Determinando o Risco de Fingerprinting em Páginas Web

Adriana R. Saraiva[1], **Adria M. de Oliveira** [1], **Eduardo L. Feitosa**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
CEP 69.077-000 – Manaus – AM – Brasil

{adriars,adriah.menezes}@gmail.com,efeitosa@icomp.ufam.ed.br

Abstract. *The fingerprinting techniques applied to Web pages to identify and track users on the Internet has become common. Although there are solutions and countermeasures against such techniques, they often come up against the need to compare codes or running in the background. In this context, this paper proposes a methodology to detect artifacts (scripts) fingerprinting in Web pages and inform the users how dangerous the page is for your privacy. The results show that although simple, the method is efficient to find fingerprinting codes and categorizes them by severity levels of user privacy.*

Resumo. *As técnicas de fingerprinting aplicadas em páginas Web para identificar e rastrear usuários na Internet têm se tornado cada vez mais comum. Embora existam soluções e contramedidas contra tais técnicas, elas muitas vezes esbarram na necessidade de comparação de códigos ou execução em segundo plano. Neste contexto, este artigo propõe uma metodologia para detectar artefatos (scripts) de fingerprinting em páginas Web e informar aos usuários o quão perigosa a página é para sua privacidade. Os resultados mostram que embora simples, a metodologia é eficaz ao encontrar códigos fingerprinting nos websites e categorizá-los em níveis de severidade quanto aos riscos à privacidade dos usuários.*

1. Introdução e Motivação

Com a ineficácia frente as plataformas móveis e a crescente preocupação com privacidade, os cookies tornaram-se obsoletos e passaram a ser limitados na Internet. Entretanto, a indústria de anúncios e publicidade online, a fim de manter a continuidade de seus negócios, investiu em novas formas de rastreio e identificação por meio do próprio conjunto de hardware/software do usuário. Esse conceito, conhecido como *Website Fingerprinting*, *Browser Fingerprinting* ou *Device Fingerprinting*, parte do pressuposto que cada usuário opera o seu próprio hardware. Assim, a identificação de um dispositivo é o mesmo que identificar a pessoa por trás dele. Segundo Nikiforakis [Nikiforakis et al. 2014], *Website Fingerprinting* faz uso de um conjunto de atributos do sistema extraídos do dispositivo do usuário, que gera uma combinação de valores únicos capazes de identificar um usuário/dispositivo.

O fato relevante e motivador para esta pesquisa é que as técnicas de *Website Fingerprinting* estão se tornando mais comuns e que os usuários sabem pouco ou quase nada sobre o assunto. Mesmo quando estão cientes de que estão sendo monitorados (como uma medida de proteção contra a fraude, por exemplo), os usuários simplesmente confiam que as informações coletadas não serão utilizadas para outros fins. Indiferente a quão fácil é

obter informações sobre o navegador e o dispositivo de um usuário, a fim de gerar uma identificação única, o fato é que existem diferentes tipos de artefatos para realizar um *fingerprinting* na Web, onde se emprega uma variedade de tecnologias.

Neste cenário, uma questão pouco explorada é o quão perigoso é um artefato desse. Assim, o objetivo deste artigo é mensurar a severidade de um artefato *fingerprinting* presente em uma página Web. Como forma de alcançar esse objetivo foram analisadas as principais formas e tecnologias para realizar um *Website Fingerprinting*. A principal contribuição desse artigo é a definição de uma simples metodologia para estimar a severidade de artefatos (scripts) de *fingerprinting* na Web, em relação a privacidade dos usuários.

O restante do artigo é organizado como segue: na Seção 2 é apresentada uma breve descrição sobre *Website Fingerprinting*, com seu funcionamento e tecnologias atualmente utilizadas. Na Seção 3 são descritas as estratégias de identificação de usuários com o uso de *fingerprinting* e também de detecção de *fingerprinting*. Nesta Seção, os trabalhos são discutidos de forma a evidenciar as escolhas/técnicas empregadas. Na Seção 4 é apresentada a metodologia proposta para estimar a severidade de artefatos (scripts) de *fingerprinting* na Web, com os detalhes de implementação. Na Seção 5 são apresentados os experimentos e os resultados obtidos com a classificação dos websites quanto ao nível de severidade da *fingerprinting*, com uma breve discussão. Por fim, a Seção 6 expõe as observações finais sobre o trabalho.

2. *Website Fingerprinting*

De acordo com a RFC 6973 [Cooper et al. 2013], o termo *fingerprint* é definido formalmente como "um conjunto de elementos de informação que caracteriza um dispositivo ou uma instância de uma aplicação" e *fingerprinting* como "o processo pelo qual um observador ou atacante identifica, de maneira única e com alta probabilidade, um dispositivo ou uma instância de um aplicativo com base em um conjunto de múltiplas informações".

Este trabalho partilha da visão que *fingerprinting* é parte de um conjunto amplo de tecnologias e técnicas usadas para identificar (ou reidentificar) um usuário ou um dispositivo, através de um conjunto de configurações, atributos (tamanho da tela do dispositivo, versões de software instalado, entre muitos outros) e outras características observáveis durante a comunicação.

De acordo com o W3C (*World Wide Web Consortium*)[1] [W3C 2014], as técnicas de *website fingerprinting* podem ser classificadas em: (1) **Passiva**, baseada nas características observáveis na comunicação HTTP, tais como: cabeçalhos, endereço IP e outras informações do nível de rede; (2) **Ativa**, que executam códigos (JavaScript mais comumente) no lado do cliente para observar as características como dados do hardware e do navegador; e (3) **Cookie-like**, onde um código ou aplicativo é instalado no lado do cliente.

Embora o uso de *website fingerprinting* esteja relacionado à identificação de usuários como medida de segurança e mecanismo anti-fraude, os impactos na privacidade dos usuários causados por essa identificação são ameaças reais de segurança. Três problemas na privacidade são apontados em [W3C 2014]: (1) **identificação do usuário**, visto que um *fingerprinting* tem o potencial de obter dados do usuário sem autorização

[1] http://www.w3c.org

prévia que o deixa exposto e vulnerável; (2) **correlacionar as atividades da navegação**, visto que os usuários podem se surpreender ao perceber que terceiros podem correlacionar suas várias visitas ao mesmo ou diferentes sites com a finalidade de elaborar um perfil; (3) **inferências sobre o usuário**, uma vez que as informações coletadas podem revelar dados sobre os quais se pode tirar conclusões sobre o usuário. O W3C [W3C 2014] afirma que tais inferências permitem oferecer e mostrar ao usuário produtos em determinada faixa de preço, o que pode ser uma forma discriminatória de publicidade.

2.1. Tecnologias

Embora as tecnologias Web visem dinamizar o uso dos conteúdos em páginas na Internet, esta seção descreve, resumidamente, como elas podem ser usadas para *website fingerprinting*. Maiores detalhes e exemplos podem ser encontrados em [Saraiva et al. 2014]

HTML5 Canvas

Canvas é um novo elemento da HTML5 que, via JavaScript, fornece acesso a um conjunto completo de funções de desenho, para que gráficos sejam gerados dinamicamente no lado do cliente. No que diz respeito a *website fingerprinting*, o uso de Canvas já foi demonstrado nos trabalhos de Mowery e Shacham [Mowery and Shacham 2012] e Kirk [Kirk 2014]. Embora pareça inofensivo, o elemento Canvas possui dois (2) métodos - $toDataURL()$ e $getImageData()$ - capazes de extrair informações da tela do navegador (dados pessoais, por exemplo).

JavaScript

JavaScript é uma linguagem de programação interpretada, implementada como parte dos navegadores para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor [Crockford 2008]. Em linhas gerais, JavaScript é usada como mecanismo de execução de um *fingerprinting* e não como a real causa do rastreio ou acesso a informações. Contudo, os trabalhos de Mowery et al. [Mowery et al. 2011] e Mulazzani et al. [Mulazzani et al. 2013] demonstraram como utilizar JavaScript para obter informações sobre o navegador e os usuários.

Adobe Flash

Adobe Flash é utilizado para entrega de um rico conteúdo na Web, no intuito de atrair e envolver os usuários. Através dele é possível exibir animações e interfaces de aplicativos, que são implantadas imediatamente em todos os navegadores e plataformas.

WebGL

WebGL (*Web Graphics Library*) é uma API multiplataforma, que traz a linguagem OpenGL ES 2.0 para a Web de forma a suportar desenhos 3D dentro do HTML [Group 2014]. Em outras palavras, oferece suporte para renderização de gráficos 2D e 3D. Por ser suportada por todos os navegadores, WebGL pode ser empregada para obter identificadores do usuário, como demonstrado no site Browserleaks [BrowserLeaks.com 2015]. Além disso, os estudos de Forshaw [Forshaw 2011] descobriram e provaram a existência de uma série de problemas de segurança com a especificação e implementação do WebGL, que permitiam desde identificar o navegador até realizar ataques de negação de serviço.

Biblioteca Modernizr

Modernizr é uma biblioteca capaz de detectar automaticamente a disponibilidade de tecnologias presentes no navegador do usuário. Ela permite, por exemplo, detectar o suporte a geolocalização, ao elemento $canvas$, aos plugins WebGL e flash, bem como o armazenamento local e armazenamento de sessão. Unger et. al [Unger et al. 2013] afirmam que o a biblioteca modernizr é uma poderosa tecnologia para obtenção de recursos disponíveis no navegador.

CSS

CSS (*Cascading Style Sheets*) é uma linguagem de estilo utilizada para descrever a aparência e a formatação de um documento em uma linguagem de marcação [Bos et al. 2011]. Foi projetada para permitir a separação do conteúdo do documento de apresentação de documentos, através de propriedades de estilo que incluem elementos de layout, cores, fontes, entre outros. No que tange *fingerprinting*, CSS tem potencial para obter vários tipos de informação diferente dos tradicionais dados de fontes.

Silverlight

Silverlight é um framework criado pela Microsoft para o desenvolvimento e execução de aplicações ricas para a Internet, com recursos e propostas similares ao Adobe Flash. Em termos técnicos, o Microsoft Silverlight é uma aplicação *cross-browser*, *cross-platform* do framework .Net. Embora o Silverlight possua várias características importantes e atraentes para os usuários, este plugin também está vulnerável as técnicas de *fingerprinting*. Assim como o Adobe Flash, Silverlight é capaz de obter informações do sistema, tais como: versão do sistema operacional, número de processadores, fuso horário, fontes instaladas, sistema, região, idioma do sistema operacional, entre outros.

3. Trabalhos Relacionados

Embora não exista uma classificação formal, a literatura sobre *website fingerprinting* pode ser dividida em: (i) identificação do usuário e/ou dispositivo e (ii) detecção de *website fingerprinting*.

Na primeira área, o trabalho de Peter Eckersley [Eckersley 2010] é considerado seminal por investigar o grau em que os navegadores modernos estão sujeitos a *website fingerprinting*. Para tanto, desenvolveu um site de teste **Panopticlick**[2] com um algoritmo capaz avaliar características do navegador e gerar um identificador único. Dos 470.161 acessos voluntários, 83,6% tiveram identificações únicas e apenas 5,3% de navegadores/usuários mantiveram o anonimato.

Boda et. al [Boda et al. 2012] seguiram a mesma ideia e desenvolveram o site Portal PET Internacional[3]. Como resultado, os autores perceberam que a correlação de listas de fontes do navegador e *UserAgents*[4] proporcionam uma base sólida para a identificação única dos sistemas operacionais Windows e Mac. Olejnik et. al. [Olejnik et al. 2013] comprovou, com cerca de 368.284 usuários, que é possível detectar, via CSS, os históricos de navegação dos usuários como ferramenta *fingerprinting*. Os resultados mostram que

[2]https://panopticlick.eff.org

[3]http://pet-portal.eu/fingerprint/

[4]De acordo com a RFC 2616 [Fielding et al. 1999], *user-agent* é um campo enviado quando o navegador faz a solicitação de uma página. Basicamente, é uma sequência de caracteres composta por um conjunto de elementos, chamados *tokens*, listados em ordem de importância de acordo com o padrão.

para a maioria dos usuários (69%), o histórico de navegação é único e que com apenas quatro (4) sites é possível identificar unicamente 97% dos usuários.

No âmbito de detecção de *website fingerprinting*, Acar et al. [Acar et al. 2013] desenvolveram um framework, denominado FPDetective, para encontrar códigos de *fingerprinting*. Para validar o trabalho, os autores empregaram 1 milhão de sites do alexa.com[5] e descobriram 404 sites que utilizavam JavaScript para *fingerprinting*. Em relação ao Flash foram encontrados 145 sites. O baixo valor nos resultados se deve ao fato dos autores usarem apenas 27 scripts conhecidos como base de comparação.

Khademi [Khademi 2014] apresentou uma solução, denominada FPGuard, para detectar e proteger os usuários contra o *fingerprinting*. Dada uma página Web em execução no navegador do usuário, o FPGuard monitora e registra sua atividade a partir de seu carregamento. Em seguida, extrai métricas relativas a cada método *fingerprinting* e constrói uma assinatura para a página Web. A avaliação foi realizada usando o top 10K do alexa.com, onde o FPGuard conseguiu identificar 9264 sites como *website fingerprinting*.

3.1. Discussão

A Tabela 1 apresenta, de forma resumida, uma comparação entre os trabalhos tanto de identificação quanto de mitigação de *website fingerprinting*. Vale ressaltar que todos valores indicados na tabela foram retirados dos trabalhos originais.

Tabela 1. Comparação das soluções de identificação e mitigação de *Website fingerprinting*

Artigo	Classificação	Tecnologias Empregadas	Propriedades do Navegador	Protótipo	Fonte de Dados
Panopticlick [Eckersley 2010]	Identificação	JavaScript e Flash	Nav, Scr, TZ, SF, HTTP	Sim	*
FPDetective [Acar et al. 2013]	Identificação	JavaScript, Flash, HTML5 e CSS3	Nav, Scr, Win, SF	Sim	alexa.com
Cross-Browser Fingerprinting [Boda et al. 2012]	Identificação	JavaScript	Nav, Scr, TZ, SF, HTTP	Sim	Panopticlick
Web Browsing History [Olejnik et al. 2013]	Identificação	CSS	Hy	Não	Não
Privaricator [Nikiforakis et al. 2015]	Híbrido	JavaScript e HTML5 Canvas	Nav e Scr	Não	alexa.com, bluecava, coinbase, petportal e fingerprintjs
FPGuard [Khademi 2014]	Híbrido	JavaScript, Flash e HTML5 Canvas	Nav, Scr, SF, TZ, HTTP	Não	alexa.com
Determinando o Risco de Fingerprinting em Páginas Web	Alerta e Classificação de Risco	JavaScript	Win, Doc, Nav, Scr, Sil, Mod, WGL, Can	Não	Diversas

Propriedades do Navegador: Nav-objeto Navigator; Scr-objeto Screen; Win-objeto Window; TZ-timezone;
Doc-objeto Document;
SF-sistema de fontes; HTTP-cabeçalho HTTP; HY-histórico;
Sil-Microsoft Silverligth; Mod-Biblioteca Modernizr; WGL-WebGL; Can-Canvas
*Não aplicável ou não informado

Independente do tipo de trabalho (classificação), os trabalhos diferem em termos do uso da tecnologia. Por exemplo, percebe-se o maciço uso de JavaScript como mecanismo de geração do *fingerprinting* enquanto o uso de Flash é bem pequeno. Em contrapartida, as soluções que empregam JavaScript e Flash acabam tendo uma maior inserção

[5]http://alexa.com

no navegador, conseguindo obter uma grande quantidade de informações, como visto no critério propriedade do navegador.

Outra observação interessante reside no fato de que existem poucas soluções acadêmicas focadas em combater *Website fingerprinting*. Com a exceção dos plugins e as duas soluções de detecção ([Nikiforakis et al. 2015] e [Khademi 2014]), todas as demais soluções visam apenas provar a existência do fato. Em outras palavras, é possível imaginar situações mais robustas e que não tenham como foco a proteção individual dos usuários. Tal fato, reforça a motivação para este trabalho. Por fim, também percebe-se na Tabela 1 que o foco da maioria dos trabalhos existentes na literatura é investigar artefatos JavaScript.

Este trabalho difere dos demais no âmbito da investigação de uma quantidade maior de termos (propriedades do Navegador e aplicações) presentes nos artefatos *fingerprinting*. Tais artefatos são classificados pelo nível de risco, o que possibilita alertar e mostrar ao usuário o perigo ao qual ele está exposto. Além disso, embora o foco da metodologia proposta aqui não ter uma implementação formal (um protótipo), ela pode ser agregada à uma aplicação, um plugin ou um módulo de um Proxy Web. Com isso, é possível verificar e classificar *fingerprinting* em páginas Web em tempo real.

4. Metodologia Proposta

Esta seção apresenta a visão geral da metodologia proposta para estimar a severidade de artefatos (scripts) de *fingerprinting* na Web a partir do conjunto de propriedades e métodos dos objetos HTML DOM e seu nível de risco à privacidade dos usuários.

4.1. Visão Geral

A Figura 1 apresenta a metodologia proposta, executada em quatro etapas: (1) Coleta de Páginas (sementes) para avaliação; (2) Extração de Scripts (códigos em JavaScript) de uma determinada página e geração de um único arquivo de script; (3) Extração de Termos (objetos, propriedades e métodos) relacionados aos artefatos comumente empregados em *fingerprinting*; e (4) Classificação da Severidade.

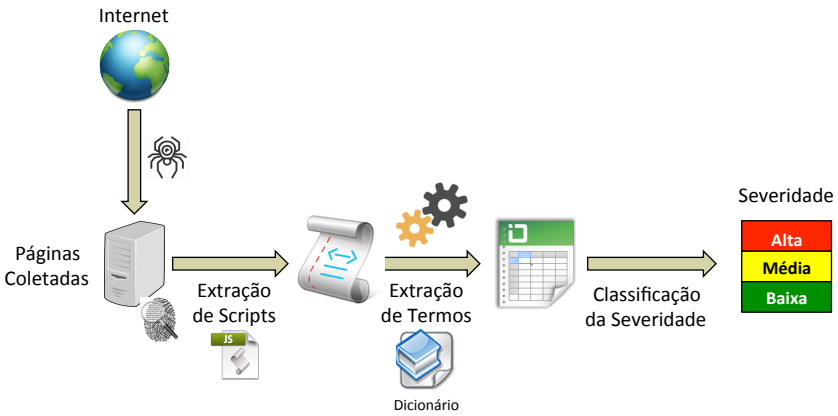

Figura 1. Visão geral da metodologia proposta

A intenção é manter o usuário informado da existência de artefatos *fingerprinting* na página acessada e qual seu nível de severidade. Este alerta permitirá ao usuário ficar

ciente da possibilidade de invasão da sua privacidade, como a captura de dados e sua posterior utilização em atividades maliciosas.

As etapas de **Coleta das Páginas** e **Extração de Script** são simples e correspondem a coleta das páginas Web e a extração dos scripts (JavaScripts) contidos nelas. Para executar essas funções, foi implementado um *crawler*, em linguagem PHP (versão 5.5.15), onde para cada site visitado, são coletados todos os scripts em JavaScript e armazenados em um único arquivo por site.

A etapa de **Extração de Termos** objetiva identificar os termos (objetos, propriedades e métodos) presentes no arquivo de script gerado na etapa anterior. Para esta etapa foi desenvolvido um script de extração na linguagem Python (versão 3.4) que realiza a leitura do arquivo único e extrai as propriedades e métodos de acordo com um dicionário previamente estabelecido. O dicionário de termos foi elaborado com base na literatura acadêmica de implementação de *fingerprinting* quanto contramedidas. Em outras palavras, trabalhos, scripts e exemplos de códigos foram avaliados e os objetos, propriedades e métodos utilizados para realizar *fingerprinting* foram selecionados. No total, o dicionário possui 58 termos.

Em relação ao processo propriamente dito, a extração dos termos é feita através do uso de expressões regulares (REGEX), um excelente recurso para casamento de padrões. O uso de REGEX permite que o script de extração seja capaz de buscar objetos, propriedades, atributos e métodos, possibilitando a descoberta de termos em até três níveis. A Tabela 3 apresenta o uso de expressões regulares na extração de termos.

Tabela 2. Expressões Regulares para Extração de Termos

Expressão Regular	Níveis
$("N2", r'([nw] * n.[nw]*)')$	Essa expressão regular busca por objetos que utilizam em sua sintaxe de chamada até dois níveis de acesso a propriedade. Exemplo: $navigator.userAgent$
$("N3", r'([nw] * n.[nw] * n.[nw]*)')$	Essa expressão regular busca por objetos que utilizam em sua sintaxe de chamada até três níveis de acesso a propriedade. Exemplo: $window.navigator.plugin$
$("N4", r'([nw] * n.[nw] * n.[nw] * n.[nw]*)')$	Essa expressão regular busca por objetos que utilizam em sua sintaxe de chamada até quatro níveis de acesso a propriedade. Exemplo: $window.navigator.plugin.name$

A quarta e última etapa é a **Classificação da Severidade**, cuja finalidade é definir o nível de risco (severidade) a qual o usuário está exposto ao acessar uma página Web. Essa etapa é baseada na metodologia para estimação da severidade de risco da OWASP[6]. Contudo, a severidade neste trabalho é calculada de forma mais simples, sem a necessidade de estimar o impacto de um evento e a probabilidade de uma vulnerabilidade ser explorada, como é necessário na metodologia da OWASP.

A ideia para classificar um artefato *fingerprinting* foi estimar o grau de envolvimento de elementos (objetos, propriedades, atributos e métodos) com ataques e intrusões na Web. Para isso, foi realizada uma detalhada pesquisa na literatura acadêmica e na área de segurança da informação (site, fóruns de discussão, blogs de desenvolvedores, comunidades hackers, entre outros), onde cada objeto, propriedade, atributo e método listados no dicionário de termos foi co-relacionado à ataques, vulnerabilidades e intrusões. Ou seja, como cada um desses elementos pode ser e é usado para explorar vulnerabilidades,

[6]Open Web Application Security Project (https://www.owasp.org) é uma entidade sem fins lucrativos e de reconhecimento internacional que contribui para a melhoria da segurança de softwares aplicativos.

violar privacidade, obter dados e realizar ataques.

Desta forma, inspirando-se na metodologia da OWASP, foram definidas três classes de severidade: Baixa, Média e Alta. Na classificação **Baixa** encontram-se os elementos capazes de fornecer informações usadas apenas para adaptações de conteúdo (tamanho da tela e a versão do navegador, por exemplo). Os objetos DOM HTML $Window$, $Navigator$ e $Screen$, bem como quase todas as suas propriedades, atributos e métodos se enquadram nessa classificação. Vale ressaltar que, até o momento, nenhum trabalho acadêmico provou ou interligou qualquer um desses elementos com ataques ou invasões de privacidade. Por isso, são considerados de baixo impacto quanto ameaça à privacidade. Contudo, no momento em que são combinados com outros elementos, podem compor uma poderosa arma de *fingerprinting* capaz de identificar unicamente o usuário e/ou dispositivo, conforme já demonstrado nos trabalhos de Eckersley [Eckersley 2010], Acar [Acar et al. 2013], entre outros.

Na classificação **Média** encontram-se os artefatos que listam os plugins instalados, os *mime-types* suportados e que verificam a presença dos plugins $SilverLight$ e $WebGL$. Khademi [Khademi 2014], Eckersley [Eckersley 2010] e Nikiforakis et al. [Nikiforakis et al. 2015] descreveram esses objetos como propensos a um médio grau de ameaça. Além disso, Unger et al. [Unger et al. 2013] afirmam que a biblioteca $Modernizr$ é uma poderosa tecnologia para obtenção de recursos disponíveis no navegador e por isso ela também é inserida nessa classificação. Em linhas gerais, a verificação dos plugins e mimeTypes existentes no navegador pode fornecer informações relevantes a um atacante, como, por exemplo, dados dos plugins bancários ou alguma informação de um software específico. Khademi[Khademi 2014], Eckersley [Eckersley 2010], [Nikiforakis et al. 2015] descreveram esses objetos como propensos a um alto e/ou médio grau de ameaça.

A classificação **Alta** é composta de quatro elementos: $canvas.toDataUrl()$, $canvas.getImageData()$, $window.history()$ e $MediaDevices.getUserMedia()$. O método $canvas.getImageData()$ retorna um objeto *ImageData* que copia os dados de pixel para o retângulo especificado em área Canvas. Na verdade, é preciso esclarecer que um objeto *ImageData* não é uma figura. Ele é parte da área Canvas e manipula a informação de cada pixel dentro daquele retângulo. O método $canvas.toDataURL()$ permite obter o conteúdo da tela do navegador. O dado retornado é uma $string$ que representa uma URL codificada que contém os dados gráficos.

O objeto $history$ contém a lista das URLs visitadas pelo usuário dentro de uma janela do navegador. É parte do objeto $window$, possui a propriedade *length*, que retorna o número de URLs no histórico, e três (3) métodos: (1) *back()*, que carrega a URL anterior no histórico; (2) *forward()*, que carrega a próxima URL no histórico; e (3) *go()*, que carrega uma URL específica do histórico. É importante notar que embora não seja um padrão definido, todos os navegadores suportam este objeto. Olejnik et al. [Olejnik et al. 2013] mostraram que o histórico no navegador pode também ser usado como meio para realizar um *fingerprinting* sem a necessidade de ajuda do lado do cliente. Para tanto, eles analisaram uma base de dados gerada quando uma falha do CSS em relação ao histórico ainda estava presente em navegadores.

Por fim, o método $MediaDevices.getUserMedia()$ solicita ao usuário permissão

para usar um dispositivo de vídeo e/ou uma entrada de áudio, como uma câmera, compartilhar a tela e/ou um microfone. Tian et al. [Tian et al. 2014] apresentam múltiplos ataques usando $MediaDevices.getUserMedia()$ que podem comprometer a confidencialidade e integridade dos usuários. Eles mostram como um atacante pode roubar dados de um usuário da tela para modificar suas contas de acesso a site populares.

Tabela 3. Níveis de Severidade

Níveis	Objetos
1	window.innerHeight; window.outerWidth; window.outerHeight; window.devicePixelRatio; document.domain; navigator.userAgent; navigator.appCodeName; navigator.appName; navigator.appVersion; navigator.appMinorVersion; navigator.buildID; navigator.browserLanguage; navigator.cpuClass; navigator.doNotTrack; navigator.language; navigator.onLine; navigator.oscpu; navigator.platform; navigator.userLanguage; navigator.product; navigator.productSub; navigator.securityPolicy; navigator.systemLanguage; navigator.vendor; navigator.vendorSub; navigator.geolocation; navigator.savePreferences; screen.availHeight; screen.availWidth; screen.availLeft; screen.availTop; screen.height; screen.width; screen.colorDepth; screen.pixelDepth; screen.bufferDepth; screen.deviceXDPI; screen.deviceYDPI; screen.logicalXDPI; screen.logicalYDPI; screen.systemXDPI; screen.systemYDPI; screen.updateInterval
2	document.referrer; document.cookie; Modernizr; Silverlight; WebGL; navigator.cookieEnabled; navigator.javaEnabled; mimeTypes; plugins; window.localStorage
3	navigator.getUserMedia; canvas.toDataURL; canvas.getImageData; window.history

5. Experimentos e Resultados

Esta Seção descreve os passos executados para classificação dos websites quanto ao nível de severidade da *fingerprinting* em baixa, média ou alta. A primeira seção relata o protocolo experimental necessário (ambiente, base de dados, entre outros), enquanto a segunda descreve o processo de aplicação da metodologia proposta.

5.1. Ambiente

Os experimentos realizados foram executados em duas máquinas. A primeira é uma estação de trabalho Intel Corei7, 3,4 Ghz, com 8 GB de memória RAM, 500 GB de disco, com sistema operacional Linux, distribuição Ubuntu 14.04. A segunda um notebook Corei5, com 8GB de memória RAM, 1Tera de HD com sistema operacional Windows 8.

5.2. Avaliações

Antes de iniciar a explicação sobre as avaliações, é necessário especificar as bases de dados (páginas Web com *fingerprinting*) utilizadas neste trabalho. Foram utilizadas três (3) base de dados.

A primeira, denominada **base de controle**, é formada por 250 Websites classificados como *fingerprinting* nos artigos de Nickforakis [Nikiforakis et al. 2015], Khademi [Khademi 2014] e Acar et al. [Acar et al. 2013]. Esta base foi empregada na análise e ajuste de uso dos termos que compõem o dicionário. A segunda base, denominada de **base canvas**, é formada por 2.127 sites listados no trabalho de Englehardt et al. [Englehardt and Narayanan 2016] que utilizam API Canvas para realizar *fingerprinting*.

A terceira e última base, denominada **base DMOZ**, é composta por 1.585 sites extraídos aleatoriamente do diretório DMOZ[7], uma base que correspondem a sites considerados benignos. Esta base foi utilizada para verificação da quantidade de termos em páginas não ligadas à *fingerprinting*.

[7]http://www.dmoz.org/

5.2.1. Base de Controle

A Figura 2 ilustra o resultado da metodologia proposta na base de controle. Em destaque, na cor vermelha, estão apresentados os sites com classificação **Alta**, perfazendo um total de 87%. Os sites $coinbase.com$ e $tumblr.com$ possuem destaque nessa base por possuírem em seus códigos referências as quatro (4) propriedades que compõe o nível 3. Na cor verde estão representados os 13% dos sites classificados como **Média**.

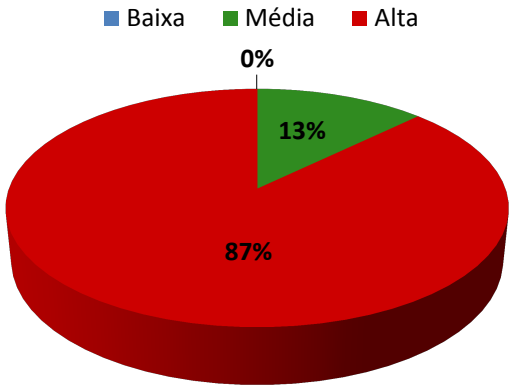

Figura 2. Classificação dos 250 Websites que compõem a Base de Controle

Percebe-se que por se tratar uma base que contêm apenas páginas empregadas para *fingerprinting* não existem artefatos de baixa severidade. Isso não significa que os elementos, objetos e métodos que compõe o Nível 1 não estejam presentes (na verdade, existem mais de 27000 métodos do nível 1 nas 250 páginas da base controle), mas sim que os elementos, objetos e métodos dos outros Níveis 2 e 3 estão presentes e representam um maior risco a privacidade do usuários.

Como forma de melhorar ainda mais o entendimento, traduzindo em números, nas 250 páginas existem 4 referências ao método $navigator.getUserMedia()$, 97 ao método $canvas.getImageData()$, 60 ao método $canvas.toDataURL()$ e 835 ao método $window.history()$.

5.2.2. Base Canvas

A Figura 3 ilustra o resultado da metodologia na base Canvas. Nesta base, observa-se a ocorrência dos 3 níveis de severidade, com maior prevalência para a classificação **Alta** (51%), visto que a base é composta por sites da base de *fingerprinting canvas*.

Diferente da base de controle, na base Canvas existem artefatos de todos os níveis. Nesta base, as páginas classificadas no nível 1 ou não possuem nenhuma referência aos elementos, objetos e métodos deste nível ou embora hajam referências, não existem referências dos outros níveis. Isso apenas comprova que as páginas que compõem esta base possuem artefatos que, de fato, apenas fazem uso correto dos métodos Canvas.

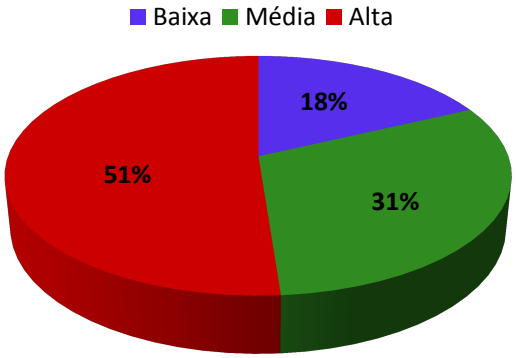

Figura 3. Classificação dos 2.127 Websites que compõem a Base Canvas

5.2.3. Base DMOZ

A Figura 4 ilustra o resultado da metodologia na base DMOZ. Ressalta-se os 33% de sites que possuem alguma das propriedades do nível 3 (no total foram detectados 522 sites para este nível).

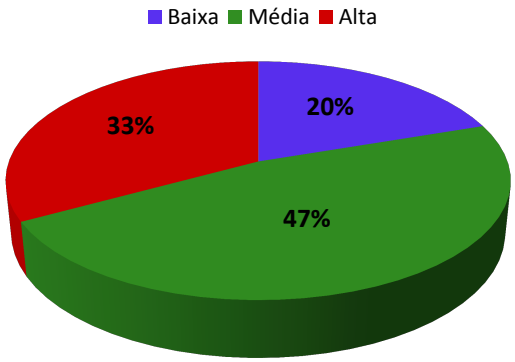

Figura 4. Classificação dos 1.585 Websites que compõem a Base DMOZ

Um aspecto interessante nos resultados desta base é o fato de sites, aleatoriamente escolhidos, de um repositório usado e reconhecido como contendo páginas benignas, apresentar artefatos de todos os níveis. Percebe-se ainda claramente que alguns destes sites tem um grande potencial para invasão de privacidade. Como forma de melhor ilustrar os resultados na base DMOZ, a Tabela 4 enumera as ocorrências de alguns dos termos que compõe o dicionário nesta base.

Os termos do **nível 1 - Baixa**, mencionados na Seção 4, são comumente usados para fornecer informações sobre o dispositivo do usuário. Por exemplo, para o site $pranapoweryoga.com$ houveram ocorrências para as propriedades $innerWidth$, $userAgent$, $screen.height$ e $screen.width$, indicando claramente a obtenção dessas informações para ajustes de tela e conteúdo.

Na coluna do **nível 2 - Média**, ganha destaque o site $bbc.com$ apresentando todos os termos dos níveis 1 e 2, mostrando que além de obter as informações de ajustes de tela, houve a verificação dos plugins e mimeTypes existentes no navegador.

Por fim, sobre as ocorrências do **nível 3 - Alta**, observa-se a prevalência dos termos de canvas como diferencial, uma vez que são usados para obtenção de informações

Tabela 4. Ocorrência de Termos encontrados nos sites da Base DMOZ

Websites	Nível 1 - Baixa					Nível 2 - Média					Nível 3 - Alta			
	innerWidth	innerHeight	userAgent	screen.height	screen.width	referrer	cookie	mimeTypes	plugins	localStorage	getUserMedia	getImageData	toDataURL	history
robertkleingallery.com	0	0	0	2	2	0	0	0	0	0	0	0	0	0
pranapoweryoga.com	1	0	3	2	2	0	0	0	0	0	0	0	0	0
joegreenephoto.com	1	5	3	1	0	0	0	0	0	0	0	0	0	0
captel.com	0	0	0	1	1	0	0	0	0	0	0	0	0	0
dynamicpost.co.uk	5	5	1	0	0	0	0	0	0	0	0	0	0	0
olddogpaws.com	0	0	2	2	3	1	36	0	3	0	0	0	0	0
ilight-tech.com	9	4	8	1	1	1	2	4	8	0	0	0	0	0
bbc.com	10	7	11	32	32	52	118	6	9	7	0	0	0	0
gourmetsleuth.com	6	6	22	6	6	12	6	16	0	0	0	0	0	0
nordicacademy.com.au	4	4	1	0	0	0	3	3	5	0	0	0	0	0
nikonusa.com	29	30	37	10	12	11	49	14	58	8	0	0	0	19
flairstrips.com	14	14	17	0	3	0	12	13	3	0	0	6	1	2
robotshop.com	8	8	13	3	3	7	13	4	6	4	0	3	1	1
fibergarden.com	13	17	9	2	2	5	20	4	5	4	0	3	1	4
newscooters4less.com	5	3	0	0	0	0	0	4	0	0	0	1	1	9

relevantes e que podem identificar o usuário unicamente na Web (conforme descrito em detalhes na Seção 4).

6. Dificuldades encontradas

Com o intuito de detectar *fingerprinting* em páginas Web, inicialmente foi pensada a utilização de grafo de dependência, grafos direcionados que representam relações de dependência entre elementos pertencentes a uma mesma estrutura. A ideia era criar um base de grafos dos scripts de *fingerprinting* que seriam comparados com os script extraídos de páginas Web em tempo real. Contudo, durante a pesquisa percebeu-se que não seria possível analisar as propriedades *fingerprinting* por meio do grafo de dependência, pois a principal finalidade do trabalho é detectar o uso de uma propriedade específica e não da estrutura como a mesma está disposta no código.

Outra ideia foi fazer uso de recuperação de informação, mais especificamente do Modelo Vetorial[8]. A ideia é que cada documento (página Web) fosse representado como um vetor de termos e cada termo possuísse um valor associado que indicasse o grau de importância (peso) deste termo em determinado documento. O ranqueamento dos scripts de *fingerprinting* foi analisado com o classificador k-NN. Contudo, nos resultados percebeu-se que todos os sites possuem algumas das propriedades de *fingerprinting*. Assim, sites rotulados como não *fingerprinting* foram ranqueados como *fingerprinting*.

Foi neste ponto da pesquisa que percebeu-se que praticamente todos os sites possuem alguma propriedade de *fingerprinting*. Assim, surgiu a ideia de classificar os sites pelo nível de severidade de acordo com o nível de risco que as propriedades detectadas nos scripts podem causar ao usuário.

[8]É um modelo onde documentos e consultas são vistos como característica num espaço vetorial n-dimensional, sendo a distância vetorial usada como medida de similaridade

7. Conclusões

As técnicas de *fingerprinting* aplicadas a páginas Web estão se tornando cada vez mais presentes e comuns, mesmo assim os usuários não são informados sobre isso. Embora as formas, os mecanismos e suas consequências sejam conhecidos, questões como a privacidade dos usuários e as formas de evitar a coleta de informações ainda são motivo de discussão. A existência de uma indústria de propaganda especializada, evoluída e financeiramente feliz, aliada ao surgimento de novos serviços e tecnologias, assim como, a constante evolução tecnológica e ao crescente número de usuários (em atividades que envolvem dados pessoais e valores) impõe novos desafios na atividade de detectar e mitigar o *fingerprinting*.

Este trabalho propôs uma metodologia para detectar artefatos (scripts) de *fingerprinting* em páginas Web e informar aos usuários o quão perigosa a página é para sua privacidade. Para tanto, o conjunto de propriedades e métodos dos objetos HTML DOM foram avaliados e o nível de risco à privacidade dos usuários foi estimado. Como forma de prová-la, várias bases de páginas Web contendo ou não *fingerprinting*, foram testadas e o resultado mostra que os artefatos para este fim são comuns e presentes em grande parte das páginas na Internet.

Referências

Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., and Preneel, B. (2013). Fpdetective: Dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 1129–1140, New York, NY, USA. ACM.

Boda, K., Földes, A. M., Gulyás, G. G., and Imre, S. (2012). User tracking on the web via cross-browser fingerprinting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7161 LNCS:31–46.

Bos, B., Celik, T., Hickson, I., and Lie, H. W. (2011). Cascading style sheets level 2 revision 1 (css 2.1) specification. W3c recommendation, W3C. `http://www.w3.org/TR/CCS2`.

BrowserLeaks.com (2015). Web browser security ? browserleaks.com. `https://www.browserleaks.com`.

Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and Smith, R. (2013). Privacy Considerations for Internet Protocols. RFC 6973 (Informational). `http://www.ietf.org/rfc/rfc6973.txt`.

Crockford, D. (2008). *JavaScript - the good parts: unearthing the excellence in JavaScript*. O'Reilly.

Eckersley, P. (2010). How unique is your web browser? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 1–18, Berlin, Heidelberg. Springer-Verlag.

Englehardt, S. and Narayanan, A. (2016). Online tracking: A 1-million-site measurement and analysis.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard). http://www.ietf.org/rfc/rfc2616.txt.

Forshaw, J. (2011). WebGL - A New Dimension for Browser Exploitation. http://www.contextis.com/resources/blog/webgl-new-dimension-browser-exploitation/.

Group, K. (2014). Webgl 2 specification. Khronos draft, Khronos Group. https://www.khronos.org/registry/webgl/specs/latest/2.0/.

Khademi, A. F. (2014). Browser Fingerprinting : Analysis , Detection , and Prevention at Runtime. (October).

Kirk, J. (2014). Canvas fingerprinting online tracking is sneaky but easy to halt. http://www.pcworld.com/article/2458280/canvas-fingerprinting-tracking-is-sneaky-but-easy-to-halt.html.

Mowery, K., Bogenreif, D., Yilek, S., and Shacham, H. (2011). Fingerprinting information in JavaScript implementations. In *Proceedings of Web 2.0 Security and Privacy 2011 (W2SP)*, San Franciso.

Mowery, K. and Shacham, H. (2012). Pixel perfect: Fingerprinting canvas in HTML5. In Fredrikson, M., editor, *Proceedings of W2SP 2012*. IEEE Computer Society.

Mulazzani, M., Huber, M., Leithner, M., and Schrittwieser, S. (2013). Fast and reliable browser identification with javascript engine fingerprinting. In *Web 2.0 Workshop on Security and Privacy*, (W2SP).

Nikiforakis, N., Acar, G., and Saelinger, D. (2014). Browse at your own risk. *Spectrum, IEEE*, 51(8):30–35.

Nikiforakis, N., Joosen, W., and Livshits, B. (2015). PriVaricator: Deceiving Fingerprinters with Little White Lies.

Olejnik, L., Castelluccia, C., and Janc, A. (2013). On the uniqueness of Web browsing history patterns. *Annals of Telecommunications - Annales Des Télécommunications*, 69(1-2):63–74.

Saraiva, A. R., Elleres, P. A., Carneiro, G. B., and Feitosa, E. (2014). Device fingerprinting: Conceitos e técnicas, exemplos e contramedidas. In *Livro de Minicursos do XIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg14)*, Belo Horizonte, MG, Brasil. SBC.

Tian, Y., Liu, Y. C., Bhosale, A., Huang, L. S., Tague, P., and Jackson, C. (2014). All your screens are belong to us: Attacks exploiting the html5 screen sharing api. In *2014 IEEE Symposium on Security and Privacy*, pages 34–48.

Unger, T., Mulazzani, M., Fruhwirt, D., Huber, M., Schrittwieser, S., and Weippl, E. (2013). Shpf: Enhancing http(s) session security with browser fingerprinting. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 255–261.

W3C (2014). Fingerprinting guidance for web specification authors. http://w3c.github.io/fingerprinting-guidance/.

