Centro Universitário de Brasília
Instituto CEUB de Pesquisa e Desenvolvimento - ICPD

HENRY MENDES DE JESUS

DETECÇÃO DE PÁGINAS DE INTERNET DESFIGURADAS

Brasília
2017

HENRY MENDES DE JESUS

DETECÇÃO DE PÁGINAS DE INTERNET DESFIGURADAS

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como pré-requisito para obtenção de Certificado de Conclusão de Curso de Pós-graduação Lato Sensu em Rede de Computadores com Ênfase em Segurança.

Orientador: Prof. Dr. José Eduardo Malta de Sá Brandão

Brasília
2017

HENRY MENDES DE JESUS

DETECÇÃO DE PÁGINAS DE INTERNET DESFIGURADAS

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Pós-graduação Lato Sensu em Rede de Computadores com Ênfase em Segurança.

Orientador: Prof. Dr. José Eduardo Malta de Sá Brandão

Brasília, ___ de _____ de 2017.

Banca Examinadora

_____

Prof. Dr. José Eduardo Malta de Sá Brandão

_____

Prof. Dr. Sylas Rodrigues Mendes

_____

Prof. Dr. Gilson Ciarallo

AGRADECIMENTO(S)

Esse trabalho foi só foi possível pela colaboração de várias pessoas. Agradeço:

- A minha família por me ensinar que o conhecimento é essencial no mundo;
- Aos professores do curso que estão sempre dispostos a ensinar e auxiliar nessa grande jornada do aprendizado;

RESUMO

No mundo moderno a Internet avançou de tal forma que ela nos torna cada vez mais dependentes dela. As páginas de Internet revolucionaram a maneira de como lidamos com nossas ações corriqueiras nos dias atuais. Organizações de todos os tipos se beneficiam disso dispondo seus produtos e serviços na rede. Porém como qualquer tipo de tecnologia que surge ela traz seus problemas. Quanto a esses problemas poderíamos resumir em ataques aos sistemas web, cada vez mais frequentes. Esse vandalismo digital tomou um grande grau de notoriedade e tornou-se uma dor de cabeça dos profissionais de TI. Diversas técnicas de ataque foram ou estão sendo desenvolvidas. Elas têm o objetivo de desfigurar ou vandalizar páginas de Internet. Existem diversos motivos para tal artimanha. É uma forma mais fácil chamar a atenção usada pelos atacantes. Para combater esse tipo de ato ilegal, foram descobertas formas de detectar esse tipo de ataque. Nesse trabalho foram apresentadas propostas de mecanismos de detecção mais conhecidos e foi desenvolvido um protótipo que utiliza uma forma deles.

Palavras-chave: Hacking. Desfiguração. Páginas de Internet. HTML

ABSTRACT

In the modern world the Internet has advanced in a such a way that it make us more dependable of it. The Internet webpages revolutionized the manner that we deal with our day-by-day common activities. Many kind of organizations acquired benefits of that by making their products and services available through Internet. But like any other IT technology it brings its problems. These problems we could synthetize on website attacks which are becoming more frequent. This digital vandalism got too much level of notoriety and became a headache for IT professionals. Many kinds of defacing techniques were developed and there are ones still being on development. Their goal is to deface or vandalize webpages. There are many causes for doing that. It is the easiest way to call for attention used by attackers. To combat against this type of illegal activity defacing detections techniques were discovered. In this academic work demonstrates most known defacing detection techniques and one prototype was developed using one of these techniques

Key words: Hacking. Defacing. Webpages. HTML.

SUMÁRIO

INTRODUÇÃO

Com o surgimento das páginas de Internet foi possível rápido desenvolvimento de negócios e serviços na Internet. Diversas tecnologias existem para manter as informações essenciais por meios dinâmicos em páginas web. Dessa maneira, surgem também aqueles que usufruem de uma forma negativa: os atacantes ou vândalos digitais. Sua justificativa para tais atos vão desde satisfazerem seus próprios egos, deixando suas assinaturas ou marcas, ou para usar como uma forma de protesto ou chamado de hackitivismo. Diariamente as organizações têm prejuízos financeiros ou a perda de sua reputação para os seus clientes. As grandes corporações são os alvos preferidos dos atacantes pelo seu alto grau de notoriedade.

Esse trabalho tem o objetivo de apontar os principais meios de detectar alterações nas páginas *web* e reduzir o tempo necessário de reação a um incidente por meio de uma automação. No capítulo 1 são descritos os tipos de ataques mais comuns que dão oportunidades para o ataque de desfiguração. No capítulo 2 são demonstradas formas eficientes de detectar desfigurações. No capítulo 3 aborda tópicos relacionados à detecção de desfigurações. E por último, será abordado o funcionamento do agente de detecção de desfigurações, descrito no capítulo 4.

OBJETIVOS

Objetivo Geral

Implementar um agente de detecção de páginas desfiguradas com intuito de demonstrar, em um caso específico, que é possível agilizar o processo de detecção de um incidente de ataque de desfiguração de páginas de Internet.

Objetivo Específico

O presente estudo acadêmico visa:

- Demonstrar situações específicas em que o software deverá funcionar.
- Demonstrar o funcionamento em detalhes do algoritmo de detecção.
- Executar o software e demonstrar o tempo de detecção obtido.
- Demonstrar limitações do software implementado.

METODOLOGIA

Para atingir o objetivo proposto neste trabalho foram realizadas pesquisas e implementação do software, obedecendo à seguinte ordem:

- Pesquisas sobre formas de ataque de páginas de Internet.
- Pesquisas sobre formas de detecção de páginas desfiguradas.
- Implementação de um software utilizando um dos casos de detecção de páginas desfiguradas.
- Teste de execução do software implementado e análise dos resultados obtidos.

No ANEXO A, constará a descrição dos passos realizados na configuração do

software implementado e um exemplo de configuração.

1 CONCEITOS

A desfiguração de uma página web, de acordo com Kanti, Richariya e Richariya (2011, p. 222), “ocorre quanto o intruso de uma forma maliciosa altera a página web substituindo o seu conteúdo com material ofensivo ou provocativo”. É um tipo de ataque que causa um grande impacto e uma grande visibilidade no mundo. Pode ser definido também como vandalismo digital o qual expõe o usuário às informações falsas até que elas sejam corrigidas novamente pelo detentor.

Esse tipo de vandalismo digital é uma das maiores ameaças às empresas que dependem dos serviços web para os seus negócios. Além de ter seu sistema web comprometido, a reputação da empresa é afetada, causando muitas vezes prejuízos financeiros. Por isso devem-se priorizar as detecções de vulnerabilidades existentes na infraestrutura de rede da organização, que possam tornar esse tipo de ataque possível.

Com o sucesso do atacante em conseguir acessar o servidor web, trará a oportunidade de surgir vários outros tipos de ataque causados. Em um servidor web comprometido o atacante, dependendo do caso, poderá acessar a rede interna(*Intranet*) da organização e até mesmo chegar a uma estação de trabalho de um colaborador para obter documentos sigilosos. Isso vai depender do modo que o servidor está configurado em uma infraestrutura de rede.

1.1 Segurança da Informação

Toda organização além de possuir ativos físicos, detêm um bem fundamental para a sua sobrevivência: a informação. E como qualquer outro ativo este deve ser adequadamente protegido contra ameaças internas e externas. Nos dias atuais roubos e vazamentos de informações são casos críticos para uma empresa. Diversos tipos de dados (estratégicos ou não) das organizações estão contidos em sistemas informatizados. O que acontece se essas informações forem roubadas ou apagadas?

Para isso surge a Gestão da Segurança da Informação: seu objetivo é estabelecer procedimentos ou processos de proteger esse valioso ativo. Segundo Ferreira “a segurança da informação protege a informação de diversos tipos de ameaças garantindo a continuidade dos negócios, minimizando os danos e maximizando o retorno dos investimentos e das oportunidades”. Existem alguns elementos que a segurança da informação preserva.

Segundo Ferreira (2004, p.6):

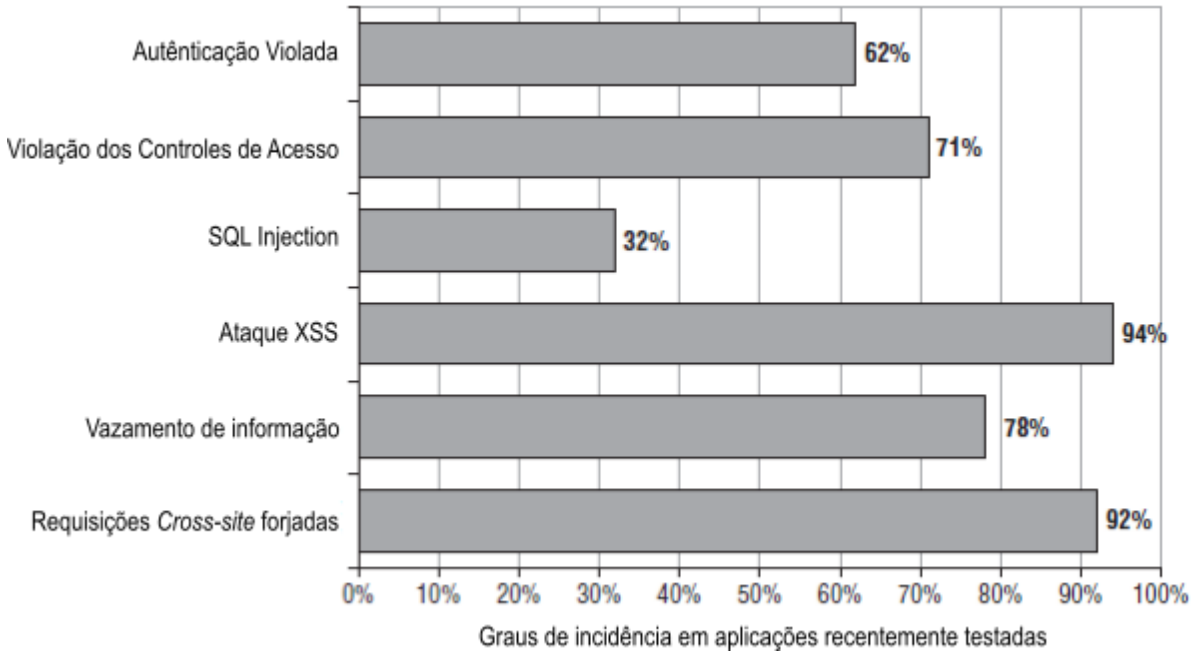
> a)Confidencialidade: somente pessoas autorizadas terão direito de acessar a informação;
>
> b)Integridade: os métodos de processamento e a consistência da informação são preservados;
>
> c) Disponibilidade: maneira de se garantir que somente usuários autorizados tenham acesso à informação e ativos correspondentes a ela;

De acordo com Krawetz (2007, p. 06) "segurança da informação é um termo relativo a um ambiente, baseado em medidas de riscos aceitáveis". Ela reduz as vulnerabilidades de um ambiente computacional seja na forma física quanto na forma lógica. Não deve-se restringir a segurança nos equipamentos(sejam firewalls, no IDS e etc.). Para Vacca (2010, p. 04) "A segurança é um processo. Não há nenhuma ferramenta que você possa configurá-la e esquecê-la por completo. Todos os produtos de segurança são tão seguros quanto as pessoas que são responsáveis pelas manutenções desses".

1.2 Desfiguração de sites web - Formas

Com advento da navegação segura através do protocolo SSL as transações de dados entre o cliente e o servidor web se tornaram mais seguras. Apesar de sua utilização em larga escala as aplicações web ainda possuem falhas de segurança. Segundo os testes de vulnerabilidades feitos por Stuttard e Pinto (2011, p. 64) foram encontrados diversos tipos de vulnerabilidades entre os anos de 2007 a 2011. Segundo o gráfico na Figura 1 eles dividiram em categorias de ataques onde esses websites são vulneráveis:

Figura 1 – Graus de incidência em aplicações recentemente testadas



Fonte – Gráfico adaptado dos autores Stuttard e Pinto (2011, p. 64)

1.2.1 *Violação de acesso ao login*

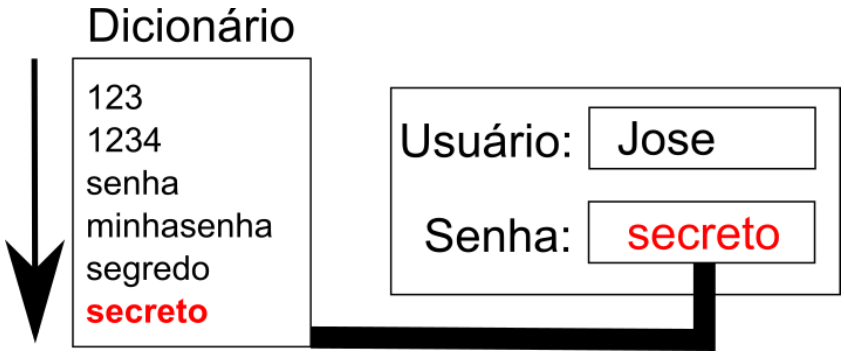
Uma forma mais objetiva de conseguir acesso a um sistema é pela porta da frente, ou seja pelo *logon*. De acordo com Stuttard e Pinto(2011, p. 64),a má implementação do mecanismo de *logon* dá oportunidade ao atacante em, conseguir com sucesso, efetuar a autenticação apenas adivinhando senhas fracas através do ataque de força bruta ou acessar o sistema sem efetuar nenhum *logon*. Geralmente o atacante tenta explorar as falhas vindas do *webmaster* ou administrador de rede,

do sítio de Internet, ao definir senhas fracas. Conseguindo essas senhas de administração da página de Internet é uma possível de desfigurar uma página pois ele age como o administrador. Vale salientar que o nome do usuário deve ser conhecido pelo atacante. Existem duas maneiras conhecidas de ataque às senhas: ataque por força bruta e por dicionário.

Ataque por Força Bruta (Brute Force Attack): para Apelbaum(2007, p. 40) é um ataque automatizado que realiza uma geração de combinações possíveis de caracteres e números para um determinado dicionário.

Ataque por dicionário (Dictionary Attack): para Apelbaum (2007, p. 40) é uma forma de adivinhar uma senha por tentativa e erro.

Figura 2 – Modo ilustrado de um ataque de força bruta



Fonte – Produzido pelo autor do trabalho

A figura acima é uma ilustração de uma ferramenta qualquer usando um ataque de dicionário. Com sucesso ela obteve a senha utilizada pelo usuário “Jose”.

1.2.2 *Violação de controles de acesso*

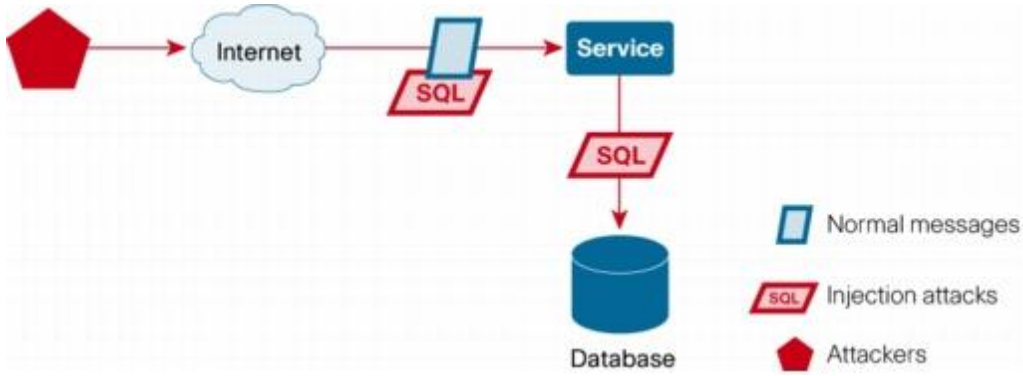
De acordo com Stuttard e Pinto(2011, p. 64) esse tipo de ataque ocorre quando uma aplicação web falha em proteger seus dados e funcionalidades dando ao atacante possibilidades de acesso a dados pessoais de outros usuários contidos no servidor dessa aplicação. Além disso, outra falha grave são os mecanismos de controle de acesso mal implementados que permitem o atacante usar ações privilegiadas do administrador do sistema com usuários comuns dentro do sistema web atacado.

1.2.3 *SQL Injection*

Sua origem vem da má implementação do código de uma aplicação web. Os principais objetivos desse ataque para Oriyano(2014, p. 290) é tipicamente um resultado de falhas nas aplicações *web* e não do banco de dados utilizado. Para ocorrer esse tipo de falha é devido à ausência de validação de dados os quais são submetidos pela aplicação web ao banco de dados.

De acordo com Stuttard e Pinto(2011, p. 08) esse tipo de ataque ocorre quando o atacante envia informações anormais (com códigos SQL) as quais interferem na interação da aplicação web com o seu banco de dados. Dessa maneira ele consegue, com sucesso, obter informações confidenciais da aplicação, interferir em sua lógica de funcionamento ou executar comando diretamente no próprio servidor de banco de dados. Dessa forma o atacante consegue desfigurar o website pois seu acesso já foi garantido pelo banco de dados. No diagrama da Figura 3 demonstra o processo de ataque via *SQL Injection*.

Figura 3 – Processo do ataque via SQL Injection



Fonte – http://www.cisco.com/c/en/us/products/collateral/application-networking-services/ace-xml-gateways/prod_white_paper0900aecd80669393.html/

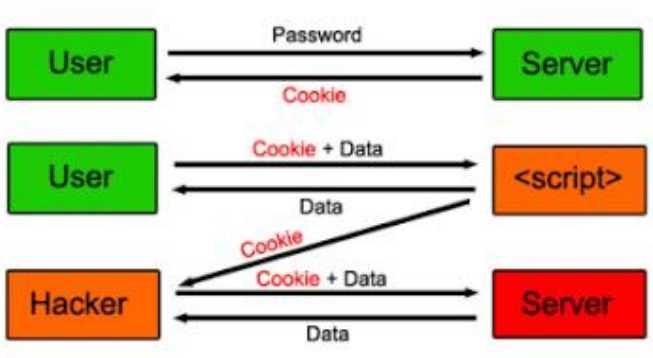
Conforme o diagrama acima o atacante pela Internet utiliza, através de mensagens normais, códigos maliciosos SQL em um sítio web. Seu objetivo é obter informações adicionais, vindas do banco de dados como nomes de usuários e senhas, por exemplo.

1.2.4 *Cross-site scripting (XSS)*

É um tipo de vulnerabilidade muito explorada nas aplicações web encontradas na Internet. Esse tipo de técnica afeta uma vasta gama de aplicações desde as mais comuns até mesmo as que possuem finalidades críticas como os sistemas bancários. O Atacante se aproveita dos meios fracos ou ausentes de validação dos dados, de uma aplicação web, para inserir códigos maliciosos que façam ele adquirir algum tipo de privilégio não autorizado.

De acordo com Oriyano (2014, p. 290) esse tipo de ataque pode ocorrer de várias maneiras. Principalmente quando os dados de algum tipo são enviados a uma aplicação web através de uma fonte não segura (na maioria dos casos em requisições web). No diagrama da figura 4 abaixo mostra dois fluxos de um acesso a uma página de Internet, um normal e um afetado pelo atacante, de um processo de ataque de XSS:

Figura 4 – Processo do ataque via XSS



Fonte – http://smartechverse.blogspot.com.br/2015/07/xss-attacks.html

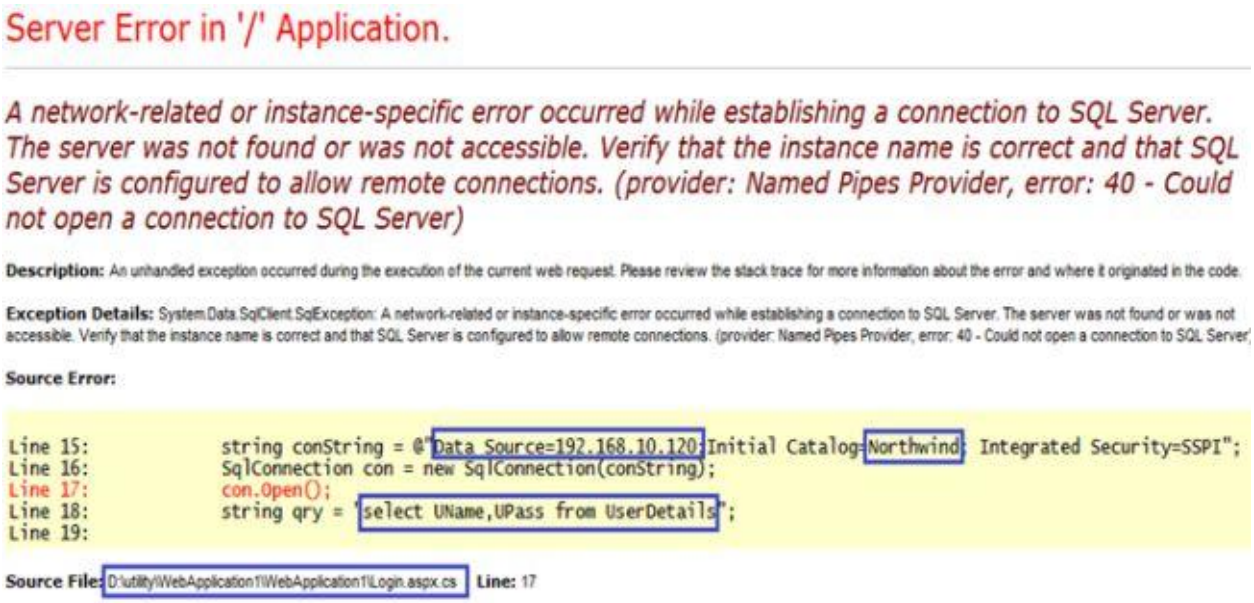
No primeiro fluxo, o usuário envia sua senha para o servidor e recebe um cookie após a autenticação. No segundo fluxo, o usuário envia as informações para o

servidor afetado por códigos maliciosos: eles interceptam os dados do usuário e enviam para o atacante.

1.2.5 *Vazamento da Informação*

De acordo com Stuttard e Pinto (2011, p. 08) é quando uma aplicação web divulga informações privilegiadas que podem ser utilizadas pelo atacante. Geralmente essas informações vêm através de uma falha em lidar com erros internos da aplicação ou algum outro tipo de comportamento. A figura 5 ilustra esse problema em um erro de execução de uma página web.

Figura 5 – Exemplo de vazamento de informação através de erro da aplicação *web*



Fonte – http://resources.infosecinstitute.com/exploiting-information-disclosure-part-1/

Segundo a figura acima ocorreu um erro de execução da aplicação web. Como não houve tratamento de erros e configuração para restringir informações confidenciais (do servidor) houve o vazamento das informações. Respectivamente: o endereço IP do banco de dados, nome do banco de dados, código SQL executado pela aplicação e por fim o script que está contido essas informações.

1.2.6 *Requisições cross-site forjadas*

Requisições cross-site forjadas ou chamado de CSRF(Cross-Site Request Forgery) não depende de modificações na página a ser utilizada pela vítima como no XSS. Segundo Shema(2010, p.28) "(..) o ataque CSRF força o navegador da vítima a fazer requisições sem o conhecimento ou aprovação dela". O CSRF utiliza meios normais de transações em um website.

Primeiramente o atacante pode utilizar diversos meios seja por um site falso ou por meio da engenharia social (e-mails). Ele se aproveita da confiança de uma sessão feita pelo usuário, em uma determinada aplicação web. Posteriormente pode utilizar os cookies temporários, da sessão criada, para fazer qualquer outra transação no futuro. Lembrando que o usuário precisa ter acessado a página falsa para ocorrer o ataque. A figura 6 é um exemplo muito utilizado pelos atacantes para efetuar transações bancárias ilícitas pela Internet usando a conta da vítima.

Figura 6 – *Processo de um ataque CSRF*

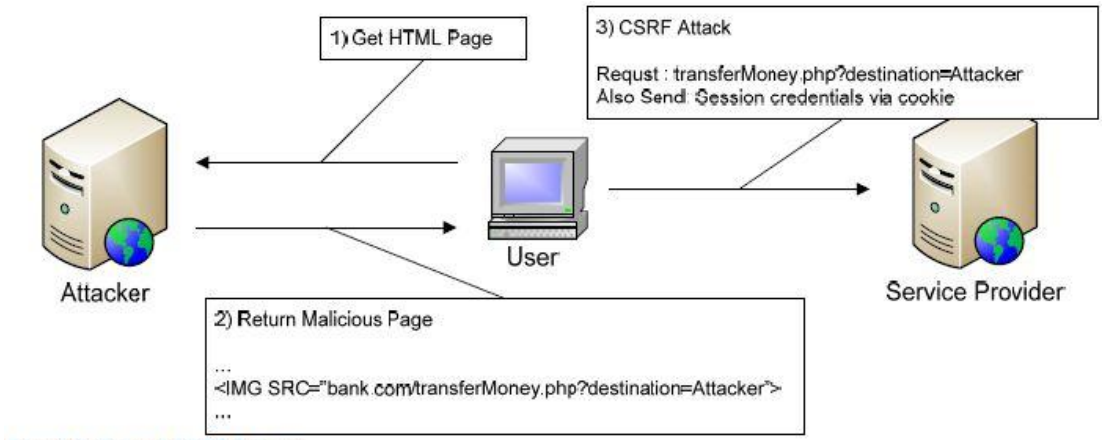


Figure 6: Typical CSRF Scheme

Fonte – http://www.cse.wustl.edu/~jain/cse571-09/ftp/social/

Conforme a figura 6, a vítima é induzida a acessar a página do atacante. Após o acesso são enviadas requisições escondidas no elemento HTML de imagem: o IMG. Esta requisição induz o Banco(*Service Provider*) a transferir o dinheiro da vítima para a conta do banco do atacante sem o consentimento do usuário de uma maneira escondida.

2 MECANISMOS DE DETECÇÃO

Existem diversas maneiras de detectar esses tipos de desfiguração de *websites*. Todas as formas que serão citadas abaixo possuem um grau de eficiência para detectar uma desfiguração. É de suma importância que haja mecanismos apropriados para detecção, pois caso ao contrário da alteração passará despercebida pelo administrador. Os mecanismos abaixo são os mais comuns para o desenvolvimento de um sistema de detecção de páginas desfiguradas. Deve-se salientar que os métodos abaixo podem ser combinados para aumento significativo na detecção de um ataque das páginas web.

2.1 Comparação por *Hash*

Essa função de assinatura é de somente um caminho, pois não há um processo de decodificação da mensagem de maneira fácil. Esse tipo de algorítmo cria uma saída com um número fixo de números de caracteres independente dos números de caracteres que são usados na entrada. Segundo Oriyano(2014, p. 68) o *hashing* permite que você facilmente detecte qualquer mudança nos dados mesmo que seja de uma pequena proporção. O resultado será um *hash* completamente diferente do original. No diagrama da figura 7 mostra a palavra “informação” sendo transformada em por um algorítmo *hash* qualquer.

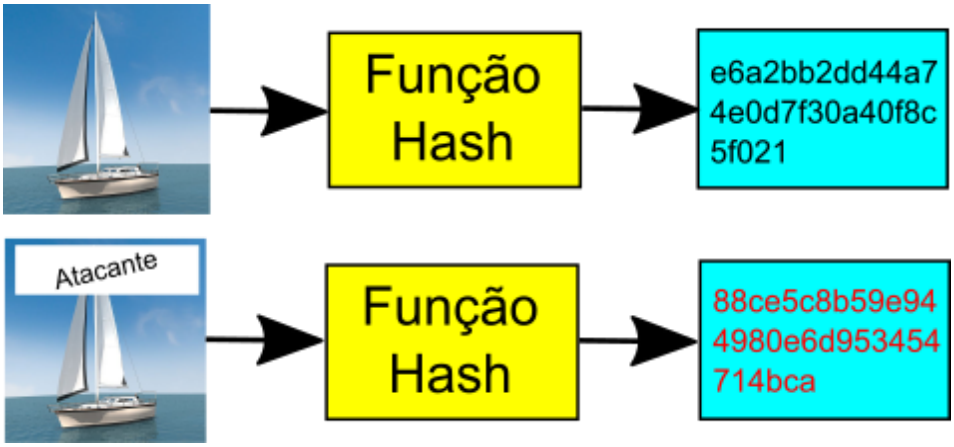
Figura 7 – Utilização de uma função *hash*

Fonte – Produzido pelo autor do trabalho

Pode-se exemplificar sua utilização em uma página de Internet gerar uma assinatura de cada conteúdo existente nela tais como imagens, scripts e etc. Dessa forma poderá haver uma comparação entre o *hash* guardado com o *hash* atual. Pode-se utilizar outras técnicas que utilizando esse algoritmo.

O diagrama da figura 6 é um exemplo de comparação de arquivos de imagem, de uma página *web*, através dos seus *hashs*. A segunda figura foi alterada com a palavra “Atacante”. Percebe-se a diferença das imagens através do segundo *hash* (em itálico).

Figura 8 – Utilização de uma função hash

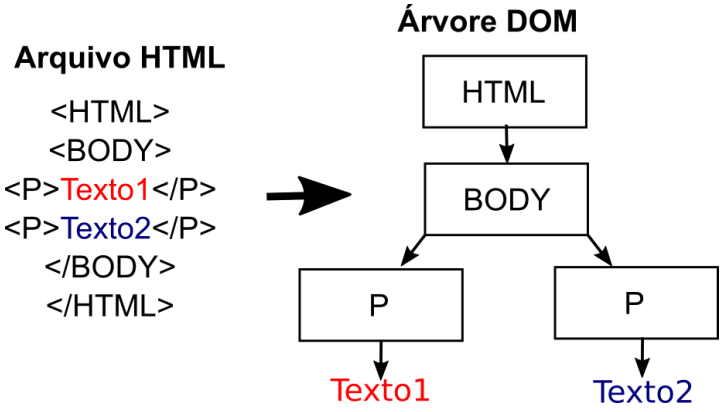


Fonte – Produzido pelo autor do trabalho

2.2 Árvore DOM (HTML)

Primeiramente, DOM é um Modelo de Objetos de Documento *(Document Object Model)*. Segundo Robie (2017) o DOM é análogo a uma árvore de objetos que compõem uma página HTML. Também é uma biblioteca que permite os desenvolvedores manipular a estrutura de um documento de hipertexto. A W3C define padrões para essa biblioteca. Para que ela seja utilizada em diversos tipos de ambientes e aplicações com qualquer tipo de linguagem de programação. A figura 9 ilustra como um arquivo com a linguagem de marcação HTML é representado em uma árvore DOM.

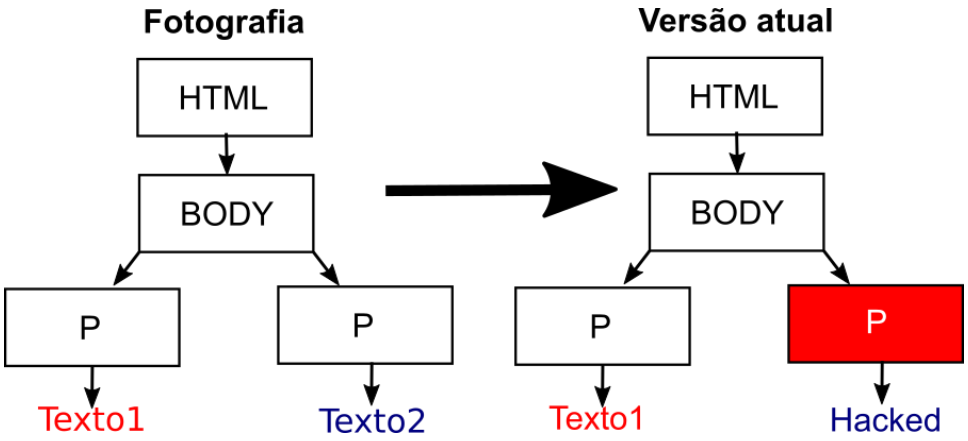
Figura 9 – Representação do HTML em uma àrvore DOM



Fonte – Produzido pelo autor do trabalho

Uma forma de conseguir detectar uma mudança, em uma página de Internet, é manter uma fotografia armazenada e comparar com uma versão recente capturada pela ferramenta de detecção. Uma varredura é feita sentido *top-down* na Árvore DOM comparando cada objeto entre os dois arquivos. No final do processo pode se definir métricas, como número de elementos alterados, para verificar se houve realmente uma desfiguração. A figura 10 demonstra como é feito o processo de comparação entres os elementos contidos nas versões da página web.

Figura 10 – Comparação entre versões da mesma página web



Fonte – Produzido pelo autor do trabalho

De acordo com a figura 10 houve uma comparação com o DOM original (fotografia) com um DOM obtido recentemente(versão atual). Nota-se a diferença do conteúdo no segundo elemento “P”. Somente um elemento pode não ser o suficiente para detectar uma desfiguração. Devem-se estabelecer critérios de comparação que envolva mais de um elemento DOM para reduzir um falso positivo.

2.3 Comparar a diferença das requisições HTML

Qualquer elemento obtido em um navegador de Internet (uma página, uma figura, um som e etc.) utiliza o protocolo HTTP o que significa “Protocolo de Transferência de Hipertexto”. De acordo com a (EVANS,2017) o protocolo HTTP é um protocolo baseado em requisição/resposta. Isso significa que o seu computador envia uma requisição para o servidor para obter um elemento e o servidor então interpreta essas informações e responde ao cliente com o envio do elemento pedido (caso ele exista). A figura 11 ilustra o funcionamento do protocolo HTTP através de uma requisição de uma página de Internet:

Figura 11 – Exemplo de requisição HTTP

```
GET /dumprequest HTTP/1.1
Host: rve.org.uk
User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:50.0) Gecko/20100101
Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://www.google.com.br/
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Fonte: http://rve.org.uk/dumprequest

A figura acima é demonstra as informações enviadas ao servidor *web*. Existem informações importantes como o seu endereço IP, o documento pedido, a versão do navegador utilizado, a página que você utilizou para fazer a requisição o idioma padrão e seus *cookies.*

Como uma requisição de um website retorna HTML pode ser armazenada uma fotografia de uma requisição normal para que seja comparada com uma requisição recente. Com isso pode ser utilizado ferramentas que façam a comparação das duas requisições, a guardada como fotografia com a requisição recente. Como há conteúdo dinâmico em cada requisição, embutido no código HTML, deve-se estabelecer um limite de aceitação de mudanças.

2.4 Palavras Comuns

Quando ocorre um caso de vandalismo digital é muito comum que os atacantes deixem assinaturas. Geralmente com termos conhecidos. Podem ser utilizados mecanismos que procurem palavras específicas como *"Hacked"* ou "Hackeado". Porém pode haver textos bem mais elaborados. A tabela abaixo é um exemplo real de assinaturas de atacantes, no ano de 2017, obtidas em WordFence.com:

Tabela 1 – Exemplo de assinaturas deixadas em páginas de Internet por atacantes

Assinatura
Hacked By MuhmadEmad
Hacked By SA3D HaCk3D
by w4l3XzY3
Hacked By Imam
Hacked By BALA SNIPER
Hacked By TheWayEnd
Hacked By GeNErAL
Hacked By RxR HaCkEr
Hacked By HolaKo
Hacked By XwoLfTn
Hacked By Dr.SiLnT HilL
hacked By Fallag Gassrini
Hacked by Shade
Hacked By chinafans
Hacked By Black Sniper
By NeT.Defacer
HackeD By Skidie KhaN
By Hawleri_hacker
Hacked by Sxtz

Fonte: https://www.wordfence.com/blog/2017/02/rapid-growth-in-rest-api-defacements/

3 TRABALHOS RELACIONADOS

Nesse capítulo, serão abordados alguns exemplos de soluções de detecção de páginas desfiguradas, encontradas em artigos acadêmicos. Muitas dessas soluções possuem uma forma mais complexa de funcionamento. Outras possuem um custo maior de desenvolvimento.

3.1 Detecção de desfigurações em um navegador de Internet

Kanti, Vineet e Vivek (2011, p.311) desenvolveram uma forma de detectar a desfiguração de páginas através de comparação de *hashes* das páginas web. Esse algoritmo calcula constantemente o *checksum* para verificar uma possível desfiguração. A tabela 3 demonstra a correlação entre os *checksums* e os *hyperlinks* de páginas *web* que são verificados:

Tabela 3 – Descrição detalhada de cada arquivo ou diretório que compõe o agente

Web Page Links	Checksum
p1	c1
p2	c2
p3	c3
p4	c4
p5	c5
p6	c6
p7	c7

Fonte – Produzido pelo autor do trabalho

O *hash* da versão original, do arquivo HTML, é armazenado em um banco de dados. Onde ci representa o código de *checksum* da página (pi). Primeiro será gerado o *checksum* para a página *web* a ser verificada. Está página se for nova será salva em um banco de dados de *checksums* como ci. Se houver mudanças na página de Internet, em seu *hash*, será calculada como nci. Será feita uma comparação entre nci e ci. Caso os dois *hashes* forem diferentes, a página será registrada como desfigurada. Os algoritmos de *hash*, para essa comparação, poderá ser MD5 ou SHA1.

O protótipo foi implementado na linguagem de programação C#.net. Ele também incluiu um navegador de Internet. O administrador precisa usar esse navegador para abrir páginas de Internet. No caso ocorra uma desfiguração, o administrador será alertado através de uma mensagem através do *browser*.

3.2 Detecção de desfigurações pela diferença dos dados

Kanti, Vineet e Vivek (2012, p.252) adicionaram a funcionalidade em detectar o ponto onde ocorreu a desfiguração da página com algoritmos que detectam diferenças.

Existe certa dificuldade em comparar dois arquivos de textos grandes. Dessa forma é mais viável comparar números em todas as linhas de texto. Se as linhas de texto forem idênticas então números idênticos serão computados. Há algumas opções como a remoção de todos os espaços e comparação de letras de forma não sensitiva. O algoritmo principal de comparação é baseado em dois tipos de métodos de detecção de diferenças. O LCS *(Conquer to Divide)* e o SMS *(Shortest Middle Snake)*.

Nesse projeto, foi demonstrada a capacidade de reduzir o *overhead*[1] no processamento de *hashes* para todas as páginas de um website. A redução ocorre quando são selecionadas algumas páginas de Internet (as mais importantes). Porém sistemas de detecção de integridade possuem limitações: eles somente servem para páginas estáticas. Os algoritmos de detecção de diferenças ajudam a localizar o ponto onde o arquivo HTML foi modificado.

[1] http://www.webopedia.com/TERM/O/overhead.html

3.3 Detecção de desfigurações através de um módulo no Apache

Verma e Sayyad (2015, p.134) desenvolveram formas de impedir que uma página desfigurada de um website seja entregue ao cliente. Para isso eles utilizaram a última versão 2.4 do servidor web Apache no sistema operacional GNU/Linux. O motivo de utilizar o Apache é sua popularidade. Ele corresponde a quase 56.2% dos websites no mundo.

Eles dividiram em dois tipos de módulos: Módulo Gerador de Conteúdo e o Módulo de Filtro. Na primeira fase foi desenvolvido o módulo gerador de conteúdo. Ele utiliza a função *handler* do Apache para receber o conteúdo da requisição HTTP. Com isso esse módulo gera um *hash* para os arquivos relacionados ao website no servidor Apache. Ambos os módulos foram compilados usando a ferramenta do Apache chamada APXS.

Na segunda etapa foi criado um banco de dados o qual mantém o arquivo HTML e seu *hash*. E por fim foi desenvolvido o módulo de filtro que compara os *hashes* armazenados do banco de dados com os gerados em tempo de execução. Outro detalhe é a configuração que deve ser realizada desses módulos no arquivo de configuração do servidor Apache.

Um exemplo prático de funcionamento é supor a existência de um arquivo HTML (sampl.html). Ele existe no arquivo raiz do servidor *web* Apache (/var/www/html/). Quando há uma requisição de um cliente para esse arquivo, o módulo implementado calcula o *hash* dessa página em tempo de execução e compara com o *hash* armazenado. Caso os dois *hashes* forem idênticos ele envia a página para o cliente. Porém a solução proposta tem limitações de detectar desfiguração de somente páginas HTML estáticas.

Da mesma forma que o trabalho de Kanti et al. (2011), a técnica não avalia alterações de imagens.

3.4 Detecção de desfigurações pelo reconhecimento de imagens

Nesse artigo, o Borgolte e Kruegel e Vigna(2015, p. 595) propõem uma forma de detecção de páginas de Internet desfiguradas por meio da Inteligência Artificial[2]. Meerkat, como é chamado, aprende por meio de aprendizado de uma rede neural por meio de assinaturas que os atacantes deixam em uma página desfigurada.

Isso é possível por meio de arquivos de imagem gerados através de um dos bancos de dados mais famosos mantido pelo THC.org. Para o aprendizado as imagens obtidas são divididas em partes menores. Elas não podem ser muito pequenas para não aumentar as chances de falsos positivos. Elas não podem ser muito grandes, pois aumentaria de uma forma considerada o tempo de aprendizado. Os métodos utilizados para detecção seriam inicialmente pela captura dos logotipos utilizados por grupos de atacantes, erros tipográficos ou gramaticais, *leetspeak*, combinação de letras e combinação de cores.

A tecnologia utilizada foi o framework Caffe, desenvolvido em Python. Ele foi utilizado devido seu alto desempenho e facilidade de uso. Em sua versão original o Caffe não fornece todas as funcionalidades. Então os responsáveis pelo projeto fizeram modificações. Foram utilizadas 125 máquinas em cluster para processamento as capturas de tela feitas de websites desfigurados. O treino da rede neural foi feito em GPUs nessas máquinas devido a sua rapidez. Após isso eles podem ser executados em CPUs.

No modo de operação, no mundo real, o Meerkat é um serviço de monitoração que alertas desfigurações de websites, através do endereço URL, fornecido pelo operador. Seu ciclo de funcionamento pode ser em minutos (ou até segundos).

De certa forma o sistema é muito eficiente na detecção de páginas desfiguradas. Porém demanda exige uma demanda grande de recursos computacionais. Seu desenvolvimento é complexo devido à utilização de Inteligência Artificial. Por utilizar redes neurais, ele requer um constante aprendizado. Caso haja novos grupos de atacantes. Outra coisa é o volume de informações. Ele necessita de um grande volume de informações para atingir sua eficiência.

[2] https://techterms.com/definition/artificial_intelligence

Portanto, a proposta de Borgolte e Kruegel e Vigna (2015) busca assinaturas de ataques enquanto a proposta do presente trabalho identifica as alterações da página web, sem a necessidade de conhecer os ataques.

3.5 Detecção de desfigurações por assinaturas de componentes web

Este projeto tem como função de detectar mudanças nos componentes de uma página web. De acordo com Gurjwar, Sahu e Tomar (2013, p.73) a solução proposta extrai componentes que compõem uma página web e verifica a integridade deles comparando a versão original o qual fica armazenado em um repositório de componentes *web*.

O projeto é composto por quatro módulos, sistema de monitoração, extração de componentes *web*, pré-processamento de textos e imagens, verificação de integridade e repositório de componentes web. Seu modo de detecção de páginas desfiguradas é a forma de uso da técnica de *polling*. Essa técnica é empregada periodicamente em certos grupos de dados para verificação de mudanças. O processo de verificação segue pela extração de como textos de imagens.

Com isso é verificado a integridade desses tipos de dados com algoritmos de hash. Nessa aplicação foram utilizados para verificação de integridade de textos e imagens algoritmos como CRC32, MD5 e SHA512. Para as imagens foram utilizados PSNR e SSIM. O resultado foi mais satisfatório utilizando o SHA512 para textos ou imagens. Pois esse algoritmo é mais preciso em detectar pequenas alterações.

A proposta de Gurjwar, Sahu e Tomar (2013) precisa armazenar muitos dados sobre os objetos monitorados, divergindo da proposta do presente trabalho, que necessita armazenar poucos dados, tornando a comparação simples.

3.6 Detecção de páginas desfiguradas em um Sistema Especialista

Nesse artigo, para resolver o problema da alta taxa de falsos positivos, os Davanzo, Medvet e Bartoli (2011, p.12522) propõem uma forma de um Sistema Especialista o qual utiliza a Inteligência Artificial. A forma mais plausível para isso foi

analisar por três meses amostras de 320 tipos de desfiguração de páginas web coletadas do site ZoneH.org. Dessa maneira, foi desenvolvido um protótipo de framework para realizar o processo de detecção. Neste *framework* encontram-se os elementos: *refiner*(refinador), *sensors*(sensores) e *agregator*(agregador).

Em seu modo de operação, o URL do site a ser detectado é utilizado como entrada para o *refiner*(refinador). Nele existem diversos tipos de sensores. Os sensores têm como objetivo de capturar os elementos da página web. Eles são utilizados para classificar esses elementos em vetores com valores numéricos. Esses vetores servem de entrada para o *agregator*(agregador). Ele fará a comparação utilizando sua base de conhecimento. Seu conhecimento analisará se a página de Internet foi desfigurada.

4 PROPOSTA DE DETECÇÃO DE DESFIGURAÇÃO DE SITES

Em uma empresa de Tecnologia da Informação, a equipe de segurança da informação precisa aumentar sua eficiência em detectar a desfiguração de páginas de Internet. Visto que há muitos sistemas web para monitorar e o processo de detecção de páginas desfiguradas é manual. Essa equipe possui a necessidade de uma automação que detecte alterações na página inicial dos sítios dos seus clientes para diminuir o tempo gasto na detecção é substituir o processo manual o qual é demorado.

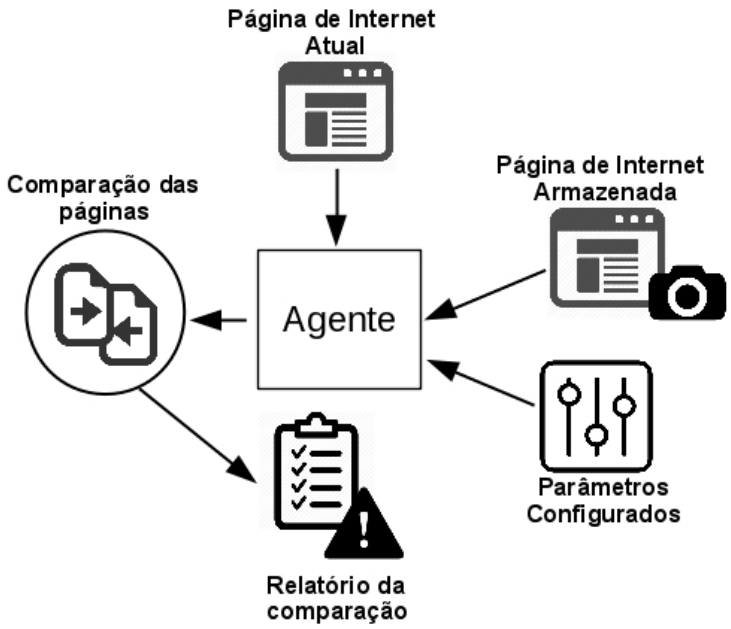
4.1 Proposta

A proposta desenvolvida utilizou um dos recursos de detecção citados: a comparação por *hash* entre arquivos. Será a divisão da página de Internet em partes e por meio de estatística, retornando um valor, em porcentagem, da alteração da página inicial. Além disso, deverá ser considerado um valor limite para alteração, pois visa reduzir os falsos positivos. O termo fotografia será utilizado para definir uma página original armazenada para comparação.

4.1.1 Funcionamento

O operador fará um cadastro da página de Internet a ser monitorada. Um agente será executado com uma frequência (configurada pelo usuário) e comparará a versão da página armazenada (uma fotografia) com uma atual. O agente fará as comparações por meio dos *hashes* gerados das partes do arquivo de imagem gerado da página web. Ele fará um alerta caso o número de modificações dessas partes seja além do limite configurado pelo usuário. A figura 12 ilustra o processo de execução do agente de detecção de páginas desfiguradas:

Figura 12 – Comparação entre versões da mesma página web



Fonte – Produzido pelo autor do trabalho

Conforme a figura acima o agente deverá considerar parâmetros configurados pelo operador (limite de alterações e intervalo de execução) e a fotografia da página de Internet armazenada como obrigatórios. Após a comparação entre a página atual e a página armazenada será gerado um relatório de alterações detectadas. O agente poderá alertar o administrador do servidor web com e-mails ou mensagens pelo celular.

4.1.2 Arquitetura

O protótipo utilizou ferramentas *Open Source* para redução de custos. Foi implementado na linguagem de programação *PHP* e alguns módulos em *Shell Script.* O processo gráfico de criação de arquivos de imagem foi feita pela ferramenta *wkhtmltoimage.* A divisão da imagem gerada, em partes, será feita pela ferramenta *ImageMagick.* Os seguintes recursos computacionais foram utilizados nos testes de serviço de monitoramento:

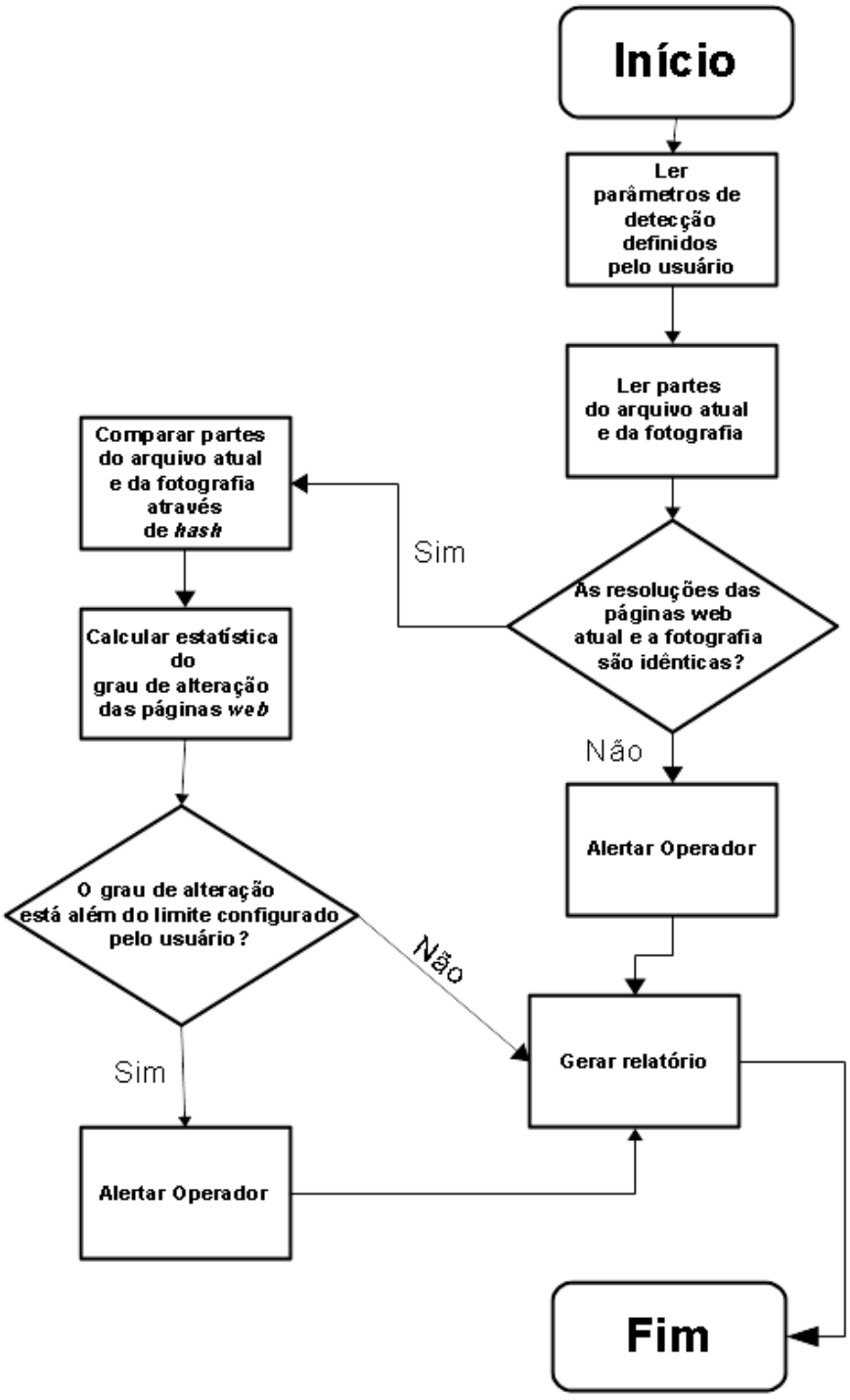
- 8 processadores x64
- 16 GB de memória RAM
- 100 GB de disco
- 1 Interface de rede Gigabit
- Sistema operacional Debian GNU/Linux 8.7.0 64 bits

Em futuros ambientes poderá ser utilizada qualquer outra distribuição Linux desde que haja as ferramentas acima disponíveis para instalação.

4.1.3 Lógica de comparação

O diagrama de fluxo da figura 13 demonstra o processo de execução do agente em detectar a página desfigurada.

Figura 13 – Comparação entre versões da mesma página web



Fonte – Produzido pelo autor do trabalho

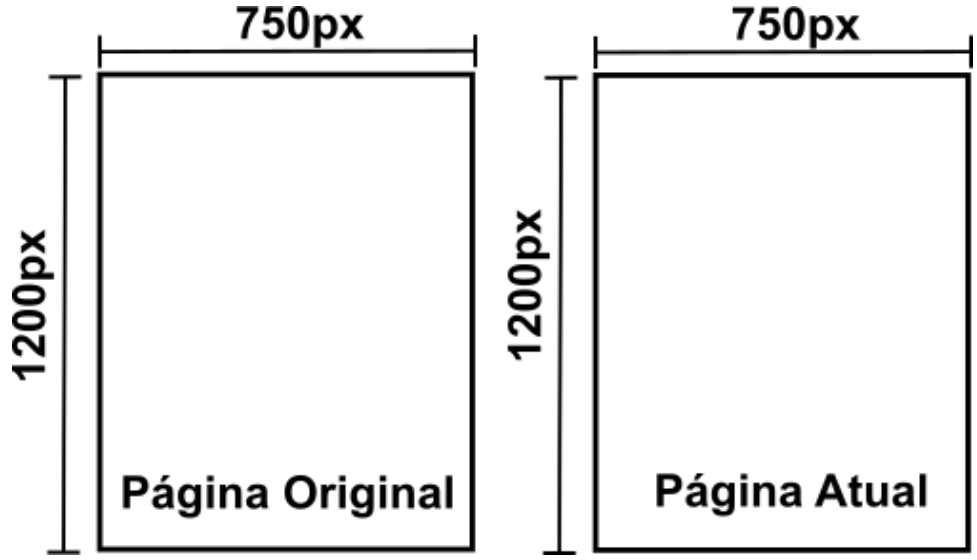
Conforme a figura acima o fluxo deve ser lido conforme as direções das setas definidas entre o “Início” e o “Fim”. Onde se lê fotografia deve-se interpretar como o arquivo HTML, armazenado pelo agente. Um detalhe importante é que existem três estados possíveis das comparações: grau de alteração além do limite, dimensões das páginas diferentes e o grau de alteração dentro do normal. Sendo que respectivamente as duas primeiras geram alertas. A geração de relatório sempre será feita pelo agente.

4.1.4 Comparação de resoluções

Antes de realizar outros tipos de comparações, mais sofisticados, o protótipo realiza uma verificação se as resoluções são iguais, entre a versão original e a atual da página de Internet. Nessa comparação, utilizam-se suas alturas e larguras como pontos de referência. Caso não satisfaça essa condição de igualdade, há uma grande possibilidade de existir uma desfiguração de página. Existem dois tipos de resultado dessa comparação.

Resoluções iguais: nesse caso quando comparação de alturas e larguras entre as duas versões da página de Internet forem iguais, há uma grande chance de página *web* estar sem alterações. A figura 14 ilustra esse cenário de comparação.

Figura 14 – Comparação entre versões da mesma página web

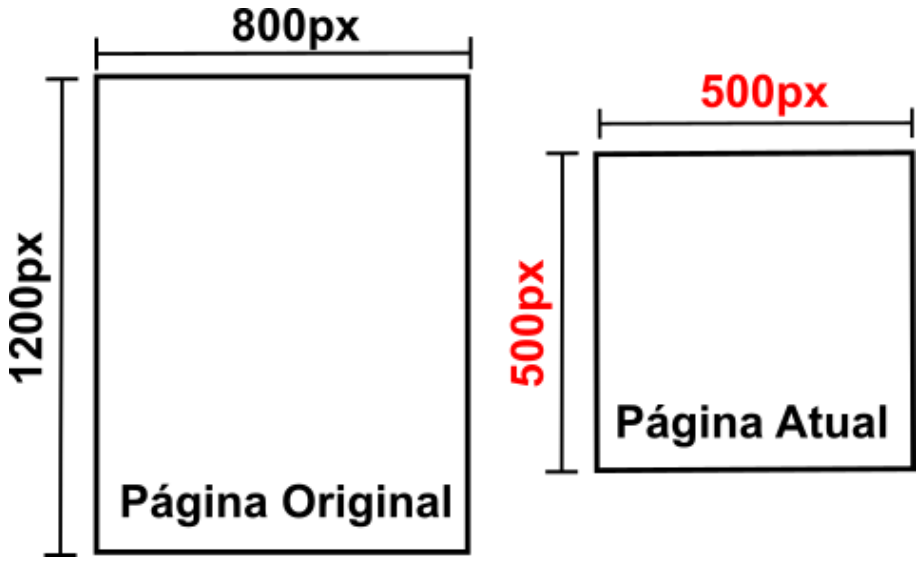


Fonte – Produzido pelo autor do trabalho

Nesse caso, conforme a figura acima, as duas páginas possuem resoluções idênticas, 750x1200, usando como referência suas dimensões, em pixels[3]. O protótipo poderá prosseguir com outros meios de teste para concluir sua execução.

Resoluções diferentes: nesse caso a diferença foi detectada. Há uma grande chance de existir uma desfiguração. Um alerta é gerado para o operador. A figura 15 ilustra esse cenário de comparação.

Figura 15 – Comparação entre versões da mesma página web



Fonte – Produzido pelo autor do trabalho

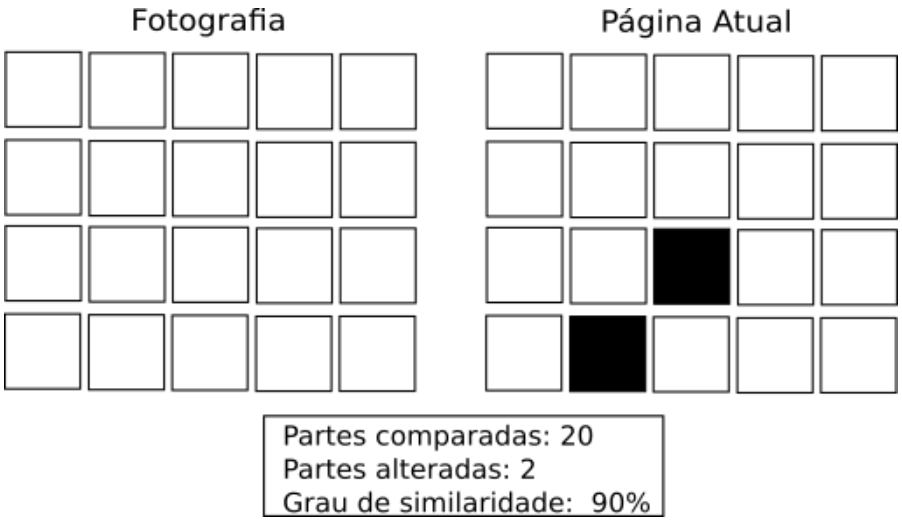
Conforme a figura acima, as duas páginas possuem resoluções diferentes. Considerando como referência a resolução original(800x1200) com a versão atual(500x500). O protótipo abortará sua execução. Antes disso será gerado um alerta ao operador, pois há uma grande chance da página atual estar desfigurada.

4.1.5 Grau de similaridade

Para realizar a detecção, o software implementado compara as partes da página web resultando no grau de semelhança. A figura 16 demonstra de uma forma simplificada o processo em obter o valor do grau de semelhança.

[3] http://www.webopedia.com/TERM/P/pixel.html

Figura 16 – Comparação entre as partes para obter similaridade



Fonte – Produzido pelo autor do trabalho

Conforme a figura acima, as páginas de Internet (atual e a fotografia) foram divididas em 20 partes ou quadrados. Na página atual foram detectadas duas mudanças. Com isso se obteve um estado dentro do aceitável de 90%. Visto que muitas páginas de Internet modernas são dinâmicas deve-se ter um limite de tolerância a qual deve ser definida pelo usuário em cada caso específico.

A figura abaixo demonstra fórmula para o cálculo do grau de semelhança, em porcentagem:

Figura 17 – Fórmula de cálculo do grau de semelhança

$$100 - \left(\frac{\text{Partes Alteradas x } 100}{\text{Partes Comparadas}} \right)$$

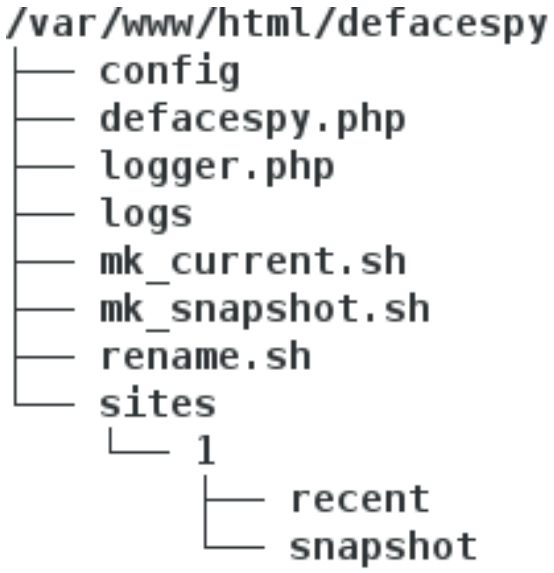
Fonte – Produzido pelo autor do trabalho

A fórmula acima calcula a estatística de quantas partes estão alteradas através da referência de alterações feitas na captura página atual.

4.2 Implantação

A implementação do protótipo é dividida em *scripts* em PHP e em *Shell Script*. O ambiente Unix ou Linux é obrigatório para seu funcionamento. A figura mostra a estrutura física do diretório onde é localizado todos os arquivos importantes para o funcionamento do agente de detecção de páginas desfiguradas.

Figura 18 – Estrutura física de arquivos e diretórios do agente



Fonte – Produzido pelo autor do trabalho

A figura acima descreve os seguintes arquivos de script, diretórios ou configuração. O quadro 1 detalha a função de cada arquivo do agente.

Quadro 1 – Descrição detalhada de cada arquivo ou diretório que compõe o agente

Arquivo/Diretório	Descrição
config	Arquivo que contém o URL do site a ser monitorado.
defacespy.php	Agente de comparação das páginas.
logger.php	Script para criação dos arquivos de logs.
Logs	Diretório onde ficam os arquivos de logs gerados.
mk_current.sh	Script para captura do estado da página atual.
mk_snapshot.sh	Script para captura do estado original da página. Gera a fotografia.
rename.sh	Script para padronizar os nomes dos arquivos de imagens gerados.

Fonte – Produzido pelo autor do trabalho

Vale salientar que os arquivos *mk_snapshot.sh, mk_current.sh e defacespy.php* são os scripts responsáveis pela detecção de páginas desfiguradas. A ordem de execução deles é da ordem que foram descritos, respectivamente.

4.3 Testes

No protótipo desenvolvido foram estabelecidos, conforme mencionado nos capítulos anteriores, três estados possíveis de comparação: grau de alteração além do limite, dimensões das páginas diferentes e o grau de alteração dentro dos padrões definidos pelo usuário. Esses estados serão exemplificados nos próximos capítulos para um maior grau de compreensão do seu funcionamento. Todos os testes realizados utilizaram uma página de Internet fictícia em um ambiente de testes.

4.3.1 Desfiguração parcial

Nessa situação a página *web* foi parcialmente alterada. Grande parte de sua estrutura definida em HTML permanece intacta. Um arquivo de imagem foi alterado pelo atacante ocasionando um grande grau de notabilidade pelos usuários desse *website*. A figura 19 demonstra um comparativo entre a página inicial, em seu estado original, e a versão modificada pelo atacante.

Figura 19 – Comparação entre versões da página web

Fonte – Produzido pelo autor do trabalho

A figura acima demonstra uma simulação de ataque onde houve uma desfiguração parcial da página da vítima. O atacante conseguiu achar um ponto no sítio onde sua alteração causou um grande grau de notabilidade. Nesse caso ele alterou o arquivo de imagem que representa o plano de fundo da página inicial. O novo plano de fundo possui a palavra “Invadido”. Porém a página desfigurada permanece no ar possivelmente com seus *hiperlinks* ainda navegáveis. Nesse caso a detecção será pelo número de alterações que foram além do limite estabelecido pelo usuário. A figura 20 demonstra o resultado dessa comparação.

Figura 20 – Comparação entre versões da página web

```
############### DefaceSpy Versao:0.0.1 ###############
[*] Dimensoes da Pagina Atual x Snapshot: OK
[*] Total de partes obtidas (Pagina Atual): 176
[*] Total de partes obtidas (Snapshot): 176
[*] Resultado da comparacao Snapshot e a versão Atual
>> Limite toleravel de alteracoes na pagina: 90%
>> Total de Partes do snapshot: 176
>> Total de Partes da versão Atual: 176
Similaridade entre o Snapshot e a versão Atual: 54.55%
Resultado: Similaridade abaixo da tolerancia de 90%.

ATENCAO! PAGINA POSSIVELMENTE DESFIGURADA.

[*] Gerado relatorio: defacespy_07052017130705.html
```

Fonte – Produzido pelo autor do trabalho

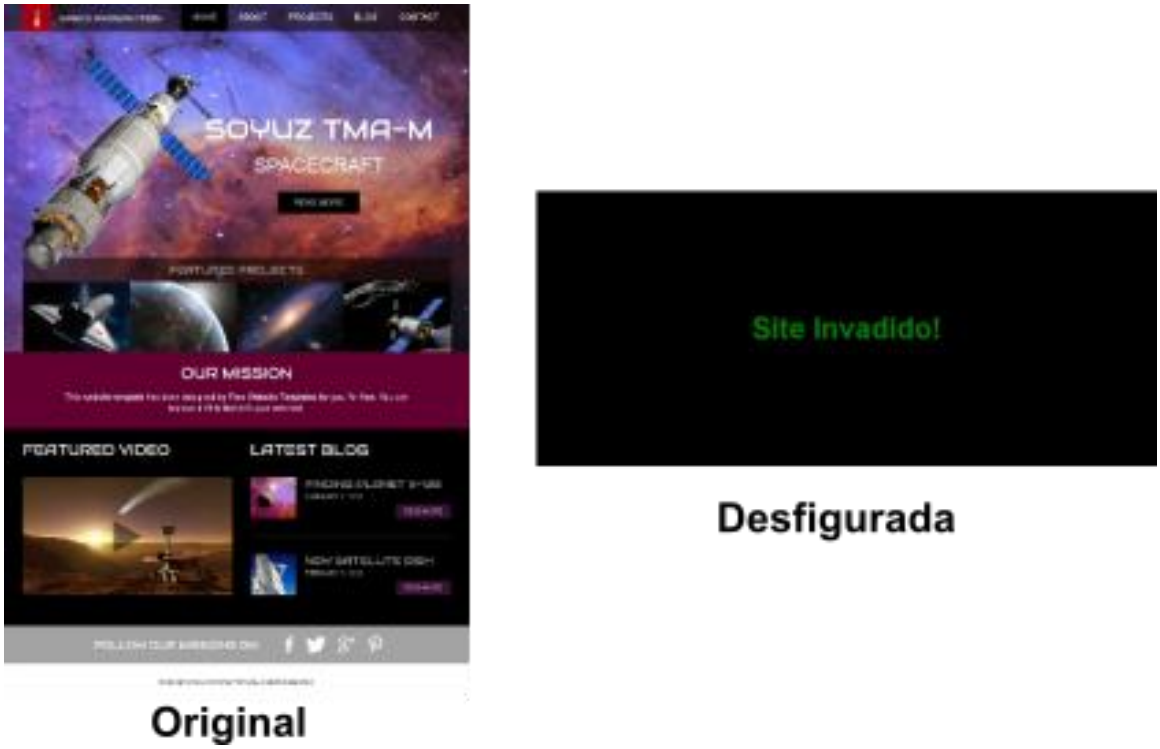
Conforme representado na figura acima, o protótipo comparou todas as partes entre as duas páginas. Elas foram divididas em 176 partes menores para haver o teste. O nível de similaridade foi de 54,55% o qual foi abaixo de 90%: limite estabelecido pelo usuário. No final um relatório foi gerado com uma pré-visualização da versão atual captura pelo protótipo.

4.3.2 Desfiguração Total

Nessa situação a página inicial da vítima foi alterada. Nesse caso o atacante poderá ter removido todas as páginas do *website*. O prejuízo é enorme visto que tecnicamente o *homepage* está fora do ar. A figura 21 demonstra um comparativo

entre a página inicial, em seu estado original e a versão, completamente modificada, pelo atacante.

Figura 21 – Comparação entre uma página original e uma desfigurada



Fonte – Produzido pelo autor do trabalho

A figura acima demonstra uma simulação de ataque onde houve uma desfiguração total da página da vítima. O atacante substituiu a página inicial original por outra completamente diferente. O grau de notabilidade é enorme pois o atacante impossibilitou sua usabilidade pelos usuários. Após o incidente houve a execução do protótipo. A figura 22 demonstra o resultado de sua execução.

Figura 22 – Resultado de uma página desfigurada por completo

```
############### DefaceSpy Versao:0.0.1 ###############
[*] Resolucoes da Pagina Atual e o Snapshot: DIFERENTES!

ATENÇÃO! PAGINA POSSIVELMENTE DESFIGURADA!
```

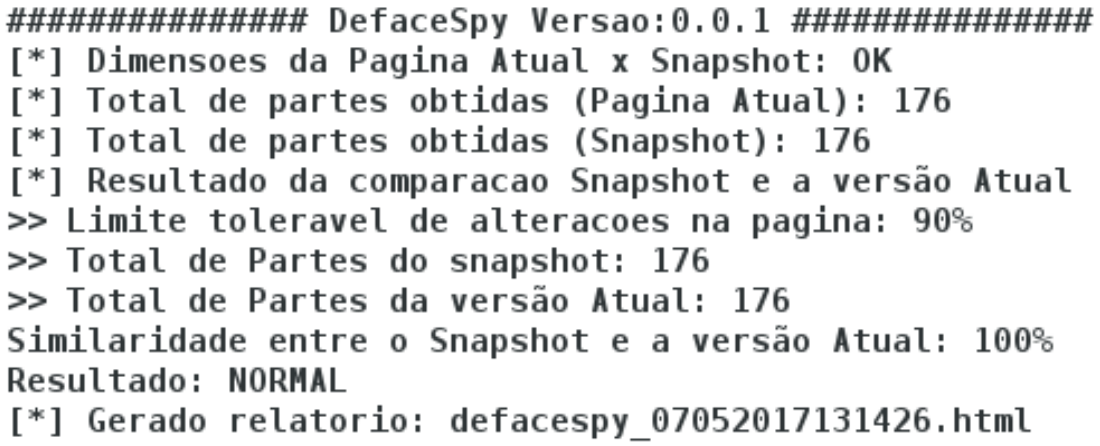
Fonte – Produzido pelo autor do trabalho

Conforme representado na figura acima, o protótipo não comparou todas as partes entre as duas páginas. O motivo foi que a validação das resoluções entre a página original e a atual. O número de partes não seria igual ocasionando um erro de execução. Como as resoluções não foram iguais, e isso foi outro pretexto para gerar um alerta ao administrador de um nível mais grave porque o atacante ocasionou uma negação de serviço. Nesse período nenhum usuário fica possibilitado de usar os serviços *online* da página atacada.

4.3.3 Nível normal de alterações

Nessa situação a página de Internet não houve alterações feitas pelo atacante. Um detalhe importante é que as páginas de hipertexto modernas são dinâmicas e alterações podem acontecer.

Figura 23 – Resultado de uma página não desfigurada



Fonte – Produzido pelo autor do trabalho

Conforme representado na figura acima, o protótipo comparou todas as partes entre as duas páginas. Elas foram divididas em 176 partes menores para haver o teste. O nível de similaridade foi acima de 90%: limite estabelecido pelo usuário. No final um relatório foi gerado com uma pré-visualização da versão atual captura pelo protótipo.

4.4 Considerações sobre os resultados

Após os testes realizados comprovou-se que há maneiras eficientes de detectar a desfiguração de páginas de Internet. O protótipo utilizou somente um tipo de detecção que foi a comparação de *hashes*. A transformação do arquivo de HTML em imagem facilitou esse processo de comparação entre arquivos de imagem e também suas divisões feitas do arquivo de imagem principal.

Porém esse método possui suas limitações em mudanças de palavras. Conforme os meios de detecção citados como o DOM seriam uma forma eficiente de aumentar a precisão em detectar essas particularidades que geralmente os atacantes podem explorar. Nota-se que existem níveis de desfiguração podendo ser textos alterados ou até toda página web. Pelos testes realizados, pode-se medir esse nível de alteração através de uma estatística simples.

Cabe ressaltar que a acuracidade dos resultados depende das ferramentas adotadas para a geração e divisão das imagens das páginas. Todas as imagens deverão ser obtidas usando as mesmas ferramentas, com os mesmos parâmetros de configuração, para evitar distorções nos dados obtidos.

Uma questão a ser tratada no futuro diz respeito à definição do indicador de similaridade em páginas dinâmicas. Para cada caso devem-se realizar ajustes, dependendo da periodicidade das atualizações de página e do grau de modificação. O uso de mecanismos de aprendizado de máquina pode ser bastante útil para a automatização do cálculo do indicador.

Umas das limitações dessa proposta é a impossibilidade de identificar alterações não visíveis da página, como a infecção de scripts para ataques aos clientes. Para esses casos será necessário incluir outras técnicas de detecção que analisassem também o código HTML da página. A associação de outras técnicas deverá ser abordada em trabalhos futuros.

CONCLUSÃO

Todos os dias surgem novos *websites* atacados, por métodos desde os mais simples até os mais complexos. Os atacantes possuem os diversos motivos para realizar esses tipos de vandalismo. O mundo moderno exige cada vez mais rapidez nos serviços *online*. Nós como usuários somos vulneráveis a qualquer tipo de ataque. Visto que dependemos cada vez mais desses sistemas *web* para realizar tarefas do dia a dia. As páginas de Internet, hoje em dia, entram na lista de ativos importantes das organizações. Muitas empresas são somente virtuais. Dessa maneira, qualquer organização que dependa da Internet pode ter prejuízos financeiros com esses tipos de ataques. Os administradores, *webdesigners* e principalmente os desenvolvedores necessitam de cada vez mais estarem atualizados para estarem cientes dos procedimentos seguros de desenvolvimento ou de administração que visam reduzir o risco de serem atacados.

Com esse trabalho acadêmico foi possível esclarecer os tipos de ataques mais comuns e métodos de detecção de páginas de Internet desfiguradas. Além disso, foi possível desenvolver um protótipo para testar um dos métodos de detecção mais conhecidos. Isso concluiu que é possível reduzir um incidente de ataque com automatizações. Com agentes configurados é possível aperfeiçoar esse processo podendo adicionar ações mais complexas indo além de alertas para o administrador. Essas ações complexas poderiam ajudar, com certa rapidez, retornar a *webpage* em seu estado normal.

O próximo passo será aumentar a eficiência de detecção de páginas desfiguradas. Poderão ser desenvolvidos outros tipos de detecção (descritos no capítulo 2) para diminuir os falsos positivos. O método de detecção DOM é o mais indicado para isso, apesar de sua complexidade no seu desenvolvimento. Ele dará à ferramenta a habilidade de detecção de palavras ou elementos do HTML que possam ter sido alterados.

REFERÊNCIAS

APELBAUM, **Yaacov. User Authentication Principles**, Theory and Practice. Fuji Technology Press, 2007.

BORGOLTE, Kevin; KRUEGEL, Christopher; VIGNA, Giovanni. Meerkat: Detecting Website Defacements through Image-based Object Recognition. In: **USENIX Security Symposium**. 2015. p. 595-610.

CISCO, **Security Within the Extensible Markup Language Infrastructure**. Disponível em: <http://www.cisco.com/c/en/us/products/collateral/application-networking-services/ace-xml-gateways/prod_white_paper0900aecd80669393.pdf>. Acesso em: 15 Abr. 2017.

DAVANZO, Giorgio; MEDVET, Eric; BARTOLI, Alberto. Anomaly detection techniques for a web defacement monitoring service. **Expert Systems with Applications**, v. 38, n. 10, p. 12521-12530, 2011.

EVANS, R. **What's in an HTTP request?.** Disponível em: <http://rve.org.uk/dumprequest>. Acesso em: 15 Mar. 2017.

FERREIRA, Fernando. **Segurança da Informação**. Brasil: Ciência Moderna, 2003.

GRAVES, Kimberly. **CEH certified ethical hacker study guide**. Estados Unidos: Estados Unidos: John Wiley & Sons, 2010.

GURJWAR, Rajiv Kumar; SAHU, Divya Rishi; TOMAR, Deepak Singh. An approach to reveal website defacement. **International Journal of Computer Science and Information Security**, v. 11, n. 6, p. 73, 2013.

JAIN, R. **Social Network Security: A Brief Overview of Risks and Solutions.** Disponível em: <http://www.cse.wustl.edu/~jain/cse571-09/ftp/social/>. Acesso em: 15 Mar. 2017.

KANTI, Tushar; RICHARIYA, Vineet; RICHARIYA, Vivek. Implementing a Web browser with Web defacement detection techniques. **World of Computer Science and Information Technology Journal (WCSIT)**, v. 1, n. 7, p. 307-310, 2011.

KANTI, Tushar; RICHARIYA, Vineet; RICHRIYA, V. Implementation of an efficient web defacement detection technique and spotting exact defacement location using diff algorithm. **Int. J. Emerg. Technol. Adv. Eng**, v. 2, 2012.

KANTI, Tushar; RICHARIYA, Vineet; RICHRIYA, V. Implementation of an efficient web defacement detection technique and spotting exact defacement location using diff algorithm. **Int. J. Emerg. Technol. Adv. Eng**, v. 2, 2012.

KRAWETZ, Neal. **Introduction to Network Security**. Estados Unidos: Charles River Media, 2007.

MAUNDER, S. **Rapid Growth in Defacements, Who was Hit, Who is Attacking.** Disponível em: <https://www.wordfence.com/blog/2017/02/rapid-growth-in-rest-api-defacements/>. Acesso em: 15 Mar. 2017.

ORIYANO. **CEH v8: Certified Ethical Hacker Version 8 Study Guide.** Estados Unidos: John Wiley & Sons, 2014.

ROBIE, J.; **What is the Document Object Model?** Disponível em: <https://www.w3.org/TR/DOM-Level-2-Core/introduction.html>. Acesso em: 17 Mar. 2017.

SHEMA, Mike. **Seven deadliest web application attacks.** Estados Unidos: Syngress, 2010.

STUTTARD, Dafydd; PINTO, Marcus. **The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws.** Estados Unidos: John Wiley & Sons, 2011.

TRIPATHI,S. **What Is XSS Attacks?** Disponível em: <http://smartechverse.blogspot.com.br/2015/07/xss-attacks.html>. Acesso em: 15 Mar. 2017.

VACCA, John R. (Ed.). **Network and system security.** Estados Unidos: Elsevier, 2013.

VERMA, Rashmi K.; SAYYAD, Shahzia. **Implementation of Web Defacement Detection**, p. 134-141, 2015.

YADAV, **A. Exploiting by Information Disclosure, Part 1.** Disponível em: <resources.infosecinstitute.com/exploiting-information-disclosure-part-1/>. Acesso em: 15 Mar. 2017.

ANEXO A

A. CONFIGURAÇÂO DO AGENTE DE DETECÇÂO DE PÀGINAS DESFIGURADAS

A.1 Instalação dos softwares necessários para o funcionamento agente

```
|root@rhino (logs)|#
|root@rhino (logs)|#apt-get install php5-cli php5-gd wkhtmltopdf imagemagik
```

A.2 Instalação dos arquivos do agente

```
|root@rhino (~)|#tar -xvf defacespy_vproto_final.tar -C /var/www/html
defacespy/sites/
defacespy/mk_current.sh
defacespy/logger.php
defacespy/
defacespy/sites/1/
defacespy/sites/1/snapshot/
defacespy/mk_snapshot.sh
defacespy/logs/
defacespy/defacespy.php
defacespy/rename.sh
defacespy/sites/1/recent/
defacespy/config
|root@rhino (~)|#
```

A.3 Configurar a página web a ser monitorada pelo agente

```
|root@rhino (~)|#echo "http://www.thespacex.com/spacex" > /var/www/html/defacespy/config
```

A.4 Configuração do limite tolerável de alterações (no script defacespy.php)

```
49
50 ###limite de tolerancia de mudancas da pagina
51 $limite_alt=90;
52
```

A.5 Geração do snapshot da página web a ser monitorada

```
|root@rhino (logs)|#/var/www/html/defacespy/mk_snapshot.sh
```

A.6 Agendamento das rotinas de execução do agente (pelo crontab)

```
#defacespy: captura da pagina na versao atual (cada 3 min)
*/3 * * * * /var/www/html/defacespy/mk_current.sh

#defacespy: geracao de relatorio (cada 5 min)
*/5 * * * * /usr/bin/php -f /var/www/html/defacespy/defacespy.php
```

GLOSSÁRIO

A

APXS. Acrônimo recursivo para Apache Extension Tool (ferramenta de extensões para o Apache). Disponível em: <https://httpd.apache.org/docs/2.4/programs/apxs.html>. Acesso em: 22 Abr. 2017.
Apache. Servidor de páginas de Internet. Disponível em: <https://techterms.com/definition/apache>. Acesso em: 22 Abr. 2017.

C

Cookie. Informação armazenada pelo navegador de Internet de um servidor web. Disponível em: <http://www.webopedia.com/TERM/C/cookie.html>. Acesso em: 22 Abr. 2017.
Checksum. Meio detecção de erros para mensagens transmitidas a um determinado destino. Disponível em: <http://www.webopedia.com/TERM/C/checksum.html>. Acesso em: 22 Abr. 2017.
C-Sharp. Linguagem de programação usada para páginas de Internet. Disponível em: <https://www.techopedia.com/definition/26272/c-sharp>. Acesso em: 22 Abr. 2017.
Caffe. Biblioteca de Inteligência Artificial para desenvolvimento de programas de computador. Disponível em: <http://caffe.berkeleyvision.org>. Acesso em: 22 Abr. 2017.
CPU. Acrônimo recursivo para *Central Processing Unit* (Unidade Central de Processamento). Responsável pela execução de aplicativos ou sistemas operacionais. Disponível em: <https://techterms.com/definition/cpu>. Acesso em: 22 Abr. 2017.
CRC. Acrônimo recursivo para *Cyclic Redundancy Checking* (Verificação Cíclica de Redundância). Forma de verificar a integridade de dados vindos de uma rede de computadores. Disponível em: <http://searchnetworking.techtarget.com/definition/cyclic-redundancy-checking>. Acesso em: 22 Abr. 2017

F

Framework. Plataforma de desenvolvimento para softwares. Disponível em: <https://techterms.com/definition/framework>. Acesso em: 22 Abr. 2017.

G

GPU. Acrônimo recursivo para *Graphic Processing Unit* (Unidade Gráfica de Processamento). Utilizada para processar gráficos tridimensionais em um ambiente computacional. Disponível em:<https://techterms.com/definition/gpu>. Acesso em: 22 Abr. 2017.

H

Hacktivismo. Ato de vandalismo em sistemas computacionais com propósitos políticos ou sociais. Disponível em: <http://searchsecurity.techtarget.com/definition/hacktivism>. Acesso em: 22 Abr. 2017.
HTML. Acrônimo recursivo para *Hypertext Markup Language* (Linguagem de Marcação de Hipertexto). Linguagem de marcação usada para desenvolvimentos de páginas de Internet. Disponível em: <http://www.webopedia.com/TERM/H/HTML.html>. Acesso em: 22 Abr. 2017.
Hash. Função usada para criar assinaturas digitais para qualquer informação. Disponível em: <http://www.webopedia.com/TERM/H/hashing.html>. Acesso em: 22 Abr. 2017.
Hyperlink. Elemento que liga um documento HTML a outro. Disponível em: <http://www.webopedia.com/TERM/H/hyperlink.html>. Acesso em: 22 Abr. 2017.
Handler. Representação interna que o Apache utiliza para lidar com arquivos. Disponível em: <https://httpd.apache.org/docs/2.4/handler.html>. Acesso em: 22 Abr. 2017.

I

Intranet. Rede corporativa que utiliza as mesmas tecnologias usadas na Internet. Disponível em: <http://www.hardware.com.br/termos/intranet>. Acesso em: 22 Abr. 2017.
Imagemagick. Ferramenta para manipulação de arquivos de Imagens. Disponível em:<http://www.imagemagick.org/script/index.php>. Acesso em: 22 Abr. 2017.
Inteligência Artificial. Meio de utilizar a forma humana de pensar em sistemas computacionais. Disponível em: <https://techterms.com/definition/artificial_intelligence>. Acesso em: 22 Abr. 2017.

L

Logon. Fornecer uma senha ou credencial para acesso a um sistema. Disponível em: <http://www.hardware.com.br/termos/logon>. Acesso em: 22 Abr. 2017.
LCS. Acrônimo recursivo para *Longest Common Subsequence* (Mais longa Subsequência Comum). Algoritmo de detecção de diferenças entre arquivos de computador. Disponível em: <https://www.ics.uci.edu/~eppstein/161/960229.html>. Acesso em: 22 Abr. 2017.
Leetspeak. Alfabeto alternativo que substitui letras por números. Frequentemente usado por *hackers*. Disponível em: <http://www.webopedia.com/TERM/1/133t_speak.html>. Acesso em: 22 Abr. 2017.

M

MD5. Acrônimo recursivo para *Message Digest 5* (sem tradução)**.** Utilizado para assinaturas digitais. Disponível em: <http://www.webopedia.com/TERM/M/md5.html>. Acesso em: 22 Abr. 2017.

O

Overhead. Utilização de todos os recursos computacionais para um determinado fim. Disponível em: <http://www.webopedia.com/TERM/O/overhead.html>. Acesso em: 22 Abr. 2017.
Open Source. Ou Código Aberto. Termo utilizado para um código fonte de um software disponível ao público. Livre de qualquer custo. Disponível em: <http://www.webopedia.com/TERM/O/open_source.html>.
Acesso em: 22 Abr. 2017.

P

Páginas Estáticas. Páginas de Internet que possuem seu conteúdo estático. Disponível em: <https://techterms.com/definition/staticwebsite>.
Acesso em: 22 Abr. 2017.
Pixel. Menor elemento de um arquivo uma imagem. Um ponto. Disponível em <https://www.webopedia.com/TERM/P/pixel.html>. Acesso em: 22 Abr. 2017.
Python. Linguagem de programação.<https://techterms.com/definition/python>. Acesso em: 22 Abr. 2017.
Polling. Forma de verificar continuamente se dispositivos ou softwares estão disponíveis para comunicação. Disponível em: <http://whatis.techtarget.com/definition/polling>. Acesso em: 22 Abr. 2017.
PSNR. Acrônimo recursivo para *Peak Signal to Noise Ratio* (relação sinal-ruído de pico). Usado para medir a qualidade de após uma compressão de imagem. Disponível em: <https://www.igi-global.com/dictionary/peak-signal-to-noise-ratio-psnr/22135>. Acesso em: 22 Abr. 2017.

R

Rede Neural. Tipo de Inteligência Artificial que imita a forma que o cérebro humano funciona. Disponível em:<http://www.webopedia.com/TERM/N/neural_network.html>. Acesso em: 22 Abr. 2017.

S

SSIM. Acrônimo recursivo para *Structural Similarity* (Similaridade Estrutural). Forma de prever a qualidade de áudio ou vídeo em meio digital. Disponível em: <http://www.imatest.com/docs/ssim/>. Acesso em: 22 Abr. 2017.
SSL. Acrônimo recursivo para *Secure Sockets Layer* (sem tradução). Cria um canal seguro (criptografado) entre o cliente e servidor. Disponível em: <http://www.webopedia.com/TERM/S/SSL.html>. Acesso em: 22 Abr. 2017.

SQL. Acrônimo recursivo para *Structured Query Language*(Linguagem de Consulta Estruturada). Usada para busca de informações em um banco de dados. Disponível em: <http://www.webopedia.com/TERM/S/SQL.html>. Acesso em: 22 Abr. 2017.
Script. Arquivo com conjunto de instruções que devem ser executadas sem a interação do usuário. Disponível em: <http://www.webopedia.com/TERM/S/script.html>. Acesso em: 22 Abr. 2017.
SHA512. Acrônimo recursivo para *Secure Hash Algorithm* (Algorítmo de Hash Seguro) de comprimento de 512 bits. Utilizado para assinaturas digitais. Disponível em: <https://www.pcmag.com/encyclopedia/term/68965/sha>. Acesso em: 22 Abr. 2017.
SMS. Acrônimo recursivo para *Shortest Middle Snake*(sem tradução). Método de detecção de diferenças entre arquivos de computador. Disponível em: <http://simplygenius.net/Article/DiffTutorial1>. Acesso em: 22 Abr. 2017.
Sistema Especialista. São sistemas que solucionam problemas que são somente resolvidos por pessoas especialistas. Disponível em: <http://www.din.uem.br/ia/especialistas/introdu.html>. Acesso em: 22 Abr. 2017.

W

W3C. Acrônimo recursivo para World Wide Web Concortium. Consórcio entre várias organizações envolvidas com a Internet. Disponível em: <https://www.webopedia.com/TERM/W/W3C.html>. Acesso em: 22 Abr. 2017.
Webdesigner. Profissional responsável pelo desenvolvimento de páginas de Internet. Disponível em: <https://techterms.com/definition/web_design>. Acesso em: 22 Abr. 2017.
Wkhtmltopdf. Conjunto de ferramentas para transformar páginas de Internet em vários tipos de arquivo de imagem. Disponível em:<https://wkhtmltopdf.org>. Acesso em: 22 Abr. 2017.
WWW. Acrônimo recursivo para *World Wide Web* (Rede Mundial de Computadores). Disponível em: <http://www.webopedia.com/TERM/W/World_Wide_Web.html>. Acesso em: 22 Abr. 2017.