

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Bruno Franco Oliveira

Vulnerabilidades em Sistemas Web

Uberlândia, Brasil

2019

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Bruno Franco Oliveira

Vulnerabilidades em Sistemas Web

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Marcelo Zanchetta do Nascimento

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2019

Bruno Franco Oliveira

Vulnerabilidades em Sistemas Web

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 18 de janeiro de 2019:

Marcelo Zanchetta do Nascimento
Orientador

Aluno

Uberlândia, Brasil
2019

Dedico aos meus pais e meus irmãos por me apoiarem em toda a jornada e estarem sempre presente na minha vida diariamente, aos meus professores, o meu singelo agradecimento a todos os profissionais da educação que cooperaram diretamente e indiretamente nas dúvidas e no compartilhar do conhecimento que contribuiu para a minha evolução pessoal e profissional.

Agradecimentos

Agradeço aos meus professores e ao meu orientador Marcelo Zanchetta do Nascimento pelo apoio nos estudos e conhecimento para o meu crescimento profissional. Sem todos vocês a realização desse projeto não seria possível.

Resumo

O estudo visa demonstrar alguns ataques e ameaças em sistemas web, as maneiras como podem aparecer falhas de segurança e como prevenir para que essas falhas sejam evitadas. Esse estudo mostra o uso de ferramentas empregadas em ataques e como poder prevení-los, utilizando mecanismos de criptografia simétrica, assimétrica, hashing, certificado digital, etc.

Palavras-chave: Criptografia simétrica, criptografia assimétrica, hashing, certificado digital.

Lista de ilustrações

Figura 1 – As principais vulnerabilidades em sistemas Web	15
Figura 2 – Um exemplo de SQL Injection.	16
Figura 3 – Um diagrama que mostra um cross site scripting.	17
Figura 4 – Aumento de vulnerabilidades em organizações da China.	18
Figura 5 – Incidentes reportado pelo CERT no Brasil em 2017.	19
Figura 6 – Incidentes reportado pelo CERT.br de janeiro a dezembro de 2017 - tipos de ataque.	20
Figura 7 – Incidentes reportado pelo CERT.br de janeiro a dezembro de 2017 - países de origem	22
Figura 8 – Teste realizado no site manancial das águas com a ferramenta skipfish.	28
Figura 9 – Teste realizado no site manancial das águas com a ferramenta nikito.	29
Figura 10 – Resultado dos testes no site manancial das águas com o skipfish.	29
Figura 11 – Resultado dos testes no site manancial das águas com o skipfish.	30
Figura 12 – Resultado dos testes no site manancial das águas com o nikito.	30
Figura 13 – Resultado dos testes no site manancial das águas com o nikito.	31
Figura 14 – Resultado dos testes no site manancial das águas com o nikito.	31

Lista de tabelas

Tabela 1 – Valores após aplicação das ferramentas para investigação de ataque e vulnerabilidade durante os experimentos.	28
--	----

Lista de abreviaturas e siglas

OWASP	Open Web Application Security Project
CSRF	Cross-Site Request Forgery
SQL	Structured Query Language
XSS	Cross-Site Scripting
IP	Internet protocol
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
TCP	Transfer Control Protocol
CERT	Centro de estudos, respostas e tratamento de incidentes de segurança do Brasil
CSR	Certificate Signing Request
XSRF	Cross site request forgery
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
IIS	Internet Information Services
WWW	World Wide Web

Sumário

1	INTRODUÇÃO	10
1.1	Objetivos	11
1.2	Organização da Monografia	12
2	CONTEXTUALIZAÇÃO	13
2.1	Internet	13
2.2	Segurança em Sistemas Web	14
2.3	Tipos de ataque	15
2.4	Mecanismos de segurança	18
2.5	Trabalhos correlatos	21
3	METODOLOGIA	24
3.1	Skipfish	25
3.2	Nikito	26
4	EXPERIMENTOS E RESULTADOS	27
5	CONCLUSÃO	32
	REFERÊNCIAS	33

1 Introdução

Com a criação da Internet e a evolução tecnológica, a demanda por sistemas web cresceu gradativamente, sendo as necessidades pela busca de novos mercados aumentou ao longo dos tempos conforme apresentado pelo (ECOMMERCEBRASIL, 2017). Baseado na história do desenvolvimento de aplicações web, pode-se notar que a Web foi criada sem grandes restrições de segurança ou políticas de segurança. Ao longo do tempo, os ataques na Internet e em sistemas conectados à Internet se tornaram mais sofisticados, enquanto a habilidade e o conhecimento exigidos para montar um ataque diminuíram (STALLINGS, 2008).

Os problemas relacionados a segurança de sistemas web tornaram-se um ponto importante a ser também tratado. A necessidade de segurança e integridade das informações e dados trafegados na rede mundial de computadores, exigiu uma procura pela segurança cada vez maior conforme estudos realizados pelos especialistas da *Open Web Application Security Project* (OWASP) a respeito das vulnerabilidades apresentadas nos últimos anos (COSTA, 2017).

Com a necessidade em manter um volume grande de dados de maneira digital, a busca pela segurança dos dados também aumentou (MELLO et al., 2006a). Então, ferramentas computacionais foram criadas para tratar a confidencialidade, integridade e disponibilidade dos dados conhecidos como (CID) (FEITOSA et al., 2012). Para que um sistema esteja funcionando corretamente, alguns mecanismos devem ser adotados para o seu funcionamento e segurança, entre eles estão:

- **A disponibilidade:** para que os dados a qualquer momento e qualquer horário possam ser acessados sem falhas e interrupções;
- **A integridade:** os dados não devem sofrer nenhuma alteração nas informações e deve haver um cuidado maior com a segurança das informações apresentadas.
- **A confidencialidade:** para com os dados, onde nenhuma pessoa ou terceiro que tenha acesso ao sistema, onde estão armazenadas as informações, possam acessar informações confidenciais sem autorização e consentimento do usuário.
- **Autorização:** técnicas de não repúdio são utilizadas, onde a intenção é garantir que quem realmente está acessando determinadas informações é o usuário final e não seja possível posteriormente negar que foi ele mesmo (MELLO et al., 2006b). Assim, um sistema será considerado seguro (CARVALHO et al., 2013).

- **Auditoria:** onde é possível saber qual usuário acessou um sistema em um determinado período de tempo e onde mexeu visando a proteção de usuários mal intencionados do sistema e possíveis acessos a informações sigilosas.

As vulnerabilidades são situações motivadas por pessoas mal intencionadas procurando falhas de segurança em uma aplicação que podem trazer prejuízo aos usuários (TAHA et al., 2017). As falhas podem ocorrer quando uma arquitetura é mal planejada, e os prazos para a entrega de desenvolvimento da aplicação são relativamente curtos para uma maior avaliação dos cenários relacionados a segurança, por muitas vezes não apresentar uma ampla variedade de casos de testes. Isso quando criados casos de testes e realizados procedimentos mais rigorosos para testar as funcionalidades da aplicação em um todo.

Com isso, uma forma de evitar as vulnerabilidades em sistema web é estudar e analisar principais erros cometidos no desenvolvimento de aplicações, configuração de serviços em sistemas Web e realizar testes para testar possíveis vulnerabilidades no sistema. Este trabalho apresenta um estudo de avaliação empírica feito nos sistemas web, utilizando ferramentas conhecidas como *scanners* para identificar as falhas presentes e assim proporcionar sugestões de correções nas possíveis falhas encontradas nos testes realizados pelos scanners.

1.1 Objetivos

O objetivo é investigar as principais formas de ataques, ameaças e vulnerabilidades em sistemas Web. Isso permitirá prevenir de maneira com que os dados não sejam expostos, interceptados, alterados e roubados. Ferramentas são empregadas para varreduras de possíveis falhas de segurança nas aplicações, que são os *scanners* para a análise de possíveis falhas de segurança e vulnerabilidades. Este trabalho irá abordar ferramentas *open source* que são os *scanners* que funcionam para testar as vulnerabilidades apresentadas em uma aplicação web. Pontualmente, pretende-se analisar:

- *Cross site scripting*;
- *SQL injection*;
- *Ajax testing*;
- *File inclusion*;
- *JS source code analyzer*;
- *Backup file check*;

Com os experimentos, serão demonstrados vulnerabilidades, ameaças e ataques de diversos tipos utilizando as ferramentas *open source* e levando em consideração o tempo para a varredura nos web sites, as falhas apresentadas ou possíveis ameaças que podem ocorrer ou não durante os testes.

1.2 Organização da Monografia

Nesse capítulo, foram apresentadas as considerações iniciais, motivação e objetivos para o estudo apresentado nesta monografia. O restante deste trabalho possui a seguinte organização:

- **Capítulo 2 - Contextualização:** apresenta a teoria e estado da arte de tópicos relacionados a segurança em aplicações Web, ataques e trabalhos correlatos;
- **Capítulo 3 - Metodologia:** apresenta os materiais e métodos utilizados para o desenvolvimento deste projeto;
- **Capítulo 4 - Experimentos e Análise dos Resultados:** apresenta os resultados obtidos em cada etapa empregada, bem como os resultados quantitativos dos experimentos;
- **Capítulo 5 - Conclusão:** apresenta as conclusões do estudo.

2 Contextualização

Nesta seção, serão introduzidos conceitos teóricos necessários para a descrição deste. Além disso, serão descritos estudos da literatura relacionados ao trabalho proposto.

2.1 Internet

Em 1950 os EUA formou a *Advanced Research Project Agency* (ARPA) que seria uma instituição para investigação, que faria parte do departamento de defesa dos EUA que tinha como objetivo a implantação de uma rede de comunicações entre locais de extrema necessidade para o sistema de defesa norte americano (GOETHALS; AGUIAR; ALMEIDA, 2000).

Era necessário uma rede eficiente, pois caso houvesse algum ataque nuclear, a comunicação seria possível normalmente. Assim foram divididos os dados em pacotes. Com essa divisão, os pacotes poderiam seguir o destino desejado nas rotas disponíveis. Essa rede experimental foi denominada de Rede da Agência para Projetos de Pesquisa Avançada (do inglês, *Advanced Research Projects Agency Network* - ARPANET), criada na década de 70, onde haviam 15 nós que interligavam 20 máquinas. A rede deveria funcionar de maneira descentralizada, onde as informações poderiam ser trafegadas de maneira independente, pois poderia enviar e receber pacotes de um ponto específico para um endereço específico.

Cada nó com um protocolo padrão para comunicação entre os demais. O termo Internet começou a aparecer em torno de 1973, pois em 1972 havia sido iniciado na ARPA a forma de interligação de redes. Já nos anos de 1980 e 1981 no período da guerra fria, cientistas, militares e universidades uniram esforços e como consequência foram criados dois segmentos de redes distintos, o (acrônimo de "Because It's Time to NETwork") ligado ao segmento universitário e a *Computer Science Network* (CSNET) ligado ao segmento científico. Com isso, houve a ampliação da rede com várias aplicações.

Em 1982, então, foi criado o protocolo padrão Protocolo de Transmissão Controlada e Protocolo de Internet (do inglês, *Transmission Control Protocol and the Internet Protocol* - TCP/IP), a partir daí a internet fez uso desse protocolo para interligação de redes. Na Internet cada computador então teria um endereço IP designado para identificação de cada computador. Posteriormente, em 1990, surge nos EUA o primeiro Provedor de Serviço de Internet (do inglês, *internet service provider* - ISP) comercial e a ARPANET deixam de existir. Em 1991, Tim Berners-Lee do CERN, na Suíça, criou uma nova estrutura de informação para o funcionamento da internet chamado *World Wide Web*

(WWW).

Antes da criação do WWW o funcionamento da internet era limitado a troca de mensagens eletrônicas, para transferir dados em tempo real e para conferências eletrônicas. A partir de 1991, foi possível a criação de servidores web, inserção de imagens digitalizadas, multimedia e textos. A proposta foi importante, pois através de uma interface, o acesso a diferentes tipos de plataformas como windows, unix e mac puderam acontecer na comunicação de dados.

Por meio do modelo cliente-servidor utilizando o protocolo *Hyper Text Transfer Protocol* (HTTP), os servidores poderiam responder as requisições de clientes e receber requisições. Assim, através da WWW, era possível armazenar e transmitir milhares de páginas web em diferentes tipos de servidores espalhados pelo mundo.

Por meio da rede mundial de computadores, a Internet, houve a integração entre as comunidades internacionais no envolvimento de pesquisa e propagação do conhecimento científico. Isso ocorreu através da difusão do conhecimento, o crescimento das informações em uma escala mundial. Então, surgiu novas oportunidades de fazer comércio, marketing e interação social.

Atualmente, a busca e a necessidade para gerir a segurança da informação dos dados trafegados pela rede mundial de computadores, têm sido de grande relevância. Estudos e técnicas foram criadas para poder assegurar o acesso a apenas aos órgãos ou pessoas responsáveis por um acesso restrito aos dados tem crescido exponencialmente.

É possível, atualmente, por meio de ferramentas fazer a varredura em busca de vulnerabilidades e simular possíveis ataques e possíveis ameaças que podem ocorrer a servidores onde apresentam falhas de seguranças (ROCHA; KREUTZ; TURCHETTI, 2012).

Atualmente, o grupo OWASP (*Open Web Application Security Project*) que é uma organização mundial sem fins lucrativos, visa demonstrar aspectos de segurança (TAHA et al., 2017).

2.2 Segurança em Sistemas Web

Com a massificação dos dados trafegados na rede mundial de computadores, ambientes corporativos passaram a observar uma maior necessidade com a segurança das informações ao serem armazenadas. Problemas como incêndios, tragédias ocasionadas por meios naturais como tsunamis, terremotos, vazamentos de informações sigilosas por terceiros através de técnicas de engenharia social, fraudes aplicadas, roubos, falhas de energia, as necessidades e o cuidado com a segurança de aplicações foram necessárias. Seja no ambiente corporativo, seja no ambiente governamental ou redes de ensino, a

preocupação com os dados trafegados começou a ser mais transparente.

Em (PEIXINHO FRANCISCO MARMO DA FONSECA, 2013), os autores relatam a respeito de vulnerabilidades como sendo brechas em um sistema computacional e quando trata-se de aplicações, essas vulnerabilidades são muitas vezes chamadas de bugs.

A segurança da informação de uma empresa garante, em muitos casos, a continuidade de negócio, incrementa a estabilidade e permite que as pessoas e os bens estejam seguros de ameaças e perigos (SANTO,).

Muitas instituições realizaram uma análise de viabilidade se os custos realizados com a segurança seriam maiores que os gastos que poderiam ocorrer em um possível ataque e roubo de informações sigilosas (SHEZAN; AFROZE; IQBAL, 2017). Devido a uma maior necessidade ao sigilo e a disponibilidade das informações de maneira segura, precauções são necessário para os diversos ataques.

2.3 Tipos de ataque

Segundo a OWASP (Open Web Application Security Project) (OWASP, 2017), uma comunidade online, onde apresenta pesquisas para auxiliar na melhoria e testes para o funcionamento da segurança arquitetural dos projetos de sistemas web. Em 2017, realizou um levantamento global com os 10 riscos de segurança na web de grande importância dos principais tipos de ataque estudados pela instituição para aplicações web, e demonstrou as maiores vulnerabilidades apresentadas nas aplicações (ver Figura 1).

OWASP Top 10 2013	±	OWASP Top 10 2017
A1 – Injection	→	A1:2017 – Injection
A2 – Broken Authentication and Session Management	→	A2:2017 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	↘	A3:2013 – Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017 – XML External Entity (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017 – Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017 – Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017 – Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	✗	A8:2017 – Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	✗	A10:2017 – Insufficient Logging & Monitoring [NEW, Comm.]

Figura 1 – As principais vulnerabilidades em sistemas Web

A lista de tipos de ataques são:

- **Injeção:** acontece quando um código malicioso pode ser injetado em uma aplicação,

um dos mais comuns é o SQL Injection, onde um usuário malicioso tenta através da entrada de dados em um formulário, tenta inserir uma instrução SQL para buscar os dados no banco de dados (SCRIPTCASE, 2017) (ver Figura 2).

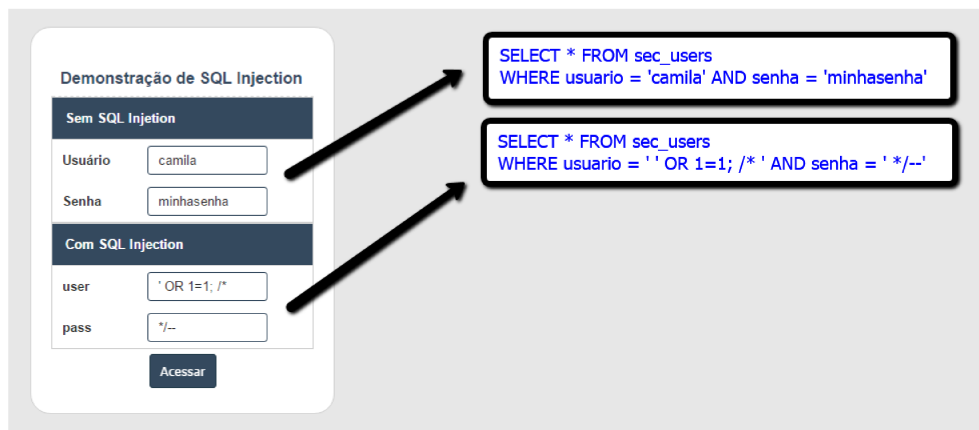


Figura 2 – Um exemplo de SQL Injection.

- **Gerenciamento de sessão e quebras de autenticação:** caso o usuário não esteja usando um canal seguro como por exemplo o SSL, usuários maliciosos podem "escutar" endereços ip com *sniffers*. É muito comum em aplicações onde não têm *token* de acesso para garantir o não repúdio do acesso dos usuários.
- **Configuração incorreta de segurança:** caso usuários descuidados ou leigos venham instalar aplicativos maliciosos como keyloggers entre outros, informações podem ser roubadas e acessos a contas podem ocorrer. Sem mecanismos como tokens de acesso e identificação do usuário, a vulnerabilidade na restrição de acesso a dados sigilosos pode ocorrer.
- **Quebras no controle de acesso:** pode ocorrer quando aplicações não apresentarem restrições de acesso a arquivos e conteúdos de sistemas, conteúdos restritos a determinados tipos de usuários quando não feito um controle de acesso por perfil, usuários mal intencionados podem usar *scanners* para procurar brechas de segurança e acessar conteúdos indevidos.
- **Cross-site scripting(XSS):** scripts são colocados e inseridos nas páginas de navegação do lado do cliente antes de uma requisição ser enviada ao servidor, podendo com isso, roubar dados da sessão do usuário. Ele pode ser subdividido em três tipos de categorias: Refletido, Armazenado, baseado em DOM (ROCHA; SOUTO; MARTINS, 2013) (ver Figura 3).
- **Exposição de dados sensíveis:** aplicações de uso de dados sigilosos como bancárias, e-mails, comércio eletrônico necessitam de uma maior segurança, pois crackers podem estar analisando dados de um determinado IP e caso não haja a criptografia

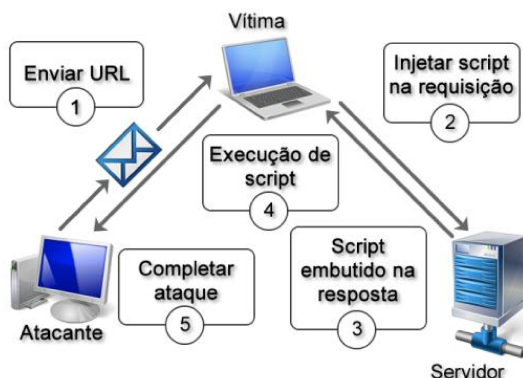


Figura 3 – Um diagrama que mostra um cross site scripting.

dos dados, as informações podem ser interceptadas através de sniffers. Algoritmos fracos de criptografia podem também contribuir para acentuação do problema, assim como funções hash com baixa chave. Senhas salvas em banco de dados sem algoritmos de hashing que funcionam como uma assinatura eletrônica, pode contribuir para que um analista de banco de dados tenha acesso a informação sigilosa e possa utilizar posteriormente. Utilizar uma conexão não criptografa em aplicações como comércio eletrônico pode contribuir para o roubo e uso de dados de cartões de crédito e informações pessoais.

- **Des-serialização Insegura:** pode ocorrer quando a lógica de negócios não faz uma validação de um objeto enviado ao servidor, com isso pode ser enviado um objeto diferente do esperado nas regras de negócio da aplicação. APIs que não fazem esse tipo de validação podem estar vulneráveis a esse tipo de ataque e prejudicar a aplicação.
- **Entidades Externas XML:** ocorre com frequência em sistemas legados, onde não há uma atualização nos dados para corrigir falhas de segurança. Com ferramentas de varredura de portas, é possível executar códigos remotamente e realizar ataques de negação de serviço (DoS), por isso as vulnerabilidades são frequentes em casos como esse, pois não há a atualização dos dados.
- **Utilizar componentes com vulnerabilidades conhecidas:** ao desenvolver uma aplicação com componentes já apresentados falhas, caso não haja uma pesquisa referente a determinados componentes, podem haver falhas de segurança por se usar uma API e componentes de terceiros que já apresentaram falhas prejudicando a aplicação.
- **Registro e Monitoração Ineficientes:** uma aplicação quando não é monitorado o acesso a cada serviço solicitado, podem haver falhas de identificação na movimentação em sua rede de acesso na aplicação e no ambiente do sistema imputado a

solicitação de dados. Caso a aplicação não possua esse monitoramento e alarmes a serem disparados em possíveis tentativas de ataques ao sistema e na varredura de dados indevidados, o sistema pode ficar vulnerável e sem a resolução do problema rapidamente, podendo ocasionar tanto danos a dados sigilosos como potenciais alterações no funcionamento da aplicação.

Segundo pesquisas realizadas pelo Instituto Ponemon, foram registrados e mostrados dados das principais vulnerabilidades conforme mostra o estudo realizado nos anos de 2012 a 2015 referente ao crescimento dos principais tipos de ataque em sua maioria na China (COELHO; MARTINS, 2018) conforme informações da Figura 4).

	2012	2013	2014	2015
SQL injection	853 (18.82%)	2,099 (25.05%)↑	15,792 (50.28%)↑	5,742 (44.87%)
XSS	770 (16.99%)	1,032 (12.32%)	1,492 (4.75%)	283 (2.21%)
Logic error	561 (12.38%)	1,048 (12.52%)↑	1,260 (4.01%)	202 (1.58%)
Sensitive data exposure	559 (12.33%)	607 (7.25%)	1,909 (6.08%)	1,403 (10.96%)↑
Broken access control	403 (8.89%)	869 (10.37%)↑	1,750 (5.57%)	515 (4.02%)
Command injection	340 (7.5%)	782 (9.33%)↑	1,624 (5.17%)	564 (4.41%)
Misconfiguration	277 (6.11%)	365 (4.36%)	1,295 (4.12%)	354 (2.77%)
Hack event	180 (3.97%)	293 (3.5%)	2,201 (7.01%)↑	1,734 (13.55%)↑
Weak password	169 (3.73%)	479 (5.72%)↑	2,046 (6.51%)↑	733 (5.73%)
File upload	168 (3.71%)	314 (3.75%)↑	562 (1.79%)	117 (0.91%)
Path traversal	111 (2.45%)	163 (1.95%)	343 (1.09%)	64 (0.5%)
Unvalidated redirects	78 (1.72%)	56 (0.67%)	136 (0.43%)	34 (0.27%)
CSRF	55 (1.21%)	150 (1.79%)↑	50 (0.16%)	16 (0.13%)
File include	8 (0.18%)	42 (0.5%)↑	371 (1.18%)↑	111 (0.87%)
Other	0 (0.0%)	78 (0.93%)↑	575 (1.83%)↑	924 (7.22%)↑

Figura 4 – Aumento de vulnerabilidades em organizações da China.

2.4 Mecanismos de segurança

A palavra criptografia se origina do grego dos radicais kriptos (oculto) e grapho (escrita) e têm como significado ser à ciência ou arte de codificar mensagens (OLIVEIRA, 2012).

Segundo pesquisas realizadas pelo Centro de pesquisa, resposta e tratamentos de incidentes de segurança do Brasil (CERT), em 2017, mostra o crescimento de possíveis falhas de segurança reportados ao CERT. O estudo realizado no Brasil no portal do CERT, mostra a quantidade de incidentes nos últimos anos (CERT-INCIDENTES, 2017) (Figura 5).

Os principais tipos de ataque apresentados pelo CERT no Brasil, em 2017, apresentam uma grande diversificação sendo a maior parcela no escaneamento (CERT-TIPOS-ATAQUE, 2017) (ver Figura 6). Na Figura 7) são apresentados em detalhes essas informações pelos países.

- **Criptografia simétrica(chave privada):** quando uma determinada mensagem é transmitida de um ponto a outro, para que do outro lado possa interpretar a mensagem, é necessária uma chave ou senha para que possa ser realizado

Total de Incidentes Reportados ao CERT.br por Ano

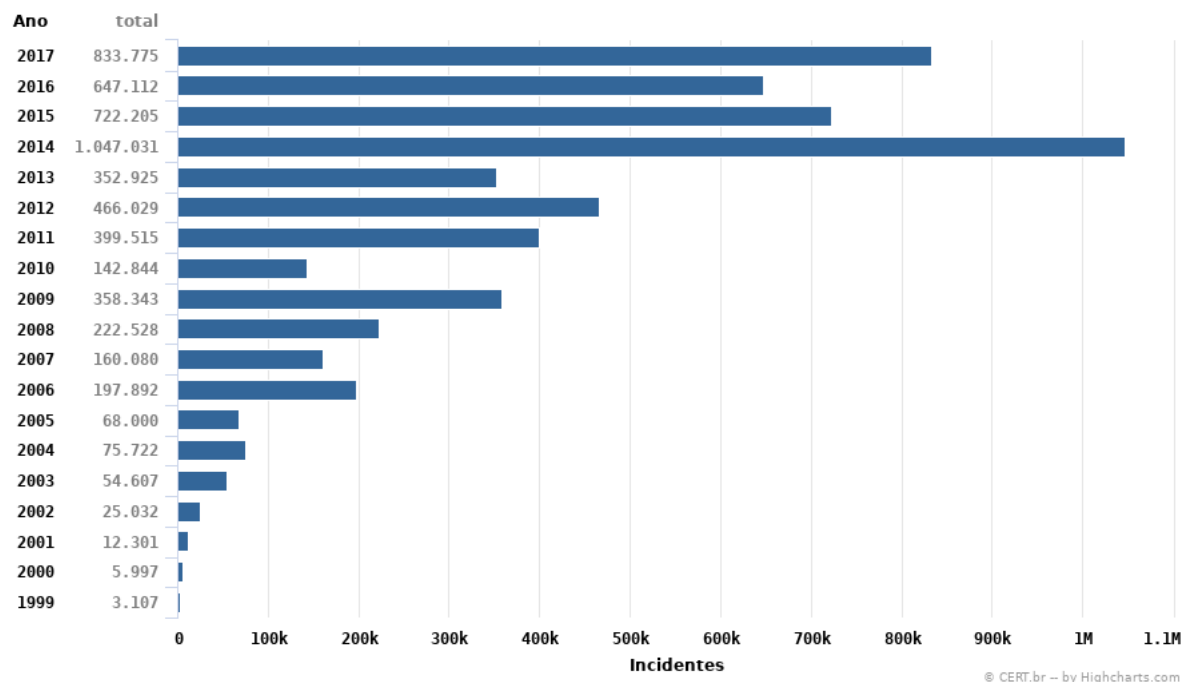
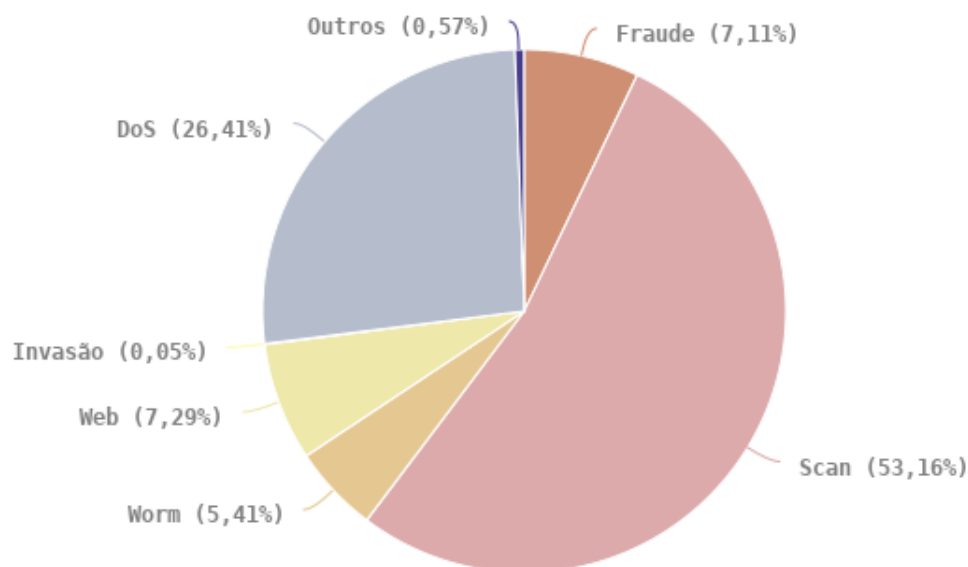


Figura 5 – Incidentes reportado pelo CERT no Brasil em 2017.

esse procedimento. Para isso é necessário que se mantenha em segurança a chave secreta para que somente ambas as partes, o emissor e receptor possam ter acesso ao conteúdo da mensagem. A sua desvantagem é que não garante a autenticidade e o não repúdio do usuário, um dos algoritmos mais utilizado atualmente é o AES pois seu consumo de memória é baixo e rápida a sua execução, tendo um bloco fixo de 128 bits e uma chave que pode variar entre 128 bits até 256 bits.

- **Criptografia assimétrica(chave pública):** modelo pelo qual as duas partes usam duas chaves assimétricas, sendo uma pública e a outra privada, as chaves não são apenas senhas, mas também por arquivos mais complexos. A chave pública pode estar disponível para qualquer parte que deseja se comunicar com a outra parte, mas apenas ambas as partes podem ter a chave privada, pois através da cifração da mensagem com a chave pública. Para se entender seu funcionamento, um exemplo seria uma mensagem fechada e guardada com um cadeado, ele simboliza a chave pública e com a chave privada você poderia abri-lo e ter acesso a mensagem. Um dos algoritmos atualmente mais utilizados é o RSA, a sua desvantagem é que o custo computacional de processamento e de memória é maior que o algoritmo AES de criptografia simétrica.
- **Certificado digital:** faz a junção de um determinado usuário a uma chave pública, onde uma chave privada é incorporada ao certificado assinado por uma pessoa de confiança. Existem unidades responsáveis pelo registro dos certifica-

Incidentes Reportados ao CERT.br -- Janeiro a Dezembro de 2017**Tipos de ataque**

© CERT.br -- by Highcharts.com

Figura 6 – Incidentes reportado pelo CERT.br de janeiro a dezembro de 2017 - tipos de ataque.

dos, no Brasil a unidade certificadora raiz é a ICP-Brasil(AC Raiz) que tem autoridade para emitir, expedir, distribuir revogar e gerenciar as auditorias de outras unidades certificadoras. São autoridades certificadoras no país: Serpro (AC-SERPRO), Caixa Econômica Federal (AC-CAIXA), Serasa Experian (AC-SERASA), Receita Federal do Brasil (AC-RFB), Certsing (AC-Certisign), Imprensa Oficial do Estado de São Paulo (AC-IOSP), Autoridade Certificadora da Justiça (AC-JUS), Autoridade Certificadora da Presidência da República (AC-PR) e Casa da Moeda do Brasil (AC-CMB). A entidade que deseja emitir o certificado gera um par de chaves criptografadas, sendo uma chave pública e outra privada em seguida gera um arquivo chamado CSR(Certificate Signing Request) composto pela chave pública da entidade solicitante com os seus respectivos dados e da unidade certificadora. Esse arquivo é assinado digitalmente com a chave privada e enviado para a unidade certificadora. Ex: "Nome da empresa, localização, algoritmo de geração e o tamanho da chave".

Em aplicações comerciais, o uso da segurança têm sido indispensável, pois cada vez mais pessoas têm usado sistemas para acessar as informações desejadas. Com isso, novas técnicas de segurança foram criadas, entre elas a cifragem

de dados, onde as mensagens trafegadas na rede mundial de computadores são cifradas através de algoritmos matemáticos, e somente o usuário após realizar o login de acesso pode acessar as informações e privilégios de determinado conteúdo ([CARVALHO et al., 2013](#)). Ainda que haja ataques como man-in-the-middle onde um cracker tente interceptar as informações, com o uso de um certificado digital as informações no canal de comunicação entre a entidade solicitante e a entidade que enviou a resposta estarão cifradas e protegidas.

- **Assinatura digital:** utiliza mecanismos de criptografia assimétrica ou de chave pública, onde os dados são assinados e criptografados a partir de uma chave privada. Qualquer pessoa pode realizar o processo da assinatura desde que tenha acesso a chave pública da pessoa, assim como uma assinatura em papel, onde a pessoa comprova a identidade do emissor. Apenas esse processo de cifragem não garante a confidencialidade dos dados, pois qualquer pessoa que tenha acesso ao uso da chave pública do usuário pode ter acesso as informações. Para garantir a autenticidade, integridade e não repudição do acesso aos dados da pessoa, algoritmos de hashing como o SHA e MD5(pouco utilizado atualmente) são baseados em funções não inversíveis, e visam garantir que até mesmo analistas que tenham acesso a base de dados não consigam ter acesso a informações de senha dos usuários. Pois, realizando o hashing da senha, a criptografando e salvando em banco apenas o hashing da senha, somente os usuários que cadastraram realmente a senha teriam acesso ao conteúdo desejado, pois a senha não é salva diretamente no banco e sim um hashing da senha digitada. Segundo o CERT, o Brasil apresenta um dos maiores índices das origens dos tipos de ataque seguido de alguns outros países ([CERT-PAISES-ORIGEM, 2017](#)).

2.5 Trabalhos correlatos

Nesta seção são tratados contribuições presentes na literatura a respeito do tema de vulnerabilidade de sistemas Web.

Segundo ([STALLINGS, 2008](#)) independente dos padrões utilizados, os protocolos de acesso utilizados sempre serão HTTP ou HTTPS a ser integrado com o design visual da aplicação. Segundo em 2018, os autores ([ASSUNÇÃO, 2018](#)), mostram em seu trabalho que uma das escolhas importante é uma ferramenta para varredura de dados que trate a quantidade dos falsos positivos e falsos negativos após a varredura dos dados ser finalizada. O falso positivo é quando não existe uma falha de segurança, mas a ferramenta de varredura detecta erroneamente uma possível falha de segurança. Já o falso negativo é quando a ferramenta efetua a varredura dos dados, mas não é identificada nenhuma

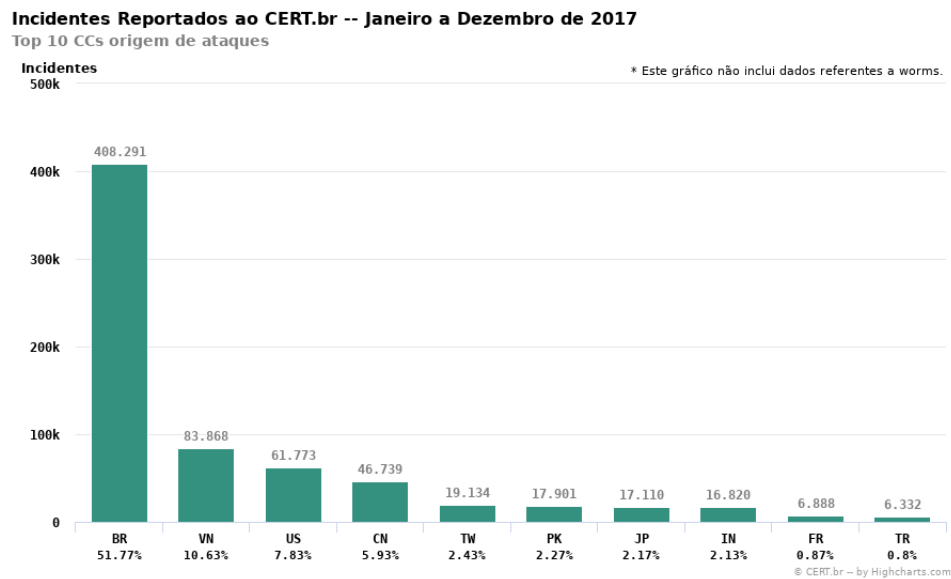


Figura 7 – Incidentes reportado pelo CERT.br de janeiro a dezembro de 2017 - países de origem

falha e segurança mesmo havendo uma determinada vulnerabilidade. Segundo o site SecTools.org, há uma lista com as principais ferramentas para varredura referente a segurança indicadas por especialistas da área. No estudo apresentado por (ASSUNÇÃO, 2018), foram utilizadas ferramentas *open source* devido a facilidade para a realização de uma possível auditoria, pois como o código fonte das ferramentas é aberto ao público, há uma transparência maior. Neste estudo, a ferramenta que apresentou o maior número de falhas de segurança na varredura dos dados tirando os falsos positivos, foi considerado mais eficiente na busca de vulnerabilidades em um ambiente controlado.

Uma das características interessantes nos *scanners* utilizados, é que a geração de logs no resultado final no processo de varredura é muito extenso. Pois apresentam diversos *warnings* (avisos) de possíveis vulnerabilidades, que podem ser divididos nas seguintes categorias alto risco, médio risco e baixo risco (FONSECA; VIEIRA; MADEIRA, 2007). Nos testes apresentados, pode-se verificar uma grande quantidade de falsos positivos apresentados pela ferramenta OWASP ZAP, onde das 119 possíveis vulnerabilidades apresentadas, quando os testes foram executados de maneira manual, apenas 72 erros foram apresentados o processo total incluindo o *crawling* e varredura foi realizado durante 20 minutos e 32 segundos. Dentro das categorias onde foram apresentados o maior número de vulnerabilidades, estão as ligadas a injeção de código, manipulação de parâmetros de autenticação e *Cross Site Request Forgeries*.

Segundo os estudos realizados por (CORDEIRO; FERNANDES, 2018) foram apresentados diversos *scanners* como *Nmap*, *Armitage*, *Metasploit* e o *Kali*

Linux foram realizados diversos testes em sites diferentes e realizados testes por diversas portas, buscando possíveis falhas de segurança. O *Nmap* após a realização da varredura e busca de algumas portas, encontrou algumas portas abertas se mostrando eficiente na procura para esse determinado tipo de teste e mostrando possíveis portas abertas que poderiam ser um canal para invasão de crackers maliciosos.

Já com o *Armitage* foram apresentados 698 exploits que são partes de software que visam tirar proveito de alguma vulnerabilidade, falha ou defeito de algum software. Com o estudo realizado pode-se mostrar que o conjunto dos dois *scanners* pode complementar os testes um do outro, um se mostrando eficaz para realizar os devidos cuidados com a infraestrutura de rede e o outro com possíveis falhas no software.

Para (SALAS; MARTINS, 2015) foram realizados testes de injeção e negação de serviços, onde foi utilizado a ferramenta soapUI com o *add-on Security Testing* para poder realizar possíveis testes de ataques tais como: *SQL Injection*, *XPath Injection*, *Cross-site Scripting (XSS)*, *Fuzzing Scan*, *Invalid Types*, *Malformed XML* e *XML Bomb*. Nos testes realizados para os tipos de ataque, foram realizados 483 ataques.

Os testes realizados procuravam efetuar mudanças aleatórias nos parâmetros dos arquivos WSDL para as requisições SOAP, nos testes de injeção os que apresentaram maior porcentagem de ataques realizados foram: *XML Bomb* com 426 alertas e 66,88%, *Cross-site Script* com 12.598 alertas e 65,93% e *Malformed XML* com 1.094 alertas e 65,51%. Nos estudos realizados, é demonstrado que mais de 97% dos serviços web quando testados, apresentam pelo menos um tipo de ataque em um ambiente emulado.

3

Metodologia

De acordo com a norma NBR ISO/IEC 27002 devido a necessidade de proteger dados sigilosos e a integridade dos mesmos de uma aplicação, atualmente, o cuidado com a segurança das aplicações vêm crescendo gradativamente. Por se tratar de informações sigilosas as regras de negócio das aplicações devem ser bem estruturadas e protegidas por possíveis ameaças ou ataques.

Neste trabalho foram utilizados softwares *open source* devido a transparência do seu funcionamento e por estarem abertos ao público sem um custo de sua utilização. Foram utilizados dois *scanners* para a varredura de dados e busca de possíveis vulnerabilidades, assim como realizado os testes em um ambiente controlado como mostrado no trabalho correlato (UTO; MELO, 2017).

O processo dos testes foi realizado na seguinte estrutura:

- 1 - Escolha das ferramentas a serem utilizadas;
- 2 - Escolha dos endereços dos sistemas web a serem realizados os testes;
- 3 - Varredura com os *scanners* buscando possíveis vulnerabilidades;
- 4 - Análise dos resultados apresentados;

Para a realização dos testes foi utilizado um ambiente controlado com uma distribuição gratuita do sistema operacional Linux Mint 18.2 cinnamon. As ferramentas utilizadas nos testes foram os *scanners* skipfish e nikito, onde são apresentados os relatórios dos resultados do tempo de execução e a sua eficácia de acordo com a quantidade de vulnerabilidades. Neste caso, o processo de escaneamento no sistema web acontece quando realizada a requisição no servidor web e listados os arquivos que fazem parte daquela aplicação web. Os experimentos realizados pelos *scanners* podem ser variados, tais como injeção de código malicioso nos arquivos para testar possíveis falhas de segurança, outros como sql injection utilizando a linguagem sql para tentar realizar uma possível consulta. O servidor onde está hospedada os experimentos é o IIS (*Internet Information Services* da Microsoft na versão 8.0. Para a realização de cada teste para as possíveis vulnerabilidades, requisições GET e POST são realizadas no servidor e em alguns testes, os parâmetros são passados ou ocultos para possíveis falhas ou configurações no servidor em cada requisição realizada.

3.1 Skipfish

Desenvolvido por Michal Zalewski, Niels Heinen e Sebastian Roschke, com licença de código Apache versão 2.0. É uma ferramenta para varredura e testes de segurança para possíveis falhas e vulnerabilidades. Possui suporte para os principais sistemas operacionais como Linux, FreeBSD, MacOS X e Windows (por meio do Cygwin) e foi escrito na linguagem C. Os experimentos foram realizados no site da manancial das águas hospedado no servidor IIS na porta 80. O skipfish se baseia nos seguintes testes:

Risco alto:

- Injeção da linguagem SQL nas requisiões GET e POST nos parâmetros;
- Injeção de comandos no lado do servidor (incluindo *blind vectors*);
- Injeção no lado do servidor de arquivos XML e XPATH (incluindo *blind vectors*);
- Vulnerabilidades no formato de string;
- Vulnerabilidades em estouro dos números inteiros;
- Requisições HTTP aceitando PUT;

Risco médio:

- Injeção de código malicioso nos arquivos javascript ou no corpo do documento para os tipos de ataque XSS;
- Dados armazenados e refletidos no cabeçalho HTTP separados nele;
- Passagem de diretórios restritos;
- Alterações nos arquivos de configurações do lado do servidor;
- Problemas nos scripts para manipulação de dados via javascript e nos arquivos css;
- Tipos MIME incorretos ou genéricos não aceitos pelo cabeçalho.
- Incorretos ou faltando *charsets* no cabeçalho.
- Armazenamento das diretivas de cache incorretas e nas respostas das configurações dos cookies;

Risco baixo:

- Listagem de diretórios;
- Redirecionamento para URL fornecida pelo invasor (podendo ser armazenada ou refletida);
- Conteúdo adicionado, fornecido pelo atacante (armazenado e refletido).

- Conteúdo adicionado não confiável externo;
- Conteúdo em sub recursos de scripts não relacionado aos scripts javascript;
- Dados de credenciais no HTTP nas URLs;
- Certificados SSL expirados ou ainda não válidos;
- Formulários HTML sem proteção XSRF;
- Certificados SSL autoassinados;
- Diferenças de nome de *host* do certificado SSL;

3.2 Nikito

É uma ferramenta para varredura e análise de dados de um servidor web. Essa ferramenta foi escrito para várias plataformas, incluindo Windows, Linux e UNIX. Foi desenvolvido na linguagem perl, foi desenvolvido no ano de 2001, em dois anos se tornou um dos *scanners* mais populares gratuitos. Em 2007, após vários anos de melhorias, foi lançada a versão 2.0 e, em 2008, David Lodge assumiu a liderança do projeto. O nikito tem suporte para SSL, proxies, autenticação de host, codificação de ataque e muito mais. Ele pode ser atualizado automaticamente a partir da linha de comando.

As principais funcionalidades do nikito que foram explorados para testes de vulnerabilidades foram:

- Configurações incorretas do servidor e do software;
- Arquivos e programas padrão;
- Arquivos e programas sem segurança;
- Servidores e programas desatualizados;

4

Experimentos e Resultados

A análise para realização no desenvolvimento dos testes foi feita baseada em *scanners* para a varredura e testes de possíveis falhas de segurança ou riscos que poderiam ocorrer. A princípio foi utilizada a ferramenta kali Linux skipfish onde foram realizados experimentos em um período de aproximadamente 32 minutos e 54 segundos buscando possíveis falhas. O skipfish funciona como um crawler, onde faz a varredura dos dados em uma aplicação web, diferente dos outros *scanners* não separa os tipos de vulnerabilidades por tópicos durante sua execução. Somente no final de todo processo permite obter um relatório com as informações do processo de varredura total da aplicação web. A Figura 9 mostra o exemplo do relatório obtido após os experimentos realizados com o servidor Web.

A utilização da ferramenta é de rápida aprendizagem e eficaz, foi utilizado o sistema operacional Linux Mint 18.2. Nos testes foram escaneados 277 arquivos, a priori foi apresentado 1 aviso de ameaça de alto risco nos testes e 4 avisos de baixo risco. A maioria dos avisos apresentados foi relacionado aos arquivos javascript da aplicação mas nenhum erro foi detectado.

Foram realizados experimentos com a ferramenta Nikto, onde o tempo para a varredura foi de 23 minutos e 20 segundos no total. Nenhum erro foi apresentado, foram escaneados 6544 itens e apenas 10 avisos apresentados. A Figura 9 mostra o relatório obtido após os experimentos realizados com o servidor Web.

A ferramenta nikto o processo de instalação e aprendizagem foi bem rápido e eficaz, atualmente é uma das ferramentas mais utilizadas segundo o site "SecTools".

Após alguns testes realizados pelos *scanners* skipfish quanto pelo nikto, os testes apresentaram bastante divergências, sendo no número de arquivos apresentados, sendo no tempo de execução ou quais apresentaram um rigor a mais nas buscas por possíveis falhas e vulnerabilidades. Sendo que em um dos *scanners* foram apresentadas possíveis vulnerabilidades, enquanto no outro nenhuma vulnerabilidade foi apresentada. A Tabela 1 ilustra as informações detalhadas após aplicação do teste.

Após o tempo de execução total dos testes implícitos realizados pelo *scanner*, foi gerado um relatório apresentando detalhadamente os resultados realizados em cada tipo de ataque pelo mesmo. As Figuras 10, 11, mostram o resultado no teste do site manancial das águas com o *skipfish*.

```

Crawl scope options:
-d max_depth      - maximum crawl tree depth (16)
-c max_child      - maximum children to index per node (512)
-x max_desc       - maximum descendants to index per branch (8192)
-r r_limit        - max total number of requests to send (100000000)
-p crawl%         - node and link crawl probability (100%)
-q hex            - repeat probabilistic scan with given seed
-I string         - only follow URLs matching 'string'
-X string         - exclude URLs matching 'string'
-K string         - do not fuzz parameters named 'string'
-D domain         - crawl cross-site links to another domain
-B domain         - trust, but do not crawl, another domain
-Z               - do not descend into 5xx locations
-O               - do not submit any forms
-P               - do not parse HTML, etc, to find new links

skipfish version 2.10b by lcamtuf@google.com
www.manancialdasaguas.com.br -

Scan statistics:
  Scan time : 0:32:54.327
  HTTP requests : 73269 (37.1/s), 268038 kB in, 17200 kB out (144.5 kB/s)
  Compression : 0 kB in, 0 kB out (0.0% gain)
  HTTP faults : 9 net errors, 0 proto errors, 2 retried, 0 drops
  TCP handshakes : 5106 total (14.3 req/conn)
  TCP faults : 0 failures, 0 timeouts, 4 purged
  External links : 3297 skipped
  Reqs pending : 0

Database statistics:
  Pivots : 277 total, 277 done (100.00%)
  In progress : 0 pending, 0 init, 0 attacks, 0 dict
  Missing nodes : 34 spotted
  Node types : 2 serv, 82 dir, 181 file, 0 pinfo, 2 unkn, 11 par, 0 val
  Issues found : 235 info, 22 warn, 5 low, 3 medium, 0 high impact
  Dict size : 58 words (58 new), 3 extensions, 256 candidates
  Signatures : 77 total

[+] Copying static resources...
[+] Sorting and annotating crawl nodes: 277
[+] Looking for duplicate entries: 277
[+] Counting unique nodes: 118
[+] Saving pivot data for third-party tools...
[+] Writing scan description...
[+] Writing crawl tree: 277
[+] Generating summary views...
[+] Report saved to '/home/bruno/Documentos/Skipfish/index.html' [0xcfd191eb].
[+] This was a great day for science!

bruno@Bruno-PC ~ $

```

Figura 8 – Teste realizado no site manancial das águas com a ferramenta skipfish.

Tabela 1 – Valores após aplicação das ferramentas para investigação de ataque e vulnerabilidade durante os experimentos.

Scanner	Endereço	Risco	Porta	Alto	Médio	Baixo	Falso positivo	Qtde de arquivos escaneados	Tempo de execução
Skipfish	www.manancialdasaguas.com.br	Alto	80	0			0	277	32 minutos e 54 segundos
		Médio	80		5		0		
		Baixo	80			3	0		
Nikito	www.manancialdasaguas.com.br	Alto	80	0			0	6544	23 minutos e 20 segundos
		Médio	80		0		0		
		Baixo	80			0	0		

Segundo os resultados apresentados após a execução do *scanner skipfish*, a informação relacionada a um risco alto para a aplicação, seria relacionado a um formulário de contato em uma página *HTML*, onde um código malicioso poderia ser embutido juntamente ao formulário de dados a ser postado e enviado ao servidor através de uma requisição *POST*.

Os baixos riscos estariam associados a uma página relacionado a localização do local

```

bruno@Bruno-PC ~ $ nikto -h http://www.manancialdasaguas.com.br -o resultados.html
- Nikto v2.1.5
-----
+ Target IP: 198.100.157.60
+ Target Hostname: www.manancialdasaguas.com.br
+ Target Port: 80
+ Start Time: 2018-12-17 17:45:21 (GMT-2)
-----
+ Server: Microsoft-IIS/8.0
+ Retrieved x-aspnet-version header: 4.0.30319
+ The anti-clickjacking X-Frame-Options header is not present.
+ Uncommon header 'x-ua-compatible' found, with contents: IE=9
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Multiple index files found: index.html, default.aspx
+ Retrieved x-powered-by header: ASP.NET
+ Server leaks inodes via ETags, header found with file /, fields: 0xe15fc749fc40ce1:0
+ Server banner has changed from 'Microsoft-IIS/8.0' to 'Microsoft-HTTPAPI/2.0' which may suggest a WAF, load balancer or proxy is in place
+ OSVDB-155: /counter/1/n/n/0/3/5/0/a/123.gif: The Roxen Counter may eat up excessive CPU time with image requests.
+ OSVDB-3093: /pls/portal/PORTAL.wmv_setting.render_css: Access to Oracle pages could have an unknown impact.
+ /gif/hp_invent_logo.gif: This device may be an HP printer/scanner and allow retrieval of previously scanned images.
+ /gif/tricolor_ink_guage.gif: This device may be an HP printer/scanner and allow retrieval of previously scanned images.
+ 6544 items checked: 0 error(s) and 10 item(s) reported on remote host
+ End Time: 2018-12-17 18:08:32 (GMT-2) (1391 seconds)
-----
+ 1 host(s) tested

```

Figura 9 – Teste realizado no site manancial das águas com a ferramenta nikto.

e integrada ao google maps, poderiam ser embutidos trechos de códigos nos arquivos javascript e fazendo um trabalho de injeção de dados maliciosos no momento do envio de dados na requisição para o servidor. Há também um baixo risco que poderia um dos arquivos javascript possuir uma correspondência de assinatura.



skipfish WEB APP SCANNER

Scanner version: 2.10b Scan date: Sun Sep 9 23:31:40 2018
Random seed: 0xcfd191eb Total time: 0 hr 32 min 54 sec 427 ms
[Problems with this scan? Click here for advice.](#)

Crawl results - click to expand:

- <http://www.manancialdasaguas.com.br/> Code: 200, length: 31846, declared: text/html, detected: application/xhtml+xml, charset: utf-8 | show trace +]
- <https://www.manancialdasaguas.com.br:80/> Fetch result: Connection error

Document type overview - click to expand:

- application/javascript (1)
- application/xhtml+xml (8)
- image/jpeg (58)
- image/png (1)
- text/css (2)
- text/plain (3)

Figura 10 – Resultado dos testes no site manancial das águas com o skipfish.

Os riscos de nível médio estariam associados a injeção de código malicioso para possíveis alterações nas configurações do servidor. Foram apresentados 103 avisos

relacionados ao cabeçalho da requisição onde poderiam ser enviados em cada requisição um *MIME Type* incorreto ou inexistente e no *charset* do cabeçalho poderiam haver problemas também relacionadas a informações inexistentes e incorretas.

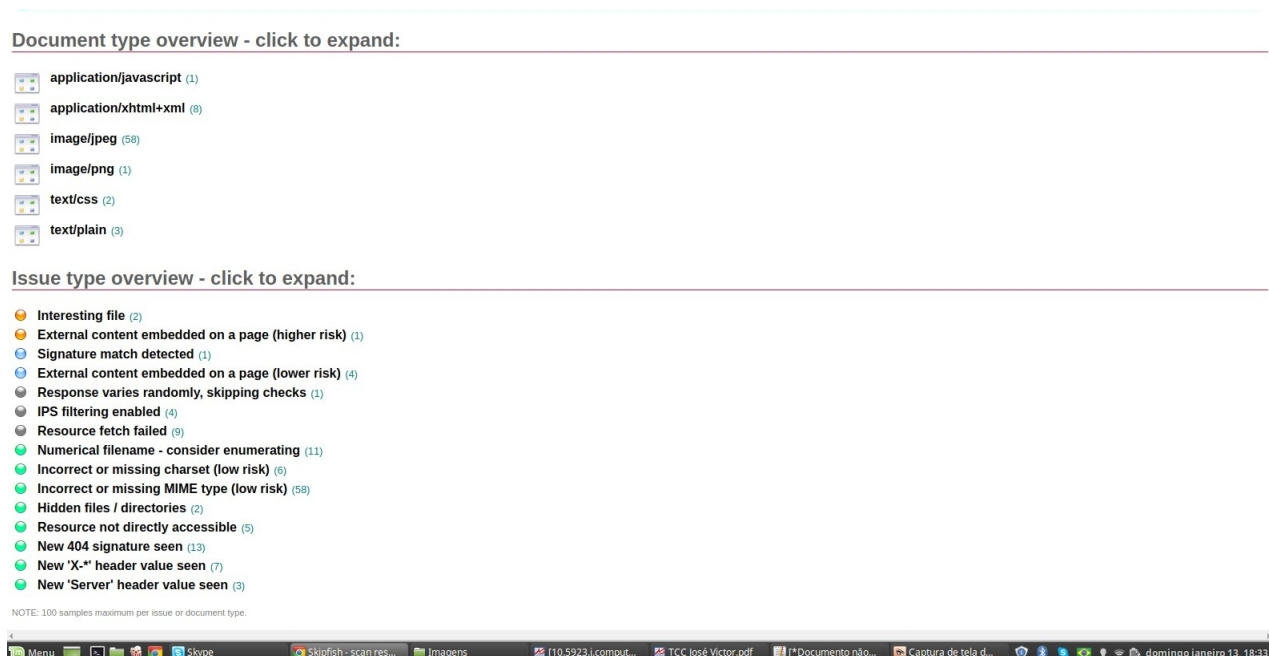


Figura 11 – Resultado dos testes no site manancial das águas com o skipfish.

Nos testes realizados pelo *scanner nikito*, quando realizada a varredura dos dados no servidor, dos testes realizados, somente foram apresentados avisos como o cabeçalho *X-Frame-Options anti-clickjacking* não está presente e um cabeçalho incomum 'x-ua-compatible' encontrado, com o conteúdo: IE = 9.

www.manancialdasaguas.com.br / 198.100.157.60 port 80	
Target IP	198.100.157.60
Target hostname	www.manancialdasaguas.com.br
Target Port	80
HTTP Server	Microsoft-IIS/8.0
Site Link (Name)	http://www.manancialdasaguas.com.br/80/
Site Link (IP)	http://198.100.157.60:80/
URI	/
HTTP Method	GET
Description	Retrieved x-aspnet-version header: 4.0.30319
Test Links	http://www.manancialdasaguas.com.br/80/ http://198.100.157.60:80/
OSVDB Entries	OSVDB-0
URI	/
HTTP Method	GET
Description	The anti-clickjacking X-Frame-Options header is not present.
Test Links	http://www.manancialdasaguas.com.br/80/ http://198.100.157.60:80/
OSVDB Entries	OSVDB-0
URI	/
HTTP Method	GET
Description	Uncommon header 'x-ua-compatible' found, with contents: IE=9
Test Links	http://www.manancialdasaguas.com.br/80/ http://198.100.157.60:80/
OSVDB Entries	OSVDB-0
URI	/
HTTP Method	GET
Description	Multiple index files found: index.html , default.aspx
Test Links	http://www.manancialdasaguas.com.br/80/ http://198.100.157.60:80/
OSVDB Entries	OSVDB-0

Figura 12 – Resultado dos testes no site manancial das águas com o nikito.

Foram apresentadas também, possíveis ameaças onde o servidor poderia vazar *inodes*

via *ETags*, mas não foi apresentado nenhuma falha de segurança que poderia haver um risco ou invasão a um servidor com este aviso. Outro aviso que foi feito, foi a respeito de caso houvesse um volume muito grande de requisições no servidor, o número de imagens poderia fazer com que houvesse uma sobrecarga no processamento do processador do servidor.

URI	/
HTTP Method	GET
Description	Retrieved x-powered-by header: ASP.NET
Test Links	http://www.manancialdasaguas.com.br:80/ http://198.100.157.60:80/
OSVDB Entries	OSVDB-0
URI	/
HTTP Method	GET
Description	Server leaks inodes via ETags, header found with file /, fields: 0xe15fc749fc40ce1:0
Test Links	http://www.manancialdasaguas.com.br:80/ http://198.100.157.60:80/
OSVDB Entries	OSVDB-0
URI	/counter/1/n/n/0/3/5/0/a/123.gif
HTTP Method	GET
Description	/counter/1/n/n/0/3/5/0/a/123.gif: The Roxen Counter may eat up excessive CPU time with image requests.
Test Links	http://www.manancialdasaguas.com.br:80/counter/1/n/n/0/3/5/0/a/123.gif http://198.100.157.60:80/counter/1/n/n/0/3/5/0/a/123.gif
OSVDB Entries	OSVDB-155
URI	/pls/portal/PORTAL.www_setting.render_css
HTTP Method	GET
Description	/pls/portal/PORTAL.www_setting.render_css: Access to Oracle pages could have an unknown impact.
Test Links	http://www.manancialdasaguas.com.br:80/pls/portal/PORTAL.www_setting.render_css http://198.100.157.60:80/pls/portal/PORTAL.www_setting.render_css
OSVDB Entries	OSVDB-3093
URI	/gif/hp_invent_logo.gif
HTTP Method	GET
Description	/gif/hp_invent_logo.gif: This device may be an HP printer/scanner and allow retrieval of previously scanned images.
Test Links	http://www.manancialdasaguas.com.br:80/gif/hp_invent_logo.gif http://198.100.157.60:80/gif/hp_invent_logo.gif
OSVDB Entries	OSVDB-0
URI	/gif/tricolor_ink_guage.gif
HTTP Method	GET
Description	/gif/tricolor_ink_guage.gif: This device may be an HP printer/scanner and allow retrieval of previously scanned images.
Test Links	http://www.manancialdasaguas.com.br:80/gif/tricolor_ink_guage.gif http://198.100.157.60:80/gif/tricolor_ink_guage.gif
OSVDB Entries	OSVDB-0

Figura 13 – Resultado dos testes no site manancial das águas com o nikito.

Apesar de um grande processamento de arquivos o nikito não demonstrou um bom tempo para a execução de todo o processo, sua utilização mostrou que a inclusão de mais ferramentas são necessárias para uma maior eficiência nos resultados para possíveis vulnerabilidades, pois alguns *scannners* são mais minuciosos do que outros para possíveis falhas de segurança.

URI	/pls/portal/PORTAL.www_setting.render_css
HTTP Method	GET
Description	/pls/portal/PORTAL.www_setting.render_css: Access to Oracle pages could have an unknown impact.
Test Links	http://www.manancialdasaguas.com.br:80/pls/portal/PORTAL.www_setting.render_css http://198.100.157.60:80/pls/portal/PORTAL.www_setting.render_css
OSVDB Entries	OSVDB-3093
URI	/gif/hp_invent_logo.gif
HTTP Method	GET
Description	/gif/hp_invent_logo.gif: This device may be an HP printer/scanner and allow retrieval of previously scanned images.
Test Links	http://www.manancialdasaguas.com.br:80/gif/hp_invent_logo.gif http://198.100.157.60:80/gif/hp_invent_logo.gif
OSVDB Entries	OSVDB-0
URI	/gif/tricolor_ink_guage.gif
HTTP Method	GET
Description	/gif/tricolor_ink_guage.gif: This device may be an HP printer/scanner and allow retrieval of previously scanned images.
Test Links	http://www.manancialdasaguas.com.br:80/gif/tricolor_ink_guage.gif http://198.100.157.60:80/gif/tricolor_ink_guage.gif
OSVDB Entries	OSVDB-0
Host Summary	
Start Time	2018-12-17 17:45:21
End Time	2018-12-17 18:08:32
Elapsed Time	1391 seconds
Statistics	6544 items checked, 0 errors, 10 findings
Scan Summary	
Software Details	Nikito 2.1.5
CLI Options	-h http://www.manancialdasaguas.com.br -o resultados.html
Hosts Tested	1
Start Time	Mon Dec 17 17:45:20 2018
End Time	Mon Dec 17 18:08:32 2018
Elapsed Time	1392 seconds

© 2008 CIRT, Inc.

Figura 14 – Resultado dos testes no site manancial das águas com o nikito.

5

Conclusão

Após ter sido realizada a análise com as ferramentas de varredura para possíveis falhas de segurança no site manancial das águas com os *scanners* nikito e skipfish, foi possível analisar mais profundamente a aplicação web e realizar testes em um ambiente controlado com o consentimento de ambas as partes. Apenas a ferramenta skipfish apresentou possíveis riscos que poderiam haver segundo os resultados encontrados. Nos testes realizados o *scanner* nikito apresentou uma maior agilidade na busca de possíveis vulnerabilidades em relação ao *scanner* skipfish, além de conseguir listar um número maior de arquivos. Porém, o skipfish conseguiu apresentar possíveis falhas que poderiam ocorrer realizando uma análise mais minuciosa dos arquivos. A intenção do estudo e dos testes foi demonstrar algumas ferramentas para teste de segurança da informação e abordar os temas principais referentes a segurança da informação e como evitar possíveis ataques ou fraudes em sistemas web. A utilização das ferramentas de segurança é importante, pois, com testes antecipados para possíveis falhas de segurança, poderiam ser evitados transtornos financeiros, processuais e a restrição do acesso a dados sigilosos de indivíduos não autorizados.

Referências

- ASSUNÇÃO, M. F. A. Análise de eficiência na detecção de vulnerabilidades em ambientes web com o uso de ferramentas de código aberto. *Projetos e Dissertações em Sistemas de Informação e Gestão do Conhecimento*, v. 4, n. 2, 2018. Citado 2 vezes nas páginas 21 e 22.
- CARVALHO, F. R. de et al. Vulnerabilidades em aplicações web. *RE3C-Revista Eletrônica Científica de Ciência da Computação*, v. 8, n. 1, 2013. Citado 2 vezes nas páginas 10 e 21.
- CERT-INCIDENTES. 2017. <<https://www.cert.br/stats/incidentes/>>. Citado na página 18.
- CERT-PAISES-ORIGEM. 2017. <<https://www.cert.br/stats/incidentes/2017-jan-dec/top-cc.html>>. Citado na página 21.
- CERT-TIPOS-ATAQUE. 2017. <<https://www.cert.br/stats/incidentes/2017-jan-dec/tipos-ataque.html>>. Citado na página 18.
- COELHO, G. M.; MARTINS, D. M. S. Comparativos de ferramentas para sql injection em aplicações web. *Seminários de Trabalho de Conclusão de Curso do Bacharelado em Sistemas de Informação*, v. 3, n. 1, 2018. Citado na página 18.
- CORDEIRO, D. L. S.; FERNANDES, S. R. Análise da vulnerabilidade de sites web. *Seminários de Trabalho de Conclusão de Curso do Bacharelado em Sistemas de Informação*, v. 3, n. 1, 2018. Citado na página 22.
- COSTA, J. V. *Análise de vulnerabilidades de segurança em portais de governos eletrônicos*. 2017. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade Federal de Uberlândia, Brazil. Citado na página 10.
- ECOMMERCEBRASIL. 2017. <<https://www.ecommercebrasil.com.br/artigos/um-panorama-e-perspectivas-de-crescimento-para-o-e-commerce/>>. Citado na página 10.
- FEITOSA, E. et al. *Análise de Vulnerabilidades em Sistemas Computacionais Modernos: Conceitos, Exploits e Proteções*. 2012. 51 p. Citado na página 10.
- FONSECA, J.; VIEIRA, M.; MADEIRA, H. Testing and comparing web vulnerability scanning tools for sql injection and xss attacks. In: *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*. [S.l.: s.n.], 2007. p. 365–372. Citado na página 22.
- GOETHALS, K.; AGUIAR, A.; ALMEIDA, E. História da internet. *Faculdade de Engenharia da Universidade do Porto, Mestrado em Gestão da Informação*, 2000. Citado na página 13.
- MELLO, E. et al. Segurança em serviços web. 01 2006. Citado na página 10.
- MELLO, E. R. de et al. Segurança em serviços web. *Livro de Minicursos do VI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Santos: SBC, p. 1–48, 2006. Citado na página 10.

- OLIVEIRA, R. R. Criptografia simétrica e assimétrica-os principais algoritmos de cifragem. *Segurança Digital [Revista online]*, v. 31, p. 11–15, 2012. Citado na página 18.
- OWASP. 2017. <https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf>. Citado na página 15.
- PEIXINHO FRANCISCO MARMO DA FONSECA, F. M. M. L. Ivo de C. *Segurança de Redes e Sistemas: Princípios e práticas*. [S.l.]: RNP/ESR, 2013. 251 p. Citado na página 15.
- ROCHA, D.; KREUTZ, D.; TURCHETTI, R. Uma ferramenta livre e extensível para detecção de vulnerabilidades em sistemas web. v. 2, p. 55–65, 07 2012. Citado na página 14.
- ROCHA, T.; SOUTO, E.; MARTINS, G. Etssdetector: uma ferramenta para detecção automática de vulnerabilidades de cross-site scripting em aplicações web. In: . [S.l.: s.n.], 2013. Citado na página 16.
- SALAS, M. I. P.; MARTINS, E. A black-box approach to detect vulnerabilities in web services using penetration testing. *IEEE Latin America Transactions*, IEEE, v. 13, n. 3, p. 707–712, 2015. Citado na página 23.
- SANTO, A. F. S. do E. Segurança da informação. Citado na página 15.
- SCRIPTCASE. 2017. <<https://www.scriptcaseblog.com.br/sql-injection-injetando-dados-a-partir-de-inputs/>>. Citado na página 16.
- SHEZAN, F. H.; AFROZE, S. F.; IQBAL, A. Vulnerability detection in recent android apps: An empirical study. In: *2017 International Conference on Networking, Systems and Security (NSysS)*. [S.l.: s.n.], 2017. p. 55–63. Citado na página 15.
- STALLINGS, W. *Criptografia e segurança de redes: Princípios e práticas*. [S.l.]: Pearson Prentice Hall, 2008. v. 4. 481 p. Citado 2 vezes nas páginas 10 e 21.
- TAHA, A. M. d. C. et al. Guia de testes de segurança para aplicações web. Florianópolis, SC, 2017. Citado 2 vezes nas páginas 11 e 14.
- UTO, N.; MELO, S. Capítulo 6 vulnerabilidades em aplicações web e mecanismos de proteção. 12 2017. Citado na página 24.