Game of Missuggestions: Semantic Analysis of Search-Autocomplete Manipulations

Peng Wang*, Xianghang Mi*, Xiaojing Liao†, XiaoFeng Wang*, Kan Yuan*, Feng Qian*, Raheem Beyah‡
*Indiana University Bloomington, †William and Mary, ‡Georgia Institute of Technology
*{pw7, xmi, xw7, kanyuan, fengqian}@indiana.edu, †xiaojing@wm.edu, ‡rbeyah@ece.gatech.edu

***Abstract*—As a new type of blackhat Search Engine Optimization (SEO), autocomplete manipulations are increasingly utilized by miscreants and promotion companies alike to advertise desired suggestion terms when related trigger terms are entered by the user into a search engine. Like other illicit SEO, such activities game the search engine, mislead the querier, and in some cases, spread harmful content. However, little has been done to understand this new threat, in terms of its scope, impact and techniques, not to mention any serious effort to detect such manipulated terms on a large scale.**

Systematic analysis of autocomplete manipulation is challenging, due to the scale of the problem (tens or even hundreds of millions suggestion terms and their search results) and the heavy burdens it puts on the search engines. In this paper, we report the first technique that addresses these challenges, making a step toward better understanding and ultimately eliminating this new threat. Our technique, called *Sacabuche*, takes a semantics-based, two-step approach to minimize its performance impact: it utilizes Natural Language Processing (NLP) to analyze a large number of trigger and suggestion combinations, without querying search engines, to filter out the vast majority of legitimate suggestion terms; only a small set of suspicious suggestions are run against the search engines to get query results for identifying truly abused terms. This approach achieves a 96.23% precision and 95.63% recall, and its scalability enables us to perform a measurement study on 114 millions of suggestion terms, an unprecedented scale for this type of studies. The findings of the study bring to light the magnitude of the threat (0.48% Google suggestion terms we collected manipulated), and its significant security implications never reported before (e.g., exceedingly long lifetime of campaigns, sophisticated techniques and channels for spreading malware and phishing content).

I. Introduction

Assume that you enter a query into Google search box. Figure 1 shows what you would see on May 12, 2017, under the search term "online backup free download". Before you can even finish typing, a list of *suggestions* pop up to help complete the query. This functionality is called *search autocomplete*, a service provided by search engines to enable users to conveniently formulate an effective query by providing only a small portion of it (called *trigger*), which is then complemented by a set of suggestions identified from common search terms

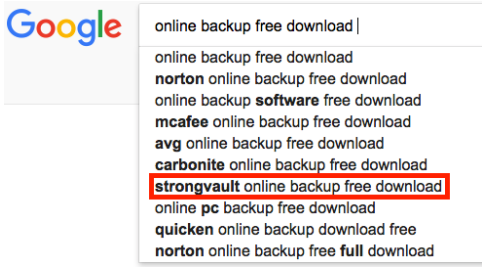


Fig. 1: Autocomplete manipulation on Google.

observed by the search engine [43]. This search assistant service, however, increasingly becomes the target for Spammers to perform illicit online promotion. As we can see in the figure, a suggestion being triggered is the name of an online backup software "strongvault", which turns out to be a potentially unwanted program. The software has malicious behaviors like installing adware, hooking the operating system and hijacking browser. However, Google recommended it as one of the most relevant suggestions for the trigger, since it was manipulated to promote the software.

Autocomplete manipulations. Such autocomplete manipulation takes advantage of the way a search engine ranks suggestions for a given trigger, which according to prior research [40], [57] and Google's own description [28], mainly relies on the popularity of queries observed from search logs. The content of the logs becomes biased when a large number of fake queries are crafted to promote a certain term (an organization, a product, etc.). This has already been done *systematically*, with the services available for selling to push promotional information (even illicit content) to the public (Section VI). Such activities are essentially a *new* type of blackhat *search engine optimization* (SEO) [15], which like other illicit SEO, not only reduces the quality of search results, with negative impacts on the search engine's reputation, but also opens an unchecked avenue for miscreants to broadly disseminate their attack content, infecting a larger victim pool than they could through free-riding the users' trust of search engines. Indeed, our research shows that almost *all* major search engines, including Google, Bing, Baidu, Yahoo! and Yandex, are victims of this attack. The parties offering such an SEO service typically have no restrict regulations about the legitimacy of the content to promote, with phishing, even malware discovered in our study (Section VI). Also, according to Google, 60% of today's searches are from mobile devices [1]. On mobile devices, due to their small form factors making them hard to type, users may highly rely on autocomplete. Therefore such an attack may cause even more damage to mobile users.

Network and Distributed Systems Security (NDSS) Symposium 2018
18-21 February 2018, San Diego, CA, USA
ISBN 1-891562-49-5
http://dx.doi.org/10.14722/ndss.2018.23036
www.ndss-symposium.org

To understand and further mitigate this emerging threat, a direct solution is to analyze the search logs, the sources for generating suggestions. This approach, however, can only be done by the search provider, like Google. Even given the logs, a thorough analysis of the massive amount of data is by no means trivial. So far, the only effort made is a prior study that utilizes Yahoo!'s search logs and a set of confirmed manipulations to identify the promotion campaigns, for the purpose of monitoring their other searches [50]. With all the resources, the study only achieves a moderate success in terms of understanding the threat (only 3,000 terms inspected), not to mention its fragility to the evasion performed by those continuously changing their IP identifiers. A truly large-scale study, involving millions of search terms, by the parties without access to search logs, cannot be done with the existing techniques. As a result, so far we know little about the real-world impacts of these illicit promotions, and could do almost nothing to protect ourselves against such a threat, in the absence of search engines' efforts.

Detecting missuggestions with semantics. In this paper, we report the *first* large-scale study on autocomplete manipulation, based upon novel techniques to automatically discover abused suggestions without access to query logs. Our approach, called *Sacabuche* (Search AutoComplete Abuse Checking), leverages a set of unique observations that lead to the features the manipulators cannot easily evade. More specifically, we found that for a query involving manipulated suggestion, there often exists *semantic inconsistency* between the *trigger phrases* (keywords inside a trigger) and their corresponding *target phrases* (keywords in the suggestion). For example, for the trigger "what is content delivery network", a keyword "kajabi" appears in its suggestion; the problem here is the semantics of the keyword is very different from "content delivery network", using the word-embedding technique [7], since the word rarely shows up together with the trigger phrases. Also observed in our research are other differentiating features: a legitimate suggested term tends to be more generic, e.g., "reviews", "list" or "companies", while a manipulated one is often more specific, for the purpose of promoting a particular product; besides, the grammatical structure between a legitimate trigger-suggestion pair and a manipulated one can differ. Using these features, we trained a classifier that automatically detect suspicious terms from the autosuggestions iteratively retrieved from Autocomplete APIs [19][14][36] provided by Google, Bing and Yahoo!, using a set of seed triggers. In this way, we avoid massively querying the search engines, which would negatively affect its performance.

This screening step effectively removes the vast majority of legitimate suggestions (from over 114 million terms down to only 1 million (less than 1%)). The remaining, however, can no longer be analyzed by only analyzing the semantics of trigger and suggestion. For example, consider the trigger "products and services example" and the suggestion "b2b products and service examples"; the term "b2b" seems to be irrelevant to the trigger, but turns out to be related. Finding such false positives needs more information, that is, the content of search results. To this end, our approach automatically crawls search engines with these suspicious terms, extracting a set of salient features from the *search results*. For example, one of such features measures is the number of results indexed by search engines: even though the manipulator could forge a large number of queries, it becomes much harder to create many relevant results indexed by the search engines; therefore this feature contributes to distinguish truly problematic suggestions from legitimate ones. Using these features (Section IV-C), we run another classifier to capture illicit suggestions. Our study shows this two-step approach is very effective, with a precision over 96.23% and a recall over 95.63%. Also, the approach enables us to massively process over 114 million suggestions, an unprecedented scale for an in-depth understanding of this emerging threat.

We note that, working on search autocomplete manipulation detection brings in the challenge of result validation, which is hampered by the difficulty in obtaining ground truth. Specifically, the detected autocompletes include a large volume (estimated 95% based on sampling) of results containing unpopular products with trigger keywords, which is a common trait of the ground truth autocompletes that we observed.

Our findings. Looking into the manipulated autocomplete results reported by Sacabuche, we are surprised to find that this new threat is indeed pervasive, having a large impact on today's Internet. More specifically, over 383K manipulated suggestions (across 257K triggers) were found from mainstream search engines, including Google, Bing and Yahoo!. Particularly, we found that at least 0.48% of the Google autocomplete results are polluted. The security implications of the attack are significant. For example, our study shows that at least 20% of these manipulation cases are used for underground advertising, promoting content as gambling and even malware. We also discovered that 3K compromised sites within top-10 search results were actually related to the manipulated autocompletes.

Also interesting is the ecosystem of autocomplete manipulation, as discovered in our study, including the promotion and evasion techniques being used, the parties involved and the way they share revenues and others. As an example, we analyzed *Seopmb*[29], a suggestion manipulation tool that automatically simulates user behaviors to generate search queries, and found that the party using the tool needs to pay rewards points to its developer whenever promoting suggestions, and the party can also receive rewards for running promotional tasks for others. Further, although the website for the tool runs in USA (Psychz Networks Ashburn), the server it communicates with is located in Hong Kong. Such communication is encrypted to evade detection. Also, such autocomplete manipulation campaigns apparently are quite successful in evading detection: the manipulated terms typically have a quite long lifetime (34 days), and also new ones counting for 3.7% of all manipulated terms appear on a daily basis. Our study further investigates the potential profit of this underground business, using estimated click-through rate, which shows that the average revenue per month for a service recipient is $95 per keyword.

Contributions. The contributions of the paper are as below:

• *New techniques for detecting missuggestions*. We designed and implemented the first system that enables the party without access to search logs to perform efficient and highly accurate missuggestion detection. Our new techniques are built on top of novel applications of NLP techniques, which significantly raise the bar for the attack and control the assumption of resources on the search engine side, a critical feature for achieving the scalability of detection.

• *New understanding of autocomplete manipulation*. Using our new techniques, we conducted the first large-scale analysis of autocomplete manipulation, through automatically discovering and evaluating over 114 million search terms from major search engines. For the first time, our study reveals the pervasiveness of this emerging threat and its significant security impacts. Further we discovered the adversary's strategy and made the first step toward understanding the ecosystem of this underground business, which is invaluable for mitigating and ultimately eliminating the threat.

Roadmap. The rest of the paper is organized as follows: Section II provides the background information; Section III describes our findings from the ground truth dataset; Section IV elaborates the design and implementation of our detection system and the evaluation of its effectiveness is described in Section V; Section VI reports our large-scale measurement study and our findings; Section VII discusses the limitations of the study and potential future research; Section VIII reviews the related prior research and Section IX concludes the paper.

II. Background

Before moving into the details of our study, here we explain how autocomplete works and the NLP technologies utilized in our research, together with assumptions made in our research.

Search autocomplete. As mentioned earlier, search autocomplete is a query prediction service, which offers suggestions based upon what others are searching for. Such a service is designed to speed up human-computer interactions, making it more convenient for users to complete her query, which is particularly useful to less technical-savvy individuals and those handicapped. In the meantime, the statistics collected from query logs guide the user to better formulate her queries.

Such an autocomplete service is built upon prediction algorithms that automatically identify relevant terms from the query logs [28]. From the information publicly available, search engines select suggestions according to the popularity of their related queries: that is, the frequencies of the phrases searched by different people and their freshness (how trending they are as popular topics). Essentially, such search predictions are meant to capture public interests manifested by human queries. So all search providers are against the activities that game their algorithms for advertising less relevant content, which is actually a new type of *blackhat SEO*. For example, Google is reported to continuously monitor suggestion Spam activities and change its prediction algorithm to make them harder to succeed[9], and also explicitly block all abusive suggestions, including links to malware, phishing and discriminative, offensive words[11], though the effectiveness of this effort can be limited (Section VI-A); similarly, Bing is trying to filter suggestion Spam, adult and offensive prediction words [8]. Detecting and understanding such autocomplete Spam is therefore the focus of our study.

Note that autocompletes may be personalized according to the client's location and search history. In some cases, people won't really come across the poisoned suggests. Although the missuggestions detected by us may not show up to some users, as a first step, we mainly looked into English suggestions (keywords in other languages also show up, see Section IV-B), changed our search locations and cleaned up search history after each query to collect suggestions as broadly as possible.

Natural language processing. To automatically identify manipulated suggestions from their semantics, we utilized a set of NLP technologies, as summarized in the following.

Word embedding. Word embedding $W : words \rightarrow V^n$ is a parameterized function mapping each word to a high-dimensional vector (200 to 500 dimensions), e.g., $W(\text{“}education\text{”}) = (0.2, -0.4, 0.7, ...)$, to represent the word's relation with other words. Such a mapping can be done in different ways, e.g., using the continual bag-of-words model [2] and the skip-gram technique [30] to analyze the context in which the words show up. Such a vector representation is designed to ensure that synonyms are given similar vectors and antonyms are mapped to dissimilar vectors. In our research, we compared the semantics meanings of different words by measuring the cosine distance between the vectors. For example, word embedding technique automatically identifies the words semantically close to "casino", such as "gambling" (cosine distance 0.35), "vegas" (0.46) and "blackjack" (0.48). We leveraged a popular word embedding tool, *Word2Vec* [34], which runs an artificial neural network to construct the model for generating the vectors.

Dependency parsing. Dependency parsing is an NLP technique for describing grammatical relations among words in a sentence. Such relations include direct object, determinant, noun compound modifier and others. Also, the content of a relative clause is further analyzed to identify the dependencies between the words it includes. The state-of-the-art dependency parser (e.g., Stanford parser [6]) can achieve a 92.2% accuracy in discovering the grammatical relations in a sentence.

Lemmatization. A natural language document always includes words in different forms, due to tenses, abbreviations and grammatical needs, e.g., "organize", "organizes", and "organizing". Further, there are derivation words with similar meanings like "slow" and "slowness". We need to find out the original form of each word, then link their appearances across different terms. This was done using *lemmatization* techniques, which reduce inflectional forms and remove inflectional endings and return the base or dictionary form of a word. A common lemmatization algorithm is morphological analysis [55] to find out the lemma (i.e., organize) for each word (e.g., "organizes", and "organizing"). The state-of-the-art algorithm (e.g., WordNetLemmatizer) can achieve 95% of accuracy [35].

Adversary model. In our research, we consider that an adversary is capable of forging a large number of queries, through various sources/locations, to promote illicit, unwanted or unrelated content. It renders the detection approach based upon IP identifiers less effective. Also, we do not assume the availability of query logs since they are hard to get (except for search service providers), hard to share (due to privacy concerns) and hard to analyze (given the massive amount of data they contain). Also, the techniques designed for this setting are meant to provide the end user immediately protection, even before the search engines start acting on the threat. On the other hand, we assume that it is difficult for the adversary to produce a large amount of web content, distribute it across many reputable websites to be indexed by search engines. This certainly needs more resources than faking queries.

III. Understanding Missuggestions

In this section, we first present an autocomplete manipulation ecosystem to explicate how missuggestions are produced. Then, we elaborate on our analysis of a set of known missuggestions and the features identified for differentiating benign suggestions and those manipulated ones. These features are utilized to build a scanner, Scacbuche, for detecting other autocomplete manipulations.

A. How is Autocomplete Manipulated: An Example

Autocomplete manipulation has been extensively used by miscreants for illicit activities. To understand how it works, we describe the ecosystem of this emerging SEO business discovered in our study, through purchasing from and interacting with the service providers. Figure 2 presents a high-level view about how such services operate, and how the different entities involved interact with each other. First, a manipulation service client (a.k.a., promoted site owner) sends a request to a service provider (❶). Such a request includes the trigger, the suggestion and the promotion period. Then, the provider creates a query task for promoting the missuggestion (trigger+suggestion) and distributes it to his crowd-sourcing operators (❷), or directly generates queries to search for the targeted suggestion (❸). Once the missuggestion successfully gets into the search engine, a user who follows the search results of the suggestion (❹) and visits the promoted site (❺) could be redirected to a phishing site (❻). Accordingly, the manipulation service provider will get the commission from the promoted site owner, while the promoted site also gain traffic that can be monetized.

As an example, *iXiala* [24] is an autocomplete manipulation system, which provides manipulation service on 19 different platforms (web and mobile), including search engines (Baidu, Sogou, 360), C2C (Taobao) and B2C (Tmall, JD) platforms. To change autocomplete results and display desired phrases, a user can simply add a block of JavaScript onto his website.

```
<script type="text/javascript" src="http
    ://www.ixiala.com/Cache/showJs/id/8
    c0e065e3d6b3e4a891c0b60b9a45cf7ce9169ca
    .html" async="async"></script>
```

The script creates a large number of iframes on the website. Whenever the site is visited, these iframes automatically fetch suggestion phrases from the *iXiala* platform, and submit great amount of suggestions to the search engines. When searching for the websites containing such promotional JavaScript code on PublicWWW [31], a public website source code database, we found around 10K websites such as *by91.com*, *ael920.com*, and *dytbj.com* using such a script.

From the suggestions promoted by the *iXiala*, such as "fresh air purification system filtech" and "Sanya hotels recommendation - *liba.com*", we found that the clients try to advertise their specific but rarely known products, which are less in line with the trigger's semantics than the benign suggestions are supposed to be. Accordingly, the search results of manipulated suggestions focus on such promoted products, which are often semantically inconsistent with the trigger.

These findings indicate that missuggestion has distinctive features, particularly semantic inconsistency between trigger terms and suggestions, and inconsistency in the search results

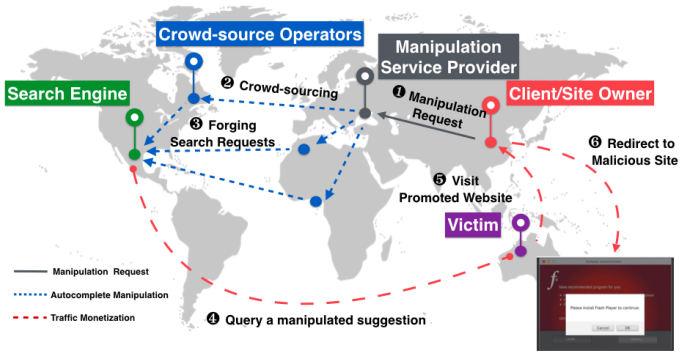


Fig. 2: Operational steps of an autocomplete manipulation ecosystem. First, a client sent manipulation request to the manipulation service provider (❶); Then, the service provider distributed the query task to his crowd-sourcing operators (❷) to search for the targeted suggestion (❸); Once a victim searched the manipulated suggestion (❹) and visited the promoted site (❺), he will be redirected to a phishing site (❻).

TABLE I: **Summary results of the datasets.**

	# of suggestions	# of linked triggers	# of result pages
Badset	150	145	295
Goodset	300	298	593
Unkown set	114,275,000	1,000,900	1,607,951

of trigger terms (without suggestions) and those for missuggestions. These features were utilized in our research for detecting the manipulated autocomplete.

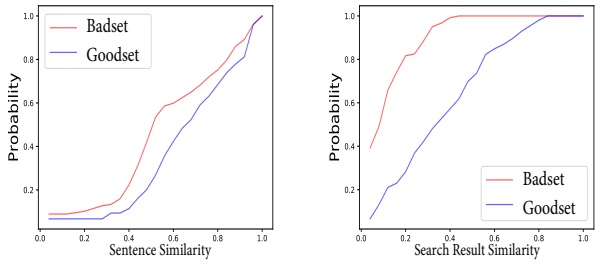
B. Features of Missuggestions

In our study, we identified a set of features that uniquely characterize missuggestions. Here we describe the high-level idea of utilizing the features to capture the manipulations.

Data collection. We first collected a set of confirmed missuggestions (called badset) and legitimate suggestions (called goodset) as well as their related search results, which are illustrated in Table I. Here we describe them in details.

• *Badset*. The badset includes 150 manipulated suggestion phrases and their corresponding trigger phrases posted online by autocomplete manipulation companies (such as Affordable Reputation Management[13] and yincmarketing[37]) to promote their services. We further validated them through manual inspections (e.g., looking for illicit content in search result pages) to ensure that they were indeed bad and live.

• *Goodset*. The good suggestions were gathered using 1,000 trigger, which were randomly chosen from 1 million most popular keywords reported by MFA [10]. The triggers cover a wide range of search interests, over ten categories (technology, education, financial service, etc.). For each trigger, we took its top 3 suggestions from the search engines. From all these 3,000 search terms (trigger+suggestion), we further selected 300 that were manually confirmed to be legitimate, using the validation criteria elaborated in Section V.



(a) **Cumulative distribution of sentence similarity per trigger/suggestion pair.** (b) **Cumulative distribution of search result similarity per trigger/suggestion pair.**

Fig. 3: **Discriminative features of missuggestions and benign suggestions.**

• *Suggestion related search results*. We collected search results using the search engine API (e.g., Google Search API), with both aforementioned trigger phrases and their search terms as inputs. For each query (either a trigger or a search term with both a trigger and a suggestion), its top 20 results indexed by the search engine were recorded, including the titles, urls and the descriptions.

Semantic inconsistency. Our key observation is that *a trigger and its suggestion are often less related when the autocomplete has been manipulated*, presumably due to the fact that a missuggestion tends to promote a specific and rarely known product, which is less relevant to the corresponding trigger. For example, "play free bingo online now at *moonbingo.com*" and "free bingo sites for us players" are both suggestions for the trigger "bingo sites play free". However, the former, which is manipulated, is more specific (promoting *moonbingo.com*, a bingo site), and therefore less similar to the trigger.

Such a semantic gap is leveraged by Sacabuche to differentiate legitimate suggestions from manipulated ones. More specifically, we utilize the Word2Vec technique to compare the semantic meanings of a trigger and its suggestion (which can have several words and a sentence-like structure), in terms of sentence similarity $\mathbf{F}_{ss}$. Given two sentences s^a and s^t, we convert them into two phrase lists $pl(s^a)$ and $pl(s^t)$ through dependency analysis. Each phrase is identified from the *dependency tree* [4] representation of a sentence, which describes the grammatical relations among different words in a sentence. Over the tree, we search for the directed paths with a length of two, which connect two non-preposition and non-article words together. All such phrases are extracted from each sentence to form its phrase list. The phrase lists of the trigger and the suggestion are then compared using the sentence kernel $\mathbf{SK}$ defined over a phrase kernel $\mathbf{PK}$, which is further built on a word embedding based word kernel $\mathbf{WK}$. We detail the sentence-level semantic inconsistency features in Section IV-B.

Such a semantic inconsistency feature was found to be discriminative in our research. Figure 3(a) compares the cumulative distribution function (CDF) of the sentence similarity between the badset and goodset. As we can see from the figures, missuggestions tend to have lower sentence similarity than benign ones: the average sentence similarity is 0.56 for the missuggestions and 0.67 for the legitimate ones.

Search Result Inconsistency. In addition to the semantic inconsistency features, we found that the search results of the missuggestions often show a remarkable inconsistency with their corresponding triggers, while the good ones will be in line with those triggers. This is because a manipulated suggestion is meant to affect the way the search engine prioritizes search results, making promoted content more prominent in the results. Figure 4 illustrated the search result inconsistency of the missuggestion and the benign suggestion. For the search result of the benign suggestion "norton online backup free download", they were similar to those of the trigger "online backup free download" (e.g., the second search result), while none of the search results of the missuggestion "strongvault online backup free download" appeared in trigger's top 20 search results.

Specifically, we measure the similarity of two ranked domains lists, one retrieved from the search engine under the trigger alone and the other under the whole query term (the trigger and a suggestion). The similarity is calculated by *Rank-Biased Overlap* (RBO) function [59], which was designed to weigh high-ranking items more heavily than those down the lists, handle nonconjointness and be monotonic with the increasing depth (i.e., ranks) [59] (see Section IV-C for details).

Figure 3(b) compares the CDF of the domain list similarity between the search result pages of the badset and goodset. As we can see from the figure, missuggestions tend to have a lower search result page similarity than benign ones in term of the domains in search result pages: the average similarity is 0.08 for the missuggestions and 0.33 for the legitimate ones.

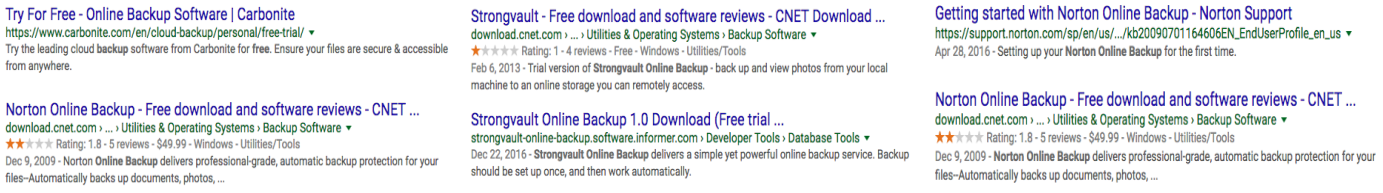
In our research, we characterize the semantic inconsistency and search result inconsistency in multiple perspectives, besides the sentence similarity and search result's domain list similarity. Such inconsistencies, fundamentally, are introduced by the attackers who tend to promote less popular products than the autocomplete service expects, making the promoted content more prominent in the results. Therefore, the inconsistencies are inherent to the attack strategies and can be hard to change.

IV. The Design of Sacabuche Framework

In this section, we elaborate the technique we used to detect missuggestions, starting with an overview of the idea behind Sacabuche, which is followed by its design and implementation details, and our evaluation of the implementation.

A. Overview

To catch manipulated suggestions, our idea is to exploit the gap between the semantics of legitimate and manipulated predictions in a scalable way. Such a gap can be immediately observed from the semantic inconsistency and other semantic features of some trigger-suggestion pairs, without even looking at their query results. In the meantime, the results provide further evidence for the presence of inconsistency: e.g., the prediction term supposed to be popular vs. the actual small number of search results reported for the term. Leveraging these observations, we build an efficient two-step detection: first filtering out the vast majority of the legitimate predictions having no signs of semantic inconsistency, without performing expensive search queries, and then analyzing the search results of a relatively small set of suspicious trigger-suggestion pairs to identify the manipulated ones.



(a) **"online backup free download".** (b) **"strongvault online backup free download".** (c) **"norton online backup free download".**

Fig. 4: **"strongvault" appears in (b) but rarely in (a), while "norton" appear both in (a) and (c).**

Architecture. Figure 5 illustrates the architecture of Sacabuche, including *Prediction Finder* (PF), *Search Term Analyzer* (STA) and *Search Result Analyzer* (SRA). PF is designed to discover a large number of autosuggestions: in particular, it iteratively queries the search engines with depth limit to 3, starting from a set of seed triggers (Section V-A) as the inputs, to derive a great number of autocompletes. These suggestions are further analyzed by STA, which looks at a set of semantic features to identify *suspicious* terms (Section IV-B). Such suspicious terms are then queried against the search engines by SRA, and their results are inspected, based upon their content features, to capture manipulated predictions (Section IV-C).

Example. Here we use an example to explain how Sacabuche works. From a seed trigger "online backup free", our approach discovers its suggestion terms "online backup free mac", "online backup free download" (①), etc., which are further used as triggers to find more suggestions "best online backup free mac" (②), "norton online backup free download" (③),"strongvault online backup free download" (④), etc., as showed in Figure 4. Among these suggestions, ③ and ④ are considered to be suspicious by STA, since there is quite a semantic distance between the triggers and their corresponding suggestions ("norton" and "strongvault" does not seem to be semantically related with the trigger ① and also these suggestions are specific. Both terms are then further queried on search engines like Google. From their search results, SRA determines, through a trained classifier, that indeed there are evidences to believe that ④ is a manipulated term: particularly, under the trigger ①, the presumably popular prediction "norton" appears frequently in the top 20 search results, while "strongvault" does not even show up among top 100 search results of the trigger. As a result, this term is reported as problematic.

B. Semantics-based Search Term Screening

As mentioned earlier, Sacabuche first analyzes the semantics of individual query terms to identify those suspicious. These terms are discovered by PF and pre-processed at this step, before they are inspected by STA. The inspection involves extraction of a set of semantic features from the terms and running of a classifier to find suspicious ones based upon the features. Here we elaborate these two steps, particularly individual features utilized by STA.

Suggestion discovery and pre-processing. The autocomplete predictions generated by search engines can be directly obtained from them, e.g., from the suggestion lists triggered by the input to their search boxes. A popular search engine typically provides an API for collecting its complete suggestions with regard to a given trigger term. In our research, we utilized over 1 million popular search keywords (see Section V-A) as "seed" triggers and ran PF to iteratively query the search engine, with all the suggestions discovered in a round serving as the triggers for the queries in the next round. In this way, our approach is able to expand these seed terms and find out not only their suggestions but those of their related terms: for example, from "free online games casino", we can get other terms in the gambling category, such as "slot machine", "roulette", "blackjack", and their suggestions.

The discovered suggestions and their corresponding triggers form search terms. Before handing them over STA, they need to be pre-processed to remove noise and further identify meaningful semantic tokens, as follows:

URL formatting. From each trigger and suggestion, our approach identifies the URLs in it and tokenizes the URLs. For examples, the link https://www.subdomain.example.com/path1/path2?q1=v1&q2=v2 is converted into the following tokens: *https www subdomain example com path1 path2 q1 v1 q2 v2*.

Content sanitization. All tokens within a query term (a trigger-suggestion pair) are then inspected to remove those less useful for the follow-up semantic analysis. We remove special words and phrases such as numbers, locations and stop words (extremely common terms such as "the"). These tokens (words or phrases) introduce noise to our semantic analysis, since they are too common to be useful for differentiating legitimate suggestions from illicit ones. Specifically, we identify the stop words from query terms using stopword datasets [32] and location tokens, which are also not informative for our purpose, from a geographical database [18], which includes the names (in different forms such as abbreviations) of over 11 million locations across all countries.

Lemmatization. We further lemmatize each word within a search term to its basic form: for example, "stopped" to "stop", "best" to "good" and "companies" to "company". Specifically, we firstly assign POS (part-of-speech) tag to each token in the given sentence (the search term) using Python NLTK package [27]. For example, we can identify "stopped" as verb and "best" as adjective using POS tagger. POS tagging is to provide the contextual information that a lemmatizer needs to choose the appropriate lemma. Then, we run a lemmatizer [35] to map a search term to its basic form.

Once STR receives pre-processed query terms, it extracts semantic features from them for detecting suspicious terms. These features characterize the semantic gap between a trigger and its suggestion term, which are supposed to be related but often less so when the autocomplete has been manipulated.

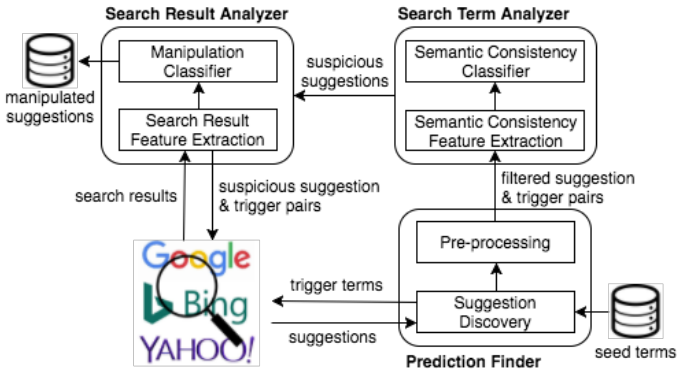


Fig. 5: Overview of the Sacabuche infrastructure.

Semantic consistency features. As mentioned before (see Section III-B), we utilized a word-embedding technique based sentence-level semantic similarity comparison to capture the semantic gap between the trigger s^a and its suggestion s^t. Given two sentences, the kernel converts them into two phrase lists $pl(s^a)$ and $pl(s^t)$, through a dependency analysis. **SK** is the sum of $PK(p_i^a, p_j^t)$, where p_i^a is a phrase in the ph^a (with $1 \leq i \leq |ph^a|$) and p_j^t is a phrase in ph^t (with $1 \leq j \leq |ph^t|$). $PK(p_i^a, p_j^t)$ is further calculated using a word kernel **WK**, by simply multiplying $WK(w_1^a, w_1^t)$ with $WK(w_2^a, w_2^t)$, and the word kernel $WK(w_i, w_j)$ directly runs the word embedding technique on word w_i and w_j respectively to convert them into vectors and then computes the cosine distance between the vectors. Once a sentence kernel value $\mathbf{SK}(s^a, s^t)$ (for s^a and s^t) is calculated, we normalize it by using $\sqrt{\mathbf{SK}(s^a, s^a)\mathbf{SK}(s^t, s^t)}$ to divide it. This normalization step is necessary for the fairness in comparing the semantic similarities across multiple sentence pairs, since the length of sentences vary, affecting their similarity values.

The whole similarity comparison is summarized as follows:

$$\mathbf{WK}(w_i, w_j) = \left[\frac{1}{2}(1 + cosineSim(w_i, w_j))\right]^{\alpha}$$

$$\mathbf{PK}(p^a, p^t) = \prod_{i=1}^{len} \mathbf{WK}(w_i^a, w_i^t)$$

$$\mathbf{SK}(s^a, s^t) = \sum_{\substack{p^a \in pl(s^a) \\ p^t \in pl(s^t)}} \lambda^2 \mathbf{PK}(p^a, p^t)$$

$$\mathbf{F}_{ss}(s^a, s^t) = \frac{\mathbf{SK}(s^a, s^t)}{\sqrt{\mathbf{SK}(s^a, s^a)\mathbf{SK}(s^t, s^t)}}$$

where λ is a decay factor of the sentence kernel, α is a scaling factor of the word kernel and F_{ss} is a feature value of sentence similarity.

We found that the standard word embedding model trained by Google Word2Vec team using part of Google News Dataset turns out to be less effective in covering autocomplete data, only 8.89% of the words in the trigger-suggestion pairs we gathered from the Google API. So in our research, we trained a different model using the training sets with Wikipedia documents in nine languages. The new model achieves a coverage of 36.15% in terms of *unique* words in our autocomplete dataset. The remaining 63.85% are all low-frequency words, counting for only 0.55% of the total words in the autocomplete dataset with average frequency of only 1.21 per query term. Further analysis of them reveals that among all the 605,556 uncovered words, more than 340K are URLs, and the remaining are rare words like "hes", "stongvault", etc.

TABLE II: **F-score of features.**

Label	Feature	F-score
$F_{ss}(s^a, s^t)$	sentence similarity	0.597
$F_{ws}(w^a, w^t)$	word similarity	0.741
$F_{if}(w^a, w^t)$	infrequent word similarity	0.653
$F_{rs}(D^a, D^t)$	result similarity	0.782
$F_{ci}(w^a, H^a, H^t)$	content impact	0.808
$F_{rp}(D^a, D^t)$	result popularity	0.632
$F_{rs}(N^a, N^t)$	result size	0.745

The sentence-level similarity alone is insufficient for detecting some manipulated suggestions. As an example, let us look at the trigger-suggestion pair: ("online backup free download", "strongvault online backup free download"). Here, an unpopular term "strongvault" is promoted onto the Google suggestion list, for advertising the malware "strongvault". The overall similarity between the trigger and the suggestion here is high, given that most of the words they include are identical. Semantic inconsistency in this case has been captured when comparing "strongvault" with other trigger words "online", "backup", "free" and "download". To identify such suggestions, Sacabuche performs a fine-grained semantic consistency check at the word-level. Specifically, for each suggested word, we run the word-similarity kernel **WK** to compare it against every trigger word, which results in a value $\underset{each\ j}{AVG}(\mathbf{WK}(w_i^a, W_j^t))$ for the word w_i^a to describe its average semantic distances with regard to all the trigger words $\{W_j^t | 1 \leq j \leq |s^t|\}$. From these values, the one with the maximum average distance is selected as the features F_{ws} to capture the word-level semantic gap between the trigger and the suggestion.

In addition to the semantic features, leveraging our observation that manipulated predictions are a set of words rarely appearing in legitimate suggestions: for example, "hes" and "stongvault", we further measure the *infrequency level* (*Infreq*) for each suggestion word as follows:

$$F_{if}(w^a, w^t) = \frac{\underset{each\ j}{MAX}(9 - log_{10}Freq(w_j^t))}{\underset{each\ i}{MAX}(9 - log_{10}Freq(w_i^a))}$$

where $Freq(w)$ is the frequency of word w in our Word2Vec model. w_j^t is each word in trigger, and w_j^a is each word in suggestion. F_{if} is utilized alongside sentence and word-level similarities as the differentiating features for detecting manipulated suggestions. Their differentiating power, in terms of F-scores, are illustrated in Table II, which were calculated using our ground truth dataset (the goodset and the badset).

Learning and classification. Using these features, we trained a support vector machine (SVM) classification model over the ground truth dataset, including 100 manipulated trigger and suggestion pairs and 150 legitimate pairs (this dataset was carefully selected so that pairs with different labels can

be distinguished without referring to the search results). This model was evaluated through a 5-fold cross validation, and achieved a precision of 94.59% and a recall of 95.89%. We further evaluated the technique over an unknown dataset, which is elaborated in Section V.

C. Search Result Analysis

The query-term screening step performed by STA is designed for scalability, filtering through a large number of autocomplete terms without actually querying the search engine. The suspicious suggestion terms discovered at this step, however, need to be further validated by looking at their search results, so as to collect more evidence to support the finding and remove false positives. This is done by the SRA module, through a set of features identified from a series of *differential analysis*, which compare the search results with and without suggestion terms. It models the observations that a manipulated suggestion tends to have a larger impact on search results, in terms of promoting its target content, but be less diverse in search results and also bring in fewer results than expected for a legitimate popular query term. *Note that even though each single feature may not necessarily fully capture manipulated suggestions, strategically leveraging multiple features in a collective manner helps achieve very high detection accuracy, as to be demonstrated in Section V.*

Search result impact. Among the suspicious suggestions reported by STA, those truly manipulated have large impacts on search results, distinguishing them significantly from what are reported by the search engine under the trigger term only. This deviation is measured in our research through two features: *result similarity* that compares the search results of a trigger with and without a suggestion and *content impact* that checks the relations between suggestion words and search content to understand their impacts on search outcome ranking.

A manipulated suggestion is meant to affect the way the search engine prioritizes search results, making promoted content more prominent in the results. As a consequence, the results reported under such a term will be less in line with those under the trigger (without suggestions), compared with the results of legitimate suggestions. To measure such discrepancy, we utilize a *Rank-Biased Overlap* (RBO) function [59] to evaluate the similarity of two ranked lists of domain names, one retrieved from the search engine under the trigger alone and the other under the whole query term (trigger +suggestion). The RBO measurement is characterized by its design to weigh high-ranking items more heavily than those down the lists, handle nonconjointness and be monotonic with the increasing depth (i.e., ranks) of evaluation [59]. Specifically, let $\{D_i^t : 1 \leq i \leq d\}$ be the element at rank i on the list D^t (the domain list for the trigger). At each depth d, the intersection of lists D^t and D^a (for suggestion) is $I(D^t, D^a)^d = D^t_{1:d} \cap D^a_{1:d}$. The proportion of two search results overlapped at depth d is defined as their agreement $A(D^t, D^a)^d = \frac{|I(D^t, D^a)^d|}{d}$. So we get the two search results' rank-biased overlaps as below:

$$F_{rs}(D^a, D^t) = (1-p)\sum_{d=1}^{\infty} p^{d-1} A(D^t, D^a)^d$$

where p is a decay value for tuning the weights for different depths d. The smaller p is, the more biased toward higher ranked items the metric becomes. Once $p = 0$, only the item on the top of the list matters. This metric is convergent and its value falls in the range [0, 1] where 0 is disjoint, and 1 is identical.

Also measuring suggestion impacts is *content impact*, which evaluates the relations between suggestion words and the titles of individual search result items (see Figure 4), with and without a given suggestion when querying the search engine. We utilize the aforementioned function, RBO, for differential analysis. The purpose is to understand whether the suggested content becomes more prominent under the suggestion words (to the extent they are more frequently present in the search result titles), which is a *necessary* condition for suggestion manipulation. More specifically, for the search results under a trigger only, our approach looks for the distribution of suggestion words across the titles of the result items, and compares it to the distribution of the same set of suggestion words over the result titles under the whole query term (the trigger and the suggestion) using RBO. Formally, let $H = \{h_i | 1 \leq i \leq |H|\}$ be a title list retrieved from a search result page, where h_i is the ith title on the list. Given a suggestion word w_i^a, its impact on h_i is 1 if the word shows up in the title and 0 otherwise. In this way, its distribution over H can be described by a binary vector $V(w_i^a, H)$. The content impact R of w_i^a is then calculated as the distance between two ranked lists: $R(w_i^a, H^a, H^t) = RBO(V(w_i^a, H^a), V(w_i^a, H^t))$. Based upon the impact of individual word in a suggestion, we identify an *content impact feature* for detecting manipulated suggestion as follows:

$$F_{ci}(w^a, H^a, H^t) = \underset{each\ i}{Min}(R(w_i^a, H^a, H^t))$$

where w_i^a is the suggestion term.

Result popularity and size. Further we found that fewer results are reported by the search engine when a truly manipulated suggestion is queried and the domains involved in the results often have low Alexa ranking. The observation is characterized by two collective features: *search result popularity* that compares the popularity of search results under both trigger and suggestions, and *search result size*, which captures the number of results that Google can find for a given query.

We define the popularity of a given domain D_i as $P(D_i) = 1 - log_{10} AlexaRanking(D_i)/7$, where $AlexaRanking(D_i)$ is the rank of a domain D_i in Alexa Top 1 million. If the domain does not show up among the top 1 million, we set its popularity to $\frac{1}{7}$. Again, here we perform a differential analysis and define a popularity vector AP with its dth element being the average popularity of the top d domains on the search result page, i.e., $AP(D) = [\underset{i \in [1:d]}{AVG}(P(D_i)) | 1 \leq d \leq |D|]$. Let AP^t be the vector for the results of querying a trigger only and AP^a be the vector for querying both trigger and suggestion. The search result popularity is calculated as follows:

$$F_{rp}(D^a, D^t) = RBO(AP^a(D^a), AP^t(D^t))$$

From each search result page, SRA further retrieves the number of the results discovered by the search engine. Based upon such information, we identify a feature F_{rs}. Let N^t be the number of results found under the trigger, and N^a be the number found under the trigger and a suggestion. The feature is calculated as $F_{rs}(N^a, N^t) = \frac{N^a - N^t}{N^t}$.

Classification and detection. The differentiation powers of these features are presented in Table II. Over these individual features, SRA trains a SVM model and utilizes it to classify the search results of the individual queries containing suspicious suggestions, as reported by STA. This two-step analysis produces a high accurate findings with a precision of 96.23% and a recall of 95.63% (Section V).

V. Evaluation

In our study, we ran our implementation of Sacabuche to automatically analyze 114 millions suggestion and trigger pairs and 1.6 million search result pages, on an R730xd server with 40 Intel Xeon E5-2650 v3 2.3GHz, 25M Cache CPUs and 256GB memories. Here we explain how the data were collected and analyzed at large scale.

A. Data Collection and Validation

To evaluate Sacabuche, we collected a large number of trigger-suggestion pairs. For this purpose, we first utilized a set of keywords to iteratively query the autocomplete services provided by Google, Bing and Yahoo!. The trigger and suggestion pairs discovered in this way were then analyzed by STA to find those suspicious. Such query terms were further used to search the search engines and gather their result pages. Altogether, we obtained 114 millions suggestion and trigger pairs and 1.6 million result pages, as illustrated in Table I.

Trigger and suggestion collection. To collect these seed triggers, we chose a set of "hot" phrases that represent popular search interests. These trigger phrases are mainly trending keywords with a large search volume, totally 1 million collected from a keyword database [10], covering a broad search interests (legal service, financial service, home service and education). However, they miss some content categories suggestion manipulators aim at, including gambling and pharmacy, etc. To address this issue, we included in the dataset 386 gambling keywords [3], [16] and 514 pharmaceutical keywords from drug lists [5].

Using these "seed" keywords, we utilized the technique mentioned in Section IV-B to iteratively discover suggestion phrases through autocomplete APIs: for each given trigger, its suggestion list is retrieved from the API and then serves as new triggers to discover their suggestions; this process continues until a search depth limit is met. In our research, we set the search depth to 2 for the trending phrases, and to 3 for gambling/pharmacy phrases, since the latter are known to be usual targets of suggestion manipulation. Such autocomplete collection was performed on daily basis for three months, from 02/09/2017 to 05/09/2017. Altogether, we collected 114 millions unique suggestion and trigger pairs.

Validation. As mentioned earlier, a study on autocomplete manipulation faces the challenge in getting ground truth, a common problem for the research on blackhat SEO [38], [58], [48], [41]. In line with these prior studies, we utilized a few indicators to manually validate the findings made by Sacabuche. Specifically, we first randomly sampled those flagged query terms (trigger+suggestion) by grouping their suggestions according to their individual categories and then randomly picked out 100 from each group for an inspection. Since the inspection cannot be done automatically, we had to manually analyze each sample (including the search results of a suggestion and related websites and content) to determine its legitimacy, based upon whether some indicators are present or not: (1) a manipulated suggestion must promote a target whose own reputation cannot make itself stand out in the search results of the trigger; (2) the manipulated suggestion and its search results often conflict with the user's original search intention. For example, for the suggestion "strongvault online backup free download", we identified the website it promotes "strongvault-online-backup.software.informer.com" in the search result pages (since the website is highly ranked and its title and search result descriptions are closely relevant to the suggestion). We found that the website does not appear in the search results of its trigger ("online backup free download"). Also when we manually examined the website, we were immediately redirected to a phishing website to download suspicious code reported to be malware by VirusTotal[33]. Therefore we consider this suggestion to be manipulated.

B. Parameter Setting

In the experiments, the parameters of our prototype system were set as follow:

- *RBO decay constant (p)*. The decay constant is a parameter for tunning the weights for different depth in rank-biased overlaps function (Section IV-C). It was set according to the convention of evaluating the RBO function: $p = 0.9$.
- *Inverse of regularization strength (C)*. Regularization is a parameter for reducing the over-fitting to the labeled data when we built the relation models using logistical regression. In our implementation, we utilized a $C = 10$, which gave the best performance among other C values from 0.3 to 15.
- *The Scaling Factor α in Word Kernel*. This factor is defined in the WK kernel to scale the word similarity value. In our implementation, we adopted $\alpha = 5$ as suggested by the original authors to have good performance in multiple datasets.
- *The Decay Factor λ in Sentence Kernel*. λ is defined in the sentence kernel $\mathbf{SK}(s^a, s^t)$ to penalize length of the given phrases. Like α in the Word Kernel, we adopted $\lambda = 1$ as recommended by the original work.

C. Results

Accuracy and coverage. We first evaluated STA over 100 manipulated trigger suggestion pairs and 150 legitimate pairs, and then examined SRA over the search result pages of 150 manipulated trigger suggestion pairs from the bad set (focusing on top 20 result items), together with 300 legitimate pairs from the good set and their search results, using a five-fold cross validation. Overall, our prototype achieved a precision of a precision of 96.23% and a recall of 95.63%.

Then, we ran Sacabuche on the unknown set with 114 millions suggestion and trigger pairs. Altogether, our STA found 1 million pairs to be suspicious. These pairs were further inspected by SRA, which reported 460K to be illegitimate. Among all those detected, 5.6% of manipulated suggestions include links to compromised or malicious websites in their top 20 search results (confirmed by CleanMX[17] and Google Safebrowsing[22]). In this step, we manually inspected 1K suggestion trigger pairs, and concluded that Sacabuche achieved a precision of 95.4% on the unknown set.

TABLE III: **Running time at different stages.** Estimated search query fetching time is 0.864s per query [23].

Sacabuche	average time (ms/pair)	Sacabuche-C	average time (ms/pair)
PF	0.29	PF	0.29
STA	2.54	-	-
Search Query	11.45	Search Query	1486.08
SRA	1.47	STA+SRA	4.01
total	15.75	**total**	1490.38

As mentioned earlier, Sacabuche is characterized by a two-step analysis (STA and then SRA), first filtering out the terms unlikely to be manipulated and then querying the search engine with the remaining terms to inspect the results. This strategy is designed to minimize the overheads incurred by the queries (reducing the number of search queries by a factor of 110 on search engines), which is crucial for making our system scalable. In the meantime, there is a concern whether the performance benefit comes with an impact on the technique's effectiveness, making it less accurate. To understand the problem, we compared our implementation of Sacabuche with an alternative solution, called *Sacabuche-C*, which queries Google for every trigger +suggestion pair and then utilizes the combined features of STA and SRA to detect manipulated suggestions. This approach is fully tuned toward effectiveness, completely ignoring the performance impact. In our research, we also trained Sacabuche-C over the labeled datasets as described in Section III and evaluated it using the five-fold cross validation. This alternative approach has a precision of 97.68% and a recall of 95.59%, which is in line with the original Sacabuche, indicating that our two-step design does not affect effectiveness in detection.

Performance. To understand the performance of Sacabuche, we measured the time it takes to process 100K suggestion trigger pairs from the unknown set, on our R730xd server using a single process. The breakdowns of the delays observed at each stage (PF, STA, and SRA) are reported in Table III. As we can see here, on average, 28.06 seconds were spent on preprocessing those 100K suggestion trigger pairs. Also, only 1,326 search queries were issued to Google. The results demonstrate that Sacabuche scales well and can easily process a large number of suggestion terms without generating too many queries. Further we looked at the performance of Sacabuche-C (Table III). As we can see, in the absence of the STA step to first filter out legitimate terms, the performance overhead became overwhelming: introducing a delay at least 94 times as large as our two-step approach, not to mention the pressure on the search engine, which makes it impossible to scale.

VI. Measurement

On the 114 million trigger-suggestion pairs in the unknown set, we performed a measurement study to understand how today's autocomplete services are being abused. Our study reveals the pervasiveness of autocomplete manipulations, with 0.4% of the suggestions we collected found to present traits typical of illicitly promoted autocompletes. More concerning is that the threat continues to evolve, becoming even more serious over time. We further looked into the ecosystem of this emerging SEO business (see Figure 2), through purchasing and interacting with the manipulation service providers. This effort leads to the discovery of their strategies, including query log pollution and utilization of compromised sites, and their revenues, as high as $569K per week. Also we report our analysis of an attack toolkit and a large-scale malvertising campaign, involving 245 manipulated suggestions and 1,672 websites.

A. Landscape

Scope and magnitude. Our study reveals that manipulated suggestion terms are pervasive across multiple search engines' autocomplete services. Of 14 million trigger phrases on Google, Bing and Yahoo!, we found that 256K were manipulated, which relate to 383K manipulated suggestions. Figure 6(a) illustrates the number of the abused suggestions we discovered on different search engines in a single day. Among them, Google is the most popular (0.48%) in our dataset, followed by Bing (0.37%) and Yahoo! (0.2%). Also, in Google autocomplete service, 16.6% of manipulated suggestions were ranked in the top 3 autocomplete results, which is 14.2% in Bing and 29.1% in Yahoo!.

We further investigated the impacts of suggestion manipulations on different search content. For this purpose, we grouped all trigger keywords into 10 categories. Table IV presents the number of manipulated suggestion terms under each category. As we can see here, such abused suggestions cover a vast range of topics, including home services, education, legal and lending products, technology, gambling and others. Among all these categories, Lending Products (4.13%) has the largest portion of trigger phrases with manipulated suggestions, which is followed by Home Services (2.47%) and Pharmaceutical Services (2.09%). When we looked at polluted triggers in the category of Lending Products, we found 536 "payday loan" related polluted triggers with manipulated suggestions such as "online payday loans get cash now - *cashnow.com*" and or "payday loans cash advances *www.quickquid.co.uk*". Note that Google bans ads for payday loans to show in search results [12]. These payday loan websites were able to promote their lending products in search engines through autocomplete manipulation.

Evolution and lifetime. To understand how this emerging threat evolves over time, we collected 67 million distinct autosuggestions from the Google Autocomplete API from 02/09/2017 to 05/09/2017. Among them, Sacabuche found 324,610 manipulated trigger-suggestion pairs on Google. Figure 6(b) illustrates the evolution of the attack, in terms of the cumulative number of newly-appearing manipulated terms observed during that period. We found that large amount of manipulated suggestions are newly appeared, with 71.3% of newly-appearing manipulated suggestions related to the newly-appearing polluted triggers. Also, we observed that on average 1.9% of trigger phrases were polluted. This number jumped to 2.1% on 03/21/2017. In general, the trend exhibits a substantial increase in the number of manipulated suggestions.

When looking into the lifetime of the manipulated suggestion terms, we were surprised by the effectiveness and stealthiness of suggestion manipulation campaigns: they have been there for a long time, without being detected. Figure 6(c) illustrates the distribution of the lifetime for those missuggestion terms, which were estimated through continuously crawling

TABLE IV: **Categories of the polluted triggers.**

Trigger Category	# of terms	# of manipulations	Volume
Lending Products	1389	1580/34629	4.13%
Home Services	17059	16712/413836	2.47%
Pharmaceutical	3715	3876/93929	2.09%
Technology	29115	32696/762548	0.91%
Auto Services	3477	3916/92220	1.08%
Education	17311	17628/423352	0.81%
Shopping	30465	32986/773087	0.72%
Gambling	413	454/10314	0.68%
Travel	6434	6554/153842	0.48%
Legal Services	14064	14084/348548	0.55%

autocomplete of the seed keywords every day from 02/09/2017 to 05/09/2017. As we can see from the figure, 39.3% of the manipulated suggestions stay on Google's suggestion list for more than 30 days, with the average lifetime of 34 days, which is comparable with the lifetimes of legitimate suggestions, that is, 63 days on average.

Manipulation content and pattern. Our research shows that multiple triggers have been utilized to promote one suggestion. We found that 20% of manipulated terms are related to more than one trigger. For example, the missuggestion "free web hosting and domain name registration services by *doteasy.com*" is associated with 123 trigger phrases such as "free web hosting without domain name" and "web hosting free domain name registration". Therefore, finding one manipulation suggestion can help to detect other polluted triggers.

Another interesting observation is that different manipulated suggestions tend to have similar grammatic structures, as shown in Table V. For example, we found that 1,446 manipulated suggestions are characterized by a pattern *trigger phrase relevant content+by+URL*, such as "custom t shirt design software and application tool by *panaceatek.com*","free web hosting and domain name registration services by *doteasy.com*" and "site hosted for free by *xfreehosting.com*". To identify such semantic patterns, we grouped the manipulated suggestions based on their content components unrelated to the semantics of trigger (such as "by+URL"). More specifically, we first removed all trigger phrases and their related phrases (identified by Word2Vec) from the suggestion. Then from the remaining suggestion content, we manually inspected the words occurring frequently across different suggestion terms (such as "by" and "-"). Table V illustrates the top 5 most popular patterns. These patterns are characterized by the locations of their promotion targets (e.g., "strongvault", "panaceatek.com"), which tend to appear at the beginning or the end of a suggestion, being connected to the rest of the term through "-", "by","at", "from", "+" and other common conjunctions. The rest of the manipulated suggestions carry their promotion targets in the middle of the terms. We found that such a promotion phrase often has a compound relation with a trigger-related phrase, such as "how does the backup software strongvault work", where "strongvault" is a compound term for "backup software".

For example, we found a malicious autocomplete campaign sharing similar manipulation content and pattern. From our dataset, we correlated 245 manipulated suggestions and 1,672 websites from their search results. These suggestions all share the same pattern: target term +"-"+ trigger relevant content, such as "hesna ims - internet marketing solutions", and once their top result items are clicked, all the redirection URLs generated have a similar format, particularly, all including a campaign ID "campaign_id=20". Our analysis shows that most top 20 search results produced by these suggestions are malicious, directing the visitors to those attack websites to download malware.

B. Autocomplete Manipulation Service

We found that manipulating suggestion has already become a booming business, with tens of services available online. To understand their promotion strategies and ecosystem, we studied such services through communicating with some of them to understand their services and purchasing the service to promote suggestion terms. In this way, we were able to get access to the tools they use, find out their strategies and collect materials for estimating their operational cost and revenues.

Manipulation techniques. We found that these services manipulate suggestions typically through two strategies: search log pollution and search result pollution, as explicated below:

• *Search log pollution*. Some of service providers we contacted (Table VI) explained to us the way they promote a given suggestion term and the tools they use. A key strategy here is to pollute search logs through crowd-sourcing. Specifically, these providers often install software on their clients' systems, which can simulate web users' behaviors to query search engines. As a prominent example, *Seopmb*[29] is one of such tools that performs automatic searches in one's browser. The clients running the tool need to pay rewards points to the provider. In the meantime, they also get rewards by either paying for them or running promotional tasks for others. In this way, a large number of clients under the same provider can work together to promote each other's suggestion keywords.

Since the illicit practice of manipulating autocomplete service can lead to SEO penalty [28], the service providers often deploy evasion mechanisms to avoid being detected. For example, Seopmb encrypts the communication between the client and the server; also interestingly, although the provider's website is hosted in USA (Psychz Networks Ashburn), the software distributed from the site communicates with a server located in Hong Kong. Also, the providers usually hire a lot of people from different locations with different IPs to perform searches manually. They can even customize such crowd-sourcing according to customers' requests. For example, if one wants to promote a suggestion for a specific geo-location, the provider can find people from that location to do the job.

To understand the effectiveness of such promotion efforts, we purchased the service from a provider *Affordable Reputation Management*[13]. This provider hired crowd-sourcing operators around the United States to perform the search. They utilized the crowd-sourcing manipulation strategy as follows: 1) search for the target suggestion phrase on Google, 2) click the client's website and 3) click any other link from this website. In this way, they generated real searches on Google and real visits to the search results. We set up a website to track the visitors with the "Referrer = google.com", assuming most of them working for the service provider (we only recorded the visiting

TABLE V: **Top 5 manipulation patterns.**

Pattern	# of manipulations	Example
trigger relevant content+target	195,738	phoenix divorce attorney sampair
target+trigger relevant content	188,238	strongvault online backup free download
target+" - "+ trigger relevant content	2,278	bd&j - los angeles personal injury lawyers
trigger relevant content+by+URL	1,446	custom t shirt design software and application tool by panaceatek.com
trigger relevant content+from+URL	427	reliable web hosting from webhostinghub.com

TABLE VI: **List of Manipulation Service Providers.**

Service Provider	Country	Supported Platforms	Cost	Time
Affordable Reputation Management	USA	Google, Bing, Yahoo!, Amazon etc.	$370 / m	1 - 3 months
Search Reputation	USA	Google, Bing, Yahoo!	$1200 / m	3 months
Google Autocomplete Changer	USA	Google, Bing, Yahoo!	$2500 (fixed)	3 months
Seopmb	China	Baidu, Sogou, 360	$1 ∼$20 / d	1 month
iXiala	China	Baidu, Sogou, 360, Taobao etc.	$2 ∼$12 / d	3 - 15 days

information without requiring user's input). We found that 102 different hosts visited our websites from 78 different locations such as California, Portland (the company's location) and Columbia. These visits happened through multiple platforms, from desktops to mobile devices, and the average number of visit was 5.4 per day. Surprisingly, the operation took effect in just one month with our suggestion phrase ranks first in the autocomplete list. The service provider claimed that this approach was difficult to detect since the searches were real and the operations performed by the crowd-sourcing operators were unrecognizable as the normal activities.

• *Search result pollution*. In addition to search log pollution, we also found that suggestions can be manipulated through polluting search results. A potential rationale for the strategy is that the ranking (e.g. PageRank) of the web pages with the keyword could impact the positioning of suggestion terms on the suggestion lists triggered by search terms [20].

In our research, we ran malware scan (Google Safebrowsing, CleanMX and VirusTotal) on the websites retrieved from manipulated suggestion phrases' search result pages, and found that 2,928 websites (related to 5.6% of suggestions) were compromised to include manipulated suggestion phrases. Among them, 39.1% were among Alexa Top 1 Million websites, such as *virginia.edu*, *liceomedi.gov.it*, and *ub.ac.id*. Figure 6(d) shows the cumulative distribution of their Alexa ranks.

Further, we found 120K webpages, which appeared on the free hosting list [26], turned out to be Spam hosting related to 9% manipulated suggestions. Here we determined that a page from the search result is Spam hosting if it looks very similar to other pages also in the results. The similarity here was measured by the cosine similarity between two pages ($>$0.98) in terms of their content's TF-IDF (term frequency-inverse document frequency) vectors [39]. Taking a close look at these pages, we discovered that several blackhat SEO techniques are used to increase their page ranking, such as keyword stuffing (i.e., the repetition of keywords in the meta tag and web page content), link farms (i.e., constructing networks of websites that link to each other) and traffic Spam (i.e., adding unrelated keywords to manipulate the relevance).

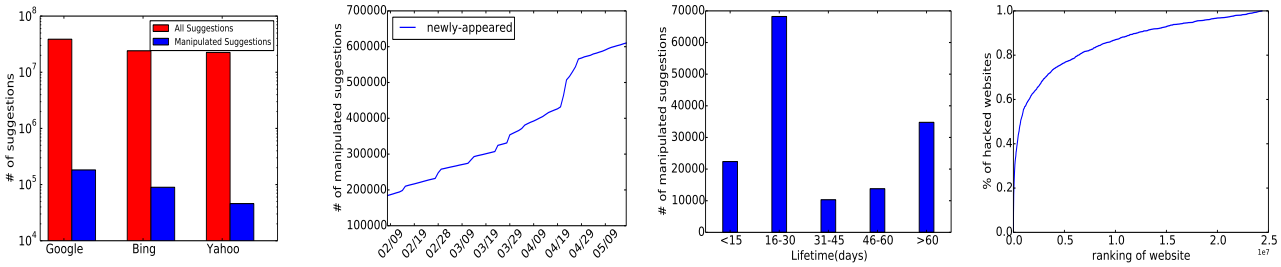
Traffic monetization. Then we looked into how the manipulated phrases are used to monetize traffic. Beside commercial promotion, we found that manipulation campaigns get profits by attracting victims to download malware or visit phishing sites, or selling the victims' traffic through an affiliate program.

We observed in our study that the parties promoted by manipulated suggestions monetize the traffic through search redirections: when a visitor clicks the link on a search result page, she will be redirected to a different site. Such redirections were detected by a Firefox add-on we implemented, which uses the browser to automatically visit each link on a manipulated suggestion's search result pages, using *google.com* as the referrer, and records all the web activities it triggers (network requests, responses and browser events). Running the crawler, we found that 315K pages related to 18% suggestions utilize search redirections to monetize traffic.

We further discovered several campaigns using suggestion manipulation. These campaigns were identified from the affiliation networks and affiliation IDs on the URLs generated by search redirections. Such information was recovered through finding the most common 3-grams on search redirection URLs, which were then manually inspected to determine whether they are affiliate IDs and if so, which network they belong to. Then, we grouped the search redirection chains according to their affiliate IDs. Table VII shows the top 3 affiliate networks utilized by most traffic campaigns. A prominent example of such networks is *alldownloads.hapc.gdn*, whose redirection doorway sites have the largest presentation (related to 245 manipulated suggestions) in the search results for all manipulated suggestions we detected. That network aims to promote potentially unwanted programs (PUP), so the adware publisher can generate pay-per-click revenues. Also, we found that some traffic abusers (promoted through suggestion manipulation) collaborate with reputable affiliate network such as *admarketplace.com*.

Revenue analysis. To understand the economic motives behind the suggestion manipulation business, we analyzed the revenue of the service providers, who profit from autocomplete promotion, and their customers, who profit from traffic monetization.

For estimating the revenue, we contacted five of them to find out the financial cost and time for running a suggestion manipulation campaign. As shown in Table VI, we found that promoting a suggestion phrase takes from $300 to $2,500 per month, depending on target search engines and the popularity of the phrase. It needs 1-3 months to have the suggestion visible



(a) **Number of manipulated suggestions on each platform.** (b) **Number of missuggestions over time.** (c) **Lifetime distribution of manipulated suggestions.** (d) **Alexa ranking distribution of hacked websites.**

Fig. 6: **Study on manipulated suggestions.**

TABLE VII: **Top 3 search redirection manipulated suggestion campaigns with most manipulated suggestions.**

Name	# of manipulations	# of links	Traffic monetization
alldownloads.hapc.gdn	245	1,672	malvertising
mobioffertrck.com	130	1,203	malvertising
admarketplace.com	93	871	affiliate network

on the autocomplete results. Hence, the average revenue for the service providers to successfully manipulate one suggestion is $2,544. While it is challenging to measure profitability of the service providers directly, we have one anecdotal data point. Through the communication with the provider *iXiala*, we found that the 10K sites requested suggestion manipulation on it, which related to a revenue of $569K per week with 465K manipulated suggestions. At the same time, *iXiala* had a group of operators, who earned a commission of $54K per week. Hence, *iXiala* earned a profit of around $515K per week.

As mentioned earlier, the recipient of the manipulation service monetizes the traffic from the victims who follow the search results produced by manipulated suggestions. We utilize the following model [48] to estimate their revenues: $R(t) = N_v(t) \times P_a \times R_a$, where the total revenue $R(t)$ during the time period t is calculated from the total number of actions taken (i.e., total number of click $N_v(t)$ times the probability of action after the click P_a), and the average revenue per action R_a.

The total number of clicks per manipulated suggestion can be estimated from individual keyword's search volumes reported by Keywordtool API [25], a mainstream search engine keyword research instrument adopted by many companies such as Microsoft and eBay. From the statistics of the 189,808 manipulations indexed by Keywordtool, we found that the search volume of manipulated suggestions is about 111 million/month in total, 584/month per keyword on average. The average cost of the keyword is $0.23 per click. Also for each search result page, there is a probability 71.3% [21] for visitors to click on a result item. Altogether, we can estimate the average revenue per month for a service recipient who monetizes traffic is $95/keyword. Note that manipulated suggestions also indirectly monetize traffic (i.e., traffic monetization through the second time of search query after querying trigger phrases). The recipients of the manipulation service also earn promotion profit even the victims do not conduct the second search.

VII. Discussion

Limitations of Sacabuche. Like any machine learning approach, our detection in theory can be evaded by making both search terms and search results mimic the benign ones. For example, an adversary may utilize a trigger with similar semantic as the legitimate search terms and pollute the trigger's all top 20 search results. However, this will very likely cause the manipulated suggestion to be less attractive to the visitors and increase the cost of the manipulation. Overall, we believe Sacabuche significantly raises the bar for the attack due to the inherent semantic inconsistency between the trigger and target as well as a set of salient features from the search results of the manipulated terms.

Again, it's worth to note that our detected autocompletes are difficult to be verified as manipulations. In our study, we consider them to be missuggestions due to the exhibition of illicit indicators as the confirmed manipulated ones. Indeed, our study shows that a large volume of the missuggested results include unpopular services or products, which are not supposed to appear on autocomplete, even when the context is right and the promoted products are legitimate.

Another limitation is the lack of ground truth and the manual efforts involved in our evaluation. To ensure our evaluation results are trustworthy, we exerted our best efforts to collect a reasonable size of manipulated examples posted by real autocomplete manipulation companies. As web security experts, we adhere to several concrete, intuitive, and easy-to-follow guidelines (Section V) when performing manual analysis. We also analyzed and observed other indicators like search term patterns as side-validation for the unlabeled data. Despite all above efforts, we admit that the validation process is laborious and may possibly include inaccuracies.

So far our work focuses on offline analysis. Nevertheless our low detection overhead (Table III) makes it entirely feasible to convert Sacabuche into an online system, for the purpose of helping users avoid autocomplete scams. The Sacabuche service can be maintained by the search engine providers or, more attractively, 3rd-party organizations (recall that Sacabuche does not require query logs). We leave this as future work.

Our current implementation of Sacabuche focuses on detecting missuggestion on search engines. In the meantime, e-commerce platforms'(such as Amazon and eBay) search services are also vulnerable to similar attacks. A natural follow-up is to develop the detector to protect those services.

Comparison with traditional blackhat SEO. We found that the manipulators always promote unpopular terms which are within the same context as the triggers. This is because it is more effective to advertise the targets and evade detection. Specifically, the autocompletes can be regarded as long-tail keywords. As studied in [48] and [47], searching for the long-tail keywords, which relate to specific terms, will make it more easy to convert the traffic to sales than generic searches. From the description of the manipulators and the missuggestions they promoted (see Section III), they indeed tend to promote specific targets in coherence with the triggers. Note that on the other hand, different from autocomplete manipulation, some traditional blackhat SEO techniques (such as keyword stuffing) pollute the search results by adding irrelevant keywords. However, they have a different goal for content promotion. Instead of achieving a higher traffic conversion rate, they tried to gain traffic from other popular keywords which are out of search intentions. In the autocomplete manipulation, users already have search intentions (as the trigger terms). As a result, the manipulators targeted a higher conversion rate by adding unpopular while coherent terms.

Lesson Learnt. Based on our findings, we identified several potentially effective mitigation strategies besides the detection efforts from the third party such as Sacabuche. First, search engine operators do have responsibilities to act more aggressively on detecting and removing manipulated suggestion. They could detect similar phrase patterns from the search logs, or degrade the positioning of autocomplete phrases with highly similar but low-quality search result contents. Further, the affiliate networks could monitor HTTP refers and identify other indications that their affiliates are engaging in autocomplete manipulation. We found that most of the affiliate networks currently have reactive policies, such as abuse reporting to restrict illicit practices of affiliates. A more proactive policy might help to mitigate the surge of autocomplete manipulation.

Responsible Disclosure. Since the discovery of manipulated suggestions, we have been in active communication with the parties affected. So far, we have reported 100 sampled manipulated phrases to Google, Bing, and Yahoo!. By now, Google has responded to our report. However, considering the size of the manipulations, the full-scale reporting and validation process takes time and is our on-going work.

VIII. Related Work

Detection on Autocomplete Manipulation. To the best of our knowledge, the work most relevant to ours was done by Liu et. al [50] who used query logs to detect promotion campaigns. The authors proposed a framework to detect promotion campaigns abusing autocomplete service, and then extended it to identify the promotion target (e.g., drugs). Their core technique is to construct and analyze a user-query bipartite graph that captures the relationship between promoters and manipulations. Their technique requires (usually proprietary) search query logs, manual annotation of thousands of terms, and even promoters' IDs (through cookies). In contrast, our NLP-based technique is fully automated and requires neither query logs nor promoters' IDs that are easy to spoof. We further conducted a large-scale measurement to reveal the pervasiveness of autocomplete manipulation and its significant security impacts.

Abusing Search Engines. Numerous malicious activities leveraging blackhat SEO have been reported in the literature. To name a few, Leontiadis et al. conducted a measurement study on search redirection attacks for online pharmacy promotion [46]. Lu et al. developed a browser plug-in for detecting malicious search redirections [51]. Moore et al. performed a measurement of the abuse of "trending" terms, which are usually obtained from popular search terms or tweets, for web search-engine manipulation and social-network spam [52]. Invernizzi et al. designed an efficient system called Evilseed to identify malicious webpages indexed by search engines, which adopts prefiltering techniques to speed up the inspection process [45]. Different from these works, our paper designed and implemented a system to perform efficient and highly accurate missugestion detection, a different type of blackhat SEO.

Attacking recommendation systems. Prior research has also reported attacks on manipulating the results of recommendation systems as well as devised mitigation strategies. For example, Xing et al. proposed pollution attacks that utilize cross-site requests to inject fake information in order to perturb web services' personalization algorithms [60]. Yang et al proposed attacks on the co-visitation recommendation systems [61]. Gelernter et al. introduced an attack to pollute the *personalized* search history and therefore the auto-suggest list of the victim [42]. The bar for the attack is high: the victim must log into the search engine service (e.g., Google account), and she has to visit the malicious website controlled by the attacker. This attack is therefore different from our studied crowd-sourcing autocomplete manipulation attack, which can be regarded as an emerging type of attack on recommendation systems.

Security Analysis Leveraging NLP. Recently, researchers leverage natural language processing for security and privacy research. Examples include analyzing web privacy policies [62], generating Android privacy policies [54], inferring mobile app permission through apps' descriptions [54], [56], detecting sensitive user input [44], [53], and website promotional infection detection [49]. Our work identifies a novel application of NLP, i.e., scalable detection of autocomplete manipulations.

IX. Conclusion

In this paper, we present the first technique that supports a large-scale semantics-based analysis of autocomplete manipulations, an emerging threat with significant security implications. Our system, called *Sacabuche*, utilizes a two-step approach to filter through a large number of trigger-suggestion pairs, based upon a lightweight NLP analysis, thereby avoiding expensive queries to search engines. Only a small set of suspicious suggestions are run against the search engines to acquire search results for a more in-depth analysis. Our study shows that this approach achieves a high effectiveness (96.23% precision, 95.63% recall) and also enables a large-scale measurement study involving 114 million query terms. Our findings reveal the significant impact of the threat, with hundreds of thousands of manipulated terms promoted through major search engines (Google, Bing, Yahoo!), spreading low-quality content and even malware and phishing. Also discovered in the study are the sophisticated evasion and promotion techniques employed in the attack and exceedingly long lifetimes of the abused terms, which call for further studies on the illicit activities and serious efforts to mitigate and ultimately eliminate this threat.

ACKNOWLEDGMENT

We are grateful to our shepherd Gianluca Stringhini and the anonymous reviewers for their insightful comments. This work is supported in part by the NSF CNS-1223477, 1223495, 1527141, 1618493, 1618898, Samsung Gift fund and ARO W911NF1610127.

REFERENCES

[1] "60 percent of searches from mobile devices," http://searchengineland.com/report-nearly-60-percent-searches-now-mobile-devices-255025.

[2] "Bag-of-words model," en.wikipedia.org/wiki/Bag-of-words_model.

[3] "Casino keywords," 50-highest-paying-adsense-keywords.blogspot.com/2006/10/casino-keywords.html.

[4] "Dependency grammar," en.wikipedia.org/wiki/Dependency_grammar.

[5] "Prescription drug lists," www.uhcrivervalley.com/pharmacy/PDL.html.

[6] "The stanford parser," http://nlp.stanford.edu/software/lex-parser.shtml.

[7] "Word embedding," https://en.wikipedia.org/wiki/Word_embedding.

[8] "A deeper look at bing autosuggest," https://blogs.bing.com/search/2013/03/25/a-deeper-look-at-autosuggest, 2013.

[9] "Google autosuggest manipulation," https://www.smartt.com/insights/google-autosuggest-manipulation-is-it-a-legitimate-strategy, 2014.

[10] "1,000,000 top high paying cpc for 2015," http://grepwords.com/1000000-top-high-paying-cpc-adwords-adsense-keywords-2015/, 2015.

[11] "Google search autocomplete," https://blog.google/products/search/google-search-autocomplete/, 2016.

[12] "An update to our adwords policy on lending products," https://blog.google/topics/public-policy/an-update-to-our-adwords-policy-on/, 2016.

[13] "Affordable reputation management," http://affordablereputationmanagement.com/, 2017.

[14] "Bing autocomplete api," http://api.bing.com/osjson.aspx, 2017.

[15] "Blackhat seo," https://en.wikipedia.org/wiki/Spamdexing, 2017.

[16] "Casino keywords-wordstream," http://www.wordstream.com/popular-keywords/casino-keywords, 2017.

[17] "Cleanmx," http://support.clean-mx.com/clean-mx/viruses.php, 2017.

[18] "Geonames," http://www.geonames.org/export/, 2017.

[19] "Google autocomplete api," http://suggestqueries.google.com, 2017.

[20] "Google autocomplete study results," https://www.wiideman.com/research/google-autocomplete/study-results, 2017.

[21] "Google organic click-through rates," https://moz.com/blog/google-organic-click-through-rates, 2017.

[22] "Google safe browsing," https://safebrowsing.google.com/, 2017.

[23] "Google search api," http://developers.google.com/custom-search/json-api/v1/overview, 2017.

[24] "ixiala," http://www.ixiala.com/, 2017.

[25] "Keyword tool," https://keywordtool.io/, 2017.

[26] "Malware domain blacklist: Dns-bh," www.malwaredomains.com/, 2017.

[27] "Natural language toolkit," http://www.nltk.org/, 2017.

[28] "Search using autocomplete," https://support.google.com/websearch/answer/106230?hl=en, 2017.

[29] "Seopmb," http://www.seopmb.com/, 2017.

[30] "Skip-gram," https://en.wikipedia.org/wiki/N-gram\#Skip-gram, 2017.

[31] "Source code search engine," https://publicwww.com/, 2017.

[32] "Stopwords in multiple languages," http://www.ranks.nl/stopwords, 2017.

[33] "Virustotal," https://www.virustotal.com/, 2017.

[34] "Word2vec," https://en.wikipedia.org/wiki/Word2vec, 2017.

[35] "Wordnet," www.nltk.org/_modules/nltk/stem/wordnet.html, 2017.

[36] "Yahoo! autocomplete api," http://sugg.search.yahoo.net/sg/, 2017.

[37] "Yinc marketing," http://yincmarketing.com/, 2017.

[38] S. Alrwais, X. Liao, X. Mi, P. Wang, X. Wang, F. Qian, R. Beyah, and D. McCoy, "Under the shadow of sunshine: Understanding and detecting bulletproof hosting on legitimate service provider networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*.

[39] C. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[40] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna, "Query suggestions using query-flow graphs," in *Proceedings of the 2009 workshop on Web Search Click Data*. ACM, 2009, pp. 56–63.

[41] M. F. Der, L. K. Saul, S. Savage, and G. M. Voelker, "Knock it off: profiling the online storefronts of counterfeit merchandise," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1759–1768.

[42] N. Gelernter and A. Herzberg, "Autocomplete injection attack," in *European Symposium on Research in Computer Security*.

[43] K. Hofmann, B. Mitra, F. Radlinski, and M. Shokouhi, "An eye-tracking study of user interactions with query auto completion," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 549–558.

[44] J. Huang, Z. Li, X. Xiao, Z. Wu, K. Lu, X. Zhang, and G. Jiang, "Supor: Precise and scalable sensitive user input detection for android apps." in *USENIX Security Symposium*, 2015, pp. 977–992.

[45] L. Invernizzi and P. M. Comparetti, "Evilseed: A guided approach to finding malicious web pages," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 428–442.

[46] N. Leontiadis, T. Moore, and N. Christin, "Measuring and analyzing search-redirection attacks in the illicit online prescription drug trade." in *USENIX Security Symposium*, vol. 11, 2011.

[47] A. A. Lew, "Long tail tourism: New geographies for marketing niche tourism products," *Journal of Travel & Tourism Marketing*.

[48] X. Liao, C. Liu, D. McCoy, E. Shi, S. Hao, and R. Beyah, "Characterizing long-tail seo spam on cloud web hosting services," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 321–332.

[49] X. Liao, K. Yuan, X. Wang, Z. Pei, H. Yang, J. Chen, H. Duan, K. Du, E. Alowaisheq, S. Alrwais *et al.*, "Seeking nonsense, looking for trouble: Efficient promotional-infection detection through semantic inconsistency search," in *Security and Privacy (SP), 2016 IEEE Symposium on*.

[50] Y. Liu, Y. Liu, K. Zhou, M. Zhang, S. Ma, Y. Yin, and H. Luo, "Detecting promotion campaigns in query auto completion," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 125–134.

[51] L. Lu, R. Perdisci, and W. Lee, "Surf: detecting and measuring search poisoning," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 467–476.

[52] T. Moore, N. Leontiadis, and N. Christin, "Fashion crimes: trending-term exploitation on the web," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 455–466.

[53] Y. Nan, M. Yang, Z. Yang, S. Zhou, G. Gu, and X. Wang, "Uipicker: User-input privacy identification in mobile applications." in *USENIX Security Symposium*, 2015, pp. 993–1008.

[54] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "Whyper: Towards automating risk assessment of mobile applications." in *USENIX Security Symposium*, 2013, pp. 527–542.

[55] M. F. Porter, "An algorithm for suffix stripping," *Program*.

[56] Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen, "Autocog: Measuring the description-to-permission fidelity in android applications," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 1354–1365.

[57] M. Shokouhi and K. Radinsky, "Time-sensitive query auto-completion," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*.

[58] J. Wadleigh, J. Drew, and T. Moore, "The e-commerce market for lemons: Identification and analysis of websites selling counterfeit goods," in *Proceedings of the 24th International Conference on World Wide Web*.

[59] W. Webber, A. Moffat, and J. Zobel, "A similarity measure for indefinite rankings," *ACM Transactions on Information Systems (TOIS)*.

[60] X. Xing, W. Meng, D. Doozan, A. C. Snoeren, N. Feamster, and W. Lee, "Take this personally: Pollution attacks on personalized services." in *USENIX Security*, 2013, pp. 671–686.

[61] G. Yang, N. Z. Gong, and Y. Cai, "Fake co-visitation injection attacks to recommender systems," 2017.

[62] S. Zimmeck and S. M. Bellovin, "Privee: An architecture for automatically analyzing web privacy policies." in *USENIX Security Symposium*.