais rápido possível. Com algumas linhas de

código ele quebrou a *tag* que o continha e executava um *script* que fazia comunicação com a aplicação *Web*.

6.2 PROTEÇÃO CONTRA ATAQUES XSS TIPO 0 (OWASP *DOM BASED* XSS *PREVENTION SHEET*)

Para proteger uma aplicação *Web* contra ataques DOM XSS, é necessário entender as diferenças de como os ataques ocorrem. Como dito nos capítulos anteriores, os ataques tipo 0 são injeções feitas no lado do cliente, ou seja, no navegador do usuário ao contrário dos outros tipos. O ataque ocorre quando o navegador está processando o conteúdo da aplicação. Quando o navegador está processando a página da aplicação *Web*, o navegador identifica que serão necessários diferentes contextos onde cada parte de código será visualizado e assim cada contexto está ligado a interpretação do HTML, das *tags* e seus atributos, segundo o artigo da OWASP (2019).

Segundo o artigo da OWASP (2019), com as *tags* certas e o fechamento de contexto correto, um código *JavaScript* pode acessar diferentes contextos dentro da página *Web*. A figura 14 mostra o uso incorreto de codificação dentro de um contexto *JavaScript* e em um sub contexto HTML:

Figura 14: Má codificação em contextos JavaScript

```
<script>
var x = '<%= taintedVar %>';
var d = document.createElement('div');
d.innerHTML = x;
document.body.appendChild(d);
</script>
```

Fonte: OWASP (2019)

Na figura 14, os erros são apresentados como a falta de codificação, tanto na parte da configuração da variável (segunda linha), quanto na parte da inserção do

código na página *Web*, que permite com que esse *script* seja alterado por um ataque DOM XSS.

6.3 UTILIZAÇÃO DE BIBLIOTECAS DE CODIFICAÇÃO

Muitas das linguagens no mercado já contam com codificadores embutidos, porém, é necessário que esse codificador esteja preparado para as adversidades das aplicações *Web*, com isso a OWASP aconselha que uma boa biblioteca de codificação pode facilitar o desenvolvimento e garantir que não haja falhas nas codificações dos dados não confiáveis.

A Microsoft disponibiliza uma biblioteca de codificação para sua plataforma .NET e sua ferramenta ASP.NET já conta com uma validação de requisições embutida. O projeto OWASP Java Encoder disponibiliza aos desenvolvedores uma biblioteca de alta performance para garantir que as aplicações sejam mais seguras.

7 CONCLUSÃO

A Internet hoje é muito importante para o cotidiano das pessoas. A todo momento se está utilizando a Internet e consequentemente aplicações *Web*. Com essa grande utilização de aplicações, novas possibilidades ficam disponíveis para os atacantes.

Analisando os estudos de caso e as estatísticas de como os ataques XSS são evidentes nas aplicações *Web*, fica claro que falta muito esforço e conhecimento dos desenvolvedores e dos produtos de aplicações. Em muitos casos, a inexperiência somada a falta de tempo para a entrega do produto final, deixa falhas e produz vetores de ataque que possibilitam um atacante invadir uma aplicação *Web*.

O intuito desse trabalho foi analisar os casos e mostrar os danos causados por falhas de segurança da informação nas aplicações *Web* e nas empresas. Na maioria dos casos, os ataques XSS são uma forma de penetração nas aplicações, não o objetivo em si. Por isso é de extrema importância fazer a prevenção desse tipo de vulnerabilidade.

Os ataques XSS podem ser uma ponte para roubo de dados sensíveis, negação de serviço, sequestro de sessão, roubo de credenciais e consequentemente outros ataques.

REFERÊNCIAS BIBLIOGRÁFICAS

AKAMAI. **An analysis of xss exploitation through remote resource injection.** 2016. Disponível em: <https://www.akamai.com/es/es/multimedia/documents/state-of-the-internet/analysis-xss-exploitation-threat-advisory.pdf>. Acesso em: 20 abr 2019.

AKAMAI. **State of the internet** security report. 2017. Disponível em: <https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2017-state-of-the-internet-security-report.pdf>. Acesso em: 20 abr 2019.

DELOITTE. **Beneath the surface of a cyberattack** a deeper look at business impacts. 2016. Disponível em: <https://www2.deloitte.com/content/dam/Deloitte/us/Documents/risk/us-risk-beneath-the-surface-of-a-cyber-attack.pdf> Acesso em: 15 abr 2019.

DIONACH. **The real impact of cross-site scripting.** 2016. Disponível em: <https://www.dionach.com/blog/the-real-impact-of-cross-site-scripting/> Acesso em: 14 maio 2019.

ERNEST & YOUNG. **The path to cyber resilience:** sense, resist, react. 2016. Disponível em: <https://www.ey.com/Publication/vwLUAssets/ey-19th-global-information-security-survey-2016%E2%80%9317-power-and-utilities-sector-results/$FILE/ey-19th-global-information-security-survey-2016%E2%80%9317-power-and-utilities-sector-results.pdf> Acesso em: 15 abr 2019.

ERNEST & YOUNG. **Get ahead of cybercrime** global information security survey. 2014. Diponível em: <https://www.ey.com/Publication/vwLUAssets/EY-global-information-security-survey-2014/$FILE/EY-global-information-security-survey-2014.pdf> Acesso em: 15 abr 2019.

FARAH, Tanjila. **Assessment of vulnerabilities of web applications of Bangladesh**: a case study of XSS & CSRF. Dhaka: North South University, 2016.

IBGE. **Acesso à Internet e à televisão e posse de telefone móvel celular para uso pessoal.** 2016. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv101543.pdf>. Acesso em: 10 abr 2019.

MOTHERBOARD TECH BY VICE. T**he Myspace worm that changed the internet forever.** 2015. Disponível em: < https://www.vice.com/en_us/article/wnjwb4/the-myspace-worm-that-changed-the-internet-forever> Acesso em: 16 mar 2019.

OWASP. **Types of cross-site scripting**, 2017. Disponível em: <https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting/> Acesso em: 10 abr 2019.

OWASP. **Cross-site scripting (XSS)**, 2018. Disponível em: <https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)/> Acesso em: 10 abr 2019.

OWASP. **Cheat sheet series.** 2019. Disponível em: <https://github.com/OWASP/CheatSheetSeries>. Acesso em: 10 maio 2019.

OWASP. **Segurança na web**: uma janela de oportunidades. 2011. Disponível em: <https://www.owasp.org/images/1/16/Seguranca_na_web_-_uma_janela_de_oportunidades.pdf>. Acesso em: 19 maio 2019.

OWASP. **Code review project.** 2017. Disponível em: <https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project> Acesso em: 28 abr 2019.

PC MAGAZINE. **Encyclopedia:** application program, 2019. Disponível em: <https://www.pcmag.com/encyclopedia/term/37919/application-program/> Acesso em: 10 abr 2019.

PONEMON INSTITUTE. **Global report on the cost of cybercrime.** 2014. Disponível em: <https://www.octree.co.uk/Documents/2014-Global-Report-on-the-Cost-of-Cybercrime.pdf>. Acesso em: 16 maio 2019.

WHITEHAT SECURITY. **Website security statistics report.** 2015. Disponível em: <https://info.whitehatsec.com/rs/whitehatsecurity/images/2015-Stats-Report.pdf> Acesso em: 16 mar 2019.