FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Classificação de Atributos através do Ganho de Informação para efeitos de Reconhecimento de Browsers

João Miguel de Carvalho Magalhães

Dissertação
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Professor Doutor Rosaldo Rossetti

Co-orientador: Mestre Pedro Fortuna

Janeiro 2011

© João Magalhães, 2011

Classificação de Atributos através do Ganho de Informação para efeitos de Reconhecimento de Browsers

João Miguel de Carvalho Magalhães

Dissertação

Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Resumo

Presentemente a maioria das pessoas já têm a possibilidade de acederem a um computador com acesso em banda larga para a Internet. Por esse motivo as pessoas passam cada vez mais tempo na Internet. Apercebendo-se desta oportunidade de negócio as empresas de publicidade começaram a publicitar os seu produtos pela Internet. Com o sucesso das campanhas, cresceu também o risco de existir fraude para terceiros lucrarem com o negócio. Como se irá discutir neste documento o *Click Fraud* tornou-se num problema para a forma de publicitar na Internet. O *Click Fraud* é um esquema onde, por exemplo, um utilizador clica em *banners* publicitários, sem qualquer interesse na publicidade, com o intuito de prejudicar a empresa publicitada ou obter algum retorno financeiro. E é neste contexto do *Click Fraud* que surgem as técnicas de reconhecimento de *browsers*. Estas técnicas podem ser usadas complementarmente com outras para identificar e combater os cliques fraudulentos que acompanham as campanhas publicitárias.

Neste documento, começamos então por fazer uma introdução dos intervenientes no projecto assim como uma introdução do problema a resolver. De forma a dar a conhecer a problemática do tema foi desenvolvido um capítulo sobre publicidade online onde foram analisados os diferentes tipo de publicidade na Internet, aos modelos de negócio existentes onde se englobou os vários modelos de receitas e o volume de negócio.

Feita esta análise foi efectuada uma análise à problemática do *Click Fraud* assim como das técnicas mais usadas para o praticar. Feita esta contextualização do problema percebe-se então qual a importância das técnicas de identificação de *browsers* para a detecção do *Click Fraud*. Assim, neste contexto surgem as técnicas de escolha de atributos como um instrumento para ajudar as ferramentas de reconhecimento de *browsers* a seleccionarem de uma forma mais rápida e eficiente os atributos mais importantes para a identificação do *browser*. Como tal serão discutidas algumas técnicas de escolha de atributos com um especial ênfase no Ganho de Informação, onde será posteriormente analisada a sua contribuição na selecção de atributos para uma ferramenta de detecção de *browsers*.

Para esse efeito foram desenvolvidos três componentes responsaveis, respectivamente, por efecutar uma análise estatística dos dados, determinar o Ganho de Informação de cada atributo e por determinar então qual o *browser* usado no clique. Os resultados obtidos, vieram confirmar a necessidade de haver uma selecção nos atributos usados, quer devido à redundância da informação, quer pela de informação reduzida de alguns atributos. A título experimental verificou-se também que o Ganho de Informação não se adequa a analisar atributos com valores distintos muito elevados pelo que foram efectuados testes para confirmar esse facto.

Abstract

Presently, most people already have the possibility of accessing a computer with broadband access to the Internet. For that reason, people are increasingly spending more time on the Internet. Having this business opportunity in view, advertising companies began to advertise their products through the Internet. With the success of such campaigns, also grew the risk of fraud where others try to profit from the business. As we shall discuss in this document, the Click Fraud has become a problem for the advertisers over the Internet. The Click Fraud is a scheme in which a user clicks on banner ads, without any interest in the advertised product, but with the intention of harming the companies advertised or get some financial return. It is in this context of Click Fraud that browser reconnaissance techniques arise. These techniques can be used complementarily with others to identify and combat fraudulent clicks from advertising campaigns.

In this document, we begin by making an introduction of the project participants as well as the problem that needs to be solved. In order to make the problems of Click Fraud known, was written a chapter on online advertising where we analyzed the different types of advertising on the Internet and the existing business models which included the various revenue models and the business revenue.

Following up this analysis, was addressed the issue of Click Fraud as well as the most used techniques for its practice. Once the contextualization of the problem is done, we realize the importance of the browser identification techniques in the detection of Click Fraud. So, in this context we can find attribute selection techniques as a tool to help to browsers reconnaissance techniques to identify faster and more efficiently the most important attributes of a browser. We will discuss some techniques for choosing attributes with a special emphasis on Information Gain and we will also analyze its contribution in the selection of the attributes for a browser detection tool.

For that purpose, were developed three components responsible, respectively, to analyze statistically the data assembled, to determine the Information Gain of each attribute and to determine which browser was used in the click. The results confirmed the necessity to determine the best attributes, either due to the redundancy of information retrieved, either due the small information retrieved by some attributes. As an experiment it was also found that the Information Gain is not an appropriate metric to use when we have attributes with a large number of distinct values.

Agradecimentos

Mais uma vez gostaria de agradecer à AuditMark por me ter prestado a possibilidade de ter realizado este trabalho dissertativo na empresa com um especial obrigado ao meu orientador Mestre Pedro Fortuna. Não poderia também deixar de agradecer ao meu colega da AuditMark, Engenheiro Pedro Antunes pelas toneladas de informação que me passou durante este trabalho. Gostaria também de agradecer ao meu orientador da faculdade, Professor Rosaldo Rossetti, pela amabilidade e paciência que teve comigo.

Para finalizar, gostaria de agradecer a todos os que me apoiaram e incentivaram em especial à Raquel por estar sempre comigo.

João Magalhães

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

WWW	World Wide Web
FEUP	Faculdade de Engenhartia da Universidade do Porto
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
TCP	Transmission Control Protocol
HTTP	Hypertext Transfer Protocol
HTML	HiperText Markup Language
DHTML	Dynamic HiperText Markup Language
JRE	Java Runtime Environment
XML	Extensible Markup Language
CSS	Cascading Style Sheets
DOM	Document Object Model
MIME	Multipurpose Internet Mail Extensions
SSL/TLS	Secure Sockets Layer/Transport Layer Security
FTP	File Transfer Protocol
RFC	Request for Comments
W3C	World Wide Web Consortium
IETF	Internet Engineering Task Force
RMI	Remote Method Invocation
API	Application Programming Interface
DPP	Data Processing Plugin
JS	JavaScript
JSTEST	Teste JavaScript
HTTPTEST	Teste HTTP

Capítulo 1

Introdução

Este capítulo tem como objectivo fazer a introdução do tema que irá ser desenvolvido assim como apresentar a empresa proponente da dissertação e as pessoas responsáveis pelo projecto. Assim será feita uma descrição do problema a resolver e serão delineados os objectivos propostos para a resolução do mesmo.

1.1 Apresentação da Auditmark

A AuditMark é uma start-up que se encontra sediada na UPTEC – Parque da Ciência e Tecnologia da Universidade do Porto – que tem como base a auditoria de tráfego proveniente de campanhas publicitárias online, sendo um dos seus principais temas de trabalho a detecção de Click Fraud em anúncios publicitários. É neste contexto que se apresenta a proposta de dissertação da AuditMark sobre o tema "Classificação de Atributos através do Ganho de Informação para efeitos de Reconhecimento de Browsers". Os orientadores da dissertação são o Professor Doutor Rosaldo Rosseti, FEUP, e o Mestre Pedro Fortuna, AuditMark.

1.2 Introdução do Problema

Actualmente o uso do computador pela população mundial tornou-se algo perfeitamente normal estando este disponível a quase qualquer um. Acompanhando esse facto, a Internet sofreu também evoluções que a transformaram num óptimo meio de comunicação. Assim verificou-se uma massificação do seu uso ao longo da última década, sendo actualmente explorados outras bases para além dos computadores, como por exemplo os telefones móveis ou as televisões.

Acompanhando essa massificação as empresas publicitárias viram a Internet como um óptimo meio para exporem os seus produtos. O sucesso inicial das campanhas publicitárias trouxe alguma atenção sobre as mesmas. Acontece que numa primeira fase foi usado um modelo de negócio muito idêntico ao que era utilizado na publicidade convencional onde se verificou que possuía falhas que poderiam ser aproveitadas por terceiros para prejudicar ou beneficiar intencionalmente a publicitação criando um novo tipo de fraude. De entre vários tipos de fraude surgiu a fraude de cliques, a Click Fraud. Esta fraude consiste em intencionalmente clicar nos anúncios publicitários de forma a sugerir um interesse no anúncio publicitário.

Para tentar detectar quando este de tipo de fraude ocorre existem algumas técnicas que se podem aplicar. De entre essas técnicas surge a técnica de reconhecimento de browsers. Esta técnica surge como uma forma de detectar qual o browser utilizado na realização do clique. Assim, utilizadores que procurem mascarar o seu browser noutros de forma a conseguir realizar Click Fraud serão detectados.

Na aplicação da técnica de reconhecimento de browsers pode ser recolhido uma quantidade de dados por clique relativamente grande. Acontece que alguns atributos que cada clique possui podem não conter qualquer tipo de informação que permita induzir qual o browser que foi utilizado. Nesse sentido surge então a necessidade de aplicar algum tipo de técnica que permita determinar e classificar quais os atributos mais importantes para um correcto reconhecimento dos browsers. Uma dessas técnicas de selecção de atributos é o Ganho de Informação que será abordado neste projecto.

Presentemente a Click Fraud é um negócio que gera milhões de dólares em prejuízos e como tal é necessário detectar e combater a mesma. A AuditMark vendo a problemática deste facto propôs o tema '' Classificação de Atributos através do Ganho de Informação para efeitos de Reconhecimento de Browsers '', que irá ser dissertado neste documento.

1.3 Objectivos

Para a realização deste projecto foram definidos os seguintes objectivos:

1. O primeiro objectivo centra-se no desenvolvimento de uma ferramenta, totalmente integrada no serviço de auditoria da AuditMark, que consiga escolher e classificar correctamente quais os melhores atributos para um correcto reconhecimento de browsers.

2. O segundo objectivo prende-se com a adaptação de uma solução já existente para o reconhecimento de browsers para que suporte o novo modelo de dados vigente na AuditMark, assim como a ferramenta desenvolvida para a escolha dos melhores atributos.

Com o cumprimento destes dois objectivos dar-se-á por concluído o projecto, integrando no serviço de auditoria da AuditMark uma ferramenta de detecção de browsers.

1.4 Estrutura do Documento

O documento encontra-se dividido em 7 capítulos.

No primeiro capítulo é feita uma introdução do projecto assim como da empresa proponente, neste caso a AuditMark. Serão também defindos os objectivos que este trabalho se propõe a alcançar.

No segundo capítulo é feita uma análise aos diferentes tipos de publicidade na Internet, com um aprofundar para a publicidade via Web. Neste contexto são analisados os modelos de negócio e de receitas existentes, assim como o volume de negócio que esta indústria movimenta. Para finalizar, é ainda discutido neste capítulo o conceito de Click Fraud, referindo-se em que é que consiste e quais as técnicas para o praticar.

No terceiro capítulo são então abordadas as técnicas de identificação de browsers, assim como realizada uma descrição superficial do funcionamento do serviço de auditoria da AuditMark.

No quarto capítulo é feita uma abordagem a algumas técnicas de escolha de atributos que já foram desenvolvidas, com uma especial análise ao Ganho de Informação.

No quinto capítulo é exibida a proposta de implementação para a resolução do problema proposto.

No sexto capítulo são mostrados os testes feitas à solução implementada, assim como será feita uma validação da mesma.

Finalizamos com o capítulo sete, que encerra este documento com as conclusões gerais deste trabalho e onde são apontadas futuras implementações e aplicações do trabalho desenvolvido noutras áreas.

Capítulo 2

Publicidade Online

A Internet é, sem dúvida alguma, uma das formas de comunicação mais populares do planeta, devendo esse facto às inúmeras facilidades e serviços disponibilizados. Actualmente é possível consultar o correio electrónico, vulgarmente conhecido como *e-mail*, fazer transferência de ficheiros, comunicar por texto, voz ou vídeo e, talvez o serviço mais popular, a utilização da World Wide Web (WWW). A WWW é um conjunto de documentos e conteúdos multimédia disponível sobre a forma de páginas Web. Estas páginas podem ser acedidas através de Uniform Resource Identifiers (URIs), único por cada página existente, que permite a um utilizador escolher qual a página que pretende visitar. De modo a facilitar o acesso ao conteúdo de uma página Web, foram então desenvolvidos browsers.

Os browsers permitem, com a inserção do URI da página Web, recolher, através do protocolo HTTP, o conteúdo da página desejada do servidor Web onde a página se encontra alojada. As páginas são geralmente desenvolvidas obedecendo ao protocolo HTML, para que possam posteriormente ser processados pelos browsers que as reconstroem para serem visualizadas pelo utilizador. A World Wide Web veio desta forma proporcionar a livre circulação da informação e ideias, adquirindo assim a popularidade que tem nos dias de hoje.

Para usufruírem dos serviços disponibilizados na Internet, na maioria dos casos, os utilizadores não necessitam de pagar nenhuma taxa à entidade que lhes presta o serviço. No entanto, a prestação do serviço tem um custo para essa mesma entidade, que cresce em conformidade com a dimensão do serviço prestado. Se aliarmos a isso a popularidade que a Internet possui, torna-se assim evidente a utilização da publicidade *online* como uma forma de suportar financeiramente essas entidades.

2.1 Tipos de Publicidade *Online*

Da mesma forma que existe vários serviços de Internet, existe também diferentes formas de se fazer publicidade *online*. Actualmente existe 2 grandes formas de publicidade *online* com base no funcionamento de dois serviços de Internet distintos, a World Wide Web e o *E-mail* [1].

2.1.1 Publicidade Via Web

Este tipo de publicidade consiste na utilização de websites como plataforma publicitária. São colocados em websites pequenos anúncios que publicitam um produto e encaminham o utilizador para o website do anunciante. Actualmente o desenvolvimento destes anúncios sofreu um forte impulso, conduzindo a que os conteúdos publicitários possam ser alterados dinamicamente com base no país do utilizador, últimas pesquisas efectuadas ou até pelos sítios Web visitados recentemente. A seguir serão descritos quais os tipos de anúncios mais comuns nos sítios Web [2].

- ***Banner*:** Este tipo de anúncio era originalmente uma imagem que pudesse cativar a atenção do utilizador colocada no sítio Web. Actualmente, este tipo de anúncio tem sofrido evoluções, com a utilização de recursos JAVA, Flash ou HTML5, de forma a conseguir cativar cada vez mais utilizadores e dinamizar a publicidade.

- ***Banner* Flutuante:** É um anúncio nos mesmos moldes do anterior, mas que se desloca ao longo da página ficando sempre visível enquanto o utilizador circula pela página.

- ***Inline Text*:** Esta forma de publicidade funciona da seguinte forma: A agência publicitária coloca, sobre determinadas palavras ou frases do sítio Web expositor, hiper-ligações para o website do anunciante. De modo a expor a publicidade, é normal colocar o respectivo texto numa cor diferente e aparecer uma pequena imagem ou animação publicitária sempre que o rato passar por cima do referido texto. Esta forma de publicidade é comum em blogs e sítios Web com muita informação escrita.

- **Anúncio em Vídeo:** Este tipo de publicidade pode ser feito de duas formas. A primeira forma consiste em colocar sobre o vídeo um pequeno banner que acompanha a reprodução do mesmo. A segunda forma é colocar vídeos publicitários antes do vídeo que o utilizador pretende visualizar. Este tipo de publicidade tem sofrido um forte impulso com o aumento da informação difundida sobre a forma de vídeo na Internet.

- ***Trick Banner*:** O funcionamento é idêntico ao *banner* tradicional com a diferença de tentar induzir em erro o utilizador de forma a tentar o mesmo a clicar no anúncio.

É usual este tipo de *banner* evidenciar semelhanças com erros do sistema operativo ou prémios pelo utilizador ser a enésima visita ao sítio Web. Geralmente este tipo de *banner* possui *scripts* de código malicioso ou conduz a sítios Web com fins maliciosos.

- ***Pop-up*:** Este método consiste em abrir novas janelas com conteúdo publicitário sempre que o utilizador circula entre páginas. Este método caiu em desuso pois os browsers adoptaram medidas para prevenir que páginas novas abrissem sem o consentimento do utilizador.

2.1.2 Publicidade Via *E-mail*

Relativamente à publicidade via *e-mail*, o processo de publicidade é igualmente simples. O anunciante envia um *e-mail* a todos os potenciais interessados no seu produto e espera por uma eventual interesse por parte do destinatário. Quando a popularidade do *e-mail* cresceu, houve um interesse enorme dos publicitários de aderir a esta forma de expor os seus produtos, já que o envio deste tipo de mensagens é um processo grátis não sendo necessário pagar nenhuma taxa por e-mail enviado à entidade que presta o serviço.

Desta forma foram criadas ferramentas para enviar mensagens publicitárias para o maior número de endereços de e-mail possível de uma forma automática. Assim sendo, logo os utilizadores viram as suas caixas de e-mail cheias de e-mails que não lhes interessava, o vulgar *SPAM*. Acompanhando as queixas dos utilizadores, as entidades que prestavam os serviços de e-mail resolveram proporcionar filtros *anti-SPAM* aos utilizadores, assim como dotar o serviço de ferramentas de detectar e neutralizar endereços que originam *SPAM*. Mais recentemente, os principais governos mundiais decidiram agir contra quem pratique *SPAM*. Em 2002, a União Europeia redige a directiva 2002/58 que proíbe o uso do *SPAM* para fins comerciais. Estabelece também que tem que partir do utilizador a intenção de receber publicidade criando-se assim as listas opt-in. É também garantido ao utilizador a possibilidade de a qualquer momento cessar a sua intenção de receber mais *e-mails* publicitários.

Devido a todos os inconvenientes que a publicidade por *e-mail* possui, para o anunciante e anunciado, esta possui na actualidade pouco peso no mercado mundial de publicidade *online*, sendo este quase totalmente preenchido pela publicidade via Web. No contexto deste trabalho interessa-nos perceber a publicidade *online* via Web pelo que esta será melhor retratada no ponto seguinte.

2.2 Modelos de Negócio

Numa forma convencional, se um utilizador tiver a intenção de publicitar um produto seu, recorre a uma agência publicitária que o ajuda a anunciar o produto ao seu mercado

alvo. De igual forma pode ser usado o mesmo modelo de negócio na publicidade *online*. As agências publicitárias detêm uma lista de sítios Web onde podem colocar os seus anúncios publicitários sendo os proprietários dos mesmos compensados financeiramente. Desta forma, podemos considerar quatro actores no processo de publicidade *online*[3]:

- **O utilizador:** São todos os indivíduos que circulam na Web podendo ou não ter interesse na publicidade *online*.
- **O anunciante:** É a pessoa ou empresa que pretende publicitar algum tipo de produto.
- **A agência publicitária:** É a empresa que o anunciante contacta para que seja efectuada a publicidade.
- **O sítio Web expositor:** É o sítio onde é feita a publicidade.

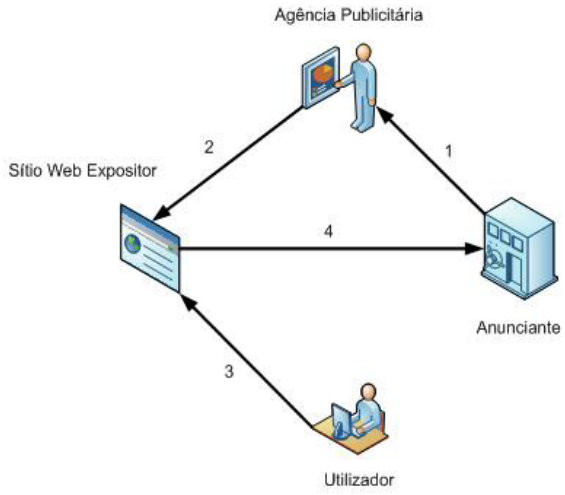

Figura 2.1: Diagrama de um modelo de negócio aplicavel à publicidade *online*.

Na figura 2.1 podemos observar como o processo de publicitar um conteúdo na Web se desenvolve.

1. O anunciante contacta a agência publicitária para publicitar a sua empresa/produto.
2. A agência publicitária faz um estudo sobre a publicidade necessária e coloca a publicidade em sítios Web parceiros mais indicados para fazer a divulgação.
3. O utilizador visualiza a publicidade e interage com a mesma.
4. O utilizador é redireccionado para o sítio Web da empresa concluindo assim o ciclo de promoção da empresa/produto.

Uma das principais funcionalidades desenvolvidas para a Web foi o desenvolvimento de motores de pesquisa. A função destes é simplesmente de organizar a informação de cada página Web e dar informação ao utilizador de quais os URIs que mais se adequam à pesquisa do utilizador. Podemos assim considerar que os motores de pesquisa se tornaram algo indispensável na Web, quer pela sua necessidade quer pela facilidade de utilização. Explorando o facto dos sítios dos motores de pesquisa obterem milhões de visitas diariamente, as empresas que os desenvolveram decidiram apostar no mercado de publicidade *online*. Este é talvez o expoente máximo de publicidade *online* onde estes sítios possibilitam diferentes opções de publicitar. Entre algumas destas opções encontram-se a possibilidade de reservar espaço na página onde é mostrada a publicidade ou até mesmo a melhoria da posição da pesquisa, estando a publicidade mostrada ligada directamente ao tipo de pesquisas que o utilizador efectua.

2.2.1 Modelos de Receita

Historicamente, os modelos de receita foram sofrendo alterações com base na experiência das agências de publicidade *online* e nos requisitos dos anunciantes. Inicialmente era aplicado um modelo de receita idêntico ao praticado pelas agências de publicidade convencionais mas, pelas próprias características da Web houve necessidade de renovar o modelo. Deste modo surgiram vários modelos de receita, onde se destacam 3 modelos: *Cost per Mile* (CPM), Cost per Click (CPC), *Cost per Action* (CPA) [3].

- ***Cost per Mille* (CPM)** - Este modelo também é conhecido por *Cost per Thousand* ou *Cost per Impression*. No modelo de receita convencional, o anunciante paga a uma empresa publicitária por cada impressão efectuada, isto é, por cada vez que a empresa publicitária coloca a publicidade nos meios expositores. Transportando este modelo para a publicidade *online*, significa que o anunciante vai pagar à empresa publicitária em conformidade com o número de vezes que a publicidade for "visualizada" por utilizadores. Esta visualização é contabilizada aquando do carregamento com sucesso da imagem publicitária. Na maioria das empresas publicitárias o acto de recarregamento da página ou outras acções externas não é considerado como uma impressão, salvaguardando assim os interesses dos anunciantes. Nalguns casos, este tipo de modelo pode ser bastante lesivo para o anunciante pois o utilizador pode não ver a publicidade, mas o anunciante irá ter que pagar por essa impressão.

- ***Cost per Click (CPC)*** - Este modelo também é conhecido por *Pay per Click* (PPC). Como o CPM não conseguia garantir as necessidades dos anunciantes, surgiu este modelo de receita que se adequa bem à realidade da publicidade *online* via Web. Ao contrário do CPM, o anunciante só irá pagar se o anúncio publicitário receber um clique e for redireccionado com sucesso para o sítio do anunciante. Desta forma é

garantido que o anunciante só irá pagar pelo número de utilizadores que clicarem na publicidade, embora, mesmo assim, não garanta que existe interesse por parte dos utilizadores. Este é o modelo de receita implementado nos motores de pesquisa onde o anunciante pede para ser colocada a sua publicidade em resposta a palavras-chave usadas nas pesquisas dos utilizadores. É a soma de todas estas mais valias que torna este modelo de receita o principal modelo usado na publicidade *online*.

- ***Cost per Action* (CPA)** - Este modelo também é conhecido por *Cost Per Acquisition*. Este modelo de receita consiste em só haver pagamento por parte do anunciante quando uma determinada acção é executada, sendo esta acção previamente acordada entre a agência publicitária e o anunciante. Este é sem dúvida o modelo que mais beneficia o anunciante, pois este consegue garantir o interesse do utilizador na sua publicidade. Para a agência publicitária este é o modelo que menos a beneficia, pois fica dependente do interesse que o produto publicitado gere no utilizador.

Existem outros modelos de receita menos importantes pelo que serão descritos a seguir de forma sucinta.

- ***Cost Per Visitor* (CPV):** O anunciante paga por cada vez que um utilizador chega ao seu sítio Web através de um anúncio publicitário.

- ***Cost Per View* (CPV):** O anunciante paga por cada vez o utilizador visualiza a publicidade. É usado nos anúncios do tipo *Pop-up* e anúncio de vídeo.

- ***Cost Per Lead* (CPL):** É uma particularização do CPA, onde o anunciante paga pelo numero de utilizadores que completa um processo de inscrição (por exemplo: *Newsletter*).

- ***Cost Per Order* (CPO):** É uma particularização do CPA, onde o anunciante paga pelo número de utilizadores que completam uma compra ou encomenda com sucesso.

- **Cost Per Engagement (CPE):** É uma particularização do CPA, onde o anunciante paga pelo número de interacções que o anúncio publicitário sofre por parte de utilizadores. Este tipo de anúncios é geralmente do tipo *banner* com pequenos jogos para cativar o utilizador.

Na figura 2.2, podemos ver a distribuição dos modelos de receita para o 1º semestre de 2010. Aqui podemos verificar a supremacia dos modelos CPC e CPA sobre os outros modelos e assim verificar a sua popularidade na sociedade [4].

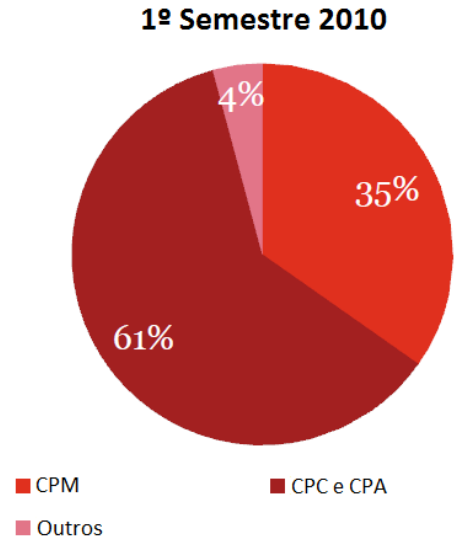

Figura 2.2: Distribuição dos modelos de receita aplicados na publicidade *online*.

2.2.2 Volume de Negócio

A popularidade da publicidade *online* deve-se essencialmente ao facto de esta ser uma forma bastante lucrativa de fazer publicidade. Segundo o estudo do *Interactive Advertising Bureau* [4], a receita com a publicidade *online* tem vindo a aumentar com um volume de negócio a ascender aos 22,6 milhares de milhão de dólares (Figura 2.3) nos Estados Unidos da América, para 2009. Apesar de nesse ano ocorrer uma ligeira quebra nas receitas, o primeiro semestre trouxe melhorias em relação às receitas em comparação com os anos anteriores, pois foi batido novo recorde para igual periodo, havendo a especulação de que 2010 seria um ano onde as receitas anuais atingiram um novo recorde. Como tal estes números vêm provar a adesão da sociedade a publicitarem os seus produtos através da Internet.

2.3 *Click Fraud*

Tal como foi abordado anteriormente o modelo de receita mais usado é o CPC, onde os anunciantes pagam pelo número de cliques que a publicidade sofreu. Este modelo, apesar da sua popularidade, possui vulnerabilidades em relação ao crime electrónico, pois devido às características do modelo não é possível classificar de uma forma inequívoca se os cliques que a publicidade sofreu foram ou não de clientes interessados na publicidade. Desta incerteza surge a *Click Fraud*. A *Click Fraud* consiste em provocar, de forma automática (usando *scripts* ou aplicações) ou manual, cliques em anúncios publicitários de forma intencional mas sem qualquer interesse na publicidade exposta. Este tipo de fraude pode ter vários objectivos, desde gerar prejuízos aos anunciantes, gerar lucros

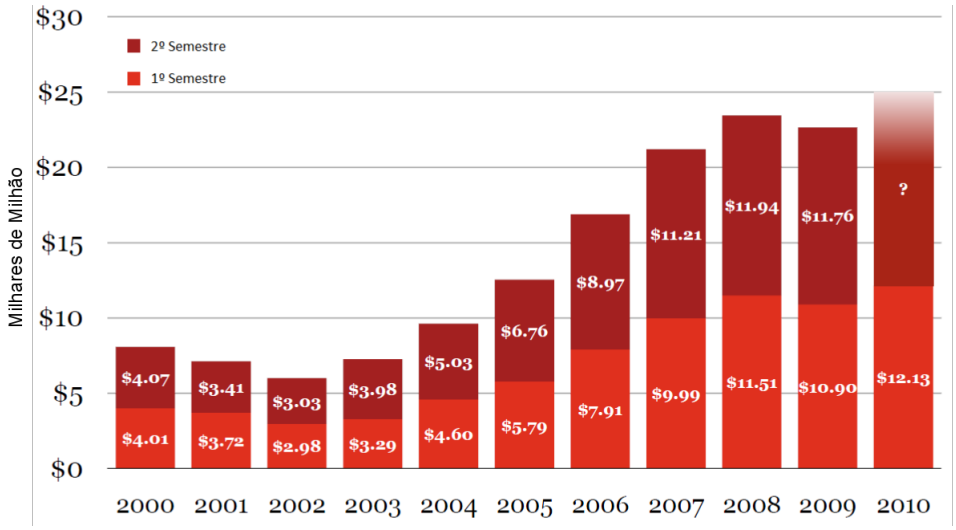

Figura 2.3: Volume de negócio da publicidade *online*.

aos sítios expositores ou tentar denegrir a qualidade da publicidade prestada. Segundo a ClickForensics, uma empresa de auditoria do tráfego Web, a percentagem de cliques fraudulentos têm vindo a aumentar gradualmente ao longo dos anos, acabando o terceiro quadrimestre de 2010 com uma taxa de 22.3% de cliques fraudulentos entre os cliques analisados. A *Click Fraud* pode ser efectuada de duas maneiras distintas: através da simulação de browsers ou através do uso de browsers. [5, 6]

- **Através da simulação de browsers:** Neste caso, os pedidos feitos ao servidorWeb nunca imitam um pedido de um browser real de uma maneira completa. Como tal é possível detectar inconsistências na estrutura do pedido que permite suspeitar do clique analisado. Uma das técnicas para verificar a consistência dos pedidos é através de técnicas de reconhecimento de browsers.

- **Através do uso de browsers:** Como o browser não é simulado o seu comportamento é autêntico no que diz respeito à estrutura do pedido. Para tal serão precisas técnicas diferentes para podermos detectar possíveis cliques fraudulentos, como por exemplo a análise aos IPs dos cliques ou a identificação dos utilizadores através das sessões Web.

De seguida serão descritas algumas das técnicas usadas para a prática de *Click Fraud* [5, 3].

- **Cliques Manuais:** A prática de *Click Fraud* através de cliques manuais consiste em o utilizador, manualmente, clicar nos anúncios dos anúncios, sendo este é o método mais básico para a prática da *Click Fraud*. Para o utilizador torna-se fácil mascarar a sua intenção alterando o seu IP com regularidade ou mudando de browser. A única

desvantagem para o utilizador é a quantidade enorme de tempo que necessita de disponibilizar para o clique nos anúncios.

- **Ferramentas de Clique Automático:** As ferramentas de clique automático são pequenos programas ou scripts que clicam de uma forma automática nos anúncios de um determinado endereço. Estes scripts são muito mais eficientes, em comparação com o método de cliques manuais, e podem ser sofisticados o suficiente para mudar de endereço IP, mudar de browser ou percorrer uma lista de *proxies* por cada clique efectuado.

- ***Click Farms*:** *Click Farms* são grupos de pessoas que são pagas para clicarem nos anúncios publicitários agindo como utilizadores vulgares. As pessoas são contratadas para clicarem nos anúncios em troca de uma pequena quantia de dinheiro. Este método de *Click Fraud* é muito difícil de ser detectado pois os utilizadores são reais e existe a possibilidade de estes circularem pelos sítios Web dos anunciantes para aumentar a credibilidade do clique na publicidade. Este método está a sofrer um forte desenvolvimento em alguns países do terceiro-mundo e países asiáticos.

- ***Botnets*:** As *Botnets* são grupos de computadores que foram infectados por *malware* e que por isso se encontram comprometidos. Aos computadores infectados é atribuída a denominação de *zombie*. Desse facto advém que se torna possível a um agente que tenha conhecimento dos computadores infectados, controlar os mesmos para fins ilícitos. No caso da *Click Fraud* é possível colocar vários computadores em diferentes partes do mundo a executar cliques em publicidade, aumentando assim a probabilidade de serem considerados como cliques lícitos. Na generalidade dos casos, os utilizadores dos computadores infectados nunca se chegam a aperceber que fazem parte de uma *Botnet*, mesmo durante um ataque.

- ***Affiliate Programs*:** Este método tem algumas semelhanças com o *Click Farms*. Neste caso, os utilizadores são seduzidos a inscreverem-se num sítio Web onde lhes é disponibilizado um número de anúncios que podem clicar por dia. Sempre que cumprir esse requisito o utilizador recebe pontos para trocar por produtos ou uma pequena quantia monetária. Desta maneira, apesar de não haver interesse por parte do utilizador, fica muito difícil de suspeitar que os cliques são, na realidade, fraudulentos.

2.4 Resumo

A publicidade *online* é sem dúvida uma mais-valia para as agências publicitárias, sendo tal comprovado pelo elevado volume de negócio por ela movimentado. A grande parte das agências publicitárias usam o modelo de receitas *Cost Per Click*, um modelo

muito popular e bem aceite pelos anunciantes, mas que possui um modelo de receitas com um problema, pois é não possível determinar com toda a certeza a intencionalidade de um utilizador que clicou no anúncio publicitário. Daí surge o problema da *Click Fraud* onde utilizadores maliciosos podem tentar ganhar dinheiro indevidamente ou prejudicar intencionalmente o anunciante. Como tal torna-se imperativo o controlo da legalidade dos cliques cobrados aos anunciantes através de técnicas de identificação de *Click Fraud*. Neste capítulo conclui-se que, apesar de o modelo *Cost per Click* ser vulnerável a *Click Fraud*, não existe nenhum sério opositor a este modelo de receita, pois é o que na generalidade traz mais benefícios para todas as partes envolvidas no negócio. Apesar de a *Click Fraud* já se encontrar bem definida, é necessário encontrar mais mecanismos de a detectar e neutralizar.

Capítulo 3

Reconhecimento de Browers

A concorrência no mercado de *browsers* originou que existam actualmente na Internet um número enorme de *browsers* com características e funcionalidades diferentes. Tal como foi discutido anteriormente no capítulo 2, a *Click Fraud* é um grave problema para a indústria de publicidade *online* pelo que se torna necessário encontrar ferramentas que permitam classificar os cliques de forma a determinar a sua validade. Uma dessas técnicas é a detecção do *browser* do cliente efectua o clique, pois permite verificar se o *browser* que originou o pedido é real e se corresponde com a mais básica informação que é possível retirar dos *browsers*. Para ser feito o reconhecimento do *browser* existem actualmente algumas técnicas que tiram partido das particularidades de cada *browser* de forma a poder encontrar uma possível identificação do *browser* que originou o clique. Neste capítulo serão abordadas as técnicas que existem para o reconhecimento de *browsers* assim como uma descrição do serviço de auditoria da AuditMark tal como descrito por Rui Polónia [7].

3.1 Técnicas de Reconhecimento de *Browsers*

O reconhecimento do *browser* é uma peça muito importante na detecção de *Click Fraud* pelo que é importante a sua investigação e desenvolvimento. A forma mais fácil obter algum tipo de informação sobre o *browser* utilizado por um utilizador é a leitura do campo user-agent. Este campo é enviado aquando o envio do método GET, no protocolo HTTP, onde se encontra descrito sobre forma de texto qual é o *browser* usado pelo utilizador.

Este método seria suficiente para a sua identificação, não fosse simples a um utilizador com intenções de mascarar a sua navegação pela Web adulterar o seu resultado, pois actualmente existem inúmeros /textitAdd-ons ou métodos para cada *browser* poder alterar

a informação enviada no campo user-agent. Como tal, este tipo de informação não deverá ser considerada para uma correcta identificação do *browser*, mas antes como um tipo de informação usado para confirmar a veracidade da informação que o *browser* transmite.

Como é possível verificar, ao aceder à Web, existem no mercado diversos *browsers* com especificidades diferentes de forma a cativar o utilizador à sua utilização. Serão essas especificidades que irão tornar possível uma distinção entre os vários *browsers* que existem no mercado. A primeira grande diferença que se verifica entre eles deriva dos diferentes motores que os *browsers* utilizam, pois são estes que determinam como a página Web irá ser processada e como é feita a comunicação com o servidor. A segunda diferença vem do suporte que cada *browser* oferece às diferentes linguagens de programação ou plataformas que existem para a execução de conteúdos Web. Esta diferença é talvez a mais importante, pois além de permitir fazer uma distinção entre *browsers* de empresas diferentes, permitirá também detectar diferentes versões de um mesmo *browser*. Em suma será a junção destas duas grandes diferenças que nos permitirá identificar com algum grau de certeza qual o *browser* utilizado na execução do clique.

3.1.1 Análise dos Cabeçalhos HTTP

Com o desenvolvimento da Web foi necessário criar protocolos que permitam a comunicação entre clientes e servidores. Como tal, surge em 1990 a primeira versão do protocolo HTTP com a versão 0.9 desenvolvida pelo britânico Tim Berners-Lee, actual director da World Wide Web Consortium (W3C), que pela primeira vez introduz o uso de URIs para a identificação de páginas Web e de hiper-ligações de forma a facilitar a navegabilidade entre páginas. Assim, em 1996 surge o RFC (*Request for Comments*) 1945 que descreve a versão 1.0 do protocolo HTTP.

Com o aumento da popularidade do protocolo, em 1999, uma parceria entre o W3C e o IETF lançam a actual versão do protocolo HTTP, versão 1.1 descrita no RFC 2616, que vem colmatar as deficiências da anterior versão em relação aos requisitos que a Web necessitava, tais como suporte a outros protocolos (SMTP, FTP, NNTP, entre outros), a possibilidade de uso de Web *proxies* e principalmente a possibilidade de troca de mensagens sem ser no formato texto, as mensagens tipo MIME (*Multipurpose Internet Mail Extension*). O protocolo HTTP é constituído por 9 métodos, descritos na tabela 3.1 que se destinam a assegurar a comunicação entre o cliente, *browser*, e o servidor Web [8].

De todos estes métodos os mais usais de serem visualizados na comunicação de um *browser* com um servidor são o comando GET e PUT.

3.1.1.1 Análise pelo método GET

O trabalho desenvolvido por David Carvalho [9], relativamente à identificação de *browsers* pelo comportamento do protocolo HTTP, sugere que os *browsers* têm com-

Tabela 3.1: Métodos HTTP

Método	Descrição
CONNECT	Usado para estabelecer uma ligação a um túnel TCP/IP. Este método é usado geralmente no uso do SSL nas ligações HTTPS.
DELETE	Este método é usado quando o cliente pretende que o servidor apague, ou mova para uma localização inacessível, um recurso identificado através do URI.
GET	O método GET é usado pelo cliente para recolher a informação disponível no URI indicado.
HEAD	Este método é usado para recolher apenas o cabeçalho HTTP.
OPTIONS	Retorna quais os métodos HTTP que o servidor suporta.
PATCH	Este método é usado para actualizar recursos.
POST	O método POST é usado quando se pretende enviar informação para ser processada para o outro interveniente da ligação.
PUT	Este método é utilizado para o envio de informação para o outro interveniente da ligação.
TRACE	Utilizado pelo cliente para determinar o que os servidores intermédios alteram no seu pedido.

portamentos diferentes na forma como constroem as mensagens a serem enviadas para o servidor Web, através do método GET. Este método possui um conjunto alargado de campos reconhecidos pela comunidade científica, mas geralmente só são declarados um número reduzido dos mesmos.

Os campos mais vulgarmente declarados pelos *browsers* Web podem ser observados na tabela 3.2. De notar que os campos enviados podem ser definidos sem qualquer tipo de entrave, pelo que na realidade existe um número indeterminado de campos que circulam pela Internet.

Numa análise a diferentes *browsers* foi então possível constatar dois tipos de diferenças aquando do uso do método GET. Foram assim detectadas diferenças na forma e no conteúdo das mensagens transmitidas, que podem ser visualizadas na figura 3.1.

- **Diferenças na forma:** Esta diferença surge essencialmente devido aos diferentes motores usados pelos *browsers*. Como foi verificado no trabalho desenvolvido por David Carvalho [9], os *browsers* não constroem o método GET exactamente da mesma forma. Entre os vários *browsers* existem assim discrepâncias no tipo de campos utilizados, onde é possível identificar campos únicos a *browsers* específicos, e na ordem que os campos assumem no pedido.

- **Diferenças no conteúdo:** Foram detectadas também diferenças no conteúdo de alguns campos do método GET. Estas diferenças são compostas essencialmente por diferenças de capitulação e diferenças na quantidade de informação transmitida ao servidor. Estas diferenças a título de exemplo podem ser detectadas no campo *Accept-Language*.

Tabela 3.2: Campos mais vulgares do método GET

Campo	Descrição
Accept	Especifica quais os tipos de mensagens que o cliente aceita por parte do servidor.
Accept-Charset	Especifica qual o conjunto de caracteres que o *browser* aceita por parte do servidor.
Accept-Encoding	Especifica quais os tipos de compactação que o *browser* aceita por parte do servidor.
Accept-Language	Especifica quais a linguagens preferidas pelo *browser*.
Connection	Especifica quais as opções desejadas da ligação.
Host	Especifica qual o Host ao qual é feito o pedido.
Referer	Especifica qual o host de onde surgiu o pedido HTTP.
User-Agent	Contem informação relativa ao *browser* utilizado.

A soma destas duas diferenças permitiu assim ao autor elevar o seu grau de certeza na identificação do *browser* usado pelo utilizador. Esta técnica por si só é bastante poderosa possuindo apenas um problema para uma utilização mais generalizada. Para ser possível efectuar este tipo de análise é necessário que o servidor guarde para ficheiro sempre que este receba mensagens com a indicação do método GET para novas ligações. Desta forma, para um servidor com um número significativo de acessos diários, a quantidade de informação a ser processada pode se tornar incomportável.

Internet Explorer

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 192.168.7.60
Connection: Keep-Alive
```

Mozilla Firefox

```
GET / HTTP/1.1
Host: 192.168.7.60
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.6)
Gecko/20040113
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,im
age/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Figura 3.1: Diferenças no uso do método GET entre Internet Explorer e Mozilla Firefox.

3.1.1.2 Análise pelo método POST

Pela mesma razão já invocada anteriormente aquando da análise do método GET, o método POST irá também ter uma construção semântica do seu conteúdo e a ordem dos seus campos em conformidade com o *browser* utilizado na comunicação [10]. Apesar do

método POST normalmente não ser utilizado pelos *browsers*, visto a sua principal função é receber dados de um servidor, não impossibilita que não possam ser geradas mensagens com este método. É assim possível através da utilização do JavaScript obrigar o *browser* a invocar o método POST para enviar ao servidor e de forma a ser possível posteriormente analisar as diferenças e conseguir uma correcta identificação do *browser*.

Como se pode verificar na figura 3.2 mais uma vez é possível detectar diferenças, quer na forma quer no conteúdo, sendo então proveitosa a derivação da técnica acima descrita para uma correcta identificação do *browser*.

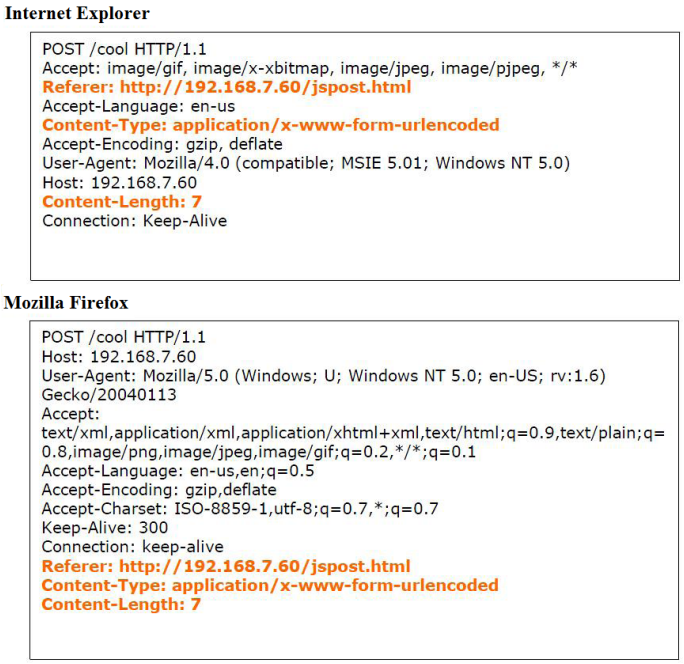

Figura 3.2: Diferenças no uso do método POST entre Internet Explorer e Mozilla Firefox.

O aliar destas duas técnicas permite assim elevar a certeza de uma correcta identificação do *browser* usado pelo utilizador. Estas técnicas por si só são bastante poderosas possuindo apenas dois problemas para uma utilização mais generalizada. Para ser possível efectuar este tipo de análise é necessário que o servidor guarde para um ficheiro toda a informação relativa à utilização dos métodos GET e POST para novas ligações. Desta forma, para um servidor com um razoável número de acessos diários, este ficheiro pode atingir facilmente valores incomportáveis.

Surge também um segundo problema que se deve à possibilidade, das aplicações que utilizam o protocolo HTTP, de poder livremente configurar os campos da mensagem que será enviada através do protocolo. Deste modo, uma análise aos tipos de campos que são registados por um *browser* podem conduzir ao error porque podem existir outras aplicações, como por exemplo *plugins* para *browser*, que transmitem através do *browser* mas com campos e valores nos campos criados/alterados por essas aplicações.

3.1.2 *Document Object Model*

O *Document Object Model* (DOM) é uma plataforma, desenvolvida pela W3C, como um conjunto de objectos que define a estrutura dos documentos XML ou HTML e a forma como estes podem ser acedidos e alterados [11]. Este tipo de estrutura das páginas Web possibilitou o aparecimento de um novo conceito, o DHTML, ou seja, o conceito de páginas Web dinâmicas. Para esse efeito é geralmente usada uma linguagem de programação, como o JavaScript ou o VBScript, para poder aceder aos vários métodos que podem ser executados sobre os objectos e as suas propriedades.

Particularmente, o JavaScript é uma linguagem de programação que pode ser executada totalmente do lado do cliente, *browser* Web, e possui a particularidade de não ser necessário uma compilação do código que vai ser executado numa página Web, sendo este antes executado directamente pelo motor do *browser* aquando do processamento do documento ou quando alguns eventos são registados. Desta forma é possível através do uso de JavaScript recolher todo o tipo de informação sobre o documento ou executar conteúdos com a ocorrência de eventos diversos como um clique rato ou o passar por cima de alguma parte do documento. A importância do DOM na identificação de *browsers* prende-se no facto deste possuir um conjunto objectos que possuem propriedades praticamente únicas de cada motor de *browser*.

O DOM, relativamente aos objectos que podem transmitir informação sobre o *browser*, é constituído pelos seguintes objectos [12]:

- **History:** Contém informação sobre o histórico de navegação do *browser*.
- **Location:** Contém informação sobre a página.
- **Navigator:** Contém informação sobre o *browser*.
- **Screen:** Contém informação sobre o ecrã do utilizador.
- **Window:** Contém informação sobre a janela no *browser*.

Da mesma forma que na análise dos cabeçalhos HTTP, existe uma propriedade que recolhe o conteúdo do *user-agent* (*Navigator.userAgent*) declarado pelo *browser*. Pelas mesmas razões que foram evidenciadas na análise dos cabeçalhos HTTP, esta propriedade não deverá ser considerada para uma correcta identificação do *browser*, mas antes como mais um método de validação da informação que o *browser* transmite. Devido aos diferentes motores que os *browsers* utilizam, estes originam diferentes comportamentos relativamente à forma de como implementam e interagem com os objectos. Como tal torna-se pertinente a comparação das propriedades e métodos de todos os objectos DOM, e não apenas dos objectos que possuem informação relativa ao *browser*, de forma a conseguir uma diferenciação dos *browsers* mais exacta.

Para além das diferenças na interpretação dos objectos, algumas empresas decidiram implementar objectos que só podem ser interpretados pelos seus *browsers*. Um exemplo disso é a propriedade *window.opera*. Esta propriedade só é reconhecida e processada pelo *browser* Opera e ,como tal, pode à partida ser usada para a determinação de um *browser* ou conjunto restrito de *browsers*.

3.1.3 HTTP *Persistent Connections*

Com a necessidade de assegurar uma conexão TCP de forma ininterrupta, foi desenvolvida a ideia do HTTP *Persistent Connections*. Este tipo de conexões foram criadas de forma evitar a abertura de uma nova ligação TCP sempre que fosse necessário o estabelecimento de uma comunicação entre o par cliente-servidor. Desta forma, este tipo de conexões vieram assegurar que a comunicação poderia ficar activa mesmo sem haver comunicação e posteriormente ser reutilizada pelo mesmo par sem haver a necessidade de estabelecer novamente uma nova ligação. Este tipo de ligações veio ajudar na redução do tráfego, especialmente nas comunicações SSL/TLS onde havia até então um exagerado número de pacotes que eram usados só para estabelecer constantemente novas ligações, degradando desta forma o desempenho do protocolo HTTP com uma desnecessária largura de banda ocupada.

Relativamente às conexões persistentes, podemos encontrar diferenças, entre *browsers*, de duas formas distintas. A primeira das diferenças deve-se ao facto de os *browsers* não declararem todos o mesmo número máximo de conexões persistentes. No caso do Internet Explorer, até à versão 8.0 (exclusive), ele só suportava duas conexões persistentes, no máximo. A segunda diferença surge através da diferença no tempo máximo que um cliente pode estar sem comunicar até a ligação ser destruída. O aliar destas duas diferenças conjuga mais um método para conseguir ajudar na detecção do *browser* utilizado [13].

3.2 Arquitectura do AuditService da AuditMark

O AuditService é o serviço que a AuditMark está a desenvolver e que tem como objectivo a análise e classificação qualitativa do tráfego Web. Será então feita uma breve explicação do serviço e as suas partes constituintes de forma a proporcionar um correcto entendimento das várias partes do processo de auditoria e de como estas se conjugam.

Devido às grandes quantidades de tráfego que é necessário auditar, o AuditService foi projectado como um sistema modular e distribuído. Este modelo de funcionamento torna possível distribuir as componentes do serviço de auditoria por várias máquinas e executar várias instâncias de cada componente simultaneamente, o que se traduz num sistema eficiente e flexível. Como é possível verificar na figura 3.3 o serviço AuditService

encontra-se dividido em três módulos distintos: O módulo de recolha de dados, o módulo Audit-PI e o módulo de processamento de dados.

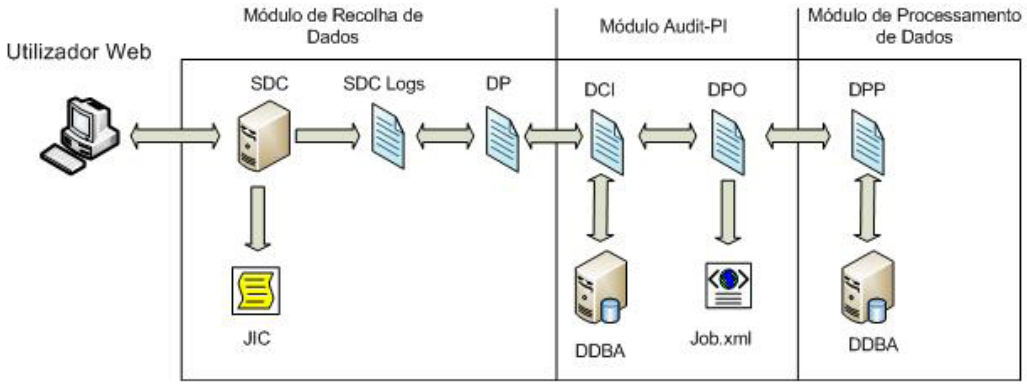

Figura 3.3: Arquitectura do AuditService da AuditMark.

3.2.1 Módulo de Recolha de Dados

Este módulo é constituído por todos os componentes destinados à recolha de dados. Fazem parte deste módulo o *JavaScript Interaction Code*, o *Server Data-Collector* e o *Data Packager*.

- ***JavaScript Interaction Code* (JIC):** O JIC é o componente responsável pela recolha dos dados Web que irão ser auditados pela AudtMark. Esta componente é constituída por vários ficheiros JavaScript que serão utilizados consecutivamente de forma a recolher a informação pretendida do utilizador.

  Para a sua execução é colocado um Web Bug[1] no servidor Web que está a ser auditado, de forma a que qualquer utilizador que visite esse servidor seja redireccionado para o servidor da Auditmark, que contém o primeiro ficheiro JavaScript, iniciando-se assim a primeira ronda de recolha de dados. Este primeiro ficheiro é responsável pela recolha da primeira parte dos dados do utilizador e da execução do próximo ficheiro JavaScript. Este processo é repetido até todos os ficheiros JavaScript terem sido executados e os dados recolhidos.

  Como o JavaScript é executado na máquina do cliente, a AuditMark teve necessidade de utilizar uma ferramenta de ofuscação de código JavaScript de forma a esconder o processo de recolha de informação do utilizador: o JScrambler.

- ***Server Data-Collector* (SDC):** O SDC tem a funcionalidade de recolher os dados transmitidos pelo JIC e transformá-los num ficheiro binário. É este ficheiro binário,

[1] Pequeno objecto que é colocado numa página Web e que pode ter como funcionalidade o despoletar de comunicações com servidores em domínios diferentes do da página Web onde está alojado. Geralmente os Web Bugs são criados a parir da inclusão de pequenas imagens no código HTML e pode ter como função garantir a monitorização da página a terceiros

o *log* SDC, que será posteriormente utilizado para a análise do tráfego que uma página Web.

- ***Data Packager* (DP):** Este componente, implementada em JAVA, tem a função de aglomerar e interpretar os *logs* SDC e transmiti-los através de uma ligação JAVA RMI para o próximo módulo do AuditService, o módulo Audit-PI.

3.2.2 Módulo Audit-PI

O Audit-PI é o módulo onde se encontram os componentes com a responsabilidade de acesso às bases de dados distribuídas, orquestração de processos, sistemas de filas, porgramador de eventos, entre outros. É também este módulo o responsável pela gestão do processamento de dados e pela gestão do fluxo de dados entre componentes do serviço. Desta forma podemos considerar o módulo Audit-PI como o módulo central do AuditService. O Audit-PI é constituído por vários componentes, mas aqui só serão retratados os componentes mais importantes para perceber o funcionamento do Audit-PI e a sua interacção com os restantes módulos.

- ***Data-Collection Interface* (DCI):** A função deste componente é receber os dados que o componente Data Packager, do módulo Recolha de Dados, transmite, inseri-los no sistema de base de dados distribuídas e notificar os restantes componentes do serviço AuditService da existência de dados novos. Para a inserção de dados no sistema de base de dados distribuídas são utilizadas stored procedures da componente Data-Access API.

- ***Data Processing Orchestrator* (DPO):** Esta é o componente responsável pela orquestração dos processos de processamento de dados. Sempre que o DCI notifica a existência de novos dados o DPO desencadeia o processamento dos dados a auditar, tendo como base ficheiros XML que indicam qual o tipo de tarefas, e ordem, que são realizadas.

- ***Distributed Database Abstraction Interface* (DDBA):** O DDBA tem como função isolar os vários componentes do serviço, do modelo de arquitectura do AuditService. Como tal é criado um PL/Proxy para as ligações ao sistema de base de dados distribuídas PostGreSQL. São também implementadas *stored procedures* para simplificar e unificar os acessos ao sistema de base de dados.

- ***Data Access API*:** Este componente é constituído pelas *stored procedures*. Como foi expresso anteriormente, as stored procedures têm como função simplificar o acesso ao sistema de base de dados. Esta simplificação deve-se à abstracção que as diversas ferramentas do serviço têm no acesso ao sistema de base de dados distribuídos, isto

é, a aplicação deixa de saber qual a localização física dos dados a serem lidos ou escritos.

3.2.3 Módulo de Processamento de Dados

- ***Data Processing Plugins* (DPP's):** Um DPP é uma tarefa, ou método, de processamento dos dados a ser auditados. Tal como foi escrito anteriormente, o DPO lê as tarefas, para o processamento dos dados a auditar, num ficheiro XML. Dessa forma, cada DPP é uma tarefa a ser realizada, sendo este o responsável pelo acesso aos dados a auditar, pelo processamento e também pelo posterior armazenamento dos resultados obtidos da tarefa.

3.3 Resumo

Algumas das técnicas de identificação de *browsers* descritas neste documento são técnicas bastante eficientes para conseguir uma resposta correcta quanto à identificação do *browser* utilizado. Tomando como exemplo a técnica da identificação do *browser* através da leitura da sua árvore de objectos DOM é possível garantir que o diferente comportamento dos *browsers* é a melhor forma de os identificar. Existem objectos DOM para quase todas as tecnologias que o *browser* utiliza. Desta forma, é sempre possível descobrir diferenças e refinar cada vez mais o grau de exactidão na identificação do *browser*. Ainda assim, apesar da eficiência na identificação, será impossível garantir que todos os cliques analisados sejam 100% válidos pois existirá sempre forma de enganar ou dificultar o trabalho de quem audita o tráfego. Um exemplo disso é a possibilidade de desactivação do JavaScript, que impossibilita a análise dos objectos DOM, ou de qualquer outra aplicação ou linguagem que possibilite a identificação do *browser* ou das suas características. De forma a concluir este capítulo será apresentada na tabela 3.3 as vantagens e desvantagens de cada técnica aqui descrita.

Tabela 3.3: Comparação das técnicas de identificação de *Browsers*

Técnica	Vantagens	Desvantagens
Método GET	Grande fiablilidade nos resultados obtidos.	Possui teoricamente um número ilimitado de diferentes campos que podem ser analisados.
Método POST	Grande fiablilidade nos resultados obtidos.	Possui teoricamente um número ilimitado de diferentes campos que podem ser analisados.
DOM	Rápido e eficiente na forma como se pode recolher a informação.	Se o JavaScript estiver desactivado este método não pode ser usado. É útil na identificação do motor de *browser*, mas perde eficácia quando procuramos apontar os resultados a um *browser* específico
Persistent Connections	Fácil verificação.	Pouco preciso. É possivel alterar facilmente o número máximo de conexões.

Capítulo 4

Técnicas de Escolhas de Atributos

Tal como descrito no capítulo anterior, para identificar correctamente qual o browser que foi usado pelo utilizador podemos utilizar diversas técnicas. Cada técnica só por si permite a recolha de um conjunto de atributos relativamente grande, onde cada um pode ter um número elevado de valores distintos. Acontece que muitos destes atributos, apesar de serem recolhidos, possuem um nível de informação que não pode ser usado na detecção de qual o browser que foi utilizado. Com vista a determinação da importância de cada um dos atributos faz sentido analisar e documentar quais as técnicas de escolha de atributos que já foram implementadas e condicioná-las ao tema do reconhecimento de browsers.

As técnicas de escolha de atributos surgem assim, neste contexto, como uma forma de evitar a análise dos atributos que contribuem pouco ou nada para uma correcta identificação de um browser. Desta forma procuramos o uso destes algoritmos de forma a conseguir aumentar o desempenho da ferramenta de reconhecimento de *browsers*, reduzindo, desta forma, a carga de processamento e memória computacional, pouco abdicando da precisão necessária para um correcto reconhecimento.

4.1 Ganho de Informação

O Ganho de Informação será a técnica usada neste trabalho e como tal será maior a profundidade dada a esta secção onde tentar-se-á descrever o seu algoritmo o mais pormenorizadamente possível.

O Ganho de Informação é um termo geralmente utilizado na área de *Machine Learning* mas que se traduz na divergência de Kullback-Leibler[14]. Utilizando duas distribuições de probabilidade diferentes, X e Z, o Ganho de Informação está definido como a quantidade de acréscimo de bits necessários a Z para que consiga utilizar X na construção das suas mensagens. Esta medida não pode ser considerada simétrica, já que o acréscimo de bits que X necessita para utilizar Z pode não ser igual ao que Z necessita para utilizar X.

Na generalidade dos casos, Z é uma distribuição de probabilidade igual a X condicionado a um dos seus resultados.

Como base do algoritmo de Ganho de Informação está o cálculo da entropia de uma distribuição de probabilidade. A entropia pode ser definida como a forma de medir o quão dispersa se encontra a distribuição de probabilidade. Esta pode ser determinada através da equação que se segue[15].

$$H(X) = -\sum_{i=1}^{n} P(X)_i \log_2 P(X)_i \tag{4.1}$$

Em que *H(X)* é a entropia da distribuição de probabilidade X, *P(X)* a probabilidade de acontecimento de cada um dos valores de X e n o número de valores distintos que X toma.

Assim, para valores de entropia baixos o seu significado traduz-se numa distribuição de probabilidade muito heterogénea, sendo formada muito provavelmente por um número pequeno de picos onde a probabilidade é elevada, sendo o resto da distribuição composta por picos de baixa relevância. Assim, torna-se evidente que quando a entropia é baixa irá traduzir-se numa maior facilidade de prever qual o valor que, neste caso, X irá tomar. Em contrapartida, quando a entropia é alta irá então traduzir-se numa distribuição muito uniforme onde todos os valores de X têm a mesma probabilidade de ocorrência, diminuindo assim a probabilidade de prever qual o valor que X irá tomar.

Tal como afirmado anteriormente, a segunda distribuição de probabilidade é uma particularização da primeira, isto é, a primeira distribuição de probabilidade condicionada a um dos seus valores. Assim:

$$H(Z) = H(X|Y) = -\sum_{i=1}^{n} P(Y)_i \sum_{j=1}^{m} P(X|Y)_j \log_2 P(X|Y)_j \tag{4.2}$$

Em que Y é um dos valores que X pode tomar.

O Ganho de Informação pode então ser definido de uma forma prática como a diferença entre entropias de duas distribuições.

$$IG(X|Y) = H(X) - H(Z) = -\sum_{i=1}^{n} P(X)_i \log_2 P(X)_i + \sum_{i=1}^{n} P(Y)_i \sum_{j=1}^{m} P(X|Y)_j \log_2 P(X|Y)_j \tag{4.3}$$

Onde *IG(X|Y)* corresponde ao Ganho de Informação que o valor Y possui para a determinação de X.

A técnica do Ganho de Informação surge assim, no âmbito do tema desta dissertação, como um algoritmo bastante viável na correcta determinação de quais os testes mais importantes para um correcto reconhecimento de browsers. De uma forma geral este algoritmo é bastante simples e como tal não se adequa para dados que se relacionam entre si, sendo então preferível a utilização de outros algoritmos mais poderosos mas consequentemente com uma carga de processamento maior.

4.2 Outras técnicas de Escolhas de Atributos

4.2.1 *Relief*

O algoritmo Relief foi demonstrado pela primeira vez por Kenji Kira e Larry Rendell onde desde logo se destacou pela sua rapidez na determinação dos atributos e, talvez o mais importante, pela sua boa imunidade a ruído nos dados a analisar. Este algoritmo é de certa forma um exemplo claro de algo diferente do que era comum implementar até à data onde na maioria dos casos eram aplicadas técnicas baseadas no Ganho de Informação ou então algoritmos complexos que analisavam um bloco de dados exaustivamente de forma a decidir quais os atributos relevantes [14].

Assim surge este algoritmo, que se baseia no principio de escolher quais os campos mais relevantes da seguinte forma: Este algoritmo aleatoriamente analisa um campo de um atributo e depois determina qual a distância mais próxima para esse atributo na classe que pretende analisar e posteriormente para qualquer outra classe, no contexto deste trabalho entenda-se classe como browser. Este processo é repetido um número de vezes definida pelo utilizador até este achar que o número de atributos seleccionados e precisão é adequado.

O objectivo deste algoritmo passa por determinar quais os atributos que são específicos de cada classe e seleccioná-los tendo em conta a sua unicidade. De forma a controlar o ruído que o algoritmo filtra, pode-se definir um limite mínimo de valores iguais que um atributo pode ter aquando da determinação das distâncias. Desta forma são ignorados os valores que se repetem muito poucas vezes, e que necessariamente não iriam ajudar na identificação dos melhores atributos. Em dados reais, a não aplicação deste filtro pode acarretar a deterioração da determinação dos melhores atributos pela inclusão de valores que podem ser forjados.

4.2.2 Selecção Baseada na Correlação dos Atributos

Esta técnica é a primeira aqui retratada que analisa blocos de atributos em vez de os tratar isoladamente. O princípio desta técnica passa por analisar blocos de dados onde determina quais os atributos mais importantes, tendo em conta não só os valores para

cada atributo, mas também a sua correlação com os outros atributos. Assumindo como exemplo a seguinte heurística: [16]

$$M = \frac{k\overline{rca}}{\sqrt{k + k(k-1)\overline{raa}}} \tag{4.4}$$

Onde M é a classificação, rca o factor de correlação entre as classes e os atributos e raa o factor de correlação entre atributos diferentes.

Analisando a heurística podemos verificar que atributos que se auto-correlacionarem muito irão aumentar consideravelmente o valor do denominador e consequentemente diminuir a classificação. Desta forma são retirados da análise todos os dados redundantes. Esta técnica por si só não permite imediatamente determinar quais os melhores atributos, mas é bastante útil na eliminação de atributos que não irão acrescentar qualquer tipo de informação para o reconhecimento dos browsers.

4.2.3 *Wrapper*

Esta técnica é a única aqui descrita que não selecciona os atributos num esquema tipo filtro onde é atribuída uma classificação a cada atributo e filtrados os que não atingem a meta estabelecida.

O princípio do *Wrapper* reside no seu algoritmo de indução. Este algoritmo é completamente transparente para quem o utiliza, funcionando como uma interface, e é responsável para determinação do subgrupo de atributos mais indicados para os dados a analisar. O algoritmo possui depois forma de validar os atributos que escolhe. Caso não sejam validados, é melhorada a escolha até serem encontrados os melhores atributos. Este facto torna o *Wrapper* como um algoritmo excelente para a determinação dos melhores atributos. Visto que, para cada bloco de dados é necessário percorrer o algoritmo de indução este naturalmente torna-se num processo lento e com uma carga de processamento enorme. Desta forma este algoritmo é inviável para largas quantidades de dados ou para quando é escolhida a rapidez sobre a precisão [17].

4.3 Resumo

De uma forma geral as técnicas aqui descritas abrangem as diferentes vertentes que podem surgir aquando da necessidade da selecção de atributos. Tal como foi descrito ao longo deste capítulo, estas técnicas tornam-se ferramentas importantes para evitar cargas desnecessárias no processamento nos dados reduzindo a análise aos atributos mais importantes de um bloco de dados.

Em resumo, o Ganho de Informação é uma ferramenta muito eficiente demonstrando rapidez na obtenção de resultados assim como efectua uma boa escolha de atributos. Esta escolha de atributo tem tendência a ficar pior consoante o número máximo de valores distintos que um atributo possui, pois não existe nativamente nenhuma forma de controlar o ruído que os blocos de dados, com dados reais, possui. Portanto, conclui-se que é necessário adoptar métodos complementares de forma a reduzir a influência do ruído introduzido pelo número excessivo de valores distintos a considerar.

Comparando com as outras ferramentas, o algoritmo do Ganho de Informação apresenta maior rapidez, pecando apenas na acertividade na escolha dos resultados. No extremo oposto encontra-se o algoritmo Wrapper que apresenta uma alta taxa de acertividade na determinação dos atributos, mas possui uma carga de processamento demasiado elevada para grandes quantidades de dados, como é o que se verifica neste trabalho dissertativo. É esse facto que condiciona a escolha do algoritmo de escolha de atributos, pois, devido às grandes quantidades de dados recolhidos e que será necessário analisar, a velocidade terá uma maior importância na escolha. Como tal comprova-se que o Ganho de Informação será, à partida, o mais adequado para este trabalho dissertativo.

Capítulo 5

Implementação

Neste capítulo será abordada a solução implementada para resolver o problema definido no capítulo 1. Para esse efeito foram desenvolvidos 3 *DPPs*. Um DPP (*Data Processing Plugin*), tal como referido no capítulo 3, é uma ferramenta que implementa uma tarefa de processamento dos dados a ser auditados. Ao principal DPP deste projecto foi atribuído o nome de *DPP Attributes Selector*, para seleccionar quais os atributos mais importantes para uma correcta detecção de *browsers*. Será também descrita a implementação efectuada ao nível da componente complementar, *DPP Browser Profiles Assembler*, que actualiza contadores de pares <atributo, valor> para cada um dos browsers, facilitando desta forma o cálculo do Ganho de Informação. De forma completar foi também implementada uma nova versão do *DPP Browser Recon*, que utilizando os campos seleccionadas pela ferramenta *DPP Attributes Selector*, atribui uma classificação ao clique. Irão também ser descritos quais os requisitos esperados da implementação assim como será ilustrada a arquitectura de alto nível da interacção entre os componentes implementados.

5.1 Requisitos

Após o levantamento do estado da arte sobre o Ganho de Informação, foi necessário estabelecer requisitos para assegurar o cumprimento e correcto funcionamento da ferramenta implementada. Como tal foram definidos dois grupos de requisitos:

- Requisitos do DPP Attributes Selector:
  - A ferramenta deverá ter o formato de um DPP e estar intregado no serviço de auditoria da AuditMark;
  - A ferramenta deverá aplicar correctamente o algoritmo do Ganho de Informação;
  - A ferramenta deverá automaticamente excluir campos que introduzem ruído no reconhecimento dos Browsers pelo DPP Browser Recon;

- Requisitos do DPP Browser Recon
  - A ferramenta deverá ter o formato de um DPP e estar intregado no serviço de auditoria da AuditMark;
  - A ferramenta deverá determinar correctamente qual o browser utilizado atribuindo para esse efeito uma classificação ao perfil de browser que melhor se assemelha com o browser utilizado;
  - A ferramenta deverá atribuir uma classificação ao perfil indicado pelo User-Agent do clique;
  - A ferramenta deverá determinar quais os cliques onde existe possibilidade de ter ocorrido Click Fraud;
  - A ferramenta deverá usar a informação determinada pelo DPP Attributes Selector para a reconhecimento dos browsers;
  - A ferramenta deverá ter uma arquitectura distribuída de forma a reduzir tempos de processamento;

5.2 Arquitectura da Implementação

Nesta secção será explicada a arquitectura de alto nível das ferramentas implementadas assim como o nível de interacção entre elas ao nível do fluxo de informação. De notar que as interacções entre os componentes não ocorrem directamente, isto é, os dados produzidos por cada ferramenta implementada é guardado numa base de dados e não transmitida directamente entre os componentes. Esta decisão prende-se essencialmente ao facto de não haver necessidade de recalcular constantemente os dados sempre que for feita uma auditoria à qualidade dos cliques no que diz respeito à informação transmitida pelo browser. Desta forma os componentes, DPP Browser Profiles Assembler e DPP Attributes Selector têm um ciclo definido para fazer a regeneração de dados.

Observando a figura 5.1, podemos ver a interacção entre cada um dos componentes. O DPP Browser Profiles Assembler está assim responsável pela análise dos dados recolhidos pelo serviço de auditoria e actualização da estatística dos mesmos. Será então essa estatística disponibilizada às outras ferramentas.

Em relação ao DPP Attributes Selector este fica responsável por determinar quais os melhores testes que podem ser utilizados partindo dos dados gerados pelo DPP Browser Profiles Assembler. Para finalizar o DPP Browser Recon utiliza os melhores testes seleccionados pelo DPP Attributes Selector conjuntamente com os dados estatísticos para cada perfil de browser criado pelo DPP Browser Profiles Assembler para poder obter a qualidade dos cliques.

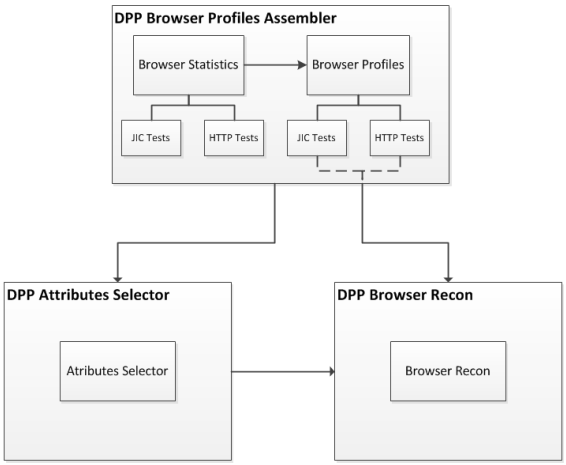

Figura 5.1: Arquitectura de Alto Nível da implementação.

5.3 Elementos da Implementação

Para a implementação da solução aqui descrita foi escolhida como linguagem de programação o JAVA. Tal como descrito no capítulo 3, a base de dados que foi utilizada é do tipo PostgreSQL. Esse facto justifica-se com o requisito da implementação que especifica que a solução deve ser construída possuindo o formato de um DPP. Desta forma garantimos à partida a integração da solução nas ferramentas do serviço de auditoria da AuditMark. De notar que aquando da realização deste trabalho dissertativo, a AuditMark procedeu à completa reestruturação do seu modelo de dados da base de dados do serviço de auditoria, originando assim a necessidade de re-implementação de alguns DPP que já existiam anteriormente. Neste caso, os DPPs que sofreram esta re-implementação foram o DPP Browser Profiles Assembler e o DPP Browser Recon. Seguidamente serão apresentados de uma forma detalhada todos os elementos implementados e explicadas as suas funções.

5.3.1 *DPP Browser Profiles Assembler*

O DPP Browser Profiles Assembler surge da necessidade de construir uma estatística sobre os dados recebidos em cada clique e depois associar essa estatística ao browser por ela indicada no campo *user-agent*. Esta necessidade não surge apenas para os elementos deste trabalho mas pode ser antes utilizada transversalmente por todos os DPPs já implementados ou a implementar. Como tal esta ferramenta foi desenvolvida em parceria com outros colegas da AuditMark que auxiliaram na definição e implementação da mesma. O DPP é então uma forma básica de organizar a estatística dos dados e foi desenvolvido dividindo o problema em duas partes:

- **Browser Statistics** - A parte Browser Statistics é a parte responsável pela construção da estatística dos dados recolhidos. A ferramenta fica assim dotada da

responsabilidade de efectuar a leitura de um conjunto de cliques ao qual posteriormente incrementa contadores por cada campo já visualizado colocando já os dados seccionados por *user-agent*. De uma forma básica esta parte da ferramenta é apenas responsável pela actualização de contadores que indicam o número de vezes que cada par <atributo,valor> foi visto por *user-agent*. Como o número de valores distintos do atributo *user-agent* é muito grande definiu-se à partida um conjunto de expressões regulares de forma a tentar agrupar browsers com *user-agents* com ligeiras diferenças mas que na realidade correspondem à mesma versão de browser.

- **Browser Profiles** - A parte Browser Profiles é a parte responsável pela aglomeração dos dados gerados pelo Browser Statistics. Os dados são recolhidos para cada browser sendo depois construída a estatística dos atributos e campos para cada browser. A estatística é apresentada segundo duas métricas. A primeira métrica é a percentagem local, isto é, a percentagem que cada par <atributo, valor> possui num determinado browser. A segunda métrica é a percentagem global que exprime a percentagem que cada par <atributo, valor> possui na globalidade dos dados.

5.3.2 *DPP Browser Attributes Selector*

Tal como explicado na introdução desta secção, esta ferramenta tem como função aplicar o algoritmo do Ganho de Informação de forma a seleccionar e quantificar quais os melhores atributos. Para proceder a esse feito foi então necessário desenvolver a ferramenta de forma a que pudesse ter acesso a todos os dados o mais rapidamente possível.

Tal como descrito no Capítulo 3, a arquitectura do sistema de base de dados encontra-se distribuída sendo a comunicação e gestão feita através de um PL/Proxy. No decorrer da experiência da própria AuditMark, um número elevado de acessos geralmente diminuía muito o desempenho do *PL/Proxy* e consequentemente procurou-se então reduzir o número de acessos à base de dados procurando desta forma fazer o tratamento dos dados em JAVA. Também no decorrer dessa experiência profissional determinou-se que é mais rápido a implementação de algoritmos sobre JAVA do que sobre a linguagem PLpgSQL que é a utilizada para a implementação de *stored procedures* na base de dados. Como contrapartida o JAVA não está desenhado para que se possa aceder de forma rápida a uma estrutura de dados. Outro facto importante prende-se com o facto de o JAVA não conseguir armazenar grandes quantidades de informação e como tal procurou-se encontrar um equilíbrio na implementação deste trabalho.

Tal como descrito no capítulo 4, a equação que representa o Ganho de Informação é a seguinte:

$$IG(X|Y) = H(X) - H(X|Y) \qquad (5.1)$$

$$IG(X|Y) = -\sum_{i=1}^{n} P(X)_i \log_2 P(X)_i + \sum_{i=1}^{n} P(X)_i \sum_{j=1}^{m} P(X|Y)_j \log_2 P(X|Y)_j \quad (5.2)$$

A implementação da solução passou sobretudo pelo cálculo da entropia de cada atributo assim como a mesma entropia, só que condicionada a cada um dos browsers. Para esse efeito foi então utilizado a estatística calculada pelo DPP Browser Profiles Assembler. Como tal foram desenvolvidas *stored procedures* que carregam para o JAVA a informação necessária para o cálculo do Ganho de Informação para cada um dos atributos, ficando então toda a parte de algoritmia situada ao nível do JAVA.

Com o desenrolar dos testes, comprovou-se a informação referida no capítulo 4 que refere que o desvio do Ganho de Informação aumenta em conformidade que o número de valores distintos para um único teste. Como tal esta ferramenta foi dotada da possibilidade de se poder definir um limite máximo ao número de valores distintos para um teste. Este facto será demonstrado no próximo capítulo.

5.3.3 *DPP Browser Recon*

O DPP Browser Recon será a ferramenta que nos permitirá validar os resultados obtidos pelo algoritmo do Ganho de Informação. Tal como afirmado anteriormente, este DPP já se encontrava implementado no modelo de dados antigo mas que, devido à reformulação para o modelo de dados novo, ainda não se encontrava adaptado. Como tal foi necessário reimplementar esta ferramenta adaptando-a já de mecanismos que utilizam a informação gerada pelo *DPP Attributes Selector*, melhorando o seu desempenho adicionalmente.

O funcionamento desta ferramenta é bastante simples. Tomando por exemplo um clique. Primeiro vamos à base de dados e escolhemos um perfil de browser. De seguida obtemos qual o valor do primeiro teste e consultamos a base de dados para recolher qual a percentagem desse valor para aquele teste condicionado aquele perfil. É também recolhida o valor da maior percentagem para aquele teste. Essas duas percentagens vão servir então para a fórmula de cálculo do browser em questão. A equação usada para esse efeito é a seguinte:

$$S = \frac{valuePercentage}{maxPercentage} \frac{IG}{T} \quad (5.3)$$

Onde S é a classificação atribuída ao teste, valuePercentage a percentagem do valor no par <teste, perfil>, maxPercentage a máxima percentagem registada no par <teste, perfil> e IG/T o peso que o teste tem atribuído pelo Ganho de Informação.

O processo é então repetido para todos os testes até que todos sejam analisados. As classificações de cada teste são somadas e é assim obtida classificação para este perfil. O

processo é então repetido para todos os perfis. No final iremos então colocar na base de dados qual o perfil que melhor pontuação obteve conjuntamente com o perfil de *browser* que o campo *user-agent* afirmava ser. É posteriormente ao compararmos estas duas classificações que podemos afirmar se de facto houve cliques fraudulentos ou não.

Tal como podemos ver no exemplo acima descrito para cada clique são necessárias milhares de cálculos de classificações para cada teste. Este facto torna esta ferramenta muito demorada, o que nos leva ao próximo requisito desta aplicação. Procurou-se então desenvolver esta aplicação de forma distribuída seguindo uma arquitectura cliente-servidor. Para o efeito foi usado a ferramenta imqbroker. Esta ferramenta incorporada no JAVA possibilita a criação de filas que permitem ser associadas a produtores ou consumidores de mensagens, criando assim o interface de comunicação entre o cliente e o servidor.

No que diz respeito às funções de cada um dos actores, o servidor fica então responsável pela segmentação dos cliques que vão ser analisados e posterior indicação aos clientes de quais os cliques que estes têm que analisar. Um cliente fica então responsável pela execução da algoritmia que permite o reconhecimento do browser e consequente inserção na base de dados dos dados que conclui.

5.4 Conclusão

As ferramentas desenvolvidas são inteiramente capazes de desempenhar as funções para o que foram criadas, ficando cumpridos todos os requisitos previamente estabelecidos. Em particular, a ferramenta *DPP Browser Profiles Assembler* permite uma organização dos dados e a determinação de dados estatísticos imensamente úteis para as ferramentas de análise dos dados do serviço de auditoria da AuditMark. A ferramenta *DPP Attributes Selector* possibilita a selecção de quais os testes mais relevantes para um correcto reconhecimento dos browsers, possibilitando assim uma redução na carga de processamento. Relativamente à ferramenta *DPP Browser Recon*, esta possibilita o correcto reconhecimento dos browsers estando distribuída segundo uma arquitectura cliente-servidor, permitindo desta forma a distribuição do processamento por várias máquinas possibilitando a redução da carga de processamento em cada máquina, assim como a redução do tempo de operação.

Capítulo 6

Testes e Validação

Este capítulo irá descrever e apresentar os testes realizados sobre as ferramentas implementas de forma a procurar uma validação dos resultados obtidos. Será validada a ferramenta DPP Browser Recon que irá servir como ferramenta de validação da ferramenta DPP Attributes Selector.

Para este efeito foram feitos testes ao desempenho, definido como o tempo necessário para fazer uma identificação do browser, e à qualidade dos resultados do DPP Browser Recon com diferentes configurações do DPP Attributes Selector. Relativamente ao DPP Attributes Selector foram ainda aplicados diferentes limites máximos ao número de atributos distintos de forma a analisar a influência desses testes nos resultados finais. Para finalizar foram efectuados testes sobre um bloco de dados reais e analisaremos como a ferramenta se comporta.

6.1 Metodologia de Teste

De modo a proporcionar uma base de testes onde podessemos analisar os primeiros resultados foram gerados 12 cliques, dos quais 8 deles são verdadeiros e 4 são forjados. Observando a tabela 6.1, podemos identificar a lista dos browsers utilizados nos testes realizados. Dos testes verdadeiros foram escolhidos os browsers mais significativos do mercado, incluindo diferentes versões para perceber como as ferramentas se comportam em relação a isso.

Relativamente aos testes falsos, 6.2, estes são testes onde foram efectuadas adulterações do user-agent como forma de falsificação do clique, assim como misturadas algumas propriedades dos browsers.A título de curiosidade foi alterado um user-agent de um Internet Explorer com a versão 7.0 para um com a versão 6.0. Esperamos desta forma testar como a ferramenta de identificação de browsers se comporta em relação a browsers da mesma família mas com versões diferentes.

ClickID	Browser
1	Internet Explorer 8.0
2	Firefox 3.6
3	Safari 4.0
4	Internet Explorer 7.0
5	Internet Explorer 6.0
6	Chrome 5.0
7	Safari 3.0
8	Opera 9.8

Tabela 6.1: Lista dos Browsers Verdadeiros.

6.2 Testes

6.2.1 Teste com todos os atributos activos

Neste teste iremos tentar determinar quais os browsers que foram utilizados nos cliques produzidos. Especificamente a este teste, não haverá qualquer tipo de restrições sobre os testes a utilizar, exceptuando apenas os testes que determinam o *user-agent*. Assim foram previamente utilizadas as ferramentas DPP Browser Profiles Assembler e DPP Browser Attributes Selector de forma a produzir os dados que servirão para a análise que irá ser produzida. Posteriormente, aplicando a ferramenta DPP Browser Recon foram obtidos os seguintes resultados expressos nas tabelas 6.3 e 6.4. As tabelas agora referidas possuem colunas que se encontram definidas da seguinte forma:

- ClickID - Esta coluna identifica qual o número que identifica o clique efectuado;
- UABrowser - Esta coluna identifica qual o browser que o clique indica que é. Este valor é obtido através da leitura do campo *user-agent*;
- UAClickQuality - Nesta coluna ficam representados qual a classificação que o *DPP Browser Reco* atrabui para o browser indicado no campo *user-agent*;
- BestBrowser - Esta coluna identifica qual o browser que possui melhor a melhor classificação atribuída pelo *DPP Browser Reco*;
- BestClickQuality - Nesta coluna fica representada a melhor classificação obtida por clique;

Como se pode observar os resultados apresentados pelo *DPP Browser Recon* são satisfatórios, em relação à determinação de qual o browser utilizado na execução do clique. Para os testes verdadeiros (tabela 6.3) as classificações atribuidas são um pouco baixas. Tal como afirmado anteriormente isto deve-se essencialmente ao facto de os resultados do Ganho de Informação perderem relevância em conformidade com o número de valores

ClickID	Browser que afirma ser	Browser Verdadeiro
9	Chrome 5.0	Safari3.1
10	Firefox 3.6	Internet Explorer 6.0
11	Inetrnet Explorer 8.0	Firefox 3.6
12	Internet Explorer 7.0	Internet Explorer 6.0

Tabela 6.2: Lista dos Browsers Falsos.

distintos de cada um dos atributos, isto é, quanto maior for o número de valores distintos que cada campo pode apresentar, maior a possibilidade de estes campos não ajudarem na determinação do browser. Observando a figura 6.1 podemos obervar no eixo das ordenadas os valores de Ganho de Informação e no eixo das abcissas o número de valores distintos. De notar que o eixo das abcissas se encontra escalado logaritmicamente.

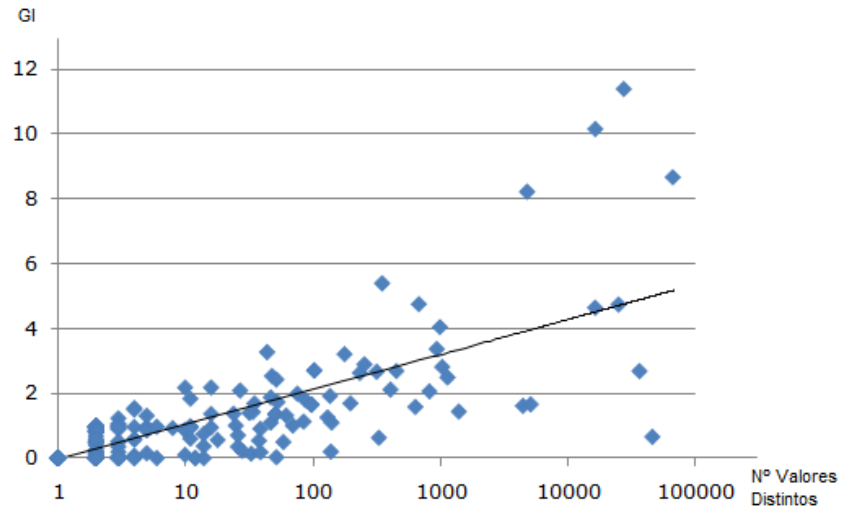

Figura 6.1: Relação Ganho de Informação vs. Número de atributos distintos.

Assim comprova-se que o valor que o Ganho de Informação pode tomar está directamente condicionado pelo número de valores distintos que um campo pode apresentar.

Na tabela 6.5 podemos observar quais os testes que obtiveram o maior número de valores distintos. Analisando mais ao pormenor cada um desses testes podemos verificar que eles à partida não acrescentam qualquer tipo de informação relativa ao reconhecimento de browsers. A título de exemplo, o teste "window.innerWidth" que retorna uma das dimensões da janela que o utilizador está a utilizar. De facto se o utilizador mudar um pouco a dimensão da janela já se irá obter um resultado totalmente diferente do primeiro, de onde se conclui que este teste não pode ser utilizado na identificação do browser. Assim, isto vem comprovar a necessidade de se estabelecer um limite máximo ao número de valores distintos que um teste pode tomar, desactivando os testes em que isso acontece.

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
1	Internet Explorer 8.0	82,1%	Internet Explorer 8.0	82,1%
2	Firefox 3.6	86,8%	Firefox 3.0	87,1%
3	Safari 4.0	83,6%	Safari 4.0	83,6%
4	Internet Explorer 7.0	81,4%	Internet Explorer 7.0	81,3%
5	Internet Explorer 6.0	85,7%	Internet Explorer 6.0	85,7%
6	Chrome 5.0	85,2%	Chrome 4.1	85,7%
7	Safari 3.0	85,8%	Safari 3.0	85,8%
8	Opera 9.8	88,2%	Opera 9.8	88,2%

Tabela 6.3: Resultados utilizando todos os testes (Testes Verdadeiros).

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
9	Chrome 5.0	52,3%	Safari 3.1	99,4%
10	Firefox 3.6	59,1%	Internet Explorer 6.0	83,4%
11	Internet Explorer 8.0	65,7%	Firefox 3.6	85,3%
12	Internet Explorer 7.0	83,4%	Internet Explorer 6.0	84,0%

Tabela 6.4: Resultados utilizando todos os testes (Testes Falsos).

Teste	Tipo Teste	Nº Valores Distintos	Ganho de Informação
document.referrer	JSTEST	66718	8,69
referer	HTTP	46184	0,66
user-agent	HTTP	36576	2,69
document.location.href	JSTEST	27542	11,42
navigator.plugins	JSTEST	25060	4,75
navigator.userAgent	JSTEST	16480	10,18
document.baseURI	JSTEST	16423	4,65
via	HTTP	5132	1,66
X-forwarded-for	HTTP	4466	1,62
Accept-language	HTTP	1401	1,44
window.innerWidth	JSTEST	1142	2,50
window.outerWidth	JSTEST	1038	2,82
window.innerHeight	JSTEST	996	4,052
X-bluecoat-via	HTTP	639	1,58
x-nai-id	HTTP	331	0,63
x-proxy-id	HTTP	318	2,67
x-fcckv2	HTTP	141	1,097

Tabela 6.5: Lista dos testes com maior o número valores distintos.

6.2.2 Testes com limite máximo de valores distintos

Como segundo teste iremos então establecer estabelecer dois limites máximos ao número de valores distintos, onde serão excluídos os testes que não cumpram os limites, e analisar os resultados que o DPP Browser Recon apresenta. Assim forma estabelecidos os limites de 100 e 50 valores distintos por teste para os cliques produzidos que se podem observar nas seguintes tabelas(6.6, 6.7, 6.8, 6.9).

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
1	Internet Explorer 8.0	93,9%	Internet Explorer 8.0	93,9%
2	Firefox 3.6	99,8%	Firefox 3.6	99,8%
3	Safari 4.0	99,8%	Safari 4.0	99,8%
4	Internet Explorer 7.0	99,9%	Internet Explorer 8.0	99,9%
5	Internet Explorer 6.0	99,8%	Internet Explorer 8.0	99,8%
6	Chrome 5.0	99,8%	Chrome 4.1	99,8%
7	Safari 3.0	100%	Chrome 4.1	100%
8	Opera 9.8	99,8%	Opera Mini 5.0	99,8%

Tabela 6.6: Resultados utilizando um limite de 100 valores distintos (Testes Verdadeiros).

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
9	Chrome 5.0	53,0%	Safari 3.1	100%
10	Firefox 3.6	80,1%	Internet Explorer6.0	97,4%
11	Internet Explorer 8.0	67,7%	Firefox 3.6	97,7%
12	Internet Explorer 7.0	97,8%	Internet Explorer6.0	98,8%

Tabela 6.7: Resultados utilizando um limite de 100 valores distintos (Testes Falsos).

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
1	Internet Explorer 8.0	96,2%	Internet Explorer 8.0	96,2%
2	Firefox 3.6	99,8%	Firefox 3.6	99,8%
3	Safari 4.0	99,8%	Safari 4.0	99,8%
4	Internet Explorer 7.0	99,9%	Internet Explorer 8.0	99,9%
5	Internet Explorer 6.0	99,8%	Internet Explorer 8.0	99,8%
6	Chrome 5.0	99,8%	Chrome 4.1	99,8%
7	Safari 3.0	100%	Chrome 4.1	100%
8	Opera 9.8	99,8%	Opera Mini 5.0	99,8%

Tabela 6.8: Resultados utilizando um limite de 50 valores distintos (Testes Verdadeiros).

Comparando as tabelas 6.3, 6.6 e 6.8 verifica-se que, com a introdução de este tipo de limites,houve um aumento das classificações atribuídas a um clique verdadeiro. Comparando as classificações atribuídas pela ferramenta (UA ClickQaulity, BestClickQuality) podemos ver que as diferenças que podem existir entre elas são muito diminutas, onde se conclui que a ferramenta detecta bem qual o browser que foi utilizado. As diferenças que

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
9	Chrome 5.0	53,0%	Safari 3.1	100%
10	Firefox 3.6	75,1%	Internet Explorer6.0	96,8%
11	Internet Explorer 8.0	67,3%	Firefox 3.6	97,4%
12	Internet Explorer 7.0	99,8%	Internet Explorer6.0	100%

Tabela 6.9: Resultados utilizando um limite de 50 valores distintos (Testes Falsos).

podem surgir na classificação devem-se a pequenos erros na detecção da versão correcta de um browser. De notar que a redução do limite máximo de 100 para 50 valores distintos não acarretou grandes diferenças nos valores obtidos.

Relativamente aos cliques forjados (tabelas 6.4, 6.7 e 6.9) verifica-se que a ferramenta identifica correctamente que o clique é falso, visto a diferença entre a melhor classificação(Best Click Quality) e a classificação atribuída ao browser que indica ser (UA Click Quality) ser grande. Tal como afirmado anteriormente a ferramenta tem dificuldades a determinar diferentes versões de browser pelo que os cliques 12 não obtiveram resultados satisfatórios, isto é, não foi determinado correctamente qual o browser utilizado no clique.

Conclui-se então que este tipo de limite é necessário para uma correcta identificação de qual o browser utilizado. Conclui-se também que os testes com um número de valores distintos muito elevados introduzem ruído na determinação do browser. Intende-se por ruído que retira certeza na determinação do browser, isto é, algo que, não introduzindo qualquer tipo de informação ao cálculo, reduz o desempenho do mesmo.

6.2.3 Testes com limite mínimo de Ganho de Informação

Nesta parte adaptar-se a ferramenta DPP Attributes Selector para só seleccionar os atributos com Ganho de Informação superior a um dado patamar. Este teste tem como propósito a análise de como a ferramenta DPP Browser Recon se irá comportar com a redução dos testes que não introduzem informação para a determinação de qual o browser utilizado. Assim irão ser introduzido 3 diferentes limites onde iremos analisar os resultados obtidos.

Relativamente aos cliques verdadeiros, analisando as tabelas 6.10 e 6.12 verifica-se uma convergência das classificações obtidas da primeira para a segunda tabela. Apesar de, na generalidade dos casos, o browser indicado pelo user-agent não ser o melhor classificado, a diferença entre as duas classificações é próxima de 0 o que indica um grau de semelhança muito próximo. Vendo a tabela 6.14 os resultados divergem e existem alguns casos onde não se consegue detectar correctamente qual o browser utilizado. Para além deste facto, o clique 1 desceu muito a sua classificação em compração aos resultados alteriores. Portanto, conclui-se neste caso que não se deverá impor limites mínimos ao

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
1	Internet Explorer 8.0	92,3%	Internet Explorer 8.0	92,3%
2	Firefox 3.6	99,6%	Firefox 3.6	99,6%
3	Safari 4.0	99,5%	Safari 4.0	99,5%
4	Internet Explorer 7.0	99,8%	Internet Explorer 8.0	99,8%
5	Internet Explorer 6.0	99,7%	Internet Explorer 8.0	100%
6	Chrome 5.0	99,6%	Chrome 4.1	99,6%
7	Safari 3.0	100%	Chrome 4.1	100%
8	Opera 9.8	99,5%	Opera Mini 5.0	99,5%

Tabela 6.10: Resultados utilizando um limite de 50 valores distintos e Ganho de Informação maior que 0,5 (Testes Verdadeiros).

ganho de Informação superior a 1, sob o risco de proceder a uma má escolha de atributos, ignorando assim testes que introduzem informação na identificação do browser.

Relativamente aos cliques forjados (tabelas 6.11, 6.13 e 6.15), continuamos a observar uma correcta determinação do browser, com uma diminuição geral das classificações atribuídas ao browser indicado pelo *user-agent* aumentando assim o grau de certeza na afirmação do clique ser falso. Analisando o clique 12, observamos que, novamente, não se conseguiu determinar a versão do browser correcto.

6.2.4 Testes de desempenho

Para efectuar os testes de desempenho, que se define como o tempo necessário para auditar um clique, foram efectuadas várias medições ao tempo de processamento de 10000 cliques pela ferramenta DPP Browser Recon com os diferentes limites antes descritos aplicados. Foram também medidos os tempos de processamento aquando da utilização simultânea da ferramenta por mais que um cliente. Essa informação pode ser consulta na tabela 6.16

Como se pode verificar, para a melhor solução obtida, isto é, limite superior estabelecido em 50 valores distintos e inferior com o ganho de informação superior a 0,75 obtemos um aumento do desempenho superior a 100

De forma a testar o desempenho da arquitectura distribuida foram primeiro colocados dois clientes a processarem 1000 cliques e posteriormente três clientes a processarem 1000 cliques, de forma a que conseguir comparar directamente com os resultados antes obtidos por um só cliente. Assim obtivemos os seguintes resultados:

Como se verifica, a distribuição da carga de processamento não obteve a melhoria prevista. Isso deve-se ao facto de, apesar de estar a trabalhar por cima de uma base de dados distribuída, no decorrer deste trabalho os dados encontravam-se todos na mesma máquina. Se aliarmos a isso os milhares de *queries* a base de dados que cada clique

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
9	Chrome 5.0	6,9%	Safari 3.1	100%
10	Firefox 3.6	55,3%	Internet Explorer6.0	100%
11	Internet Explorer 8.0	69,6%	Firefox 3.6	99,6%
12	Internet Explorer 7.0	99,5%	Internet Explorer 8.0	100%

Tabela 6.11: Resultados utilizando um limite de 50 valores distintos e Ganho de Informação maior que 0,5 (Testes Falsos).

necessita de fazer no contexto do DPP Browser Recon é normal que exista que a limitação se transfira para os acessos à base de dados.

6.3 Conclusões

Os testes que foram efectuados comprovaram a eficiência do algoritmo do Ganho de Informação para uma correcta determinação dos testes necessários para o reconhecimento de browsers. Testaram-se várias soluções possíveis de redução dos campos a utilizar, onde se verificou que o Ganho de Informação não consegue tratar o ruído dos dados a auditar. Para contornar esse efeito foi assim necessário implementar limites aos testes com um número de valores distintos muito elevado.

Relativamente aos testes dos limites inferiores chegou-se a conclusão que existe um conjunto de testes com valores próximos de 1 que ajudam no processo de reconhecimento e como tal devem ser mantidos em consideração.

Em termos de desempenho da ferramenta esta apresenta melhorias consideráveis relativamente a quando todos os testes são incluídos na análise assim como um melhor desempenho numa arquitectura distribuída. Os testes sobre a arquitectura distribuída, levam a testar futuras soluções para a distribuição de carga assim como futuras soluções para uma distribuição dos dados por várias máquinas.

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
1	Internet Explorer 8.0	92,2%	Internet Explorer 8.0	92,2%
2	Firefox 3.6	99,8%	Firefox 3.6	99,8%
3	Safari 4.0	99,4%	Safari 4.0	99,4%
4	Internet Explorer 7.0	99,7%	Internet Explorer 8.0	99,7%
5	Internet Explorer 6.0	99,7%	Internet Explorer 8.0	100%
6	Chrome 5.0	99,6%	Chrome 4.1	99,6%
7	Safari 3.0	100%	Chrome 4.1	100%
8	Opera 9.8	99,4%	Opera Mini 5.0	99,4%

Tabela 6.12: Resultados utilizando um limite de 50 valores distintos e Ganho de Informação maior que 0,75 (Testes Verdadeiros).

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
9	Chrome 5.0	0,8%	Safari 3.1	100%
10	Firefox 3.6	11,6%	Internet Explorer 8.0	100%
11	Internet Explorer 8.0	69,5%	Firefox 3.6	99,5%
12	Internet Explorer 7.0	99,4%4	Internet Explorer 8.0	100%

Tabela 6.13: Resultados utilizando um limite de 50 valores distintos e Ganho de Informação maior que 0,75 (Testes Falsos).

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
1	Internet Explorer 8.0	79,5%	Internet Explorer 8.0	79,5%
2	Firefox 3.6	90,5%	Firefox 3.6	90,4%
3	Safari 4.0	90,4%	Internet Explorer 8.0	100%
4	Internet Explorer 7.0	85,3%	Internet Explorer 8.0	95,0%
5	Internet Explorer 6.0	94,8%	Internet Explorer 8.0	100%
6	Chrome 5.0	93,3%	Chrome 4.1	100%
7	Safari 3.0	100%	Chrome 4.1	100%
8	Opera 9.8	90,7%	Opera Mini 5.0	100%

Tabela 6.14: Resultados utilizando um limite de 50 valores distintos e Ganho de Informação maior que 1 (Testes Verdadeiros).

ClickID	UABrowser	UAClickQuality	BestBrowser	BestClickQuality
9	Chrome 5.0	0,0%	Safari 3.1	100%
10	Firefox 3.6	2,9%	Internet Explorer 8.0	95,0%
11	Internet Explorer 8.0	62,0%	Firefox 3.6	100%
12	Internet Explorer 7.0	95,3%	Internet Explorer 8.0	100%

Tabela 6.15: Resultados utilizando um limite de 50 valores distintos e Ganho de Informação maior que 1 (Testes Falsos).

Teste	Tempo (segundos/clique)
Todos os testes	3,83
< 100 valores distintos	2,99
< 50 valores distintos	2,96
Ganho Informação > 0.5	1,94
Ganho Informação > 0.75	1,85
Ganho Informação > 0.1	1,18

Tabela 6.16: Tempos de processamento para cada um dos testes.

Clientes	Tempo (segundos/clique)
1	1,85
2	1,56
3	1,47

Tabela 6.17: Tempos de processamento para vários clientes.

Capítulo 7

Conclusões e Trabalhos Futuro

7.1 Conclusão

Este documento foi desenvolvido com o objectivo de estudar a problemática do reconhecimento de *browsers* e solucionar a necessidade do aumento de desempenho deste tipo de ferramentas. De forma a contextualizar o tema da dissertação foi desenvolvido um capítulo sobre publicidade *online* onde foram analisados os diferentes tipo de publicidade na Internet. Devido ao peso que a publicidade na Web possui, foram analisados ainda os modelos de negócio existentes onde se englobou os vários modelos de receitas e o volume de negócio que esta forma de publicidade atinge. Feita esta análise é assim possível compreender como os fenómenos de fraude na Web, como o *Click Fraud*, surgem. Assim foi apresentada um descrição desta problemática e assim como das técnicas mais usadas para o praticar. Feita esta contextualização do problema percebe-se então qual a importância das técnicas de detecção de *browsers* para a detecção do *Click Fraud*.

Depois desta análise concluída, procedeu-se à investigação das técnicas de detecção de *browsers* que já foram desenvolvidas. Consequentemente, foram descritas três técnicas que permitem identificar diferenças entre pedidos que um servidor Web recebe e, dessa forma, criar perfis para a identificação do *browser* utilizado. Complementarmente foi descrito duma forma simples a arquitectura do serviço de auditoria da AuditMark.

Após esta problemática ter ficado bem definida, estudou-se a importância e a necessidade de algoritmos de escolha de atributos aplicados em ferramentas de reconhecimento de *browser*. Discutiu-se sobre os vários temas dando especial destaque para o algoritmo de Ganho de Informação.

Depois do estado da arte ter sido totalmente levantado, passou-se para a implementação da solução. Relembrando os objectivos, este projecto tinha a função de proporcionar o desenvolvimento de uma ferramenta que, com a forma de um DPP e totalmente integrada no serviço de auditoria da AuditMark, conseguisse escolher correctamente quais

os melhores atributos para um correcto reconhecimento de *browsers*. O segundo objectivo deste trabalho era reimplementar uma solução já existente para a identificação de *browsers* para que suportasse o novo modelo de dados vigente na AuditMark. Para tal foram implementados três componentes, responsáveis, respectivamente, por uma análise estatística, pela determinação de quais os melhores atributos para um reconhecimento de *browsers* e , como esperados, pelo reconhecimento de *browsers*.

Vendo os resultados dos Testes e Validações, verifica-se que ambos os objectivos foram cumpridos onde se obtiveram resultados que comprovaram a importância de um algoritmo de escolha de atributos na dinâmica do reconhecimento de *browsers*, assim como permitiu perceber o comportamento do algoritmo de ganho de Informação para atribuitos com um conjunto de valores muito distintos. Assim é possível concluir este trabalho com bons resultados.

7.2 Trabalhos Futuros

Este trabalho pode continuar a ser desenvolvido em várias áreas distintas. Podemos procurar utilizar outros algoritmos de selecção de atributos, de forma a conseguir perceber qual o melhor algoritmo para a realidade da AuditMark, procurando obter assim cada vez melhores resultados. Podemos ainda tentar capacitar o algoritmo do Ganho de Informação de métodos que permitam minimizar o ruído atributos com muitos valores distintos, aumentando desta forma o grau de eficiência do mesmo. Para finalizar na área do Ganho de Informação podemos ainda testar novos limites e procurar definir novos limites que permitam refinar melhor ainda a escolha efectuada.

Relativamente à parte do reconhecimento de *browsers* podemos procurar novas técnicas para o reconhecimento de *browsers*. De uma forma complementar, devemos ainda estar atentos a pequenas particularidades que possam surgir na forma como cada browser se comporta.

7.3 Outras Aplicações

Relativamente a outras áreas de aplicação deste trabalho, o aperfeiçoamento deste tipo de algoritmos pode ser bastante útil para ramos como a Medicina, onde se pode aplicar este mesmo conceito para determinar quais as patologias que podem conduzir a uma doença ou na área da Ciência onde podemos procurar saber quais os factores que mais determinam um certo aparecimento de um fenómeno físico.

No campo do reconhecimento de browsers, podemos usar o mesmo para possibilitar servidores de disponibilizar páginas específicas para cada browser. Este facto torna-se ainda mais relevante se analisarmos a nova realidade dos browsers móveis, que possuem particulares no tipo de plataformas que suportam assim como no tamanho do ecrã disponível para a visualização da página.

Referências

[1] Robbin Zeff Brad Aronson. *Advertising on the Internet*. John Wiley and Sons, second edition edition, 1999.

[2] Allen Stern. What forms of online advertising are acceptable today? [sl:sn], 2008.

[3] Bruno Ferreira. A survey of click fraud techniques. Technical report, Instituto Superior de Engenharia do Porto, Julho 2008.

[4] PricewaterhouseCoopers. Iab internet advertising revenue report. 2010 First Half-Year Results, [sl].

[5] Click Forensics. Clickforensics. q4 2009 click fraud rate is down. or up. depends. Disponível em http://blog.clickforensics.com, acedido pela última vez a 29 de Janeirol de 2010.

[6] N. Kshetri. The economics of click fraud. *Security & Privacy*, pages 45 – 53, Maio - Junho 2010.

[7] Rui Polónia. User tracking through web sessions. Technical report, Faculdade de Engenharia da Universidade do Porto, Janeiro 2010.

[8] J. Mogul H. Frystyk L. Masinter P. Leach e T. Berners-Lee R. Fielding, J. Gettys. Http - rfc 2616. Disponível emhttp://www.w3.org/Protocols/, acedido pela última vez a 18 de Junho de 2010.

[9] David T. Carvalho. Utilização de reconhecimento de web browsers para detecção de cliques fraudulentos. Technical report, Faculdade de Engenharia da Universidade do Porto, Julho 2008.

[10] Shreeraj Shah. Browser identification for web applications. [sl:sn], 2005.

[11] Ray Whitmer e Lauren Wood Philippe Le Hégaret. Document object model. Disponível em http://www.w3.org/DOM/, acedido pela última vez a 18 de Junho de 2010.

[12] W3Schools. Javascript and html dom reference. Disponível em http://www.w3schools.com/jsref/default.asp, acedido pela última vez a 18 de Junho de 2010.

[13] P. Cao Z. Wang. Persistent connection behavior of popular browsers. Technical report, Department of Computer Sciences, University of Wisconsin, 1998.

[14] Kamber Han. *Data Mining - Concepts and Techniques*. Morgan Kaufmann, second edition edition.

[15] Robert Gray. Entropy and information theory. Technical report, Information Systems Laboratory, Electrical Engineering Department, Stanford University, 2009.

[16] Mark Hall. Correlation-based feature selection for machine learning. Technical report, University of Waikato, Abril 1999.

[17] Pederson Yang. A comparative study on feature selection in text categorization. [sl:sn].