MITIGATING INFORMATION MANIPULATION

A Thesis
Presented to
The Academic Faculty

by

Xinyu Xing

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
December 2015

MITIGATING INFORMATION MANIPULATION

Approved by:

Wenke Lee, Advisor
School of Computer Science
Georgia Institute of Technology

Nick Feamster
Department of Computer Science
Princeton University

Mustaque Ahamad
School of Computer Science
Georgia Institute of Technology

Hongyuan Zha
School of Computer Science
Georgia Institute of Technology

Michael Bailey
Department of Electrical and
Computer Engineering
University of Illinois at Urbana-Champaign

Date Approved: 21 August 2015

Dedicated to my lord

Without you, I cannot reach this stage.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Wenke Lee, and my co-advisor Nick Feamster, for their understandings, support, and encouragement throughout my graduate studies in the program of information security. I have been amazingly fortunate to have them as my advisors. Without their constructive criticism, rapport and limitless capacity for knowledge, such an accomplishment would have been impossible for me to achieve.

I am also deeply grateful to Dr. Roberto Perdisci for his insights and more importantly, the opportunities to work with him, from which I learned tremendously. His encouragements and belief in students warmed my heart even when I was disheartened. To my colleagues and friends, Wei Meng, Byoungyoung Lee, Yunhui Zheng, Anmol Sheth and Udi Weinsberg, thank you all for your support, kindness and comforting throughout the ups and downs of my graduate studies. To my family back in China, I thank you for your unconditional love and support. To my dear wife, Sui Huang, I could not thank you enough for your support, patience and encouragements. Having you in my life is God's biggest blessing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The advent of information services introduces many advantages, for example, in trade, production and services. While making important decisions today, people increasingly rely on the information gleaned from such services. Presumably, as such, information from these services has become a target of manipulation.

During the past decade, we have already observed many forms of information manipulation that misrepresents or alters reality. Some popular manipulation – we have ever witnessed on the Internet – include using black hat SEO techniques to drive up the ranking of a disreputable business, creating disinformative campaigns to conceal political dissidence, and employing less-than-honest product assessments to paint a rosy picture for inferior wares. Today, emerging web services and technologies greatly facilitated and enhanced people's lives. However, these innovations also enrich the arsenal of manipulators.

The sheer amount of online information available today can threaten to overwhelm any user. To help ensure that users do not drown in the flood of information, modern web services are increasing relying upon personalization to improve the quality of their customers' experience. At the same time, personalization also represents new ammunition for all manipulators seeking to steer user eyeballs, regardless of their intents. In this thesis, I demonstrate a new unforeseen manipulation that exploits the mechanisms and algorithms underlying personalization. To undermine the effect of such manipulation, this thesis also introduces two effective, efficient mitigation strategies that can be applied to a number of personalization services.

In addition to aforementioned personalization, increasingly prevalent browser extensions augment the ability to distort online information. In this thesis, I unveil an overlooked but widespread manipulation phenomenon in which miscreants abuse the privilege of browser extensions to tamper with the online advertisement presented to users. Considering that online advertising business is one of the primary approaches used to monetize free online services and applications available to users, and reckless ad manipulation may significantly roil advertising ecosystem, this thesis scrutinizes the potential effect of ad manipulation, and develops a technical approach to detect those browser extensions that falsify the ads presented to end users.

Although the thesis merely discusses several manipulation examples in the context of the Internet, the findings and technologies presented in this thesis introduce broad impacts. First, my research findings raise Internet users' awareness about pervasive information manipulation. Second, the proposed technologies help users alleviate the pernicious effects of existing information manipulation. Finally, accompanying the findings and technologies is publicly available open-source software and tools that will help an increasing number of users battle against the growing threat of information manipulation.

CHAPTER I

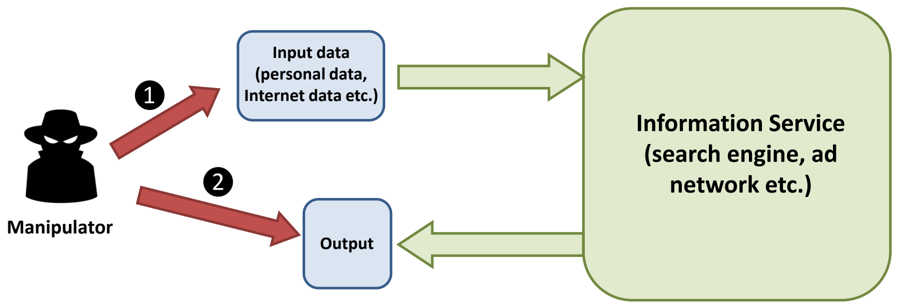
INTRODUCTION

1.1 Thesis Overview

Information services permeated almost every aspect of people's lives. With the help of information services, we share information, establish relationships, exchange ideas, and even conduct business on the Internet. The wide adoption of these information services has also attracted malicious parties who seek to use the services as entry points to gain unlawful benefits. Information manipulation is a new emerging frontier in cyber security. It denotes all attempts by adversaries to distort information with the goal to influence opinion, thought, or action.

Information manipulation can take many shapes and forms. Indeed, we have already observed many forms of information manipulation on the Internet. For example, scammers have long used cloaking as a technique to drive up search engine rankings of disreputable businesses or products, leading users to make a poor economic decision. Similarly, public relation companies leverage massive disinformation campaigns to directly influence customer purchase decisions.

Unlike most traditional attacks, which typically aim to take control of computational resources or sensitive data, information manipulation targets human minds and their ideas. Given an information service, manipulators can control information presented to users in a variety of ways. As is shown in Figure 1, attackers can manipulate the input data to an information service, in order to control the output presented to users with the goal of influencing the users to make decisions and take actions that benefit the attackers at the expense of the users. In addition, manipulators can directly control what Internet content users are able to see and how they see



Input data (personal data, Internet data etc.)
1
2
Manipulator
Output
Information Service (search engine, ad network etc.)

Figure 1: The target of information manipulators: ① the input data to the information service, and ② the output from the information service.

them in order to influence the users' minds and actions. Left untackled, information manipulation can result in serious consequences.

In this thesis, my work aims to counter information manipulation on the Internet. In achieving this goal, I rely on the following approaches:

- identifying and understanding unforeseen information manipulation schemes;
- building infrastructure to quantify and evaluate the potential influence of information manipulation;
- applying the resulting insights to the invention and design of mitigation approaches and
- building system prototype to overcome information manipulation.

More specifically, this thesis focuses on two types of unforeseen information manipulation. In one type of manipulation, manipulators control the input data to an information service. To be more specific, attackers manipulate the personal data that personalization systems rely upon and thus alter the output of these systems in predictable ways. Another type of manipulation directly targets the output from information services. In particular, attackers abuse the privilege of browser extension to manipulate the advertisement presented to users. In this thesis, I quantify the potential influence of both manipulation schemes. To mitigate the first type of manipulation, I built `BobbleX`, a client side tool that creates multiple user personas and leverages sentiment analysis techniques to expose distorted or incomplete information. In attempt to overcome the second type of manipulation, I developed `Expector`, a system that automatically identifies browser extension with advertisement manipulation activities. In the following sections, I summarize the main contributions of this thesis in turn.

1.2 Pollution Attack: Manipulating Personal Input

Modern Web services routinely personalize content to appeal to the specific interests, viewpoints, and contexts of individual users. Ideally, personalization allows sites to highlight information uniquely relevant to each of their users, thereby increasing user satisfaction—and, eventually, the service's bottom line. Unfortunately, as I demonstrate in Chapter 3, the personalization mechanisms currently employed by popular services have not been hardened against attack. I show that third parties can launch pollution attack that manipulates them to increase the visibility of arbitrary content—whether it be a new YouTube video, unpopular product on Amazon, or low-ranking website in Google search returns. I demonstrate that attackers can inject information into individual users' profiles on these services, thereby perturbing the results of the services' personalization algorithms. While the details of my exploits are tailored to each service, the general approach is likely to apply quite broadly. By demonstrating the attack against three popular Web services, I highlight a new class of vulnerability, whereby an attacker can significantly affect a user's experience for that service without the user's knowledge.

1.3 Mitigating Pollution Attack

Guided by the understanding of pollution attack, Chapter 4 introduces two mitigation strategies that aim to disclose information prejudices attributable to pollution attack. These two strategies leverage sentiment analysis to raise users' awareness about information prejudices. Using different user personas to scrutinize the output of an information service, one mitigation strategy can also facilitate users to identify possibly invalid input data. I demonstrate these two strategies on Google personalized search. In particular, I developed `BobbleX`, a client side tool that (1) allows users to see how Google returns to them differ from the results that are returned to other users; (2) raises users' awareness about information prejudices hidden behind search results; (3) facilitates users to identify invalid searches.

1.4 Mitigating Direct Information Manipulation

In Chapter 5, the thesis switches the gear to the second type of information manipulation. In particular, this chapter discusses browser extension that increasingly abuses its privilege to manipulate the ad presented to users. I present Expector, a system that automatically inspects and identifies browser extensions that abuse their privileges and inject ads in webpages that a user browses. Expector is designed to be scalable and accurate. I evaluate Expector across more than 18,000 Chrome browser extensions and are successfully able to detect 292 extensions that inject ads with a 4% false positive and 3% false negative rate. I present a detailed study of the ad injecting practices and find that many of these extensions violate the store policies, participate in fraudulent activities, and degrade their users' trust. Finally, using HTTP and DNS traces collected from large enterprises and ISP networks, I provide a detailed operation and revenue estimation of the ad networks that participate in delivering ads to these extensions.

CHAPTER II

RELATED WORK

In the past decades, there are many information manipulation attacks. Though sharing the same goal with the aforementioned manipulation, they take different measures, such as controlling publicly available information or its access. In this chapter, we summarize these information manipulation attacks and analyze their detection or mitigation mechanisms.

2.1 Controlling Publicly Available Data

Information services typically gather and process publicly available information across the Internet. For example, search engines hunt for text in publicly accessible documents offered by web servers, and rank these documents that match a given search string using various ranking algorithms. Blackhat SEO (bSEO) is one typical manipulation attack, which targets publicly accessible documents and influences their presentation to searchers. More specifically, it utilizes a huge amount of spam webpages to promote the ranking of certain target webpages in search engines [48, 58]. Another manipulation attack is opinion spamming. In particular, manipulators post a large number of fake comments and reviews on publicly available platforms with the goal of altering users' beliefs and influencing their decisions [51].

To detect the manipulation attacks above, the most common solution is to harness graph-based methods and relational modeling to identify manipulated information. In the case of pinpointing spam webpages [27,33,67,98,99], one typical approach is to analyze link dependencies among webpages. Regarding weeding out phony reviews, technical endeavors [51, 55, 63, 64, 68, 69, 87] primarily focus on developing effective anomaly detection approaches by using complex relationships among reviewers, reviews, and entities. The aforementioned personalization manipulation does not need a massive spam campaign. Thus, techniques that address bSEO and opinion spamming are unlikely to be effective against this new class of manipulation.

2.2 Controlling Data Access

In another form of information manipulation, manipulators compromise infrastructures underlying information services and thus control the access to information. Internet censorship is such an attack [23, 93]. In particular, it controls what can be accessed, published or viewed on the Internet. Internet censorship is typically carried out by government for the purpose of suppressing politically incorrect information. In some cases, individuals and organizations could also engage censorship for religious or business reasons. In general, Internet censorship happens in the communication layer, where manipulators block the access to certain information. A similar manipulation attack is communication hijacking. Different from Internet censorship, manipulators do not block content access but hijack and manipulate information as it propagates to users. One typical example of such manipulation is to insert additional advertisements on the webpages served to end users [88]. Another example is to remove unwanted search results from a user's set of results.

Past studies on mitigating these manipulation attacks focus on solutions that provide *computational trust*, e.g., establishing encrypted tunnels or proxies to circumvent Internet censorship [45, 50, 60, 89, 90]. In personalization manipulation, manipulators do not need to take the control of the infrastructure underlying information services. Thus, techniques that provide computation trust are not applicable to the new problem space. In this thesis, I invent a new type of mechanism to provide *behavioral trust*, i.e., a scheme that will ensure the content that a user receives satisfy the user's utility function or expectation.

CHAPTER III

POLLUTION ATTACKS: MANIPULATING PERSONAL INFORMATION

The economics of the Web ecosystem are all about clicks and eyeballs. The business model of many Web services depends on advertisement: they charge for prime screen real estate, and focus a great deal of effort on developing mechanisms that make sure that the information displayed most prominently is likely to create revenue for the service, either through a direct ad buy, commission, or, at the very least, improving the user's experience. Not surprisingly, malfeasants and upstanding business operators alike have long sought to reverse engineer and exploit these mechanisms to cheaply and effectively place their own content—whether it be items for sale, malicious content, or affiliate marketing schemes. Search engine optimization (SEO), which seeks to impact the placement of individual Web pages in the results provided by search engines, is perhaps the most widely understood example of this practice.

Modern Web services are increasingly relying upon personalization to improve the quality of their customers' experience. For example, popular websites tailor their front pages based upon a user's previous browsing history at the site; video-sharing websites such as YouTube recommend related videos based upon a user's watch history; shopping portals like Amazon make suggestions based upon a user's history at the site; and search engines such as Google return customized results based upon a wide variety of user-specific factors. As the Web becomes increasingly personal, the effectiveness of broad-brush techniques like SEO will wane. In its place will rise a new class of schemes and outright attacks that exploit the mechanisms and algorithms underlying this personalization. In other words, personalization represents a new attack surface for all those seeking to steer user eyeballs, regardless of their intents.

In this chapter, we demonstrate that contemporary personalization mechanisms are vulnerable to exploit. In particular, we show that YouTube, Amazon, and Google are all vulnerable to the same general form of cross-site scripting attack that allows third-party websites to alter the customized content the services return to users who have visited a page containing the exploit. We do not claim that this attack is either the most powerful—although we demonstrate it is quite effective—broadly applicable, or hard to defeat. Rather, we offer it as a first example of an entire class of attacks that we believe will soon—if they are not already—be launched against the relatively unprotected underbelly of personalization services.

Our attack exploits the fact that a service employing personalization incorporates a user's past history (including browsing, searching and purchasing activities) to customize the content that it presents to the user. Importantly, many services with personalized content log their users' Web activities whenever they are logged in regardless of the site they are currently visiting; other services track user activities on the site even if the user is logged out (e.g., through a session cookie). We use both mechanisms to pollute users' service profiles, thereby impacting the customized content returned to the users in predictable ways. Given the increasing portfolio of services provided by major players like Google and Amazon, it seems reasonable to expect that a large fraction of users will either be directly using the service or at least logged in while browsing elsewhere the Web.

We show that our class of attack, which we call *pollution attacks*, can be extremely effective on three popular platforms: YouTube, Google, and Amazon. A distinguishing feature of our attack is that it does not exploit any vulnerability in the user's Web browser. Rather, it leverages these services' own personalization mechanisms to alter the user's experience. While our particular implementation employs a form of cross-site request forgery (XSRF) [22], other mechanisms are possible.

The ability to trivially launch such an attack is especially worrisome because it indicates the current approach to Web security is ill-equipped to address the vulnerabilities likely to exist in personalization mechanisms. In particular, today's Web browsers prevent exploits like cross-site scripting and request forging by enforcing boundaries between domains though "same origin" policies. The limitations of these approaches are well known, but our attack represents a class of exploits that cannot be stopped by client-side enforcement: in an attempt to increase the footprint of its personalization engine (e.g., Google recording search queries that a user enters on a third-party page), a service with personalized services is providing the cross-site vector itself. Hence, only the service can defend itself from such attacks on its personalization. Moreover, enforcing isolation between independent Web sessions seems antithetical to the goal of personalization, which seeks to increase the amount of information upon which to base customization attempts.

This chapter makes the following contributions:

- We describe pollution attacks against three platforms—YouTube, Google, and Amazon—that allow a third party to alter the personalized content these services present to users who previously visited a Web page containing the exploit.
- We study the effectiveness of our attack on each of these platforms and demonstrate that it (1) can increase the visibility of almost any YouTube channel; (2) dramatically increase the ranking of most websites in the short term, and even have lasting impacts on the personalized rankings of a smaller set of sites, and (3) cause Amazon to recommend reasonably popular products of the attacker's choosing.
- More broadly, our attack and its effectiveness illustrates the importance of securing personalization mechanisms in general. We discuss a number of implications of our study and the ways in which websites can attempt to avoid similar vulnerabilities in the future.

The rest of the chapter is organized as follows. Section 4.2 provides a general overview of pollution attacks on personalized services. Sections 3.2, 3.3, and 3.4 introduce specific attacks that can be launched against YouTube, Google, and Amazon, respectively, and report on our success. We discuss limitations of our work and possible defenses in Section 3.5 before concluding in Section 5.7.

3.1 Overview and Attack Model

In this section, we present a brief overview of personalization as it is employed by popular Web services. We then present a model of pollution attacks, which we apply to three different scenarios later in the chapter – YouTube, Amazon, and Google.

3.1.1 Personalization

Online services are increasingly using personalization to deliver information to users that is tailored to their interests and preferences. Personalization potentially creates a situation where both the service provider and the user benefit: the user sees content that more closely matches preferences, and the service provider presents products that the user is more likely to purchase (or links that the user is more likely to click on), thus potentially resulting in higher revenues for the service provider.

The main instrument that a service provider can use to affect the content that a user sees is modifying the *choice set*, the set of results that a user sees on a particular screen in response to a particular query. The size of a choice set differs for different services. For example, YouTube shows the user anywhere from 12–40 videos; Amazon may show the user up to five sets of recommended products; Google's initial search results page shows the top ten results. Figure 2 shows several examples of choice sets on different sites.

A service's *personalization algorithm* affects the user's choice set, in response to a query. The choice set that a personalization algorithm produces depends on a user query, as well as a number of auxiliary factors, including the universe of all possible content and the user's search history. Previous work has claimed that many factors, ranging from geography to time of day, may affect the choice set that a user sees. For the purposes of the attacks in this chapter, we focus on how changes to a user's *search history* can affect the choice set, holding other factors fixed. In particular, we focus how an attacker can pollute the user's search history by generating false clicks through cross-site request forgery (XSRF). We describe these attacks in the next section.

3.1.2 Pollution Attacks

The objective of a pollution attack is to affect a user's *choice set*, given a particular input. In some cases, a user's choice set appears before the user enters any input (*e.g.*, upon an initial visit to the page). In this case, the attacker's goal may be to affect a default choice set. Figure 3 shows an overview of the attacker's goal: in summary, the attacker aims to affect the resulting choice set by altering the user's history with false clicks, using cross-site request forgery as the attack vector. This attack requires three steps:

1. *Model the service's personalization algorithm.* We assume that the attacker has some ability to model the personalization algorithm that the site uses to affect the user's choice set. In particular, the attacker must have some idea of how the user's past history affects the user's choice set. This information is often available in published white papers, but in some cases it may require experimentation.

2. *Create a "seed" with which to pollute the user's search history.* Given some knowledge of the personalization algorithm and a goal for how to affect the choice set, the attacker must design the seed with which to affect the user's choice set. Depending on the service, the seed may be queries, clicks, purchases, or any other activity that might go into the user's search history. The

optimal seed is one that can maximally affect the user's choice set with minimal overhead.

3. *Inject the seed with a vector of false clicks.* To pollute a user's history, in most cases we require that the user be signed in to the site. (For some services, pollution can take place even when the user is not signed in.) Then, the attacker can use a mechanism to make it appear as though the user is taking action on the Web site for a particular service (*e.g.*, clicking on links) using a particular attack vector.

In the following sections, we explore how an attacker can apply this same procedure to attack the personalization algorithms of three different services: YouTube, Amazon, and Google search.

3.2 Pollution Attacks on YouTube

In this section, we demonstrate our attack on YouTube. Following the aforementioned attack steps, we first understand how YouTube uses the watch history of a YouTube user account to recommend videos by reviewing the literature [38]. Second, we discuss how to prepare seed data (seed videos) to promote target data (target videos belonging to a specific channel). Third, we introduce how to inject the seed videos to a YouTube user account. Finally, we design experiments and validate the effectiveness of our attack.

3.2.1 YouTube Personalization

YouTube constructs a personalized list of recommended videos based upon the videos a user has previously viewed [38]. In particular, YouTube attempts to identify the subset of previously viewed videos that the user enjoyed by considering only those videos that the user watched for a long period of time. Typically, YouTube recommends videos that other users with similar watching histories have also enjoyed.

In particular, YouTube tracks the co-visitation relationship between pairs of videos; namely, how likely is a user who watched and enjoyed (i.e., watched a substantial portion of) video X to watch and enjoy video Y. In general, there may be more videos with co-visitation relationships (which we term “recommendable”) to a user’s history than there is display area, so YouTube selects those recommendable videos with high rankings. YouTube will not, however, recommend a video the user has already watched.

YouTube displays recommended videos in the suggestion list placed alongside with a playing video (e.g., Figure 6) and in the main portion of the screen at the end of a video (Figure 2(a)). A suggestion list appearing alongside a video typically contains 20–40 suggested videos, 2 of which are recommended based upon personalization. At the end of a video, YouTube shows an even more concise version of the suggestion list containing only 12 of the video from the full list; these may or may not contain the personal recommendations.

3.2.2 Preparing Seed Videos

YouTube organizes videos into channels, where each channel corresponds to the set of uploads from a particular user. In our attack, we seek to promote a set of target videos (Ω^T), all belonging to the same YouTube channel, C. To do so, we will use an additional set of seed videos (Ω^S) that have a co-visitation relationship with the target videos. By polluting a user’s watch history with videos in Ω^S, we can coerce YouTube into recommending videos in Ω^T. There are two ways to obtain Ω^S: we can identify videos with pre-existing co-visitation relationships to the target videos, or we can create the relationships ourselves.

Existing Relationships. In the simplest version of the attack, the attacker identifies existing videos to use as the seed set. For example, given a target video set (Ω^T) belonging to channel C, the attacker could consider all the other videos in the channel, $C - \Omega^T$, as candidate seeds. For every candidate video, the attacker checks which videos YouTube recommends when a fresh YouTube account (a YouTube account with no history) watches it. Note that YouTube allows its users to view their recommended videos at `http://www.youtube.com/feed/recommended`. If the candidate video triggers YouTube to recommend a video in Ω^T, then the attacker adds the injected video to seed video set Ω^S.

In general, this process can be used to identify seed videos for every target video in Ω^T. The attacker is not yet ready, however, to launch the attack because a YouTube video in Ω^S may trigger YouTube to also recommend videos not in Ω^T. To address this issue, the attacker can simply add these unwanted videos to the seed video set Ω^S because YouTube does not recommend videos that the user has already watched. (As we show later, the attacker will convince YouTube that the user watched, but *did not* enjoy, these unwanted videos, so their inclusion in Ω^S will not lead to additional recommendations.)

Fabricating Relationships. For some videos, it may be difficult to identify a seed set Ω^S that recommends all of the elements of Ω^T due to lack of co-visitation relationships for some of the target elements. Instead, an attacker that is willing to upload her own content to use as the seed set can simply create co-visitation relationships between this content and the target set. In particular, an attacker uploads a set of videos (Ω^0) and establishes co-visitation relationships between Ω^0 and Ω^T through crowd-sourcing (e.g., Mechanical Turk or a botnet): YouTube visitors need only watch a video in Ω^0 followed by a video in Ω^T. After a sufficient number of viewing pairs (we show in Section 3.2.4.1 it doesn't take many), the attacker can use videos in Ω^0 as the seed set.

3.2.3 Injecting Seed Videos

To actually launch the attack and inject seed videos into a victim's YouTube watch history, an attacker can harness XSRF to forge the following two HTTP requests for each video in the seed set: (1) `http://www.youtube.com/user_watch?plid=<value>&video_id=<value>`, and (2) `http://www.youtube.com/set_awesome?plid=<value>&video_id=<value>`, where `plid` and `video_id` correspond to the values found in the source code of the seed video's YouTube page. The first HTTP request spoofs a request from the victim to start watching the seed video, and the second convinces YouTube that the victim watched the video for a long period of time. Note that both HTTP requests are required in order for videos in Ω^S to trigger the recommendation of videos in Ω^T, but only the first HTTP request is needed to prevent the recommendation of unwanted videos.

3.2.4 Experimental Design

We wish to evaluate the effectiveness of our attack both in controlled environments and against real YouTube users. We start by validating the effectiveness of our approach in the simplest scenario where our attack is launched to promote pre-existing YouTube channels through existing co-visitation relationships. We then consider the scenario where an attack seems to upload and promote content from her own channel. Finally, we conduct a small-scale experiment to demonstrate the attack works against a volunteer set of real YouTube users.

3.2.4.1 New Accounts

We first promote existing YouTube channels by launching our attack against victims with fresh YouTube user accounts. Fresh YouTube user accounts allow us to confirm the effectiveness of our approach in the absence of other, potentially countervailing influences, i.e., other personalized recommendations based on a user's existing history.

We begin by selecting 100 existing YouTube channels at random from among the list of the top 2,000 most-subscribed channels published by VidStatsX [86]. For each of the selected YouTube channels, we randomly select 25 videos from the channel as the target video set, used the method described in the previous section to identify a seed video set, and then injected the seed videos to a fresh YouTube account.

Secondly, we consider promoting new content by creating our own YouTube channel and similarly attacking fresh YouTube accounts. Our YouTube channel contains two 3-minute videos. We select one of the videos as a one-element target video set and used the other as the seed set. We created a co-visitation relationship by embed both videos on a Web page and recruiting volunteers to watch both videos sequentially. We obtained 65 and 68 views by existing YouTube users for our seed and target video respectively.

3.2.4.2 Existing Accounts

Finally, we study the effectiveness of our pollution attack against real YouTube user accounts. We recruit 22 volunteers with extensive pre-existing YouTube watch histories. In order to limit the inconvenience to our volunteers, we limit our study to attempting to promote one moderately popular YouTube channel based upon existing co-visitation relationships. We select a moderately popular account for two reasons: first, a wildly popular channel is likely to be recommended anyway, so extracts little benefit from our attack. Conversely, an entirely new channel requires a certain amount of effort to establish the co-visitation relationships as described above and we have limited volunteer resources.

Based on these parameters, we arbitrarily selecte channel *OnlyyouHappycamp*. We believe this selection is a reasonable candidate to be promoted using our attack due to the following reasons. First, compared to popular channels, most videos in *OnlyyouHappycamp* have low view counts (about 2,000 view counts per video on average) and the number of subscribers to the channel is a similarly modest 3,552.

Both of these are easily achievable by an attacker at fairly low cost[1]. Second, most videos in *OnlyyouHappycamp* are 22 minutes long, which makes them well suited for promotion. As we explain in Section 3.2.5.1, the length of a target video plays a significant role in its likelihood of being selected from among the recommendable videos.

Similar to the experiments with new accounts, we randomly select 15 target videos from channel *OnlyyouHappycamp*, identified a seed set, and injected the seed videos into the volunteers' YouTube accounts. After pollution, the volunteers are asked to use their accounts to watch three videos of their choice and report the suggestion list displaying alongside each of their three videos.

3.2.5 Evaluation

We evaluated effectiveness of our pollution attacks by logging in as the victim user and viewing 114 representative videos[2]. We measure the effectiveness of our attack in terms of *promotion rate*: the fraction of the 114 viewings when at least one of the target videos was contained within the video suggestion list. Recall that the list contains at most two personalized recommendations; we call it a success if one or both of these videos are among the target set.

3.2.5.1 New Accounts

Our attacks are universally successful in promoting target videos from each of the 100 selected existing channels. In other words, each time we target a fresh account by injecting seed videos targeting a particular channel, we observe the target videos in the suggestion list for each of the 114 videos. Because these are fresh accounts, there is no other history, so our targeted videos always occupy both of the personalized

[1]According to the prices in underground markets such as `freelancer.com` and `fiverr.com`, 40,000 view counts and 10,000 subscribers cost $15 and $30 US dollars, respectively.

[2]We actually attempted to view 150 videos drawn at random from a trace of YouTube usage at our institution over the course of several months. Unfortunately 36 of the videos were no longer available at the time of our experiment.

recommendation slots.

In addition, we observe the the particular target videos shown in the suggestion video list varies, even when we viewing the same video using the same victim YouTube account. In other words, every target video has a chance to be promoted and shown on the suggestion video list no matter which video a victim plays. Figure 4 shows the frequency with which each of the 25 target videos for a representative channel, *lady16makeup*. In an attempt to explain this variation, we compute (1) the Pearson correlation between the showing frequencies and the lengths of the target videos for each channel (ρ_t); (2) the Pearson correlation between the showing frequencies and the view counts of these target videos for each channel (ρ_{cnt}). We find the average Pearson correlation values are medium ($\rho_t = 0.54$) and moderate ($\rho_{cnt} = 0.23$), respectively. This suggests that both the length and view count of a target video influence its recommendation frequency, but the length of a target video is a more significant factor.

Because screen real estate is precious, and users typically focus on the first few items of a list, we report on the position within the suggested video lists that our targeted videos occupy when they are promoted. We observe that the two target videos are usually placed back-to-back on the suggestion list. Moreover, as shown in Figure 5, YouTube usually places our target videos among the top few spots of a victim's suggestion list: In our tests with new accounts, the target videos are always recommended and placed on the top 12, which means they also appear at the end of viewed videos. This is particularly significant, as it implies that our target videos are shown even if a victim finishes watching a YouTube video on a third-party Website (which typically embeds only the view-screen portion of the YouTube page, and not the full suggestion list).

Our attacks were similarly completely successful in promoting newly uploaded content. As a control, we also signed in as non-polluted fresh YouTube accounts and, unsurprisingly, did not find any of our new content among the videos in the suggestion list. In other words, the videos were recommended exclusively because of our attacks; our experiments were sufficiently small that we did not lead YouTube to conclude that our content was, in fact, universally popular. Figure 6 shows a sample screenshot comparing the suggestion lists from a victim account and another, non-exploited fresh account. Finally, while we omit the full distribution due to space constraints, we find that one of our target videos occupies the top suggestion slot while viewing 80 out of the 114 test videos.

3.2.5.2 Existing Accounts

Unsurprisingly, our attacks were somewhat less successful on real YouTube accounts. 14 out of the 22 volunteer YouTube users report that at least one of our target videos from channel *OnlyyouHappycamp* appeared in the suggestion list during each of their three video viewings, a 64% promotion rate.

To understand why we were able to exploit some accounts and not others, we asked our volunteers to share their YouTube watch histories. Ten of our volunteers shared their histories with us and allowed us to sign in to their YouTube accounts to conduct a further study. The number of videos in the watch histories of the 10 volunteers ranged from a few hundred to ten thousand. Figure 7 shows the relationship between the number of watched videos in a watch history and the number of times that at least one of our target videos is displayed along with a playing video. While there appears to be an intuitive decreasing trend (i.e., the longer the history an account has the more resistant it is to pollution), there are obvious outliers. For example, one account with almost 3,500 previous viewings in its history succumbed to our attacks almost 80% of the time.

Consistent with the Pearson coefficients reported earlier, we find that the success of our attacks depends on the rankings and lengths of the videos that are otherwise suggested based upon a user's history. In particular, we observe that the majority of the videos recommended to users for whom our attacks have low promotion rates have longer lengths and more view counts than our target videos, while the videos that YouTube recommends based on the watch history of the user with 3,500 previous viewings have shorter lengths than our target videos (though they generally have higher view counts than our targets).

While we believe our attack clearly demonstrates that YouTube's personalization mechanism is subject to exploit, it is not clear how long lasting such exploits would be, nor how easy they are to defend against. In our experiments, volunteers watch arbitrary YouTube videos right after being attacked, but we believe our pollution attacks on YouTube are likely to last for a significant period of time. While YouTube does not explicitly disclose how time factors into their recommendation system (if at all) [38], analysis of volunteers' watch histories indicates that a YouTube video that was watched as long as two weeks prior is still used for generating recommended videos.

3.3 Google Personalized Search

In this section, show how history pollution attacks can be launched against Google's search engine. The goal of our attack is to promote a target webpage's rank in the personalized results Google returns for an arbitrary search term by injecting seed search terms into a victim's search history.

3.3.1 How Personalized Search Works

Search personalization uses information about users including their previous query terms, click-through data and previously visited websites to customize results. The details of Google's personalization algorithms are not public, but many previous studies have explored aspects of personalized search [28,36,41,56,59,71,76,78,80–82]. We describe two classes of personalization algorithms: *contextual personalization* and *persistent personalization.* According to recent reports [73, 74], many search engines including Google, Bing, and Yahoo! apply both types of personalization.

Contextual personalization constructs a short-term user profile based on recent searches and click-throughs [36, 80]. When a user searches for “inexpensive furniture” followed by “maternity clothes,” Google’s contextual personalization algorithm typically promotes search results that relate to “inexpensive maternity clothes” for the next few searches. In contrast, persistent personalization uses the entire search history—as opposed to only recent searches—to develop a user profile [59, 78]. Personalization that occurs over the longer term may not affect a user’s search results as dramatically, but can have longer-lasting effects for the results that a user sees. For example, searching for “Egypt” using different accounts may result in two distinct result sets: one about tourism in Egypt and the other related to the Arab Spring.

3.3.2 Identifying Search Terms

Given the differing underlying algorithsm that govern contextual and persistent personalization, an attacker needs to select different sets of seed search terms depending on the type of attack she hopes to launch.

Contextual Personalization. For the contextual personalization attack, the keywords injected into a user’s search history should be both relevant to the promoting keyword and unique to the website being promoted. In particular, the keywords should be distinct with respect to other websites with close ranking positions so that the only website promoted is the target website and no other closely-related websites. Presumably, an attacker promoting a specific website will be familiar with the website and know what keywords best meet these criteria. However, good candidate keywords can be obtained using automation by examining a website’s `meta` keyword tag. While Google no longer incorporates `meta` tags into their ranking function [97], the keywords listed in the `meta` keyword tag provide a good summary of the page’s content.

Persistent Personalization. Carrying out a persistent personalization attack requires a different method of obtaining keywords to inject. In this case, the size of the keyword set should be larger than that used for a contextual attack in order to have a greater impact on the user's search history. Recall that contextual attacks only impact a user's current session, while persistent attacks pollute a user's search history in order to have a lasting impact on the user's search results. Good candidate keywords are found using the Google AdWords tool, which takes as an input a search term and URL and produces a list of about one hundred related keywords. Ideally, an attacker would pollute a user's search history with each of these terms. Doing so takes time, however. We determined that an attacker can safely inject roughly 50 keywords a minute using cross-site request forgery; more rapid search queries are flagged by Google as a screen-scraping attack. For this study we assume an attacker can inject a maximum of 25 keywords into a user's profile, but the number of keywords can increase if the user stays on a webpage longer than 30 seconds. Not all keyword lists return by the AdWords tool actually promote the target website, however. The effectiveness of this attack likely depends on several factors, including the user's current search history, time, etc. In Section 3.3.5, we evaluate the effectiveness of this attack under different conditions.

3.3.3 Injecting Search Terms

Similar to the attacks on YouTube, the attack on Google's personalized search also harnesses XSRF to inject the seeds. For example, an attacker can forge a Google search by embedding `https://www.google.com/search?hl=en&site=&q=usenix+security+2013` into an invisible iframe. Injecting search terms into a Google user's account affects the search results of the user's subsequent searches. The exact number and set of search terms to inject differs depending on whether an attacker carries out a contextual or persistent personalization attack.

3.3.4 Experimental Design

To cleanly study the effects of our proposed attacks on contextual and persistent search personalization, we conduct most of our experiments using Google accounts with no search history. To validate whether our results apply to real users, we also conduct a limited number of tests using accounts that we construct to mimic the personae of real users.

To quantify the effectiveness of our attack in general, we must select an unbiased set of target Web pages whose rankings we wish to improve. We build two test corpora, one for attacks on contextual personalization, and one for attacks on persistent personalization. We attempt to promote existing Websites using only their current content and link structure; we do not perform any SEO—white or black hat—on Websites before conducting our attacks. We believe this represents a conservative lower bound on the effectiveness of the attack, as any individual Website owner could engineer the content of their site to tailor it for promotion through search history pollution.

3.3.4.1 Contextual Pollution

We start by scraping 5,671 shopping-related keywords from `made-in-china.com` to use as search terms. We then enter each of these terms into Google one-by-one to obtain the top 30 (un-personalized) search results for each. Because some of our search terms are related, not all of these URLs are unique. Additionally, we cannot hope to improve the URLs that are already top-ranked for each of the search terms. This leaves us with a corpus of 151,363 URLs whose ranking we could hope to improve.

Because we could not manually inspect each of these Websites to determine appropriate seed search terms, we instead focus on the subset of them that include a `meta` keyword tag. For the approximately 90,000 such sites, we extract the provided keywords or phrases, exactly as provided by the Website. Clearly many of these keywords are generic and will appear in a wide variety of Websites—to launch the attack, we require keywords that are unique to the Website we wish to promote (at least relative to the other URLs returned in response to the same query). Hence, we ignore any keywords that are associated with multiple URLs in the same set of search results.

After this procedure, we end up with 2,136 target URLs spanning 1,739 different search terms for which we have a set of 1–3 seed keywords to try to launch a contextual pollution attack. The average search term has 1.23 results whose ranking we will try to improve. Figure 8 shows the distribution of the original rankings for each of these target Webpages; the distribution is skewed toward highly ranked sites, perhaps because these sites take care in selecting their `meta` tag keywords.

3.3.4.2 Persistent Pollution

Once again, we begin by selecting 551 shopping-related search terms and perform Google searches with each of the search terms to retrieve the top 30 search results. As opposed to the contextual attack, where we searched for keywords that differentiate the results from one another, we aim to find search terms that will be associated with the Website and search-term pair for the long term.

As described in Section 3.3.2, we use a tool provided by Google AdWords to obtain a list of keywords that Google associates with the given URL and search term. Constructing related keyword lists for each of the 29 search returns (again excluding the top hit, which we cannot hope to improve) and 551 search terms yields 15,979 distinct URLs with associated lists of keywords.

For each URL, we select 25 random keywords from the AdWords list for 25 distinct trials. If a trial improves a URL's ranking, we then test the persistence of the attack by performing 20 subsequent queries, each with a randomly chosen set of Google trending keywords. These subsequent queries help us verify that the URL promotion is not just contextual, but does not vanish when a user searches other content. If, after all 25 trials, we find no keyword sets that promote the URL's ranking and keep it there for 20 subsequent searchers, we deem this URL attempt a failure. If multiple keyword sets succeed, we select the most effective (i.e., the set of 25 keywords that induces the largest ranking improvement) trial to include in the test set.

3.3.5 Evaluation

In this section, we seek to quantify the effectiveness of search history pollution by simulating attacks that seek to promote the target Websites identified in the previous section. To scope our measurements, we consider the effectiveness of the attacks only for the set of search terms that we identified; it is quite possible, of course, that our pollution attacks also affect the rankings of the targeted URLs for other search terms.

When measuring the effectiveness of our attack, we use two different criteria, depending upon a Website's original position in the search results. In the case of URLs that are already on the front page of search results (i.e., results 2–10), we consider the pollution attack successful if it increases the ranking of a URL at all. For the case of URLs that are on subsequent pages, however, we consider the attack successful only if the attack moves the URL to the first page of search results, since improved ranking on any page that is not the first page is not likely to have any real utility.

3.3.5.1 Top-Ranked Sites

For our 2,136-page contextual attack test corpus, of the 846 pages that appeared on the front page prior to our attack, we improved the ranking of 371 (44%). The persistent attack was markedly less effective, as only 851 (17%) of the 4,959 test cases that originally appeared on the first page of the search results survived the persistence test (i.e., they remained promoted after 20 random subsequent queries).

In both cases, however, the probability of success depends greatly on the original ranking of the targeted URL. For example, moving a second-ranked URL to the top-ranked position for contextual personalization succeeded 1.1% of the time, whereas moving a tenth-ranked URL up by at least one position succeeded 62.8% of the time. Similarly, for attacks on persistent personalization, moving a second-ranked URL to the top succeeded 4.3% of the time, and moving a tenth-ranked URL to a higher-ranked position succeeded 22.7% of the time. These results make sense, because second-ranked sites can only move into the top-ranked position, whereas sites that are ranked tenth can move into any one of nine higher spots.

To illustrate this effect, and illuminate how far each webpage is promoted, Figure 9 shows the PDF of an improved webpage's rank after contextual history pollution, based upon its position in the non-personalized search results. We observe that most webpages are promoted 1 to 2 spots via our contextual pollution attack. However, there also exist many left tails, meaning that some webpages are promoted to very high ranks even if they are originally ranked very low. Similarly, Figure 10 shows the distributions for each result ranking for those websites whose rankings were improved by a persistent history pollution attack. Here, the distributions appear roughly similar (although the absolute probability of success is much lower), but it is hard to draw any strong conclusions due to the small number of promoted sites of each rank for either class of attack.

3.3.5.2 The Next Tier

The remaining 1,290 of our test websites for the contextual attack started on the second or third page of search results. By polluting a user's search history with the unique `meta` tag keywords associated with each site, we promoted 358 of them (28%) to the front page. Interestingly enough, Figure 9(j) shows that these websites have a higher chance of ending up at the very top of the results than those pages that started at the bottom of the first page in our data set. We suspect this phenomenon results from the choice of keywords used in pollution: because their original rankings were low, it will take a distinguishing keyword to move one of the webpages to the front page at all. If such a keyword is capable of doing so, it may be more likely to be sufficiently distinguished to move the page nearly to the top of the results.

Here, the results from the persistent test set are markedly different. Figure 10(j) shows that sites starting on the second or third page are extremely unlikely to end up at the very top of the result list due to a persistent history attack. Indeed, only 80 (less than 1%) of the 11,020 attacks that attempted to promote a website appearing on the 2nd or 3rd page of results was successful in moving it to the front page (and keeping it there). From this result we can clearly see that persistent search history attacks are generally best launched for sites that are already highly ranked, as opposed to contextual attacks, which can help even lower-ranked sites.

3.3.5.3 Real Users

Finally, we further evaluate the effectiveness of our successful attacks on ten volunteers' accounts with extensive pre-existing search histories. We find that, on average, 97.1% of our 729 previously successful contextual attacks remain successful, while only 77.78% of the persistent pollution attacks that worked on fresh accounts achieve similar success. Presumably the remainder are counter-acted by biases in the user's own search histories. The disparity between the test sets is likely due to the increased noise in the long-term search history. Contextualized attacks rely only on a small set of recent search terms to alter the personalized search results, which is unlikely to be affected by a user's search history. In contract, persistent poisoning relies on a much larger portion of a user's search history. If relevant keywords are already present in a user's search history, it may render some poisoning search terms ineffective. In any event, both class of attacks are relatively robust even when launched against users with long search histories.

3.4 Pollution Attacks on Amazon

Of the three services, Amazon's personalization is perhaps the most direct and obvious to the end user. On the one hand, it makes pollution-based attacks less insidious, as they will be plainly visible to the observant user. On the other, of the three services, Amazon has the most direct monetization of its screen real estate—users may directly purchase the goods from Amazon—so it is possible that any exploitation of Amazon's personalization can be profitable to an enterprising attacker.

Amazon tailors a customer's homepage based upon the previous purchase, browsing and searching behavior of the user. Amazon product recommendations consider each of these three activities individually, and explicitly labels its recommendations according to the aspect of the user's history it used to generate them. For our purposes, we focus on the personalized recommendations Amazon generates based upon the browsing and searching activities of a customer because manipulating the previous purchase history of a customer may have unintended consequences.

3.4.1 Amazon Recommendations

In general, Amazon displays on a customer's homepage five separate recommendation lists that are ostensibly computed based on the customer's searching and browsing history. Among the lists, four of them are computed based on the products that the customer has viewed lately (view-based recommendation lists) and the last is computed based upon the latest search term the customer entered (a search-based recommendation list). For each of the view-based recommendation lists, Amazon uses co-visitation relationships between products to compute corresponding recommended products (here the co-visitation relationship between a product pair on Amazon means how often two products are purchased together). In other words, Amazon uses each of the four most recently viewed products to compute a personalized view-based recommendation list. For the recommendation list that is computed based on the latest search term of a customer, the recommended products are the top-ranked results for the latest search term.

A key distinction between the operation of Amazon's personalization and the other two services is that, aside from purchase-based personalization, Amazon's personalization is controlled by the user's Web browser as opposed to the service itself. Because customers frequently do not sign into their accounts when browsing on Amazon, both the latest viewed products and search term of the customer are stored in session cookies on the user's browser rather than in profiles on Amazon servers.

3.4.2 Identifying Seed Products and Terms

Because Amazon computes the view and search-based recommendation lists separately, the seed data needed to exploit each is necessarily different.

Visit-based pollution. To promote a targeted product in a view-based recommendation list, an attacker needs to identify a seed product as follows. Given a targeted product that an attacker wishes to promote, the attacker visits the Amazon page of the product and retrieves the related products that are shown on Amazon page of the targeted product. To test the suitability of these related products, the attacker can visit the Amazon page of that product and then check the contents of the Amazon home page. If the targeted product is shown in a recommendation list, the URL of the candidate related product can be used as a seed to promote the targeted product.

Search based. To promote a targeted product in a search-based recommendation list it suffices to identify an appropriate search term. If automation is desired, an attacker could use a natural language toolkit to automatically extract a candidate keyword set from the targeted product's name. Any combination of these keywords that successfully isolates the targeted product can be used as the seed search term for promoting the targeted product. For example, to promote product "Breville BJE200XL Compact Juice Fountain 700-Watt Juice Extractor", an attacker for example can inject search term "Breville BJE200XL" to replace an Amazon customer's latest search term through XSRF.

3.4.3 Injecting Views/Searches

As with attacks on the previous two services, the attacker embeds the Amazon URLs of her desired seed items or search queries in her own website. For example, if one seed search terms is "Coffee Maker", the seed URL would be something like `http://www.amazon.com/s/?field-keywords=Coffee+Maker`. Similarly, an attacker could embed the URL of a seed product into an invisible *img* tag as the *src* of the image. Once a victim visits the attacker's website, Amazon would believe that she has made that search query (viewed that item) and, unless the victim clears her cookies, customize her Amazon homepage based upon this search. In particular, these will be the most recent activities the next time the victim visits Amazon, so they will be shown first among the recommendation lists.

3.4.4 Experimental Design

To evaluate the effectiveness of our attack, we conduct two experiments. Our first experiment measures the effectiveness of our attack when targeted toward popular items across different categories of Amazon products. The second quantifies the effectiveness of our attack on randomly selected, mostly unpopular Amazon products.

3.4.4.1 Popular Products

Amazon categorizes sellers' products into 32 root categories. To select products from each category, we scrape the top 100 best-selling products in each category in January 2013, and launch a separate attack targeting each of these 3,200 items.

3.4.4.2 Random Products

To evaluate the effectiveness of attempting to promote arbitrary products, we also select products randomly. In particular, we download a list of Amazon Standard Identification Number (ASIN) [25] that includes 75,115,473 ASIN records. Since each ASIN represents a Amazon product, we randomly sample ASINs from the list and construct a set of 3,000 products currently available for sale. For every randomly selected product in the list, we also record the sale ranking of that product in its corresponding category.

3.4.5 Evaluation

Because Amazon computes search and visit-based recommendations based entirely upon the most recent history, there is no need to experiment with real user's Amazon accounts. Hence, we measure the effectiveness of our attack by studying the success rate of promoting our targeted products for fresh Amazon accounts.

3.4.5.1 Promoting Products in Different Categories

To validate the success of our attack on each targeted product, we check whether the ASIN of the targeted product matches the ASIN of an item in the recommendation lists on the user's customized Amazon homepage.

Figure 11 illustrates the promotion rate of target products in each category. The view-based and search-based attacks produce similar promotion rates across all categories (about 78% on average). Two categories stand out as yielding markedly lower promotion rates than the rest: Gift-Cards-Store and Movies-TV (achieving 5% and 25%, respectively).

In order to understand why these categories yield lower promotion rates, we analyze the top 100 best selling products for each category. For Gift-Cards-Store, we find that there are two factors that distinguish gift cards from other product types. First, the gift cards all have very similar names; therefore, using the keywords derived from the product name results in only a small number of specific gift cards being recommended. Second, we find that searching for any combination of keywords extracted from the product names always causes a promotion of Amazon's own gift cards, which may imply that it is more difficult to promote product types that Amazon competes with directly.

Further investigation into the Movies-TV category reveals that Amazon recommends TV episodes in a different way. In our attempts to promote specific TV episodes, we find that Amazon recommends instead the first or latest episode of the corresponding TV series or the entire series. Because we declare a promotion successful only if the exact ASIN appears in the recommendation lists, these alternate recommendations are considered failures. However, we argue that these cases should be counted as successful because the attack caused the promotion of very similar products. Therefore, we believe that for all categories except for Gift-Cards-Store, an attacker has a very high chance of successfully promoting best selling products.

3.4.5.2 Promoting Randomly Selected Products

We launch pollution attacks on 3,000 randomly selected products. We calculate the *Cumulative Success Rate* of products with respect to their rankings. The Cumulative Success Rate for a given range of product rankings is defined as the ratio of the number of successfully promoted products to the number of target products in that range.

Figure 12 shows the cumulative promotion rate for varying product rankings for the two different types of pollution attacks. As the target product decrease in popularity (i.e., have a higher ranking position within their category) pollution attacks become less effective. However, this is a limitation of Amazon recommendation algorithms, not our attack. For products with very low rankings, there is little chance that they have been bought by many users. Hence, they may have very few and weak co-visit and co-purchase relationships with other products. It is difficult to say precisely without further study, but it appears that products ranking 2,000 or higher within their category have at least a 50% chance of being promoted by a visit-based pollution attack, and products with rankings 10,000 and higher have at least a 30% chance to be promoted using search-based attacks.

3.5 Discussion

There are several limitations with our current study. Most notably, the scale of our experiments is modest. However, since in most instances we randomly selected the target items, we believe that the results of our experiments are representative, and they illustrate the substantial potential impacts of pollution attacks. Similarly, our specific attacks are fragile; there are a number of steps each service could take to defend themselves.

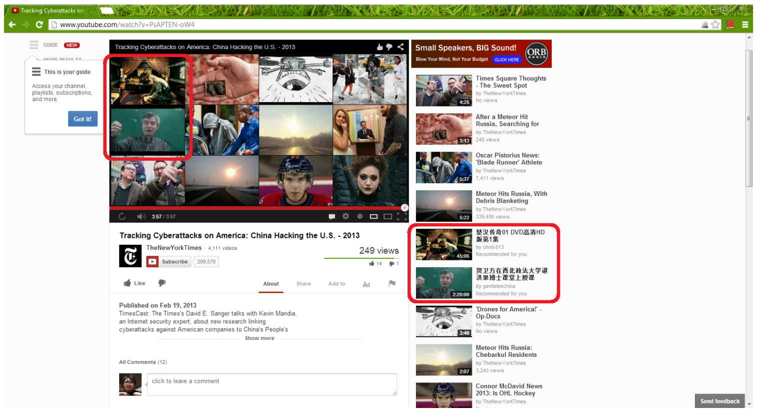
A possible defense against our pollution attacks is that cross-site request forgery can be stopped by requiring that requests to a website carry tokens issued by the site. However, the result is that cross-site user information and behaviors cannot be harvested for personalization. This goes against the current trend of constantly increasing the scope of data collection by websites. Thus, a positive impact of our work may be that (some) websites will begin to consider the trade-offs between the security and benefits of personalization.

YouTube in particular uses two separate HTTP requests to track a YouTube's user watching activity, and these are entirely independent from the actual streaming of the video. One straightforward defense mechanism against our pollution attacks is to monitor the time between the arrivals of the two HTTP requests. If YouTube finds the interval is substantially less than the length of the video, it could ignore the signal. However, an attacker can always inject a short video or control the timing of the HTTP requests in an effort to bypass such a defense mechanism. Anecdotally, we find that an injected short video can be used to promote multiple longer videos. For example, watching one 2-second video [75] causes YouTube to recommend several long videos.

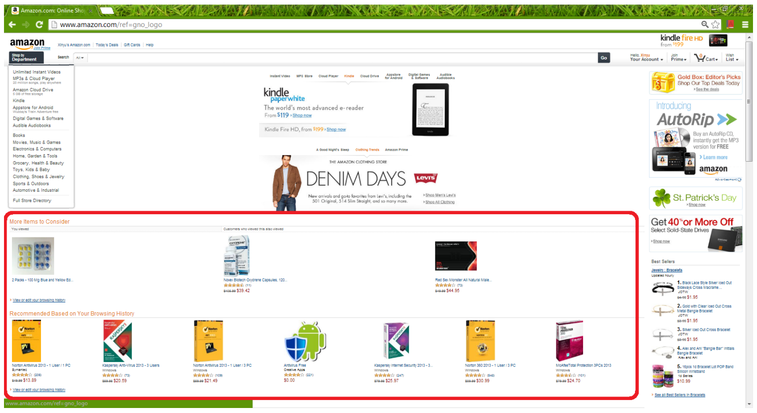
3.6 Conclusion

In this chapter, we present a new attack on personalized services that exploits the fact that personalized services use a user's past history to customize content that they present to the user. Our attack pollutes a user's history by using cross-site request forgery to stealthily inject and execute a set of targeted browsing activities in the user's browser, so that when the user subsequently accesses the associated service specific content is promoted. We illustrate how an attacker can pollute a user's history to promote certain content across three platforms. While our attack is simple, its impact can be significant if enough users' histories are compromised.

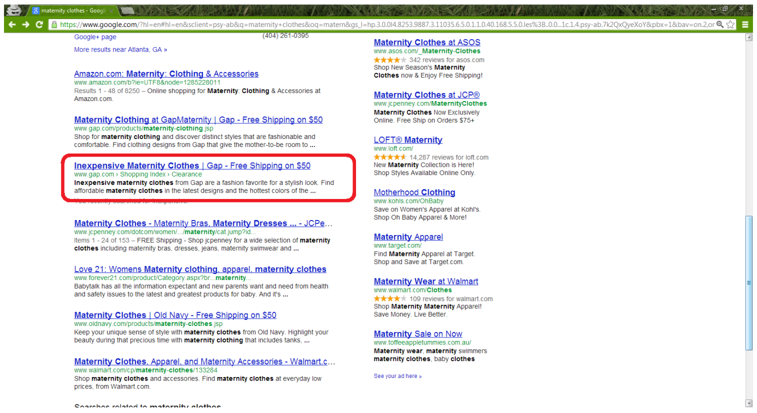
As personalization algorithms and mechanisms increasingly control our interactions with the Internet, it is inevitable that they will become the targets of financially motivated attacks. While we demonstrate pollution attacks only on YouTube, Google, and Amazon, we believe that our methods are general and can be widely applied to services that leverage personalization technologies, such as Facebook, Twitter, Netflix, Pandora, etc. The attacks we present here are just (the first) few examples of potentially many possible attacks on personalization. With increasingly complex algorithms and data collection mechanisms aiming for ever higher financial stakes, there are bound to be more vulnerabilities that will be exploited by motivated attackers. The age of innocence for personalization is over; we must now take up this new challenge.


www.youtube.com/watch?v=PsAPTEN-oW4
Tracking Cyberattacks on America: China Hacking the U.S. - 2013
TheNewYorkTimes
249 views
Small Speakers, BIG Sound!
Times Square Thoughts - The Sweet Spot
After a Meteor Hit Russia, Searching for
Oscar Pistorius News: 'Blade Runner' Athlete
Meteor Hits Russia, With Debris Blanketing
'Drones for America!' - Op-Docs
Meteor Hits Russia: Chebarkul Residents
Connor McDavid News 2013: Is OHL Hockey
Send feedback
Published on Feb 19, 2013
TimesCast: The Times's David E. Sanger talks with Kevin Mandia, an Internet security expert, about new research linking cyberattacks against American companies to China's People's

(a) Customized YouTube.



(b) Customized Amazon.


Google+ page
More results near Atlanta, GA »

Amazon.com: Maternity: Clothing & Accessories
www.amazon.com/b?ie=UTF8&node=1285228011
Results 1 - 48 of 8250 – Online shopping for Maternity: Clothing & Accessories at Amazon.com.

Maternity Clothing at GapMaternity | Gap - Free Shipping on $50
www.gap.com/products/maternity-clothing.jsp
Shop for maternity clothing and discover distinct styles that are fashionable and comfortable. Find clothing designs from Gap that give the mother-to-be room to ...

Inexpensive Maternity Clothes | Gap - Free Shipping on $50
www.gap.com › Shopping Index › Clearance
Inexpensive maternity clothes from Gap are a fashion favorite for a stylish look. Find affordable maternity clothes in the latest designs and the hottest colors of the ...

Maternity Clothes - Maternity Bras, Maternity Dresses ... - JCPe...
www.jcpenney.com/dotcom/women/.../maternity/cat.jump?id...
Items 1 - 24 of 153 – FREE Shipping - Shop jcpenney for a wide selection of maternity clothes including maternity bras, dresses, jeans, maternity swimwear and ...

Love 21: Womens Maternity clothing, apparel, maternity clothes
www.forever21.com/product/Category.aspx?br...maternity...
Babytalk has all the information expectant and new parents want and need from health and safety issues to the latest and greatest products for baby. And it's ...

Maternity Clothes | Old Navy - Free Shipping on $50
www.oldnavy.com/products/maternity-clothes.jsp
Keep your unique sense of style with maternity clothes from Old Navy. Highlight your beauty during that precious time with maternity clothing that includes tanks, ...

Maternity Clothes, Apparel, and Maternity Accessories - Walmart.c...
www.walmart.com/cp/maternity-clothes/133284
Shop maternity clothes and accessories. Find maternity clothes at everyday low prices, from Walmart.com.

Maternity Clothes at ASOS
www.asos.com/_Maternity-Clothes
342 reviews for asos.com
Shop New Season's Maternity Clothes now & Enjoy Free Shipping!

Maternity Clothes at JCP®
www.jcpenney.com/MaternityClothes
Maternity Clothes Now Exclusively Online. Free Ship on Orders $75+

LOFT® Maternity
www.loft.com/
14,287 reviews for loft.com
New Maternity Collection is Here! Shop Styles Available Online Only.

Motherhood Clothing
www.kohls.com/OhBaby
Save on Women's Apparel at Kohl's. Shop Oh Baby Apparel & More!

Maternity Apparel
www.target.com/
Find Maternity Apparel at Target Shop and Save at Target.com.

Maternity Wear at Walmart
www.walmart.com/Clothes
109 reviews for walmart.com
Shop Maternity Maternity Apparel! Save Money. Live Better.

Maternity Sale on Now
www.toffeeappletummies.com.au/
Maternity wear, maternity swimmers maternity clothes, baby clothes

See your ad here »

(c) Customized Google.

Figure 2: Websites with personalized services (personalized services tailor the data in the red rectangles).

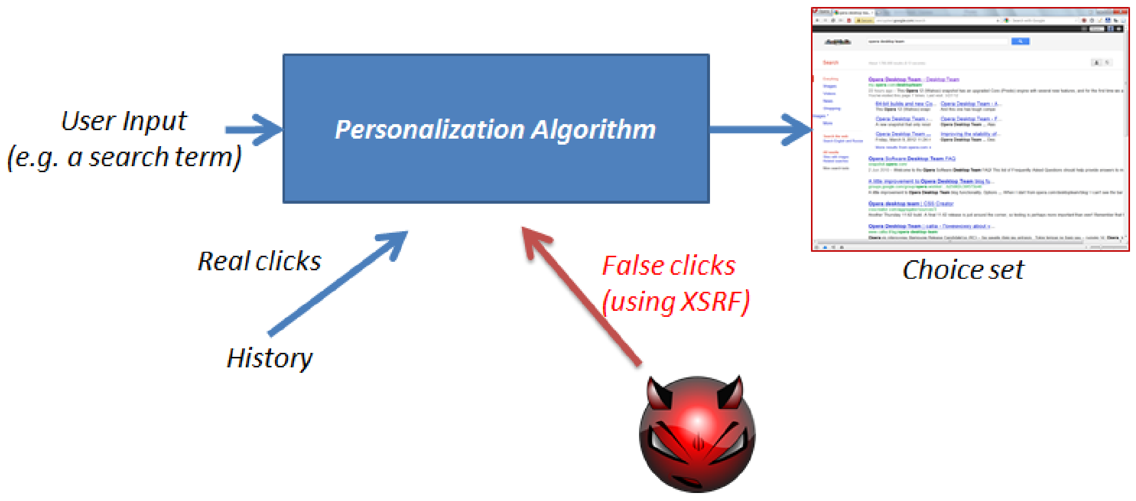


Figure 3: Overview of how history pollution can ultimately affect the user's choice set.

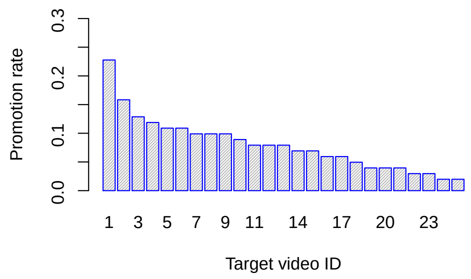
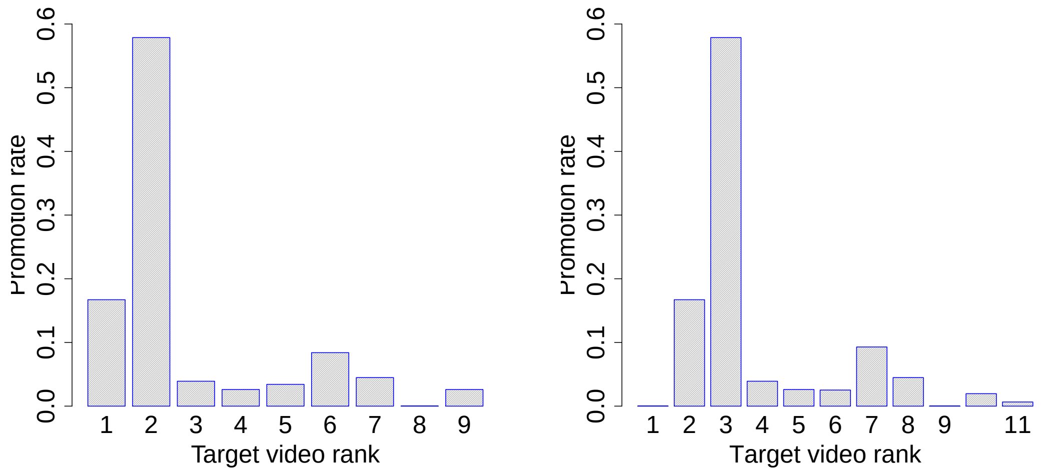


Figure 4: The promotion rate for each of the 25 target videos in channel *lady16makeup*. Two videos were recommended in each of the 114 trials.



(a) Higher ranked video (b) Lower ranked video

Figure 5: Distribution of the suggestion slots occupied by each of the two successfully promoted target videos.

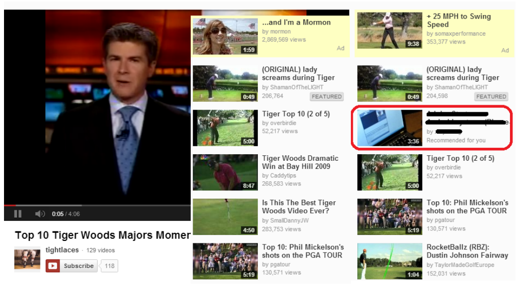


Figure 6: Suggestion lists before (left) and after (right) a pollution attack against a fresh YouTube user account. The video highlighted in red is our uploaded video.

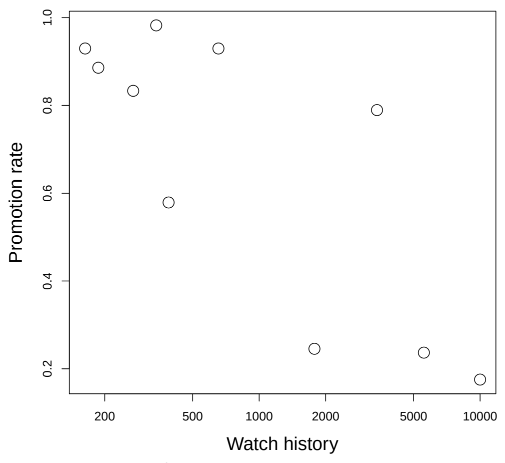


Figure 7: Promotion success rates for 10 real YouTube user accounts with varying watch history lengths.

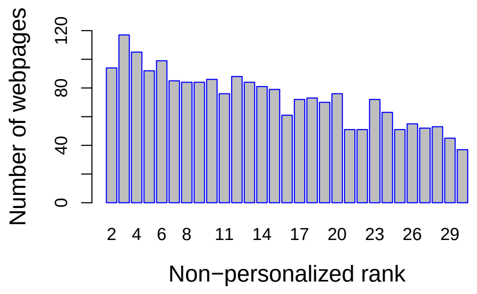
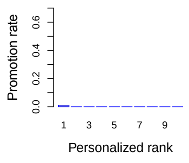
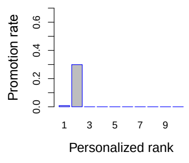


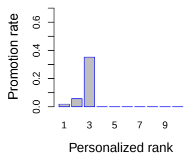
Figure 8: Google's original rank distribution for the 2,136 webpages whose ranking we attempt to improve with contextual search history pollution.



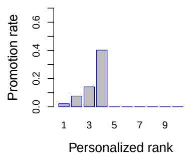
(a) Non-personalized rank = 2



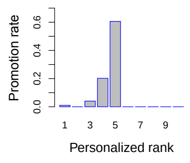
(b) Non-personalized rank = 3



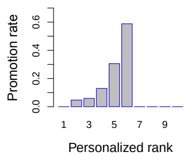
(c) Non-personalized rank = 4



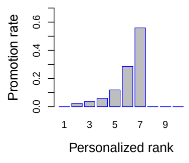
(d) Non-personalized rank = 5



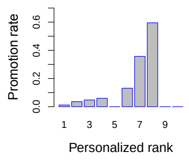
(e) Non-personalized rank = 6



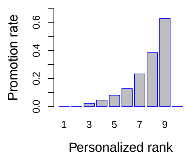
(f) Non-personalized rank = 7



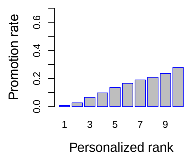
(g) Non-personalized rank = 8



(h) Non-personalized rank = 9

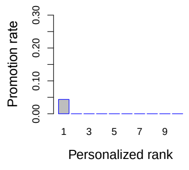


(i) Non-personalized rank = 10

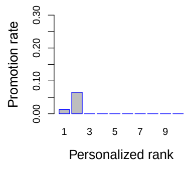


(j) Non-personalized rank > 10

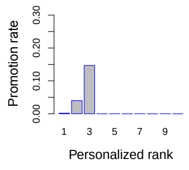
Figure 9: Promotion rates of promoted Google search rankings for successful contextual history pollution attacks.



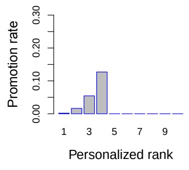
(a) Non-personalized rank = 2



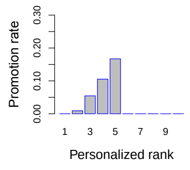
(b) Non-personalized rank = 3



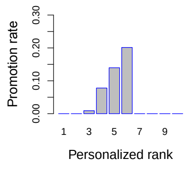
(c) Non-personalized rank = 4



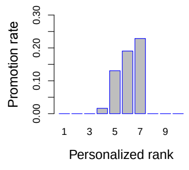
(d) Non-personalized rank = 5



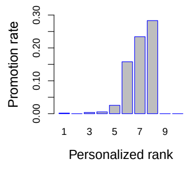
(e) Non-personalized rank = 6



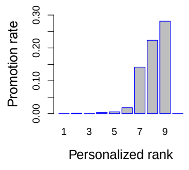
(f) Non-personalized rank = 7



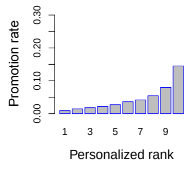
(g) Non-personalized rank = 8



(h) Non-personalized rank = 9



(i) Non-personalized rank = 10



(j) Non-personalized rank > 10

Figure 10: Promotion rates of promoted Google search rankings for successful persistent history pollution attacks.

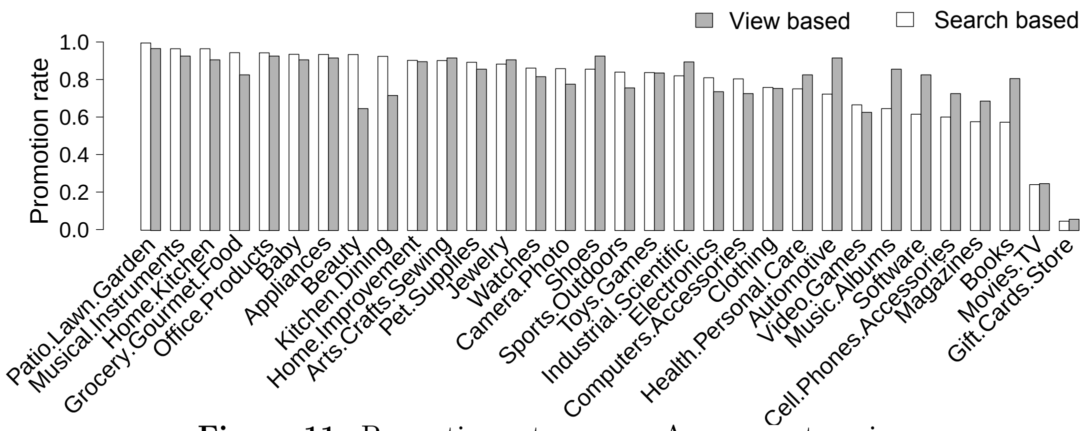


Figure 11: Promotion rates across Amazon categories.

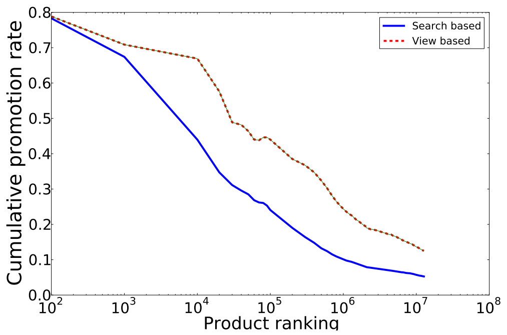


Figure 12: Cumulative promotion rates across varying product ranks for different Amazon pollution attacks.

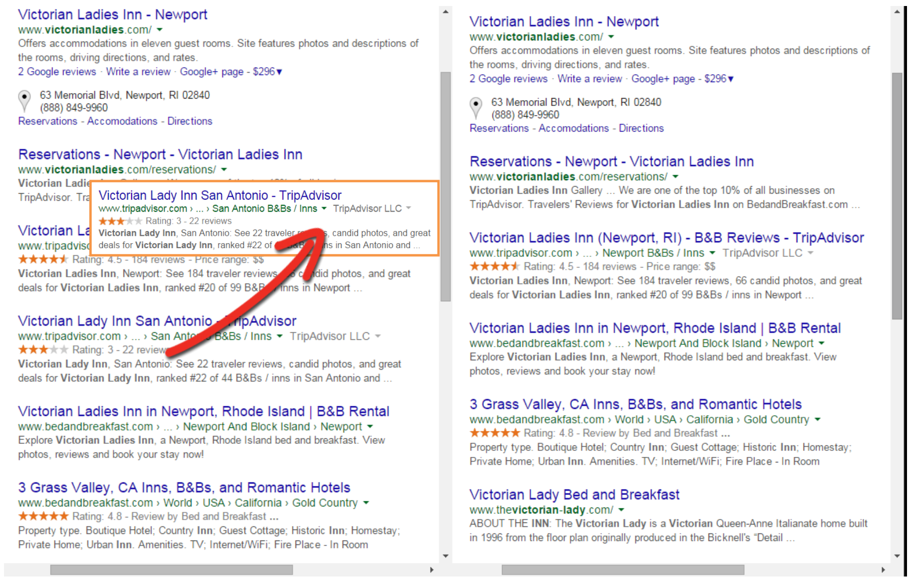
CHAPTER IV

MITIGATING PERSONAL INFORMATION MANIPULATION

Given the ever-expanding user base, personalization is a key weapon for many web services. For example, each search engine seeks to tailor search results not only to the query term, but the end user herself [28, 31, 46, 56, 59, 71, 76, 78, 82, 83], based upon the user's past search history, clicks, geographic location, device type, and other features that may help identify the user's preferences and predispositions [46]. Ideally, personalization identifies results that closely match the user's preferences and intent, and, therefore improve user satisfaction and, ultimately, increase revenue for the service providers.

In practice, web personalization can hide certain important information from users, and distort their perception of content on the web. As is discussed in Chapter 3, personalization services can be exploited by pollution attack to deliver contents intended by attackers [92]. In particular, this new form of attack uses the general form of cross-site scripting attack to contaminate users' profile and alter the customized content in predictable ways. Therefore, we can imagine that an adversary may leverage this attack to impact a user's mind and action. As is shown in Figure 13, inconsistent search results arise when a user searches for "a victorian lady inn" before and after her search history gets contaminated. In this example, we observe that pollution attack results in the negative information about the victorian lady inn to be suppressed from the user's search results. Our intuition suggests this may sway the user's perception concerning this business.

In this chapter, our goal is to put aside aforementioned information prejudices. We


Victorian Ladies Inn - Newport
www.**victorianladies**.com/
Offers accommodations in eleven guest rooms. Site features photos and descriptions of the rooms, driving directions, and rates.
2 Google reviews · Write a review · Google+ page - $296
63 Memorial Blvd, Newport, RI 02840
(888) 849-9960
Reservations - Accomodations - Directions

Reservations - Newport - Victorian Ladies Inn
www.**victorianladies**.com/reservations/
Victorian Ladie
TripAdvisor. Tra

Victorian Lady Inn San Antonio - TripAdvisor
www.tripadvisor.com › ... › San Antonio B&Bs / Inns TripAdvisor LLC
★★★☆☆ Rating: 3 - 22 reviews
Victorian Lady Inn, San Antonio: See 22 traveler , candid photos, and great deals for **Victorian Lady Inn**, ranked #22 of ns in San Antonio and ...

Victorian La
www.tripadvisc
★★★★★ Rating: 4.5 - 184 reviews - Price range: $$
Victorian Ladies Inn, Newport: See 184 traveler reviews, ndid photos, and great deals for **Victorian Ladies Inn**, ranked #20 of 99 B&B inns in Newport ...

Victorian Lady Inn San Antonio - ripAdvisor
www.tripadvisor.com › ... › San Anto B&Bs / Inns TripAdvisor LLC
★★★☆☆ Rating: 3 - 22 review
Victorian Lady Inn, San Antonio: See 22 traveler reviews, candid photos, and great deals for **Victorian Lady Inn**, ranked #22 of 44 B&Bs / inns in San Antonio and ...

Victorian Ladies Inn in Newport, Rhode Island | B&B Rental
www.bedandbreakfast.com › ... › Newport And Block Island › Newport
Explore **Victorian Ladies Inn**, a Newport, Rhode Island bed and breakfast. View photos, reviews and book your stay now!

3 Grass Valley, CA Inns, B&Bs, and Romantic Hotels
www.bedandbreakfast.com › World › USA › California › Gold Country
★★★★★ Rating: 4.8 - Review by Bed and Breakfast **...**
Property type. Boutique Hotel; Country **Inn**; Guest Cottage; Historic **Inn**; Homestay; Private Home; Urban **Inn**. Amenities. TV; Internet/WiFi; Fire Place - In Room

Victorian Ladies Inn - Newport
www.**victorianladies**.com/
Offers accommodations in eleven guest rooms. Site features photos and descriptions of the rooms, driving directions, and rates.
2 Google reviews · Write a review · Google+ page - $296
63 Memorial Blvd, Newport, RI 02840
(888) 849-9960
Reservations - Accomodations - Directions

Reservations - Newport - Victorian Ladies Inn
www.**victorianladies**.com/reservations/
Victorian Ladies Inn Gallery ... We are one of the top 10% of all businesses on TripAdvisor. Travelers' Reviews for **Victorian Ladies Inn** on BedandBreakfast.com ...

Victorian Ladies Inn (Newport, RI) - B&B Reviews - TripAdvisor
www.tripadvisor.com › ... › Newport B&Bs / Inns TripAdvisor LLC
★★★★★ Rating: 4.5 - 184 reviews - Price range: $$
Victorian Ladies Inn, Newport: See 184 traveler reviews, 66 candid photos, and great deals for **Victorian Ladies Inn**, ranked #20 of 99 B&Bs / inns in Newport ...

Victorian Ladies Inn in Newport, Rhode Island | B&B Rental
www.bedandbreakfast.com › ... › Newport And Block Island › Newport
Explore **Victorian Ladies Inn**, a Newport, Rhode Island bed and breakfast. View photos, reviews and book your stay now!

3 Grass Valley, CA Inns, B&Bs, and Romantic Hotels
www.bedandbreakfast.com › World › USA › California › Gold Country
★★★★★ Rating: 4.8 - Review by Bed and Breakfast **...**
Property type. Boutique Hotel; Country **Inn**; Guest Cottage; Historic **Inn**; Homestay; Private Home; Urban **Inn**. Amenities. TV; Internet/WiFi; Fire Place - In Room

Victorian Lady Bed and Breakfast
www.the**victorian-lady**.com/
ABOUT THE **INN**: The **Victorian Lady** is a **Victorian** Queen-Anne Italianate home built in 1996 from the floor plan originally produced in the Bicknell's "Detail ...

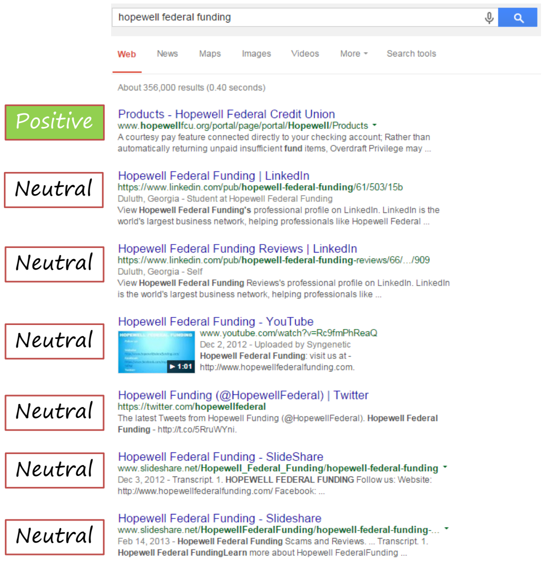
Figure 13: Inconsistent search results before and after pollution attacks.

approach this goal by two different strategies. As pollution attack is typically against individual users and at a high cost when carried out against a large number of users, our first strategy seeks to use a crowdsourcing approach to undermine the influence of pollution attack. In particular, this approach aggregates information from multiple users and presents information with different viewpoints to the victim. Different from the first strategy, our second strategy copes with information prejudices by facilitating users to strip away invalid data items from their profiles. In particular, this strategy leverages a wider variety of personas to identify the data items possibly contaminated and make them visbile to the point where a victim can determine their validity.

We demonstrate the aforementioned strategies on Google personalized search. In particular, we prototyped BobbleX – a client side tool that helps users to undermine the impact of information prejudices. The evaluation of BobbleX shows that it has a false positive rate of 0.02 when it remains its true positive rate at 0.8.

4.1 Defense Strategies

In this section, we describe and discuss the aforementioned strategies in detail.



hopewell federal funding

Web News Maps Images Videos More Search tools

About 356,000 results (0.40 seconds)

Positive
Products - Hopewell Federal Credit Union
www.hopewellfcu.org/portal/page/portal/Hopewell/Products
A courtesy pay feature connected directly to your checking account; Rather than automatically returning unpaid insufficient fund items, Overdraft Privilege may ...

Neutral
Hopewell Federal Funding | LinkedIn
https://www.linkedin.com/pub/hopewell-federal-funding/61/503/15b
Duluth, Georgia - Student at Hopewell Federal Funding
View Hopewell Federal Funding's professional profile on LinkedIn. LinkedIn is the world's largest business network, helping professionals like Hopewell Federal ...

Neutral
Hopewell Federal Funding Reviews | LinkedIn
https://www.linkedin.com/pub/hopewell-federal-funding-reviews/66/.../909
Duluth, Georgia - Self
View Hopewell Federal Funding Reviews's professional profile on LinkedIn. LinkedIn is the world's largest business network, helping professionals like ...

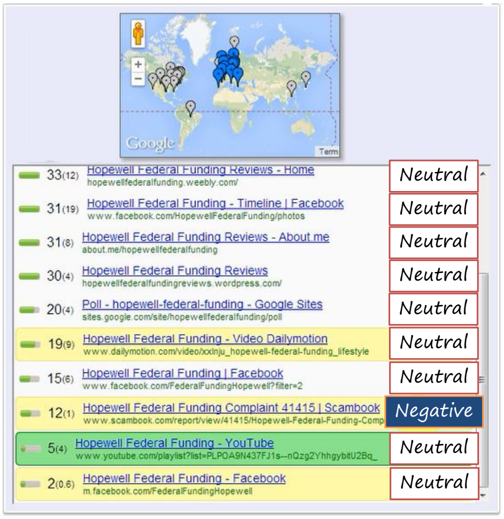
Neutral
Hopewell Federal Funding - YouTube
www.youtube.com/watch?v=Rc9fmPhReaQ
Dec 2, 2012 - Uploaded by Syngenetic
Hopewell Federal Funding: visit us at - http://www.hopewellfederalfunding.com.

Neutral
Hopewell Funding (@HopewellFederal) | Twitter
https://twitter.com/hopewellfederal
The latest Tweets from Hopewell Funding (@HopewellFederal). Hopewell Federal Funding - http://t.co/5RruWYni.

Neutral
Hopewell Federal Funding - SlideShare
www.slideshare.net/Hopewell_Federal_Funding/hopewell-federal-funding
Dec 3, 2012 - Transcript. 1. HOPEWELL FEDERAL FUNDING Follow us: Website: http://www.hopewellfederalfunding.com/ Facebook: ...

Neutral
Hopewell Federal Funding - Slideshare
www.slideshare.net/HopewellFederalFunding/hopewell-federal-funding-...
Feb 14, 2013 - Hopewell Federal Funding Scams and Reviews. ... Transcript. 1. Hopewell Federal FundingLearn more about Hopewell FederalFunding ...

(a) Search results customized to a user.



33(12) Hopewell Federal Funding Reviews - Home
hopewellfederalfunding.weebly.com/
Neutral

31(19) Hopewell Federal Funding - Timeline | Facebook
www.facebook.com/HopewellFederalFunding/photos
Neutral

31(8) Hopewell Federal Funding Reviews - About.me
about.me/hopewellfederalfunding
Neutral

30(4) Hopewell Federal Funding Reviews
hopewellfederalfundingreviews.wordpress.com/
Neutral

20(4) Poll - hopewell-federal-funding - Google Sites
sites.google.com/site/hopewellfederalfunding/poll
Neutral

19(9) Hopewell Federal Funding - Video Dailymotion
www.dailymotion.com/video/xxlnju_hopewell-federal-funding_lifestyle
Neutral

15(6) Hopewell Federal Funding | Facebook
www.facebook.com/FederalFundingHopewell?filter=2
Neutral

12(1) Hopewell Federal Funding Complaint 41415 | Scambook
www.scambook.com/report/view/41415/Hopewell-Federal-Funding-Comp
Negative

5(4) Hopewell Federal Funding - YouTube
www.youtube.com/playlist?list=PLPOA9N437FJ1s--nQzg2YhhgybitU2Bq_
Neutral

2(0.6) Hopewell Federal Funding - Facebook
m.facebook.com/FederalFundingHopewell
Neutral

(b) Search results crowdsourced from others.

Figure 14: Search results with sentiment.

4.1.1 Strategy 1

The first strategy is accomplished by a crowdsourcing approach. When a user submits a query to a personalization service, this approach catches her query and distributes it to other participants. The participants then pass the user's query to the same service along with their own profiles. Due to the difference in profiles, the personalization service tailors and returns different output to the participants who then forward them to the user. After retrieving information from multiple participants, the user obtains not only a set of information tailored to herself but also a set of information crowdsourced. Using an opinion mining technique, the crowdsourcing approach further analyses information in each set and compares viewpoints between the sets. When detecting a significant difference in opinions, this approach raises the user's awareness by showing information with different opinions. To illustrate this viewpoint comparison, Figure 14 shows an example where we attach sentiment to each search result. From Figure 14(a), we can observe that the result set expresses a *mostly neutral and slightly positve* proposition, while the overall sentiment in the another set is *mostly neutral and slightly negative.* In this case, the result with a negative label is absent from the user's results, and therefore, the crowdsourcing approach automatically presents this missing result to the user.

Intuition suggests that the crowdsourcing approach could undermine information prejudices influenced by pollution attack because an attacker would need to contaminate the profiles of many participants in order to bypass this mitigation strategy. However, this approach may exhibit some false positives because personalization algorithms themselves can over-customize a user's output and narrow her perspectives even though no attacker exploits personalization services. In this work, such false positives neither indicate the ineffectiveness of the crowdsourcing approach nor threaten to overwhelem users because the users needs to stay mindful of information prejudices regardless of the attribution behind the prejudices.

4.1.2 Strategy 2

As pollution attack inserts invalid data to user profiles, our second strategy aims to facilitate users to strip away invalid data. It utilizes a data correlation mechanism to identify possibly invalid data in a user's profile and then makes these possibly invalid data visible to the point where the user can easily verify their validity. Given customized output with an extreme viewpoint, in particular, the correlation mechanism cultivates a wider variety of personas, each of which is an account containing a data subset in the user's profile. As each persona consists of similar but not identical data subset, the correlation mechanism can query the personalization service with different personas and obtain different output. By comparing each of these output with the one that the personalization service tailors to the user herself, the correlation mechanism can identify the data items that attribute to the output with the extreme viewpoint. For example, assuming a user observes customized output with an extreme viewpoint, but never seen it in one persona, those disjoint data items should

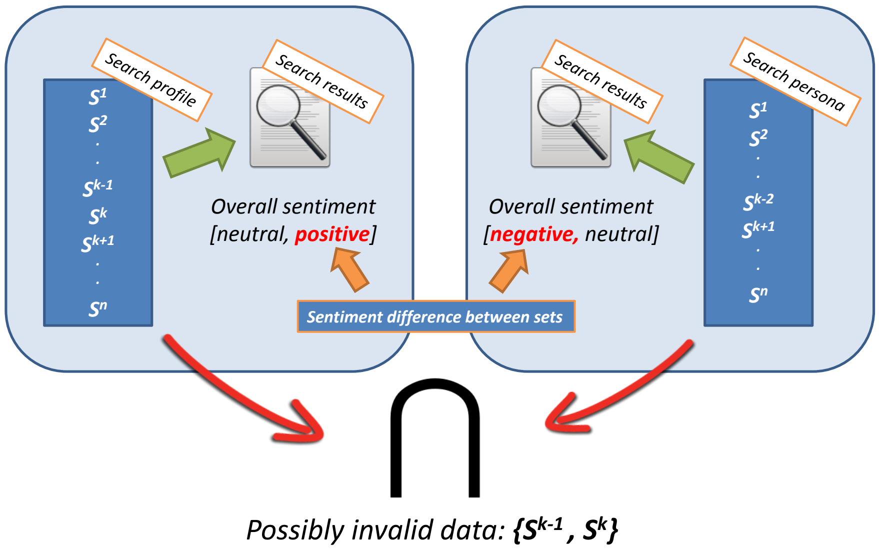

Search profile
Search results
S^1
S^2
S^{k-1}
S^k
S^{k+1}
S^n
Overall sentiment [neutral, positive]
Search results
Search persona
S^1
S^2
S^{k-2}
S^{k+1}
S^n
Overall sentiment [negative, neutral]
Sentiment difference between sets
$\cap$
Possibly invalid data: $\{S^{k-1}, S^k\}$

Figure 15: Identifying possibly invalid data items using a search persona.

hold such output to account. Similar to the first strategy, customized output with an extreme viewpoint may not be necesarily attributed to pollution attack. Instead of correcting one's profile automatically, this strategy therefore presents suspicious data items and leaves the data validation to the end users.

4.2 Prejudice Mitigation on Google Personalized Search

In this section, we demonstrate our prejudice mitigation strategies on Google personalized search. In particular, we build a client side tool – BobbleX which accomplishes the aforementioned two strategies as follows.

To accomplish the first strategy, BobbleX crowdsources search results by distributing a user's query to other participants, who then use their own search profiles (i.e., search accounts) to perform the same search and return their results to the user. Then, BobbleX uses a sentiment analysis approach to assign sentiments to search results. These results include the user's own results as well as the corresponding results crowdsourced. Based on the sentiments assigned, BobbleX compares the overall viewpoints in the two result sets and presents to the user the results with different sentiments. In particular, BobbleX intersects distinct sentiments included in each set and outputs the results with disjoint sentiments. Take Figure 14 for example again. In this specific case, BobbleX outputs the result with the negative sentiment because it is absent in the user's result set where the distinct sentiments include *neutral* and *positive*.

BobbleX accomplishes the second strategy by first assigning sentiment to the results in the set tailored to the user. Then, it examines the overall sentiment of that set. If the results in that set are not distributed across the entire spectrums of sentiment, it means the results express an extreme viewpoint and may sway the user's perception. Thus, BobbleX further performs the same search using predetermined personas, and compares the overall sentiment of each result set with that of the user's result set. When detecting difference in sentiment between two sets, BobbleX intersects the data items in the search persona and profile, and eventually outputs disjoint data items. The disjoint data items are responsible for the extreme viewpoint presented to the user and might be inserted by attackers for influencing the user's decision. As a result, BobbleX presents these disjoint data items to the user who can then examine and even discard them if they are indeed invalid. Figure 15 illustrates an example of identifying possibly invalid data in a user's profile by using a search persona.

4.3 Analysing Sentiment in Search Results

As is discussed in the section above, both of the prejudice mitigation strategies rely on sentiment analysis to assign sentiment to search results. In this section, we introduce a sentiment analysis approach. In particular, I begin with describing the challenges of sentiment analysis in the context of prejudice mitgiation. Following that, we then propose a sentiment analysis approach.

4.3.1 Technical Challenges

Since users usually navigate to landing pages quickly from their search pages, BobbleX needs to examine prejudices in search results in advance of their navigation. Or put it in another way, BobbleX needs to perform sentiment analysis efficiently.

In general, the sentiment of a search result may lay beneath its landing page. To pull it to the surface, a sentiment analysis approach needs to examine the landing pages of search results, and then apply sentiment-based classification to catgorize the pages into *positive, neutral* or *negative.*

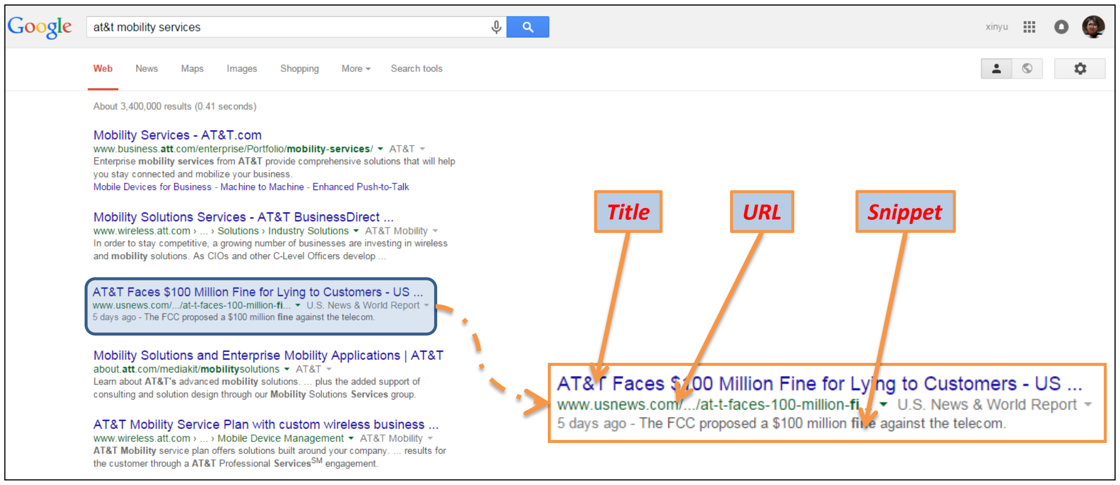
Many prior classification approaches can be used to categorize pages into sentiment [32, 34]. However, they are typically time intensive and not applicable to information prejudice mitigation. Given a page, the prior approaches refine information from its neighboring pages, which usually follows a long spell.

4.3.2 Sentiment Analysis

Considering the challenge mentioned above, we propose a sentiment-based classification approach that does not rely on any neighboring pages. Instead, our approach works as follows. First, it distills informative text from search results and their landing pages. Based on the informative text distilled, it then extracts and selects features. Using these features, our approach utilizes a supervised machine learning method to build classifiers for sentiment categorization.

4.3.2.1 Distilling Informative Text

As is shown in Figure 16, a search result consists mainly of three segments - a page title, a landing page URL and a snippet. The generation of the page title and snippet is completely automated and takes into account the content of the landing page. Therefore, they represent not only the summary of a landing page but sometimes reflect the proposition hidden behind the landing page. For example, the highlighted search result in Figure 16 can quickly lead users to draw a negative impression on



About 3,400,000 results (0.41 seconds)

Mobility Services - AT&T.com
www.business.**att**.com/enterprise/Portfolio/**mobility-services**/ AT&T
Enterprise **mobility services** from **AT&T** provide comprehensive solutions that will help you stay connected and mobilize your business.
Mobile Devices for Business - Machine to Machine - Enhanced Push-to-Talk

Mobility Solutions Services - AT&T BusinessDirect ...
www.wireless.att.com › ... › Solutions › Industry Solutions AT&T Mobility
In order to stay competitive, a growing number of businesses are investing in wireless and **mobility** solutions. As CIOs and other C-Level Officers develop ...

AT&T Faces $100 Million Fine for Lying to Customers - US ...
www.usnews.com/.../at-t-faces-100-million-**fi**... U.S. News & World Report
5 days ago - The FCC proposed a $100 million **fine** against the telecom.

Mobility Solutions and Enterprise Mobility Applications | AT&T
about.**att**.com/mediakit/**mobility**solutions AT&T
Learn about **AT&T's** advanced **mobility** solutions. ... plus the added support of consulting and solution design through our **Mobility** Solutions **Services** group.

AT&T Mobility Service Plan with custom wireless business ...
www.wireless.att.com › ... › Mobile Device Management AT&T Mobility
AT&T Mobility service plan offers solutions built around your company. ... results for the customer through a **AT&T** Professional **Services**^{SM} engagement.

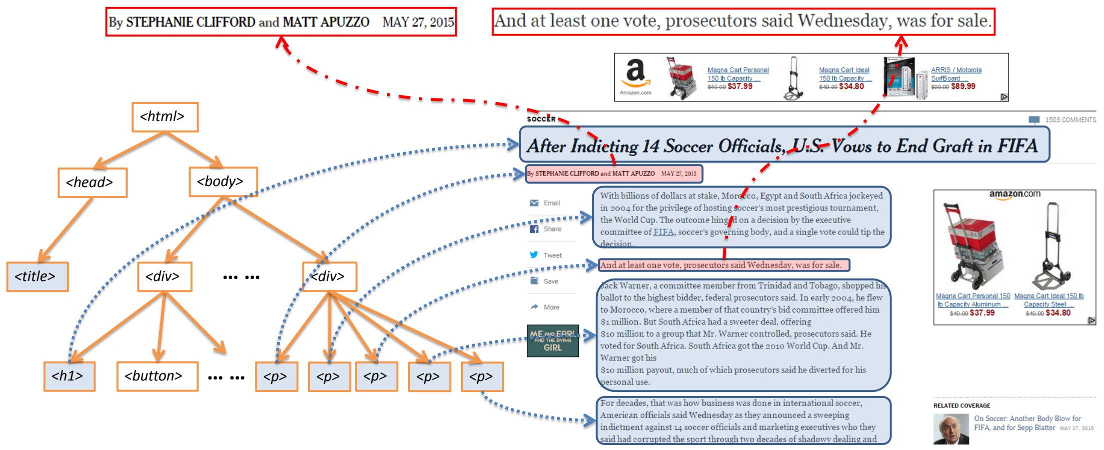
AT&T Faces $100 Million Fine for Lying to Customers - US ...
www.usnews.com/.../at-t-faces-100-million-**fi**... U.S. News & World Report
5 days ago - The FCC proposed a $100 million **fine** against the telecom.

Figure 16: The search results that may leave negative impressions to searchers.

AT&T. In this work, we therefore distill text from both the page title and snippet in a search result. In addition, the domain name shown in a landing page URL occasionally reflects the sentiment of a search result. For example, the search result of a commercial product from `scambook.com` or `ripoffreport.com` can leave a negative impression to its potential customers. As a result, we also extract domain names from landing page URLs.

To categorize search results, we further examine and extract content from landing pages because they carry the opinions that page editors intend to convey. Considering most landing pages are structured in an HTML format, where useful information is often accompanied by a large amount of noise such as banner advertisement or navigation bars, we develop the following approach that discards noisy information items and only extracts informative content from landing pages.

In general, the landing page of a search result is structured in either a PDF or an HTML format. Empirically, PDF-formatted landing pages do not contain aforementioned nosiy information. As a result, we retrieve text in PDF-formatted landing pages without refining. In particular, we retrieve text by using PDF extractor *slate 0.4* [77]. For any PDF page formatted as an image, we utilize an OCR (optical character recognition) technique [61] to convert it into editable docuemtn and then


<html>
<head>
<body>
<title>
<div>
... ...
<div>
<h1>
<button>
... ...
<p>
<p>
<p>
<p>
<p>
Amazon.com
Magna Cart Personal 150 lb Capacity ... $40.00 $37.99
Magna Cart Ideal 150 lb Capacity ... $40.00 $34.80
ARRIS / Motorola SurfBoard ... $90.00 $89.99
SOCCER
1503 COMMENTS
Email
Share
Tweet
Save
More
ME AND EARL AND THE DYING GIRL
amazon.com
Magna Cart Personal 150 lb Capacity Aluminum ... $40.00 $37.99
Magna Cart Ideal 150 lb Capacity Steel ... $40.00 $34.80
RELATED COVERAGE
On Soccer: Another Body Blow for FIFA, and for Sepp Blatter MAY 27, 2015

By STEPHANIE CLIFFORD and MATT APUZZO MAY 27, 2015

And at least one vote, prosecutors said Wednesday, was for sale.

After Indicting 14 Soccer Officials, U.S. Vows to End Graft in FIFA

By STEPHANIE CLIFFORD and MATT APUZZO MAY 27, 2015

With billions of dollars at stake, Morocco, Egypt and South Africa jockeyed in 2004 for the privilege of hosting soccer's most prestigious tournament, the World Cup. The outcome hinged on a decision by the executive committee of FIFA, soccer's governing body, and a single vote could tip the decision.

And at least one vote, prosecutors said Wednesday, was for sale.

Jack Warner, a committee member from Trinidad and Tobago, shopped his ballot to the highest bidder, federal prosecutors said. In early 2004, he flew to Morocco, where a member of that country's bid committee offered him $1 million. But South Africa had a sweeter deal, offering
$10 million to a group that Mr. Warner controlled, prosecutors said. He voted for South Africa. South Africa got the 2010 World Cup. And Mr. Warner got his
$10 million payout, much of which prosecutors said he diverted for his personal use.

For decades, that was how business was done in international soccer, American officials said Wednesday as they announced a sweeping indictment against 14 soccer officials and marketing executives who they said had corrupted the sport through two decades of shadowy dealing and

Figure 17: Informative content blocks and their corresponding HTML tags.

retrieve text from it.

For landing pages in HTML, we refine their content by following a heuristic approach. A landing page in HTML corresponds to a DOM tree where tags are internal nodes and detailed text, images or hyperlinks are the leaf nodes. Given an HTML-formatted landing page, there is no straighforward approach that can get rid of internal nodes representing noisy information items. Fortunately, we observe that noisy information items - such as navigation bars or advertisement banners - are usually structured in *<div>* and *<span>* tags while informative items typically correspond to the following tags.

- *<p>* tag defines a paragraph of an article;
- *<h1>*$\cdots$ *<h6>* tags define the headings of an article, where *<h1>* defines the most important heading and *<h6>* defines the least important one;
- *<title>* tag is required in all HTML pages and defines the title of an article;
- *<meta name="description">* tag specifies a short description of an HTML page.

We therefore extract text from these tags. Figure 17 shows a webpage where the information corresponding to these tags are highlighted. As we can observe in the figure, some noisy information items (e.g., the navigation bar and advertisement banners) are successfully excluded.

Following the information extraction above, we could obtain the text with informative content. However, the extracted text still contain some noisy information which may hamper sentiment analysis. In Figure 17, we discover some hightlighted text (such as the editing information about the article) are not useful for sentiment analysis. To eliminate such text, we futher use a statistical language model [72] which outputs a probability for each text block base on how likely that text block is to be a sentence in English. In particular, we set up a probability threshold to further get rid of less informative content. Taking the two information blocks highlighted in Figure 17, for example, a statistical language model outputs a higher probability for information block "*And at least on vote, prosecutor said Wednesday, was for sale*" than information block "*By STEPHANIE CLIFFORD and MATT APUZZO May 27, 2015*". Using the predetermined threshold, we can easily exclude the second information block.

Using the aforementioned information extraction approach, we can accurately distill informative text from most landing pages. However, we discover such an approach sometimes is not effective for some pages whose editors use other HTML tags to lay out their informative content. Rather than using *<p>* tags to structure paragraphs of an article, for example, an HTML page editor could utilize *<table>* and even *<div>* tags to lay out the article. So, we further examine and retrieve the detailed text under these tags.

HTML tags like *<div>* can be used for structuring noisy and informative text blocks. To filter out informative text block from these tags, we extend our aforementioned heuristic approach. In particular, we compare the class attributes in these tags with those in tags that we have already selected as informative. Using the aforemented heuristic approach, for example, we have already extracted informative text from tag *<p class="story-body-text story-content"> ... </p>*. We can extract its class attribute "*story-body-text story-content*" and search similar strings in other tags[1]. The reason is that page editors usually use *class attribute* to describe the type of an information block and we can use their descriptions about informative text blocks to find more informative text under other HTML tags. For example, string *story-body-text story-content* indicates that the text under this *<p>* tag is the body text of a story which includes the opinion of the story and can be useful for sentiment analysis. So, a similar string in other HTML tag indicates the text under the tag is also useful for sentiment analysis, e.g., text in tag *<div class="body-text">...</div>*.

4.3.2.2 Extracting Features

Having the informative text from search results and corresponding landing pages, we now explore the features that are likely to have a positive impact on sentiment analysis.

Given a sentiment-based classification problem. one instinctive solution is to apply existing text categorization approaches [85, 94, 95] which represent documents – in our case, search results and their landing pages – as a native feature space consisting of the unique words that occur in documents. As we will show in Section 5.3, this solution cannot provide us with optimal classification performance because the words that occur in different parts of search results and landing pages may contribute differently to classification. Consider word "complaint" appearing in the domain domain in one URL (e.g., `http://www.complaintsboard.com/complaints/macy039s-furniture-and-bedding-c56442.html`), and in the text of a landing page (e.g., *... Questions? Comments? Complaints? We'll quickly get back to you with the information you need.*). In the first case, we know the site includes consumer news and database of rated complaints submitted by users. So, the content from the site

[1]In this work, we compute the similarity of two strings using the Levenshtein distance [65].

is highly likely to discuss a disreputable business and any search results linking to the site are likely to leave negative impressions to searchers. In the second case, the landing page is for customer services. Though it contains a word with a negative sentiment, the landing page does not convey a negative message to searchers when they visit the page. As a result, distinguishing where words occur may improve the accuracy of sentiment analysis.

We create word features that indicate which part the word comes from. In particular, we begin with filtering out stop words – the most common words in a language such as *the, to, that, etc.* Then, we take each of the remaining words from informative text and attach its corresponding HTML tag because the combination of word and its HTML tag indicates the origin of the word. Given a <p> tag with informative text *"And at least one vote, prosecutor said, was for sale"*, for example, we extract word features including *<p>least, <p>vote, <p>prosecutor, <p>sale.* For search results (that we do not extract informative text based on their HTML tags), we attach tag *<search-title>* and *<search-snippet>* to the correpsonding segments of each search result. Finally, we extract the domain name of each URL, partition it into meaningful word units and attach tag *<url>* to each word unit. For example, URL `http://www.complaintsboard.com/complaints/park-west-gallery-c68078.html` can be segmented into feature *<url>complaints* and *<url>board.*

4.3.2.3 Feature Selection and Classification

Using word features extracted above, we can create a feature space with tens or hundreds of thousands of word and HTML tag combinations. According to a prior study [96], such a high dimensionality of feature space can hinder many learning algorithms. Therefore, we use a term frequency-inverse document frequency (tf-idf) to reduce the dimenionality of our feature space. In particular, we compute term frequency (tf) and inverse document frequency (idf) for each unique aforementioned word feature in our training corpus and remove from the feature space those word features whose tf and idf product is less than some predetermined threshold. We represent each search result using the reduced feature space and then apply an Support Vector Machine (SVM) learning algorithm to train a classification model. In Section 5.3, we will provide detailed evaluation for our sentiment-based classification approach.

4.4 Generating Personas

As is discussed in Section 4.2, one of our prejudice mitigation strategies uses predetermined personas to identify possibly invalid data in one's search profile. In this section, we discuss the technical challenges of persona generation, and then describe a practical approach to address these challenges.

4.4.1 Technical Challenges

A search persona is a subset of the past searches in a user's search profile. Ideally, it needs to be generated in a dynamic way. For example, BobbleX detects prejudices in search results when the user performs a search. To identify the past searches attributed to the bias, one solution is to build a persona "petri dish" – by duplicating all the searches in the user's search profile except searches that might be relevant to the user's outcome – and then cultivate new personas from it.

However, such an approach is not practical. According to our observation and prior studies [29, 49], search personalization relies on many signals to tailor one's search results. Some important signals include search terms, user clicks, geographic region and timing. When duplicating searches from one's profile, a persona generation approach can easily copy past search terms and maybe user clicks by repeating the searches and clicking their results in a new search account. However, this process follows a long spell because the data needed in a persona should be dynamically determined and generating one persona needs to duplicate a sheer number of searches.

In addition, generating a persona in this dynamic way neglects the timing factor that search personalization relies on, which jeopardizes the effectiveness of identifying invalid data.

4.4.2 Persona Generation

To address the challenge above, we introduce a practical approach that builds personas by sacrificing the dynamics of persona generation.

Pollution attack inserts invalid searches to a victim's search profile through cross-site scripting. One crucial characteristic of such searches is that they are performed through third party websites. As a result, we can identify possibly invalid searches by building personas in the following ways.

Our persona generation approach first creates multiple accounts and synchronizes the user's search activities whenever performed. In this way, we obtain multiple copies of a user's profile. When identifying possibly invalid data in the user's profile, our persona generation approach pinpoints those searches performed through third party websites, and creates all possible personas by removing them accordingly. Given a search profile ($T \in \{s_1, s_2, \cdots, s_i, \cdots, s_j, \cdots\}$) containing 2 searches (s_i and s_j) performed on third party websites, for example, our approach creates three possible personas including $T - \{s_i\}$, $T - \{s_j\}$ and $T - \{s_i, s_j\}$. Using this approach, we successfully create personas without ignoring the timing factor because the persona generation follows search activity synchronization and discards unwanted searches accordingly.

It is worth of noticing that the number of profile copies needed depends on the amount of searches performed through third party websites. That would suggest an exponential number of copies. As we will show later in Section 5.3, however, users seldom perform searches through third party websites in their everyday lives, which indicates personas are needed only when pollution attack happens. Considering pollution attack has not been pervasive on the Internet, we only need to create a fixed amount of personas in practice.

4.5 Implementation

In this section, we present the design and implementation of our BobbleX as follows.

When a user executes a search query for another with his own profile (i.e., Google history), Google search adds that query to his profile. As adding query in such a way incurs profile contamination, we design BobbleX in a non-intrusive manner. By non-intrusiveness, we meant BobbleX does not incur any changes to one's search profile when performing searches for others. To accomplish this, we modify PhantomJS [70] - a scripted web browser that can automate web page interactions. In particular, we augment PhantomJS with the ability to automatically access one's Google search profile and pause Google logging when BobbleX needs to perform searches for others.

BobbleX needs to distribute a user's searches to other partcipants, who then use their own Google search accounts to perform the same searches and return their results to the user. To do this, we develop a Chrome browser extension. It passively collects search queries and forwards them to a central server. The central server then coordinates other participants to execute the queries contemporaneously.

To analyse search results and their landing pages, we implement BobbleX by further extending PhantomJS. In particular, our extended PhantomJS fetches landing pages and extracts necessary information in the background. Last but not the least, we implement persona generation by augmenting PhantomJS with an ability to selectively duplicate a user's searches.

4.6 Evaluation

The effectiveness of BobbleX relies upon two parts - the sentiment analysis approach and persona generation. In this section, we evaluate BobbleX by exploring the performance of our proposed sentiment analysis approach and the feasibility of our persona generation approach.

4.6.1 Data Set

We use two different data sets to evaluate BobbleX. One data set is used for evaluating sentiment analysis, while the other is for examining the feasibility of our persona generation.

To evaluate our sentiment analysis approach, we randomly sampled 1,350 Google searches in the category of business from a large dataset collected by Bobble [91]. From a crowdsourced platform – Mechanic Turk [62], we recruit volunteers to label the results of these searches. In particular, each search result is manually categorized into *positive, neutral* or *negative* based on their sentiment. Eventually, we obtain a data set including 15% results with a negative opinion, 5% results with a positve opinion and the rest with a neutral opinion. We evenly splitted this data set into two subsets, one half of them for classifier trainning and the rest for its testing.

To evaluate our persona generation approach, we collected 1,259,949 searches from 6,482 Google users and recorded how these searches are performed. From a public available seach engine NerdyData [66], we also obtained a webpage corpus including 40,144,820 webpages.

4.6.2 Performance of Sentiment Analysis

We evaluate our sentiment-based classification approach by comparing it with a few other classification approaches. We apply these approaches to categorize search results with a negative viewpoint, and show their performance in Figure 18. As is shown in the figure, we present their performance in a Receiver Operating Characteristic (ROC) plot. We observe our sentiment-based classification approach has an extremely low false positive rate (approximately 0.02) when it remains a true positive rate of 0.8. This low false positive indicates that our BobbleX can accurately detect search prejudice. In other words, users can rely on BobbleX to identify the biases hidden

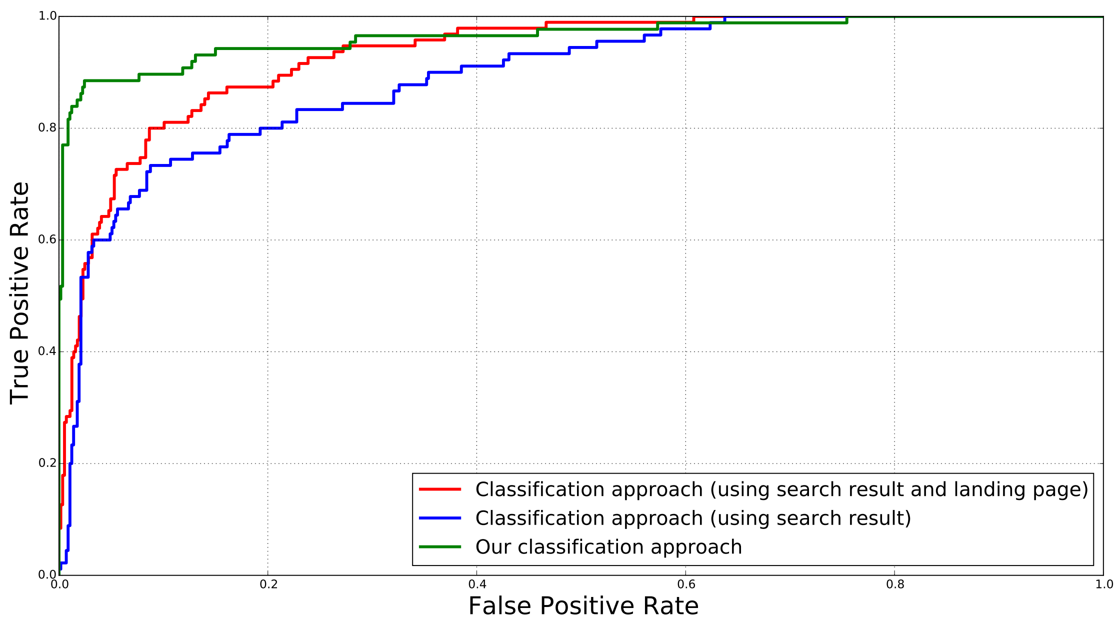
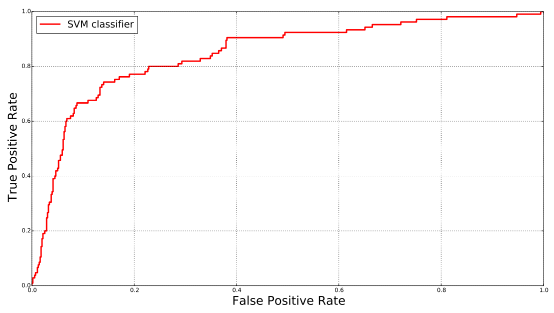


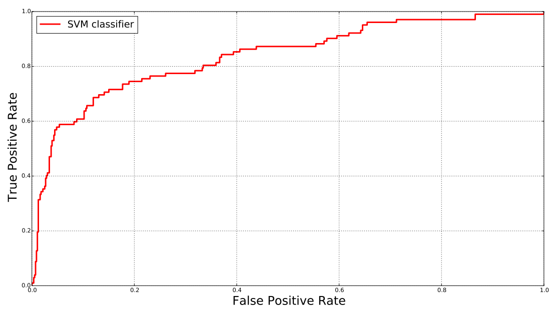
Figure 18: The ROC plot of three classification approaches.

behind his or her search results.

In addition, Figure 18 shows the classification performance of two other approaches. In one approach, we build a baseline classifer. It uses the text in search results only. As is shown in the figure, this baseline approach has lower classifcation performance in terms of false positive and true positive rates. The reason behind this low performance is simply that the text in search results alone only provide limited distinguishable information for sentiment classification. To further verify this argument, we rebuild this baseline classifer. In particular, the improved classifer uses both the text in search results and the text in their landing pages. As is shown in Figure 18, we observe that this improved classifer shows better classification performance than the baseline classifer. This further indicates that the information in landing pages can be used for boosting classification performance. However, we also observe that our sentiment-based classification approach outperforms this improved classifier. This indicates that information in landing pages are typically noisy and removing information nosie from landing pages can significantly improve the performance in sentiment



(a) Classifier for positive sentiment analysis.



(b) Classifier for neutral sentiment analysis.

Figure 19: The ROC plots of other binary classifiers.

classification.

As is mentioned in Section 4.3, we use three binary classifiers to categorize search results into *positve, neutral* or *negative.* As a result, we also evaluate our sentiment analysis approach per classifier. Figure 19 shows the classification performance of the other two binary classifiers. Comparing with the classification performance for negative sentiment analysis shown in Figure 18, we observe the other two classifers perform in lower performance. One reason is that search results with a negative opinion contain higly distinguishable words which we can use to generate distinguishable features and benefit this classifer for negative opinion categorization. Another reason is that the search results with neutral and positive opinions share many common words and we cannot select highly distinguishable features from them for classifying search results in these two opinions.

4.6.3 Feasibility of Persona Generation

As is discussed in Section 4.4, the effectiveness of persona generation relies on the number of searches made through third party websites. Intuition suggests that our persona generation approach may not be a feasible solution if users typically perform many searches through third party websites. So, we evaluate our persona generation by examining how frequent a user performs searches on third party websites and how many webpages allow a user to perform searches through them.

We answer these two questions by using the aforementioned large-scale data set. Our measurement shows only 0.0367% (462 out of 1,259,949) searches are performed through third party websites and only 0.1068% (42,888 out of 40,144,820) webpages allow users to perform searches. These two percentages in low values indicate that users rarely perform searches through third party websites and our persona generation approach does not need to create many personas when pollution attack does not happen.

4.7 Conclusion

In this chapter, we introduce two mitigation strategies that undermine the impact of information prejudices. We show that an effective and efficient sentiment analysis technique can facilitate information prejudice mitigation. By integtrating it to our crowdsourcing and persona correlation strategies, the impact of information prejudices can be significantly undermined. Though we demonstrate our mitigation strategies on Google personalization search, they can be applied to a variety of personalization services. Our proposed mitigation strategies do not require server side modification and can be implemented as client-side software.

CHAPTER V

MITIGATING DIRECT INFORMATION MANIPULATION

Browser extensions (or “add-ons”) have became a significant driving force behind web browsers, often offering much needed missing functionalities to their users. Extensions are available for all modern browsers, including Chrome, Firefox, Internet Explorer, Safari and Opera, and are now beginning to appear in mobile browsers [9]. Extension “stores” usually contain tens of thousands of different offerings, with some extensions reaching a user base in the millions.

To monetize their work, extension developers may decide to display ads to their users, thus producing revenue through ad impressions and clicks. Naturally, miscreants have found ways to heavily abuse this practice. For example, a “shady” advertisement network may attempt to purchase a popular extension from its developers. Once the extension has been acquired, a (silent) update containing ad-injection code can be pushed to the extension users, thus inflating (often fraudulently) the ad revenue for the ad network [4].

One of the main challenges is that some of these extensions try to operate in the “grey area” of extension development policies, to reduce the risk of being removed from the extension stores while increasing their revenues. This seems to be one of the driving factors that urged Google to update their Chrome extension policies in December 2013, making them more strict with regards to ads [14]. According to [6], developers are not allowed to build a Chrome extension “that requires users to accept bundles of unrelated functionality...for example, display product ratings and reviews, but also injects ads into web pages”. This policy implies that the ad-injection functionality must be implemented as a separate extension, enabling users to clearly identify the developer's intention and to install that functionality only with explicit consent.

Unfortunately, with tens of thousands of extensions and a constant stream of updates, enforcing such policies without introducing a significant delay is challenging. Currently, extension stores (e.g., Google's Chrome extensions store) set a low bar for introducing new extensions, and for providing updates to existing extensions. Instead, they adopt a crowdsourced approach, relying on their users to identify misbehaving extensions – users can rate extensions, review them, and report abusive extensions. For example, some of these reports recently led to the removal of extensions from the Chrome store [13]. Similarly, there exist other third-party websites (e.g., Extension Defender [8]) that keep track of the "reputation" of browser extensions, again relying on users to report different types of misbehaviors.

These crowdsourced manual efforts are slow and limited, and react to violations only *after* they have already affected many users. Furthermore, this process does not scale well, as extension stores grow and are expected to expand to mobile browsers [5]. Therefore, the implications of misbehaving extensions with respect to user trust, privacy and security may become even more severe in the near future.

An alternative solution employed by extension stores, e.g., Firefox add-ons [11], is a rigorous process in which editors manually inspect various aspects of the extension. The problem is that this process is extremely slow (Firefox addon approval process can take up to 10 days [3]), and obviously it does not scale with the increase in the number of extensions. Furthermore, as we later show, we identified ad-injecting Chrome extensions that appear also in the FireFox add-on store, indicating that even this rigorous process is not sufficient.

To address these issues, we introduce `Expector` (Extension Inspector), a system that automatically inspects browser extensions looking for signs of ad injection capabilities. Our main goal is to quickly sift through thousands of extensions to identify those that are most likely in violation of the advertisement practices policies set by browser vendors. `Expector` operates using both static and dynamic analysis techniques. It first identifies domains and events that trigger the ad injections, and then compares the DOM structure resulting from visiting websites with and without the analyzed extension. It identifies DOM elements introduced or altered by the extension that is suspicious as containing ads and examines network traffic to identify patterns associated with ad networks.

Unlike the current manual efforts, `Expector` is highly scalable and operates without human intervention. By integrating `Expector` in the extension approval process, a browser vendor can better *prioritize* further analysis for extensions that are suspected to violate the policies. In turns, this can help to significantly improve the quality of their extensions, and to potentially prevent abusive extensions from ever reaching the users.

The main contributions of this chapter are as follows:

- We present `Expector`, a novel system that automatically identifies browser extensions capable of injecting ads on webpages. `Expector` works by installing an *instrumented* extension within a virtual browser. By correlating the network traffic generated by the browser with changes that the extension performs over the DOM elements of various HTML pages, `Expector` is able to identify various forms of ad injection.

- Our current implementation of `Expector` enables the automatic analysis of Chrome extensions. We ran `Expector` on all extensions (almost 18,000) available on the Chrome extension store, and identified 292 extensions that embed ad-injection capabilities, installed and used by millions of users. `Expector` is accurate, with low 3.6% false-positives and 3% false-negatives.

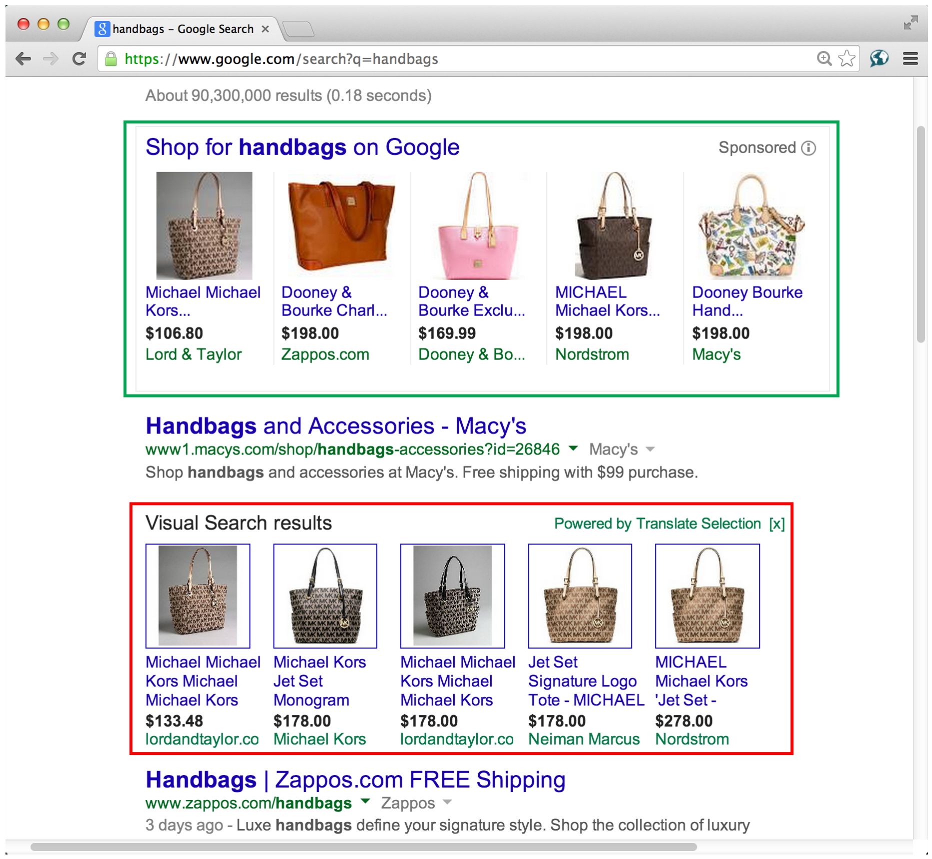
- We present a detailed study of the ad-injection ecosystem from browser extensions. Specifically, we select 223 very popular extensions from those identified by `Expector`, and detail their behavior, malpractices, and revenue structure. We show that by participating in various fraudulent activities, the ad networks can generate large revenues, in par with many legitimate ad networks.

The reminder of this chapter is structured as follows. In Section 5.1 we provide essential background and terminology. We detail the design of `Expector` in Section 5.2, and show the results from applying it to the Google Chrome extension store in Section 5.3. We then perform a detailed case study for a few popular extensions identified by `Expector` in Section 5.4, discuss potential future improvements in Section 5.5, related work in Section 5.6 and conclude the chapter in Section 5.7.

5.1 Background and Motivation

Browser extensions are programs that enhance the functionalities of the browser. These programs are written in HTML, JavaScript and CSS and are hosted in online stores like the Chrome web store [12] and Mozilla Ad-Ons store [11]. Extensions interact with the current webpage loaded in the browser by injecting JavaScript to read or modify the DOM structure of the webpage, communicating with external servers outside of the origin using `XMLHttpRequest` objects, and leveraging the browser APIs and features. The permissions requested by the extension are listed in a manifest file that is reviewed by the user at installation time. As is evident from this description, browser extensions hold significant privileges which leaves the door open to malpractices with security and privacy risks.

Browser extension developers often user advertising as way to monetize the "free" extension provided on the web store. While this practice is legitimate, there has been a growing number of incidents where extension developers abuse the privileges and the policies of the online stores by injecting ads on webpages that users browse. These


handbags – Google Search

https://www.google.com/search?q=handbags

About 90,300,000 results (0.18 seconds)

Shop for **handbags** on Google — Sponsored

Michael Michael Kors...	Dooney & Bourke Charl...	Dooney & Bourke Exclu...	MICHAEL Michael Kors...	Dooney Bourke Hand...
$106.80	$198.00	$169.99	$198.00	$198.00
Lord & Taylor	Zappos.com	Dooney & Bo...	Nordstrom	Macy's

Handbags and Accessories - Macy's
www1.macys.com/shop/**handbags**-accessories?id=26846 Macy's
Shop **handbags** and accessories at Macy's. Free shipping with $99 purchase.

Visual Search results — Powered by Translate Selection [x]

Michael Michael Kors Michael Michael Kors	Michael Kors Jet Set Monogram	Michael Michael Kors Michael Michael Kors	Jet Set Signature Logo Tote - MICHAEL	MICHAEL Michael Kors 'Jet Set -
$133.48	$178.00	$178.00	$178.00	$278.00
lordandtaylor.co	Michael Kors	lordandtaylor.co	Neiman Marcus	Nordstrom

Handbags | Zappos.com FREE Shipping
www.zappos.com/**handbags** Zappos
3 days ago - Luxe **handbags** define your signature style. Shop the collection of luxury

Figure 20: A Google search injected with ads from "Translate Selection" Chrome extension.

practices impact end-users (popup ads, intrusive and inappropriate ads), publishers (decrease in ad revenue), and other ad-networks (ads replaced with other ads). In the rest of the section we describe the three primary type of ad injection malpractices we observed through our study of Chrome browser extensions.

5.1.1 Ad Injection Practices

Ad Injection on Search Result Pages. Consider the screenshot in Figure 20, showing the result page of a Google search for the query "bags". The user has installed the "Translate Selection" Chrome extension whose primary purpose is to help users quickly translate the selected text between different languages. However, this extension bundles with it the functionality to inject ads in the search results returned to the user. The top highlighted region in Figure 20 shows the standard Google sponsored ads. The bottom highlighted region shows another set of ads of bags with links to online stores. In this case, the extension clearly violates the Chrome extension policy as it bundles its core functionality with injecting – the Chrome policy states that “if two pieces of functionality are clearly separate, they should be put into two different extensions”. Furthermore, the organic search results returned by Google are pushed further down the screen, and potentially not visible all together on smaller screen sizes.

Ad Injection on Retail Websites. Another type of ad injection practice is related to injecting ads on webpages related to online retail (amazon.com, ebay.com, etc.). When a potential shopper browses a product on an online retail website, the extension sends the context of a browsing session to a third-party advertisement service, which retrieves similar or related products. These products are shown in the form of an ad to the user, overlayed on the existing website content. As we shall see in Section 5.3, these ads are injected by extensions that are not related to online shopping (e.g. News and Weather, Sports, Developer Tools, etc.) which again violates the above stated Google Chrome extension policy. Additionally, these extensions potentially also impact the revenue of the online retailer by showing users ads from competing or similar products.

Ad Injection on Unrelated Websites. The third type of ad injection practice consists of extensions that aggressively insert ads on almost every webpage that the user browses. This practice can often degrade the user’s experience by having ads pop up or annoying ads. Furthermore, it can potentially reduce the ad revenue of publishers as users may end up clicking on the injected ads or the injected ads may be overlaid on top of the publisher’s ad slots.

5.1.2 Ad Injection JavaScript Libraries

It is relatively straightforward for extension developers to monetize their extension through ad injection. Similar to existing ad networks and ad exchanges, there exists a thriving market of ad networks that provide JavaScript ad injection libraries for extension developers to integrate with their applications. These libraries inject ads on webpages by modifying the DOM structure of the HTML and inserting additional HTML iframes that contain the injected ad content. The ad provider may trigger the ad injection only on websites that are related (e.g., retail websites) and/or inject ads when a user performs a specific operation on this websites (e.g., hover on a product image).

5.1.3 Goal and Challenges

Our primary goal is to develop a system that automatically detects extensions that participate in ad injection practices out of the several thousand extensions that exist in online stores. Unlike existing approaches that rely on crowdsourced feedback and manual code inspection approaches, we seek to reduce the extent of human intervention required to analyze the ad injection practices of extensions by providing the extension store operators sufficient insight into the ad injection practices of the extension. In some cases compliance checking can be fully automated (e.g., ads served through invisible iframes, pop-unders or out of screen positioning), while others might require further manual inspection (e.g., camouflaging ads in order to trick users to click on them).

Achieving the above described goals requires addressing the following key challenges that we identified by manually inspecting a large set of extensions.

Identifying Triggering Websites As previously described, some extensions ubiquitously inject ads on every website that the user visits, while others are contextual and operate only on specific websites, e.g., Google search results. In order minimize the search space of possible webpages that the ad injection is triggered, it is important for the system to directly identify the domains on which the ad injection is triggered.

Identifying Injected Ads Injected ads alter the DOM structure of the website. Although identifying these changes is relatively easy, attributing the changes to an extension is not trivial, mainly because websites often have some logic that changes their DOM structure upon consecutive reloads. For example, a news website can decide to show (or not show) a Stocks Price bar based on the time of day. Furthermore, an extension can alter the DOM structure as part of its core functionality, which often not ads related.

Identifying Triggering Events In order to provide better user experience, or alternatively, make ad injection less intrusive and detectable, some extensions are activated based on user interactions, e.g., hovering over a product photo. This logic is encoded in the JavaScript library or be fetched in runtime from the third-party ad network. Our system needs to identify these triggering events, and simulate them so that the extension performs its ad-injection functionality.

5.2 Expector Design and Implementation

In this section we present the design and implementation of `Expector`, a system that can automatically detect and characterize ad injection practices used by browser extensions. The design of `Expector` addresses the challenges listed earlier in a way that enables scalable and lightweight checking of browser extensions. `Expector`'s current design is able to detect a wide range of ad injection practices with high accuracy. As we show in Section 5.3, `Expector` has a false positive and negative rate of 3.6% and 3.0%, respectively. We implement `Expector` for the Chrome browser and describe the implementation below.

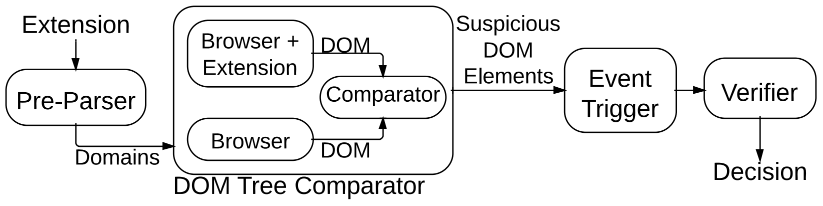


Figure 21: `Expector` design and process flow. The `Expector` contains four primary components: Pre-Parser, DOM Tree Comparator, Event Trigger, and Verifier. Given an extension, `Expector` determines if the extension is adware.

5.2.1 Design Overview

Figure 21 provides an overview of `Expector` which consists of four primary components – Pre-Parser, DOM Tree Comparator, Event Trigger, and Verifier. `Expector` operates by processing one extension at a time (though multiple instances can run in parallel). The Pre-Parser loads the extension into a browser and extracts all the JavaScript code that is loaded at runtime. It performs static analysis of the code to extract the potential triggering domains. Then, `Expector` launches two browser configurations; one without any extension and the another with the extension being investigated. For both configurations, `Expector` loads the domains detected by the pre-parser. After each page is fully loaded, the DOM Tree Comparator outputs a set of DOM elements that only exist in the webpage loaded by browser with the extension installed. These DOM elements are flagged as the potential elements used by the extension to inject ads. The Event Trigger identifies all events associated with these elements (e.g., onClick, onMouseOver, and onMouseMove, etc.) and triggers these events. Triggering these events could potentially result in new DOM elements being injected. `Expector` clicks on every new DOM element, while the Verifier monitors all HTTP requests made by the Event Trigger, and detects patterns that are commonly used by ad networks. We now describe each component in detail.

5.2.2 Pre-Parser

The primary goal of the pre-parser is to identify the potential domains that trigger the extension to inject ads, minimizing the need for exhaustively testing many domains. The pre-parser spawns a browser instance with the extension installed. In addition, it installs a virtual proxy between the browser's JavaScript engine and the extension. Using the remote debugging protocols supported by browsers (e.g., Chrome [17]), the virtual proxy intercepts all the function invocations related to JavaScript executions. Once the pre-parser obtains the JavaScript code executed by the extension, it searches for references to one of the top-10 top-level domain names [21] (e.g., `.com`, `.net`, `.org`). We limit the search to these top-10 top-level domains because we assume that extensions that are triggered by specific websites will most likely target popular websites, almost all of which are included in the top-10 domains.

5.2.3 DOM Tree Comparator

The goal of the DOM Tree Comparator is to identify the suspicious DOM elements that are potentially inserted by the extension for ad injection. A naive approach is to directly compare the HTML source of two different pages – one with the extension loaded, and another one without. However this approach will result in a large number of false positives as most webpages load a large amount of dynamic content that can be served by different hosts on each reload of the webpage. For example, each reload of a webpage can potentially result in ads served from different ad providers.

To address this, the DOM tree comparator uses *both* the DOM structure as well as the content of the DOM elements to identify potential elements injected elements. This is achieved by the following steps:

1. We assume that all hosts that serve content on the webpage *without* the extension installed are trusted and generate a list of trusted host names by loading the webpage multiple times without the extension.

2. We load the webpage using two instances of the browser (with and without the extension loaded) and record the DOM tree.
3. The DOM tree structure is flattened into a list of elements by performing a pre-order traversal over the corresponding DOM tree, i.e., for each node in the list, its descendants are located at its right side and its ancestors are at the left side of the node. Each item in the list consists of the tag name of the DOM element and the domain associated with the DOM element (if such domain exists). For example, an iframe tag like `<iframe src=‘http://www.amazon.com/product/B00BWYQ9YE/’/>` is encoded as a two tuple `(‘iframe’, ‘www.amazon.com’)` after transformation.
4. In order to compute the difference between the two lists, we used a variant of the *longest common subsequence (LCS) problem* algorithm, which is used by the well-known `diff` command. The modified LCS algorithm outputs nodes that are present only in the webpage loaded with the extension installed.
5. Finally, these elements are further processed and only the elements whose host name is not presented in the trusted list are used as input to the event trigger component.

Overall, we find that the above process yields very few false positive and false negative and enables us to identify DOM elements that are only added by the extension. The one caveat for this process is that we cannot identify DOM elements that are dynamically generated, e.g., create an iframe containing ads and inject it to the DOM only after the user hovers on a product photo. However, in our experiments we did not find extensions that use this behavior, probably since dynamically creating elements upon user interactions might create a visual lag, whereas simply having the DOM element already injected, and then simply controlling its visibility in code is immediate.

5.2.4 Event Trigger

The primary goal of the event trigger is to visit the ad landing pages potentially associated with the identified DOM elements through redirects. To this end, the event trigger process the suspicious DOM elements as follows. For all DOM elements that have a JavaScript event handler to process user events, like onClick, onMouseOver, etc., we instruct the browser to execute the corresponding JavaScript function. For all events that have a URL associated with the DOM element, we instruct the browser to visit the URL. Note that we ignore iframes as the URL points to the source of the iframe. Finally, for all the remaining elements (elements with no URLs or iframes), we instruct the browser to trigger a click event to emulate a user clicking inside the element. For all the above cases, if the element contains an ad, the advertiser's website will be loaded usually through a series of redirects, first to one or more ad networks and then to the ad landing page.

A caveat to the above described event trigger mechanism is that some extensions inject ads only after some time has elapsed since the user installed the extension. We assume these practices are used to reduce user dissatisfaction from the injected ads. We manually inspected these extensions and found that they mostly retrieve timing information using the system APIs. We therefore "tricked" these extensions by setting the system clock before we install an extension to one year backwards, then install the extension, launch the browser and then set the system clock to one day in the future.

5.2.5 Verifier

Finally, the verifier validates that the triggered DOM elements are indeed associated with ads. A unique characteristic of an ad element is that after an ad is clicked there is a redirection pattern, oftentimes to multiple websites. These redirections enable ad networks to monitor ad clicking for analytics and billing purposes. This redirection pattern is unique to ads and simply detecting it is sufficient to identify that the corresponding DOM element injects ads.

To detect such redirection patterns, we implemented a lightweight Chrome extension, which is loaded before the extension is installed, that logs all HTTP requests made by the browser. HTTP requests issued as a result of the Event Trigger actions are analyzed by the verifier which searches for redirections. A redirection is detected when an event results in more than one domain in the traffic trace, and the last domain (e.g., the ad landing page) is not the same as the original web site (i.e., the one that contains the DOM element). If such redirection is identified, the extension is marked as an adware.

5.2.6 Implementation

We implement `Expector` using Node.js [15] and Selenium [19], a tool that enables browser automation (the entire setup is executed without human intervention). We use Node.js to spawn a Chrome instance and load a Chrome extension for pre-parsing. Using the remote debugging protocol of Chrome, we configure a virtual proxy working as a Chrome Developer Tool that intercepts all the JavaScript function invocations. Specifically, we listen on all *Debugger.scriptParsed* events and log all the JavaScript code parsed by the `V8` JavaScript engine. The rest of the components of `Expector` are implemented with Selenium and the LCS algorithm is implemented in Python.

To ensure fast and scalable processing of a large number of Chrome extensions, we deploy `Expector` across 60 Linux Debian 7 virtual machines running on a 32 core server with 128 GB RAM. This setup enabled us to process all 18,030 extensions in the Chrome webstore in less than three days.

5.3 Evaluation

In this section, we provide a detailed evaluation of `Expector`. We evaluate the accuracy of `Expector` to detect ad injecting extensions (adware) by characterizing the

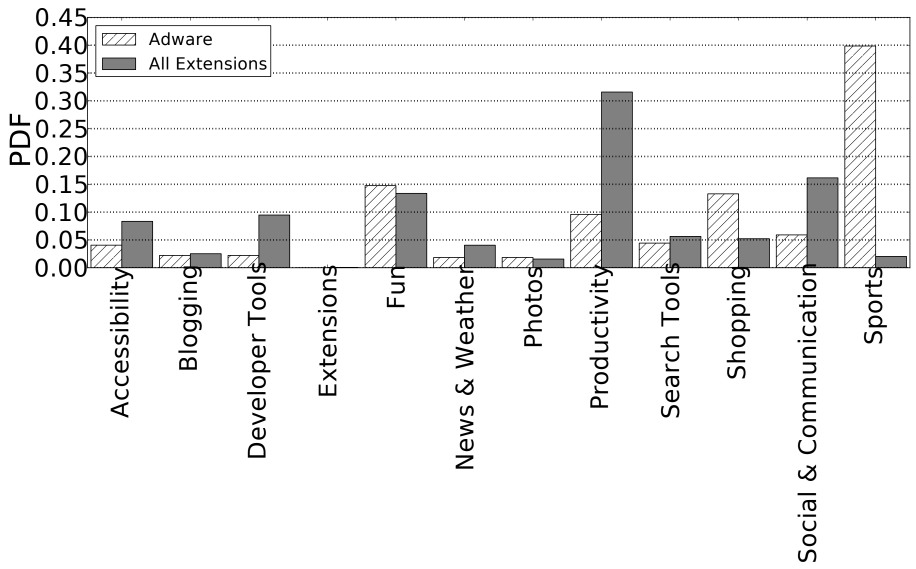


Figure 22: The distribution of Chrome extensions across different categories. All Extensions span the 18,030 extensions we crawled from the Chrome store, and Adware span the 292 extensions identified as adware by `Expector`.

false positive and negative rates using two different datasets. Using metadata associated with the extensions, we further characterize the ad networks and extension developers that publish such extensions. Finally, we discuss the adware takedown strategy that Google probably relies upon.

5.3.1 Datasets

Dataset to characterize false positives. We evaluate the false positive rate of `Expector` by testing all the available extensions on the Chrome web store followed by manual verification. To this end, we developed a crawler that downloads all extensions that are listed on the Chrome Web Store [12] along with the extension's meta data, i.e., developer account, extension category, number of active users, rating, description, and user reviews. We ran our crawler during March 2014, and obtained 18,030 Chrome extensions. Figure 22 shows the distribution of these Chrome extensions across different categories, with the top three categories being "Productivity", "Social & Communication and "Fun".

Dataset to characterize false negatives. In order to study the false negative rate of `Expector` we use a crowdsourced list of Chrome extensions that are tagged as adware. This list is provided by Extension Defender [1] that enables users to submit extensions as potential adware which are reviewed and verified manually by the curators of this list. In March 2014, this Extension Defender had 78 Chrome extensions that were tagged as adware.

5.3.2 False Positives

Out of the 18,030 extensions downloaded from the Chrome web store, 108 extensions (0.6%) failed to be processed by `Expector`. By inspecting the code of these extensions we found that these extensions utilize native binary code [18] that was not ported by the extension developer to Linux. All other 17,922 extensions were successfully evaluated by `Expector` using the setup within three days highlighting the ability of `Expector` to scale to a large number of extensions.

`Expector` reported 303 extensions as adware, which accounts for 1.7% extensions on Google Chrome store. We manually installed each of these and inspected their source code, and found that 292 of these are indeed adware. This indicates a very low false positive rate of 3.6%. We further analyzed the 11 extensions that `Expector` incorrectly identified and find that `Expector` flagged them as users were re-directed to webpages controlled by the extension developer that contained ads. For example, parental control extensions like *Anti-Porn Pro* and *No Xvideos* maintain a blacklist of websites. When users navigate to these webpages, the extension re-directs the user to a webpage hosted by the extension developer showing a warning message. This webpage hosts ads which are detected by `Expector` and consequently flags the extension as adware.

5.3.3 False Negatives

We begin the analysis of false negative rate by first manually verifying the crowd-sourced list of extensions tagged as adware. Surprisingly, we find that out of the 78 extensions only 34 of them were manually verified as adware. This initial analysis highlights the limitations of the crowdsourced approach where curators cannot manually check every update to the extension source code and verify user complaints. Consequently, we postulate that these 44 extensions may have disabled the ad injection functionality after being flagged as adware, and Extension Defender failed to remove them from their list.

Out of the 34 manually verified adware extensions, `Expector` was able to correctly detect 32 of these. We further analyzed the two extensions that `Expector` missed. The first one uses a Windows DLL file, which is not compatible on Linux, but could have been processed if `Expector` was deployed on Windows. We do not count this as a false negative. The second extension operates on Facebook, and uses a more sophisticated triggering method. It requires the user to visit Facebook.com, scroll to the bottom of the page, and wait for a 10 seconds before injecting ads. `Expector` correctly identified the triggering website and the timeout event, but failed detecting the scroll event. These complex triggering events are difficult to identify, however, as our analysis shows, they are quite rare as they target a very specific user interaction, thus reducing the number of ads the extension injects (and lowering their revenue). Consequently, `Expector` has a low false negative rate of 3.0% (1/33).

5.3.4 Deploying Expector

`Expector` identified 292 adware extensions in the Chrome store, which is notably higher ($9\times$) than the crowd-sourced approach used by Extension Defender. Considering a 3% false negative rate, deploying `Expector` in the online approval process of uploading extensions to the webstore will almost mitigate adware from entering the

Developer id	Adware	Total Ext.
brandthunder.com	142	172
everplexmedia.com	7	8
Chrome	6	65
extensions.czechian.net	5	7
www.extensions-for-chrome.com	4	9
itworks	4	7
professional-extensions.czechian.net	3	4
–Monster–	3	4
Ext	2	2
bionic	2	4

Table 1: The top-10 adware developers.

Chrome webstore, thereby greatly improving its quality, while incorrectly rejecting only 3.6% of the submitted extensions. Developers of these incorrectly flagged extensions can appeal and request a manual verification, requiring a small amount of human intervention by the webstore operators.

5.3.5 Characterizing Adware

In this section we characterize the 292 adware that our system correctly reports to understand their category distribution, developers authoring such extensions and usage of ad networks.

Extension Categories. Intuitively, we expect the adware extensions to belong to the "Shopping" category in order minimize the risk of violating Google's policies that prohibits the bundling of different functionalities. For example, a shopping related extension could potentially inject ads on webpages with the goal of improving the user's online shopping experience. Figure 22 shows the fraction of all extensions and adware extensions in each category. Surprisingly, we find that only 14% of the adware extensions are categorized as "Shopping". The remaining span different categories, with "Sports" holding the largest percentage of adware (40% of the extensions we identified as adware).

Adware Developers. We study the developers of the adware extensions. Overall,

Domain	Extensions	User base
www.superfish.com	199	1,235,379
fastestjs.info	1	1,235,379
txtsrving.info	11	594,565
secure.demand-go.com	3	594,565
o.yieldsquare.com	9	488,043
www.imgclck.com	14	344,821
go.redirectingat.com	23	332,694
service.pricegong.com	2	308,084
mc.yandex.ru	147	125,860
xtensionplus.com	7	97,250

Table 2: The top-10 ad networks that adware contact.

Number of Ad Networks	Number of Adware	Adware (%)
1	74	27.21
2	40	14.71
3	12	4.41
4	145	53.31
5	0	0
6	1	0.37

Table 3: The distribution of adware across the number of network domains that they contact.

there are 11,738 distinct developers that publish extensions on the Chrome store. Out of these, 118 are developers of one or more of the 292 adware extensions. Only 16 of them (13.6%) publish more than one adware extension (determined by the developer ids). Table 1 shows the top 10 developers of adware along with the number of extensions that are flagged as adware. We observe that not all of the developer's extensions are adware. This is because, ad networks for extensions require a minimum number of users to actively use the extension (e.g., `superfish.com` requires a minimum of 50,000 active users) and not all the developer's extensions may have these many active users. This effectively makes it difficult for the webstore operators to directly ban the developer as not all the extensions may be adware.

Ad Networks. In order to study the ad networks used to deliver ads, we processed the network logs stored by the Verifier module, and extracted the domains used by adware for fetching ads. We found 67 unique ad network (domains) in total. We further want to assess the user-base of each ad network, i.e., how many users each ad networks serves ads to. We use the following simple heuristic that provides a conservative estimate of the number of users; among all the adware extensions that fetch ads from a given ad network domain, we assign the ad-network the maximum user-base among these extensions (making the assumption that all users overlap). Table 2 shows the top-10 ad networks that adware contact. We observe that a majority of the ad networks are used by multiple extensions, with superfish.com and yandex.ru leading with 199 and 147 adware extensions, respectively. Finally, Table 3 shows the number of ad networks used by adware. The table shows that more than 50% of the adware contact at least three ad networks, the vast majority (145 extensions) contact four networks, and one extension, *TrustedShopper*, partners with 6 different networks. We attribute this to the developers' attempt to maximize their financial gains and reduce their reliance on a single ad network.

5.3.5.1 Extension Removal

We now characterize the extent to which Google purges adware extensions from the webstore. We crawled the Chrome store again in May 2014 (roughly 6 weeks after our initial crawl), and checked the availability of the adware identified by `Expector`. We found that out of the 292 adware detected only 33 (11.3%) are no longer available for download (see Table 4). Finding the root cause behind these extension removals is not trivial – the extensions may have been removed by the developer or Google could have removed these extensions due to policy violations related to ad injection or other malicious activity. By manually verifying these extensions we do not find any explicit malicious behavior. We also observed that 15 of the 33 extensions belonged to the same developer. Given these observations we hypothesize that these extensions likely

Extension id	# of users	Developer
hnmhpjbejpnnaffkpmebeagdiidibjfa	40775	almond.foier
dgkodfmpfeankonfpmeiecfcdnnplfoh	6	brandthunder.com
npfginbbkdjnoomjoipnafkgaclcipdc	3	brandthunder.com
dkdjknkhdfkcjdkfgfdlhglfeheefkgc	61	brandthunder.com
kgechjpjeifpccomemjclcndpfhebgph	2	brandthunder.com
flcgogbgmgihhcjgakodpgkndoefjikl	9	brandthunder.com
ahmbpbdpkcknchhlbhehkhgddeebjbaa	3	brandthunder.com
mcienhknfimilomfokgenohgephcmpnh	3	brandthunder.com
jklamffdbhkmijgiimpkdnihbmicomml	19	brandthunder.com
elgiklnblmlckpkmigdmlmcmjfpiekgc	8	brandthunder.com
gcoeaodmmobklbpcajdiohbfgedafobe	25	brandthunder.com
fdnepimfkhmamfefpagkplhbjflogjmo	43	brandthunder.com
inchelgepobcoidngomhcljobbjdhmok	35	brandthunder.com
egghpghdddlhclmmiaaihmgnggeaeehb	2	brandthunder.com
hddfbbleodlbkcgbonkfoknoodnknifo	3	brandthunder.com
nghobmofokaoniogeeoglgkghcppbkfo	0	brandthunder.com
hlcpjandbihelclgoinjineogmpifpdl	1299	Chrome
adgebcppjdjaoinhpiacddeokhiammkl	198	Chrome
anniokiiijccmdokaojcjocfjigdncjp	16	Chrome
fggbmkfalaofcfdmidbjphgjcmaafkgh	61	Chrome
hhffdhobhcodehjbdkgnmopbaifcaecl	64	Chrome
gngecdocapndgkmopgklkiafoeoinlpl	81	Chrome
ocafaidocnigmmhbhffbhidhedgckcfd	384	everplexmedia.com
dlbblkmidojcmjafbeddanofmckfbmbb	6	hotelo
ignkobhlkpgjcpkfgfohhdgdaldfaoni	48357	leon.munifield
jieojmaflakccpfnjiibbkajdplbhhcc	681	ownweather.com
gbmkfanjlljhcfcnejpecajlpkfmckpi	18	peoplesbankal.eversave.com
dblpdenjapnbppelpaflghkfblfhgobo	8385	professional-extensions.czechian.net
lgbmknmpendafnnkibphfmeeljfdomgk	6644	weatherww.com
akmlaclbbenlpkimkhbmgpfampphciab	216	www.bizrate.com
cdfjbkbddpfnoplfhceolpopfoepleco	70774	xnotifier.tobwithu.com
cpofbbgoaeoiiagmlfkkipjmggkedgic	7446	xtensionplus.com
gnpkfejhbjbfidphpdfeokopdigeinkf	164	xtensionplus.com

Table 4: 33 Chrome extensions that Google took down due to ad injection bundle.

violated Google's policies related to ad injection.

In addition to the above analysis, we uploaded an extension to the Chrome store that contains almost the same code as one of the extensions that were taken down by Google that performed ad injection – *NASA*. Though we reported our uploaded extension to Google on 10 May, at the time of the writing this extension is still available for download on Chrome store. Overall, this further implies that the current manual approach to identify adware is not sufficient, resulting in many adware extensions and developers that fraudulently generate ad revenue.

5.4 Case Studies

In this section, we seek to characterize the ad metrics (CTR and number of impressions) of the ad networks that deliver ads to adware extensions. As shown in Section 5.3, these ad networks are not among the major ad networks (e.g., DoubleClick, Yahoo!, etc.) and hence not well studied. We focus on 4 popular ad networks that were contacted by 223 extensions identified as adware by `Expector`– `i.txtsrving.info`, `superfish.com`, `imgclck.com` and `xtensionplus.com`.

5.4.1 Dataset

In order to perform this study we use two datasets provided to us by a large ISP, a security company and two universities, all of which monitor their users' network traffic.

HTTP Traffic. We collected the HTTP traffic log of tens of thousands of users from two university networks and tens of enterprises. For each request we obtained the source and destination IP address, and the complete HTTP request header including the user's cookie identifier (if present), requested URL path and referer. This dataset spans 2 weeks in April 2014.

DNS Traffic. A large ISP provided us the DNS queries performed during 2 weeks

Ad Network	Request pattern	Request type	Affiliate ID field	User identifier
TxtSrving	`cdncache1-a.akamaihd.net/loader/...pid=...`	fetch ad	pid	Cookie
	`i.txtsrving.info/kwdu?...subid=...`	fetch ad	subid	Cookie
	`p.txtsrving.info/click?..subid=...`	click	subid	Cookie
SuperFish	`www.superfish.com/ws/findByUrl.action?...userid=...dlsource=...`	fetch ad	dlsource	userid
	`www.superfish.com/ws/offerURL.action?...userid=...dlsource=...`	click	dlsource	userid
ImgClck	`www.imgclck.com/supp0rt/www/delivery/afr.php?...&beacon=...`	fetch ad	beacon	Cookie
XtensionPlus	`xtensionplus.com/display.htm?...&pi=...`	fetch ad	pi	Cookie

Table 5: The HTTP request patterns originated from adware for each ad network. The table shows the purpose of the request (fetch an ad or report that a user clicked on an ad), the field used to identify the affiliate (adware), and the method used by the ad network to track the user.

in April 2014 against the domains of the 4 ad-networks we study. The DNS traffic log contains A-type DNS records from around 20 million daily users located in the US. Each DNS query contains the IP address of the requesting host (user) and the requested domain. We note that the ISP has a very low DHCP churn rate on a daily basis, thus it is very unlikely that a single host will have more than a single IP on a daily basis.

Mapping Adware to Traffic Logs. Our first task is to associate the traffic logs to specific adware extensions. To this end, we need to identify the URL patterns issued by the extensions we study, and locate these patterns in the traffic logs.

We identify patterns in the URLs that each adware generates by inspecting the source code, and extract the affiliate IDs which are unique IDs issued by the ad networks to their customers. Table 5 lists some of the traffic patterns used by the adware extensions, along with the fields we use to identify the affiliate IDs (leading us to the corresponding adware) contained in the HTTP requests. Additionally, the Table also shows the method used to associate unique users to the requests. Using these patterns, we filtered our datasets to only include traffic that matches the adware we study. To further validate that these requests originate from the adware and not from users visiting pages that make similar requests to the ad networks, we inspected the HTTP referrer field and found that none of the referres were from the domains that adware contacts. This implies that the requests are generated by the adware

	`www.imgclck.com`	`xtensionplus.com`	`superfish.com`	`txtsrving.info`
% of IPs	0.003	0.017	0.168	0.008
Total ad requests	22,935	3,983	9,741	2,342
Avg Daily requests / user	102.30	54.71	8.19	8.44

Table 6: Statistics regarding the four domains used by adware for ad injections, showing the percentage of unique IPs (out of the 20 million IPs in the DNS dataset) that resolve each domain, the total ad requests and the average daily ad requests per user from each domain in the HTTP dataset.

extensions rather than other user browsing activity.

In order to draw broader conclusions on the reach of these ad networks, i.e., number of users they serve, we correlate the IP address of hosts that contact ad networks in the HTTP dataset with the DNS logs. We found that all enterprise IP addresses that exhibit adware network patterns in the HTTP dataset also appear to query the ad networks domains in the DNS logs (the university hosts use a different DNS server). Together with the observation that these domains do not serve any regular content, leads us to believe that DNS queries to these 4 ad networks is indeed the result of an adware and not a user explicitly visiting their domain. This enables us to use the large DNS logs and estimate the user-base of these ad networks.

5.4.2 Click Stealing

We first study the adware extensions that contact the ad network `txtsrving.info` for fetching ads. As shown in Table 2, we identified 11 such adware, the most popular of which has 594,565 active users. TxtSrving injects ads into a page by adding *<a>* tags to various text elements in the original HTML. These tags are used as hyperlinks, usually placed in pages to connect it with other relevant content. However, TxtSrving connects the text with their ads, so an unwary user that clicks on the links, probably expecting some related content, actually clicks on an ad. This form of ad injection can be seen as a “click stealing”, essentially tricking users into clicking ads despite lack of intent [37], potentially leading to a clickthrough that is substantially higher than that obtained by more legitimate practices.

To further study the clickthrough rate of such ads, we measured the number of ad requests made to the domain `txtsrving.info`. As shown in Table 6, we identified 2,342 distinct ad requests, all originating from 11 different extensions (identified by their unique affiliate IDs). Out of these, we found 149 clicks on injected ads (identified by an HTTP request to the domain of the ad with the referrer being the ad network). This yields a 6.36% clickthrough rate, which is significantly higher than the average 0.22% clickthrough rate reported by Google's Double Click [7].

5.4.3 Impression Fraud

Next, we study the adware that serve ads delivered from `xtensionplus.com`. These adware fetch ads and display them in 13 invisible iframes every time a user visits a page. This kind of injection strategy is a clear form of impression fraud [79], essentially causing advertisers to pay for ads that were never seen by real users.

As shown in Table 6, we found 3,983 ad requests to `xtensionplus.com`, 3,701 (91%) of which originate from Chrome browsers and the rest from Firefox, all using 3 unique affiliate IDs.

5.4.4 Additional Ad Injection Practices

Finally, we discuss two examples of adware that seem to violate extension policies in a more subtle ways. While we may not have enough evidence to categorize the ad injection practices of these extensions as clearly fraudulent, they do present some "shady" traits that we believe are worth highlighting.

5.4.4.1 www.imgclck.com

Extensions that use ImgClck `www.imgclck.com` typically place a banner ad at the bottom of a user page. Many webpages are very long and oftentimes implement some method of "infinite" scroll (e.g., Facebook, Pinterest, etc.), therefore a user might never really reach the bottom of the page [20]. This means that an ad positioned in

Regions	CPC	Internet users
US & Canada	\$0.22	284,056,300
UK, FR, Germany	\$0.16	177,631,638
Italy, Spain	\$0.10	69,402,475
Austria, Belgium, Switzerland, Denmark, Ireland, Netherlands, Norway, Sweden	\$0.12	59,444,750
Brazil	\$0.08	99,357,737
Australia, New Zealand	\$0.12	22,003,709

Table 7: Superfish cost per click per geographic region and the corresponding Internet user population.

	Chrome	Firefox	IE	Safari	Total
Ad display	4,169	4,700	861	11	9,741
Affiliate ids	56	37	27	3	76
User ids	264	187	85	7	543

Table 8: Statistics of observed traffic to superfish.com across 4 browsers.

the bottom of the page is counted as an impression, but might never be seen by a real user, thus might also be considered a type of impression fraud. As shown in Table 2, there are 14 adware contacting domain `www.imgclck.com`, the most popular of them being used by 344,821 users.

Although positioning a banner in the bottom of a page is not necessarily fraudulent, we found that ImgClck embeds DoubleClick ads into the iframe they provide to the extension. The iframe with the DoubleClick ads is then injected to arbitrary pages, which is in direct violation to DoubleClick's policy that strictly prohibit placing ads by anyone other than the page owner [2].

5.4.4.2 superfish.com

Superfish is an ad-network that shows ads for products based on visual similarity to other products. Extension developers use them on e-commerce websites (e.g., `Amazon.com` and `eBay.com`), so that users that shop for a product get ads to similar-looking products. Superfish styles the ads so that they blend well with various websites. Figure. 20 shows an example of Superfish ads injected into Google search result,

Browsers	# of overlapped affiliate ids
Chrome & Firefox	22
Chrome & IE	20
Chrome & Safari	2
Chrome & Firefox & IE	18
All 4 browsers	1

Table 9: Superfish affiliate overlap between Chrome and other browsers.

when the user issues a search for *women handbags*. As shown in Table 2, we identified 199 adware that use Superfish to display ads, with the most popular extension being used by 1,235,379 users. Using our DNS dataset, we found that roughly 0.2% of the 20 million unique IP addresses resolve `superfish.com`. Similar to the other ad network domains, this domain has no content to offer to regular Internet users and there is a one-to-one mapping between adware and DNS traffic, thus we assume that these DNS queries can be attributed to adware extensions.

Since Superfish is a popular provider of ads, we seek to estimate the revenue it generates from delivering ads to extensions. Using our HTTP traffic dataset, we found 9,741 requests to fetch ads from `superfish.com`, issued by 543 distinct users. Of the 9,741 ad injections, 107 were clicked, implying a clickthrough rate of 1.098% (not as high as the click fraud performed by TxtSrving, but still significantly higher than DoubleClick's). We found 76 unique affiliate IDs from these requests, and out of those originating from Chrome browsers, we found 56 affiliate IDs that are present in the source code of the adware identified by `Expector`. Table 8 and 9 summarize our findings.

Table 6 shows the number of unique IP addresses resolving the domain `superfish.com`, and our user-base estimates of the adware that uses Superfish. Using these estimates and the aforementioned data, we can estimate the annual revenue of Superfish-based adware. We contacted Superfish and obtained their CPC rates, which are presented in Table 7. We limit our estimates to the Internet population that reside in the regions for which we have CPC data. Using the HTTP requests to `superfish.com`, we estimate that on a daily average, 19.69% of the 543 users we identified receive on average 8.19 ads from Superfish. Overall, based on the above DNS dataset analysis, assuming that 0.2% of each region's Internet population is using an extension with Superfish ads, we estimate that the total annual amount Superfish pays to the adware developers is approximately $1.5 million.

This indicates that although we are only able to ballpark the revenue adware developers make from ad networks like Superfish, we find that this practice is indeed prevalent and fosters a large and profitable ecosystem of ad networks and extension developers.

5.5 Discussion

To the best of our knowledge, `Expector` is the first system that automatically detects browser extensions that utilize ad injections. As most other automated detection systems, `Expector` has some limitations, which we discuss below along with possible directions for future work.

Native Code. Extension developers that want to improve the performance of their code, or hide specific implementation details, can use the NPAPI to embed native code into their extensions. In our current implementation, `Expector` runs on Linux, therefore extensions that use native code for non-Linux OSes cannot be executed. However, note that due to portability concerns NPAPI is being phased out [16], which will not impact `Expector` in the long run.

Evading Detection. Once a tool like `Expector` is adopted, adware developers may use various evasion techniques to avoid being detected. These practices are already used by malware, for example to detect dynamic analysis environments [24]. Similarly, we showed in Section 5.3 that an adware targeting Facebook evades automated detection by requiring complex user interaction. Obviously, `Expector` will have to evolve and keep track of these evasion techniques.

Obfuscation. `Expector` analyzes the JavaScript code of an extension to extract the domain names that trigger the adware. Therefore, adware developers may obfuscate their JavaScript code to prevent detection. However, since the end-goal of adware developers is to maximize their revenue, most of these extensions operate on popular websites. Therefore, an alternative approach to our Pre-Parser, is to exhaustively try a large list of popular websites, and see if any of these triggers the extension. All of the adware we studied, either operated on all websites, or were triggered in extremely popular websites, such as Google, Amazon and Facebook.

Replacing Publisher IDs. `Expector` operates by looking for DOM changes introduced by an extension. Therefore, a sophisticated adware can detect DOM elements with existing tags and change the publisher ID into their own publisher ID, thereby diverting the profits from the original publisher to themselves. However, such a trick is difficult to implement and easily detected by the ad networks themselves (mismatch between the referred domain and the publisher ID).

5.6 Related Work

In this section, we discuss three lines of work most related to ours – (1) adware analysis, (2) detection of malicious JavaScript, and (3) detection and mitigation of malicious browser extensions.

Adware. Edelman *et al.* [42] provide an overview of the adware ecosystem, studying the ad networks, exchanges, and practices used by ad injectors. In a more recent work [43], the same authors study fraud in online affiliate marketing, and how adware takes part of such frauds. Both works focus more on the business aspects of the ecosystem, and unlike our work, they do not attempt to provide a system for automatic detection of adware, but instead extract the participants of manually selected adware extensions.

Several tools have been recently developed to assist users remove adware from Chrome [8, 10]. These tools operate by comparing extensions installed by a user against a list, oftentimes crowd-sourced, of known adware, and alerting the user if such extensions are found. Such detection approach is limited by the accuracy and completeness of its list, and cannot handle new adware that was not yet reported. `Expector` does not rely on manually curated lists, making it highly suitable for analyzing extensions in the approval process, before they ever reach users.

Malicious JavaScript Detection. Several works proposed generic methods for detecting of malicious code [26,26,35,39,40]. Chugh *et al.* [35] present an information flow based approach for inferring the impact that the JavaScript code has on the website, using both static and dynamic analysis.. In [26], the authors introduce VEX, a framework that performs static taint analysis to track explicit leaks resulting from potential privilege escalations in browser extensions. A different form of static analysis was recently proposed by Kashyap and Hardekopf [52] for creating security signatures of extensions, which describe the usage of critical APIs and information flows of an extension. A recent dynamic information flow analysis was presented by Bichhawat *et al.* [30], tracking both explicit data and implicit control flows of JavaScript bytecode in WebKit with moderate overheads.

Despite great improvements of JavaScript code analysis, adware typically do not use sensitive APIs or specific coding practices, thus simple ad-hoc methods, such as those used by `Expector`, are likely to be more suited to the problem domain.

Browser Extension. A series of security issues in browser extensions have been studied for years, mostly focusing on preventing user data leakage through malicious extensions [44, 53, 54, 57, 84] and protecting against privilege abuse of extensions [47]. The security model of Chrome extensions was criticized in [57]. The authors showed that extensions introduce attacks on the Chrome browser itself, and proposed a enforcing micro-privileges at the DOM element level. Egele *et al.* [44, 53] proposed several detection solutions to spyware that silently steals user information. By leveraging both static and dynamic analysis techniques the authors show that it is possible to track the flow of sensitive information as it is processed by the web browser and any extension installed. [47] showed that many Chrome extensions are over-privileged, which can potentially cause security risks. To address the shortcomings of existing extension mechanisms, the authors propose a comprehensive new model for extension security.

Although the security model of browsers with respect to extensions has improved over the years, creating a security model that mitigates adware while maintaining a decent user experience is extremely difficult. This is mostly because adware do not typically require more privileges than other non-adware extensions, namely accessing the DOM of visited pages and network access. In addition, adware often start as a non-adware extension, and then, when the user-base is sufficiently large morph into adware. Enforcing fine-grained control on every update is doable, but will significantly hurt user experience of the vast majority of non-adware extensions.

5.7 Conclusions

In this chapter we present `Expector`, a system that automatically inspects and identifies browser extensions that perform ad injection. We implemented `Expector` for Chrome, ran it on the entire 17,931 extensions in the Chrome store and identified 292 extensions that inject ads, a detection rate that significantly outperforms the current crowd-sourced detection process. Through rigorous evaluation, we found that `Expector` is capable of identifying various forms of ad injection practices with low 3.6% false positive rate and 3% false negative rate. Using an indepth study of the practices employed by the ad injecting extensions we detected, we found that many of these extensions violate the browser policies, participate in fraudulent activities and abuse their users' trust. As part of future work we plan on extending `Expector` beyond Chrome, so that it can process extensions of other popular browsers. Furthermore, our study lays the foundations to analyze web apps, which are very similar to extensions for increasingly popular browser-based OSes, such as Chrome OS and FireFox OS.

CHAPTER VI

CONCLUSION AND FUTURE WORK

This thesis explores information manipulation in the context of the Internet. It shows that, with the advent of new services and technologies, Internet-scale information manipulation has been pervasive and rampant. Motivated by this finding, the thesis further introduces some technical approaches that help users raise awareness about information manipulation and – more important – undermine their pernicious effects. This thesis is pioneering research about information manipulation in the context of the Internet. In addition to the proposed technologies that empower users against adversarial information manipuluation, my findings have been guiding service providers to revamp their services for better defending against malicious information control.

The information manipulation schemes discussed in the thesis represent only the tip of the iceberg. As emerging technologies spread both online and offline, a increasing number of manipulators would add a growing list of attempts to misuse or exploit the mechanisms and algorithms underlying new technologies regardless of their intents. For example, the availability of Internet-connected telematics devices has brought personalization into traditional automobile insurance industry, where mainstream insurers use a telematics device to track driving habits of people and customize their insurance rates. Intuition suggests presenting low-risk driving habits to an insurance provider would allow a customer to enjoy significant saving on insurance. As a result, it is highly likely that car insurance providers become the next target of manipulators who attempts to tamper with driving habits and deceive insurers with the goal of exchanging potential savings on insurance.

To cope with unforeseen manipulation, in the future, there is a need for developing new detection and mitigation solutions. The nature of information manipulation is that, attackers compromise the integrity of data that end users or service providers consume. As a result, future efforts may focus on developing a variety of sophisticated mitigation solutions to verify data integrity. One possible solution is to correlate those vulnerable data points with additional data that end users or service providers cannot directly consume but can leverge to assess data integrity. Take the manipulation of car insurance rates for example. Insurers may integrate additional sensors – such as MEMS gyroscope and accelerometers – to their telematics devices for the purpose of gathering additional data that attackers cannot easily falsify. Although sensing data are typically too noisy to divine driving habits, insurers may use them to assess the validity of other data items.

REFERENCES

[1] "Abusive extension submission." `http://extensiondefender.com/submit.php`.

[2] "Ad placement policies." `https://support.google.com/adsense/answer/1346295#Ads_on_the_same_page_or_site_as_another_publisher`.

[3] "Add-on policies - review process." `https://addons.mozilla.org/en-US/developers/docs/policies/reviews`.

[4] "Adware vendors buy Chrome Extensions to send ad- and malware-filled updates." `http://arstechnica.com/security/2014/01/malware-vendors-buy-chrome-extensions-to-send-adware-filled-updates/`.

[5] "Android add-ons." `https://addons.mozilla.org/en-US/android/`.

[6] "Developer Program Policies." `https://developer.chrome.com/webstore/program_policies#extensions`.

[7] "Display Benchmarks." `http://www.richmediagallery.com/resources/benchmarks/`.

[8] "Extension defender." `http://extensiondefender.com/`.

[9] "Extensions for firefox for android." `https://developer.mozilla.org/en-US/Add-ons/Firefox_for_Android`.

[10] "Extshield notifies you if you're running an adware extension." `http://lifehacker.com/chrome-protector-notifies-you-if-youre-running-an-adwa-15053714`

[11] "Firefox add-ons." `https://addons.mozilla.org/en-US/firefox/`.

[12] "Google Chrome Web Store." `https://chrome.google.com/webstore/`.

[13] "Google Removes Two Chrome Extensions Amid Ad Uproar." `http://blogs.wsj.com/digits/2014/01/19/google-removes-two-chrome-extensions-amid-ad-uproar/?mod=WSJBlog&utm_source=twitterfeed&utm_medium=twitter`.

[14] "Keeping Chrome Extensions Simple." `http://blog.chromium.org/2013/12/keeping-chrome-extensions-simple.html`.

[15] "Node.js." `http://nodejs.org/`.

[16] "Npapi plugins." `https://developer.chrome.com/extensions/npapi`.

[17] "Remote Debuggin Protocol, Google Developers." `https://developers.google.com/chrome-developer-tools/docs/debugger-protocol`.

[18] "Saying goodbye to our old friend npapi." `http://blog.chromium.org/2013/09/saying-goodbye-to-our-old-friend-npapi.html`.

[19] "Selenium automates browsers." `http://docs.seleniumhq.org/`.

[20] "Unfolding the fold (insights into webpage scroll)." `http://blog.clicktale.com/2006/12/23/unfolding-the-fold/`.

[21] "Web Technology Surveys." `http://w3techs.com/technologies/overview/top_level_domain/all`.

[22] "Cross-site request forgeries." `http://shiflett.org/articles/cross-site-request-forgeries`, 2004.

[23] "Conceptdoppler: A weather tracker for internet censorship," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, (New York, NY, USA), pp. 352–365, ACM, 2007.

[24] "Invasion of malware evading the behavior-based analysis," *AhnLab MDS Whitepaper*, February 2014.

[25] "Amazon.com product identifiers." `http://archive.org/details/asin_listing`.

[26] Bandhakavi, S., King, S. T., Madhusudan, P., and Winslett, M., "Vex: Vetting browser extensions for security vulnerabilities," in *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, (Berkeley, CA, USA), pp. 22–22, USENIX Association, 2010.

[27] Becchetti, L., Castillo, C., Donato, D., Baeza-YATES, R., and Leonardi, S., "Link analysis for web spam detection," *ACM Trans. Web*, vol. 2, pp. 2:1–2:42, Mar. 2008.

[28] Bennett, P. N., Radlinski, F., White, R. W., and Yilmaz, E., "Inferring and using location metadata to personalize web search," in *The 34th Annual ACM SIGIR Conference*, SIGIR '11, ACM, 2011.

[29] Bennett, P. N., White, R. W., Chu, W., Dumais, S. T., Bailey, P., Borisyuk, F., and Cui, X., "Modeling the impact of short- and long-term behavior on search personalization," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, (New York, NY, USA), pp. 185–194, ACM, 2012.

[30] Bichhawat, A., Rajani, V., Garg, D., and Hammer, C., "Information flow control in webkit's javascript bytecode," in *Principles of Security and Trust*, vol. 8414 of *Lecture Notes in Computer Science*, pp. 159–178, Springer Berlin Heidelberg, 2014.

[31] "More personalization on bing with adaptive search." `http://www.youtube.com/watch?v=CgrzhyHCnfw`.

[32] Calado, P., Cristo, M., Moura, E., Ziviani, N., Ribeiro-Neto, B., and Gonçalves, M. A., "Combining link-based and content-based methods for web document classification," in *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, (New York, NY, USA), pp. 394–401, ACM, 2003.

[33] Castillo, C., Donato, D., Gionis, A., Murdock, V., and Silvestri, F., "Know your neighbors: Web spam detection using the web topology," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, (New York, NY, USA), pp. 423–430, ACM, 2007.

[34] Chakrabarti, S., "Data mining for hypertext: A tutorial survey," *SIGKDD Explor. Newsl.*, vol. 1, pp. 1–11, Jan. 2000.

[35] Chugh, R., Meister, J. A., Jhala, R., and Lerner, S., "Staged information flow for javascript," in *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '09, (New York, NY, USA), pp. 50–62, ACM, 2009.

[36] Daoud, M., Tamine-Lechani, L., and Boughanem, M., "A session based personalized search using an ontological user profile," in *The 24th Annual ACM Symposium on Applied Computing*, SAC '09, ACM, 2009.

[37] Dave, V., Guha, S., and Zhang, Y., "Viceroi: catching click-spam in search ad networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, CCS '13, (New York, NY, USA), pp. 765–776, ACM, 2013.

[38] Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B., and Sampath, D., "The YouTube video recommendation system," in *Proceedings of the 4th ACM conference on Recommender Systems*, RecSys '10, (New York, NY, USA), pp. 293–296, ACM, 2010.

[39] Dhawan, M. and Ganapathy, V., "Analyzing information flow in javascript-based browser extensions," in *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09, (Washington, DC, USA), pp. 382–391, IEEE Computer Society, 2009.

[40] Djeric, V. and Goel, A., "Securing script-based extensibility in web browsers," in *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, (Berkeley, CA, USA), pp. 23–23, USENIX Association, 2010.

[41] Dou, Z., Song, R., and Wen, J.-R., "A large-scale evaluation and analysis of personalized search strategies," in *The 16th International Conference on World Wide Web*, WWW '07, ACM, 2007.

[42] Edelman, B. and Brandi, W., "The ad networks and advertisers that fund ad injectors," in ***http://www.benedelman.org/injectors/***, 2013.

[43] Edelman, B. and Brandi, W., "Information and incentives in online affiliate marketing," in *HBS Working Paper*, 2013.

[44] Egele, M., Kruegel, C., Kirda, E., Yin, H., and Song, D., "Dynamic spyware analysis," in *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, ATC'07, (Berkeley, CA, USA), pp. 18:1–18:14, USENIX Association, 2007.

[45] Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., and Karger, D., "Infranet: Circumventing web censorship and surveillance," in *Proceedings of the 11th USENIX Security Symposium*, (Berkeley, CA, USA), pp. 247–262, USENIX Association, 2002.

[46] "Making search more relevant." http://www.google.com/goodtoknow/data-on-google/more-relevant/.

[47] Guha, A., Fredrikson, M., Livshits, B., and Swamy, N., "Verified security for browser extensions," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, (Washington, DC, USA), pp. 115–130, IEEE Computer Society, 2011.

[48] Gyöngyi, Z. and Garcia-Molina, H., "Link spam alliances," in *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pp. 517–528, VLDB Endowment, 2005.

[49] Ho, S. Y., Bodoff, D., and Tam, K. Y., "Timing of adaptive web personalization and its effects on online consumer behavior," *Info. Sys. Research*, vol. 22, pp. 660–679, Sept. 2011.

[50] Houmansadr, A., Nguyen, G. T., Caesar, M., and Borisov, N., "Cirripede: Circumvention infrastructure using router redirection with plausible deniability," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, (New York, NY, USA), pp. 187–200, ACM, 2011.

[51] Jindal, N. and Liu, B., "Opinion spam and analysis," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, (New York, NY, USA), pp. 219–230, ACM, 2008.

[52] Kashyap, V. and Hardekopf, B., "Security signature inference for javascript-based browser addons," in *Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization*, CGO '14, (New York, NY, USA), pp. 219:219–219:229, ACM, 2014.

[53] Kirda, E., Kruegel, C., Banks, G., Vigna, G., and Kemmerer, R. A., "Behavior-based spyware detection," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, (Berkeley, CA, USA), USENIX Association, 2006.

[54] Li, Z., Wang, X., and Choi, J. Y., "Spyshield: Preserving privacy from spy add-ons," in *Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection*, RAID'07, (Berlin, Heidelberg), pp. 296–316, Springer-Verlag, 2007.

[55] Lim, E.-P., Nguyen, V.-A., Jindal, N., Liu, B., and Lauw, H. W., "Detecting product review spammers using rating behaviors," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, (New York, NY, USA), pp. 939–948, ACM, 2010.

[56] Liu, F., Yu, C., and Meng, W., "Personalized web search by mapping user queries to categories," in *Proceedings of the eleventh international conference on Information and knowledge management*, CIKM '02, ACM, 2002.

[57] Liu, L., Zhang, X., Inc, V., Yan, G., and Chen, S., "Chrome extensions: Threat analysis and countermeasures," in *NDSS*.

[58] Lu, L., Perdisci, R., and Lee, W., "Surf: detecting and measuring search poisoning," in *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, (New York, NY, USA), pp. 467–476, ACM, 2011.

[59] Matthijs, N. and Radlinski, F., "Personalizing web search using long term browsing history," in *The Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, ACM, 2011.

[60] Mohajeri Moghaddam, H., Li, B., Derakhshani, M., and Goldberg, I., "Skypemorph: Protocol obfuscation for tor bridges," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), pp. 97–108, ACM, 2012.

[61] Mori, S., Nishida, H., and Yamada, H., *Optical Character Recognition*. New York, NY, USA: John Wiley & Sons, Inc., 1st ed., 1999.

[62] "Mechanic turk." `https://www.mturk.com/mturk/welcome`.

[63] Mukherjee, A., Liu, B., and Glance, N., "Spotting fake reviewer groups in consumer reviews," in *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, (New York, NY, USA), pp. 191–200, ACM, 2012.

[64] Mukherjee, A., Liu, B., Wang, J., Glance, N., and Jindal, N., "Detecting group review spam," in *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, (New York, NY, USA), pp. 93–94, ACM, 2011.

[65] Navarro, G., "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, pp. 31–88, Mar. 2001.

[66] "Nerdydata." `http://nerdydata.com/`.

[67] Ntoulas, A., Najork, M., Manasse, M., and Fetterly, D., "Detecting spam web pages through content analysis," in *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, (New York, NY, USA), pp. 83–92, ACM, 2006.

[68] Ott, M., Choi, Y., Cardie, C., and Hancock, J. T., "Finding deceptive opinion spam by any stretch of the imagination," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, (Stroudsburg, PA, USA), pp. 309–319, Association for Computational Linguistics, 2011.

[69] Pang, B. and Lee, L., "Opinion mining and sentiment analysis," *Found. Trends Inf. Retr.*, vol. 2, pp. 1–135, Jan. 2008.

[70] "Phantomjs website." `http://phantomjs.org/`.

[71] Qiu, F. and Cho, J., "Automatic identification of user interest for personalized search," in *The 15th International Conference on World Wide Web*, WWW '06, ACM, 2006.

[72] Rosenfeld, R., "Two decades of statistical language modeling: Where do we go from here," in *Proceedings of the IEEE*, 2000.

[73] "Bing results get localized & personalized." `http://searchengineland.com/bing-results-get-localized-personalized-64284`.

[74] "Google now personalizes everyone's search results." `http://searchengineland.com/google-now-personalizes-everyones-search-results-31195`.

[75] "A 2-second video." `http://www.youtube.com/watch?v=UPXK3AeRvKE`.

[76] Sieg, A., Mobasher, B., and Burke, R., "Web search personalization with ontological user profiles," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, ACM, 2007.

[77] "slate 0.4 : Python package index." `https://pypi.python.org/pypi/slate`.

[78] Sontag, D., Collins-Thompson, K., Bennett, P. N., White, R. W., Dumais, S., and Billerbeck, B., "Probabilistic models for personalizing web search," in *The Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, ACM, 2012.

[79] Springborn, K. and Barford, P., "Impression fraud in online advertising via pay-per-view networks," in *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, (Berkeley, CA, USA), pp. 211–226, USENIX Association, 2013.

[80] Sriram, S., Shen, X., and Zhai, C., "A session-based search engine," in *The 27th Annual International ACM SIGIR Conference*, SIGIR '04, ACM, 2004.

[81] Tan, C., Gabrilovich, E., and Pang, B., "To each his own: personalized content selection based on text comprehensibility," in *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, ACM, 2012.

[82] Teevan, J., Dumais, S. T., and Horvitz, E., "Personalizing search via automated analysis of interests and activities," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, ACM, 2005.

[83] Teevan, J., Liebling, D. J., and Geetha, G. R., "Understanding and predicting personal navigation," in *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, WSDM '11, ACM, 2011.

[84] Ter Louw, M., Lim, J., and Venkatakrishnan, V., "Enhancing web browser security against malware extensions," *Journal in Computer Virology*, vol. 4, no. 3, pp. 179–195, 2008.

[85] Tzeras, K. and Hartmann, S., "Automatic indexing based on bayesian inference networks," in *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, (New York, NY, USA), pp. 22–35, ACM, 1993.

[86] "Youtube channel, subscriber, & video statistics." `http://vidstatsx.com/`.

[87] Wang, G., Xie, S., Liu, B., and Yu, P. S., "Review graph based online store review spammer detection," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, (Washington, DC, USA), pp. 1242–1247, IEEE Computer Society, 2011.

[88] Weaver, N., Kreibich, C., Dam, M., and Paxson, V., "Here Be Web Proxies," in *Passive and Active Measurements Conference (PAM)*, (Los Angeles, CA, USA), March 2014.

[89] Weinberg, Z., Wang, J., Yegneswaran, V., Briesemeister, L., Cheung, S., Wang, F., and Boneh, D., "Stegotorus: A camouflage proxy for the tor anonymity system," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), pp. 109–120, ACM, 2012.

[90] Wustrow, E., Wolchok, S., Goldberg, I., and Halderman, J. A., "Telex: Anticensorship in the network infrastructure," in *Proceedings of the 20th USENIX Conference on Security*, SEC'11, (Berkeley, CA, USA), pp. 30–30, USENIX Association, 2011.

[91] Xing, X., Meng, W., Doozan, D., Feamster, N., Lee, W., and Snoeren, A. C., "Exposing inconsistent web search results with bobble," in *Passive and Active Measurement*, (Los Angeles, CA), pp. 131–140, September 2014.

[92] Xing, X., Meng, W., Doozan, D., Snoeren, A. C., Feamster, N., and Lee, W., "Take this personally: pollution attacks on personalized services," in *Proceedings of the 22nd USENIX conference on Security*, pp. 671–686, USENIX Association, 2013.

[93] Xu, X., Mao, Z. M., and Halderman, J. A., "Internet censorship in china: Where does the filtering occur?," in *Proceedings of the 12th International Conference on Passive and Active Measurement*, PAM'11, (Berlin, Heidelberg), pp. 133–142, Springer-Verlag, 2011.

[94] Yang, Y., "Expert network: Effective and efficient learning from human decisions in text categorization and retrieval," in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, (New York, NY, USA), pp. 13–22, Springer-Verlag New York, Inc., 1994.

[95] Yang, Y. and Chute, C. G., "An example-based mapping method for text categorization and retrieval," *ACM Trans. Inf. Syst.*, vol. 12, pp. 252–277, July 1994.

[96] Yang, Y. and Pedersen, J. O., "A comparative study on feature selection in text categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, (San Francisco, CA, USA), pp. 412–420, Morgan Kaufmann Publishers Inc., 1997.

[97] "Google does not use the keywords meta tag in web ranking." `http://www.youtube.com/watch?v=jK7IPbnmvVU`.

[98] Zhang, J. and Gu, G., "Neighborwatcher: A content-agnostic comment spam inference system," in *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS'13)*, February 2013.

[99] Zhou, D., Burges, C. J. C., and Tao, T., "Transductive link spam detection," in *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '07, (New York, NY, USA), pp. 21–28, ACM, 2007.