SWM: Cloaking Detection through Simhash-based Website Model

Ruian Duan*, Weiren Wang*, Ji Zhang**, and Wenke Lee*

*Georgia Institute of Technology
**Google Inc.

Abstract

Cloaking, used by spammers for the purpose of increasing exposure of their websites, as well as to circumvent censorship, has been a nostorious spamming Search Engine Optimization (SEO) technique. Recently, besides SEO, we also identify a rising trend of employing cloaking in search engine marketing. The motivation of cloaking is to hide the true nature of a website by delivering blatantly different content to users versus censors. Cloaking is popular due to low setup cost and lack of efficient detection technique.

In this paper, we propose Simhash-based Website Model, or SWM, a new way to fuzzy hash the content and layout of a website and model the dynamics of a website. We apply SWM in cloaking detection, and achieve comparable accuracy and much higher efficiency compared with past approaches. In order to understand the incentives behind cloaking, we systematically categorize and review the detected samples, and find 9% of them are malicious websites, and 80% of them are phishing sites. With this strong relationship between cloaking and maliciousness of a website, we discuss deployment of SWM in cloaking detection and propose a novel model to detect cloaking through crowdsourcing with user privacy guarantee.

1 Introduction

Cloaking is a technique to deliver blatantly different content to users versus censors on the web. Search engine is the main entry for user to get online today, and therefore, attackers who want to promote illegal services or do phishing and malware hosting is willing to have their page ranked high in search engine. On the other hand, search engine develops algorithms to rank pages and penalize these pages. In order to evade detection, attackers use cloaking to hide the true nature of their website. Similarly, in search engine marketing, advertisers pay search engine for specific advertisements and get their page displayed on the first page of search results. Attackers abuse this service to deliver non-compliant content to user. The foundamental reason of SEO and SEM cloaking is that, search engine is just processing information over the Internet, with no control over it, hence it is hard to tell whether the copy of website that they see are actually what user sees.

In Search Engine Optimization (SEO) and Search Engine Marketing (SEM), spammers use user agent, referer, IP etc based cloaking techniques to avoid inspection. Among the cloaking techniques, IP cloaking is very difficult to detect. There are two major challenges in cloaking detection in SEO and SEM. First challenge is revealing the uncloaked content. There are various commercial tools, such as blackjack's, noIPfraud, and wpCloaker [14] that are efficient in performing cloaking. They provide different strategies of cloaking, including user agent, referer, redirect, IP etc. Regarding IP cloaking, they provide services to periodically refresh the list of crawlers' IP addresses. Therefore, unrevealing cloaked content is an ongoing war between cloakers and censors. Second challenge is differentiating cloaking from naturally dynamic changes of the same page. Dynamic content are allowed by SEO and SEM, however, blatantly deliver different, unrelated content to search engine spider versus normal user is not allowed.

In order to address the first challenge, existing detection approaches pretend to be real user, e.g. set referer, user agent, use multiple IPs or proxy to hide identity. But attackers could identify them and incrementally blacklist IPs. It is an endless war between cloakers and spiders. Intuitively, an alternate is to collect data from user side directly. However, it is hard to collect sufficient data for comparison while protecting user privacy. Past work usually go through the documents multiple times and compare page difference to address the second challenge.

This work employs simhash to address the two challenges. Charikar's simhash [3] is a special signature of feature set. It is a class of Locality Sensitive Hash (LSH)

that has been extensively used in near duplicate in search engine. It is an random project based algorithm that maps high dimentional data to fixed bits while maintaining the property that, hamming distance between the resulting bits is an estimation of similarity between the original feature set.

We propose Simhash-based Website Model(SWM) to model the dynamics of a web page and use the outlier detection ability of SWM to do cloaking detection. The basic idea is to generate two fuzzy hash corresponding to text and dom tree for each website (fuzzy signature), and compare the differences between spider copy and user copy. In order to model dynamics of a website, we retrieve spider copy multiple times and leverage the fact that hamming distance is euclidean distance (L1 norm) to learn patterns. SWM cloaking detection performs comparable in terms of accuracy (95% true positive rate with 5% false positive rate) and much higher efficiency compared to past approaches.

Then, we manually classify and illustrate the incentives behind cloaking, results show that 9% of cloaking sites are malicious and 80% of cloaking sites are phishing. With strong relationship between cloaking and phishing sites and malicious download, we propose a novel model to collect simhash from user side, with low overhead and privacy guarantee provided by RAPPOR. By crowdsourcing the fuzzy signatures of what is actually shown to user, we can address both challenges elegantly.

The remainder of this paper is structured as follows. Section §2 provides a technical background on cloaking and simhash, and related work in cloaking detection. Section §3 introduces methodlogies, including the simhash-based website model and the cloaking detection system. Followed by evaluation of our model and comparison against previous approaches in Section §4. Section §5 explains our detection result in SEO and SEM and motivate the deployment. Section §6 describes server-based and crowdsource-based deployment, proposes the frameworks and corresponding pros and cons.

2 Related Work

Search engine, as a most innovative technology introduced in late 20th century, has widely influenced our relationships. Search engine used their own page ranking algorthim to rank and indexed websites. In this way, users could find information by entering the search terms into the search enginee. To get better rankings and accurate indexed on search engines such as Google, websites are encouraged by Google to optimize their content. This is a technology called search engine optimization (SEO). SEO policies were introduced by search engine later to maintain a fairness of search enginee ranking. Recently, some websites break the SEO policies to get better rankings on websites because of the large profits. These technology are called Black Hat SEO, which is used to get better rankings on search engine but break the SEO policy. From the SEO policy, one of the most efficient methods to get better page rankings is to update the content of website frequently [15]. Following this rule, cloaking, a Black Hat SEO method, is invented. Cloaking serves a blatently different information to users and Google to maintain their profits and get better page rank. To be clear, we give a general example. Websites provide a frequently updated information to the Google so that they could get better rank. On the other side, they send a different information to users to get profits.

2.1 Example

To concrete the idea of cloaking, a specific cloaking example, malicious downlowd cloaking, is introduced. We entered the term ”Game Texas Holdem Poker Online - Dj w” in Google. Google returned a set of search results as showned in figure 1. From the figure 1, we see that the top 6 search results are all under domain ”dje,.com.ol”. If we disguised ourselves as normal users without setting HTTP proxy agent and clicked the links, the broswers sent an HTTP request with Google reference. The website received our HTTP request and found Google reference. The website redirected several times, eventually landed to a page and automatically download malicious software. On the other hand, we reconfiguerd our browsers and set Google-Bot as our HTTP agent. We revisited the website. This time, the website provide us totally different information without redirecting and malicious downloading. From two scenarios, the cloaking idea is clear. Websites provide legal information to Google to get good page rank and avoid censorships. On the other side, websites provide different information with redericting and malicious downloads to get profits from users.

2.2 Search Engine Marketing

Priviously, we briefly talked about search engine optimization. In this section, we introduce another area called search engine marking(SEM) where enourmous cloaking websites exist. According to wikipedia [2], the term SEM is used to mean pay per click advertising, particularly in the commercial advertising and marketing communities which have a vested interest in this narrow definition. To be clear, search egnine marketing defines the advertisements shown on search engines after searching terms. Accoding to wikipedia [2], boundary beween SEO and SEM are sometimes not clear. However, when studying the cloaking, the SEO and SEM are clearly different. The difference betwen them is because that the policies and rules on SEO and SEM are different. On SEM, the

policy are usually strict and commertial-related. Further, the cloaking-incentives on SEO and SEM are different. On SEM, websites are more likely to provide illegal medicines and services. On SEO, websites are inclined to do malicious downloading and phishing. Further, the strategy on cloaking detection on SEO and SEM are different. The difference is crucially decided by ranking mechanisims. In SEO, though the page rank algorithm is not fully public, it is well known that rankings are related to content, page rank and visit traffic. In SEM, the rank algroithm depends on real time biding system. Websites need to bid the "pay per click" price on the real time biding system. The one with the the higher price will be shown in higher priority. The terms with large commerical potential ask higher price. Because of the difference between SEM and SEO, the strategy to detect cloaking should change. The cloaking detection algorithm should change such as collecting commercial related terms, crawling adverstisements, checking the SEM policy(Google Ads Policy). To ensure "Pay per click" mechanisms, in other words, websites won't pay much extra money because of cloaking detection, we introduced a new model and "click counting" mechanisms. Traditional methods failed to do this adjust on SEM. Thus, the traditional methods are inefficient detect cloakings in SEM.

2.3 Cloaking Types

A website could differentiate search engine crawler from user web browser, from the following data:

2.3.1 IP address

Attackers blacklist a list of IP addresses and simply block their access.

2.3.2 HTTP user-agent header

Attackers identify bots by looking at user-agent, serve different content.

2.3.3 HTTP referer header

Referer tells landing page, where user is coming from, they may only allow access for users with referer http://www.google.com.

2.3.4 Cookie

This is used as a complementary method to referer, in order to track whether user is shown malicious content before. If so, then subsequent accesses are bad as well. If not, set cookie and disallow access to bad stuff.

This is mainly observed on websites, that try to provide illegal services. The intuition of doing this is to track past user visit, and provide further service to the victim.

2.3.5 Number of visit times

The website shows malicious content to a specific IP only once.

2.4 Previous Work

2.4.1 Simhash

[7] did a large-scale evaluation of algorithms in finding near-duplicate web pages and results show that Simhash performs better than Minhash (text shingling).

Previous work on simhash. [11] employ simhash to detect near duplicate at scale.

2.4.2 Cloaking Detection

The major challenge in claoking detection, is to differentiate dynamic pages from blatantly different content. Comparing document word by word is very expensive and slow, therefore, various ways to test similarity of documents are proposed. [8] considered cloaking as one of the major search engine spam techniques. [12] proposed a method of detecting cloaked pages from browsers by installing a toolbar. The toolbar would send the signature of user perceived pages to search engines. [17] use statistics of web pages to detect cloaking, [4] detected syntactic on the most popular and monetizable search terms. They showed that monetized search terms had a higher prevalence of cloaking than popular terms. Referrer cloaking was first studied by [16]. They found a large number of referrer cloaking pages in their work. [10] use tag based methods,

[15] extended their previous efforts to examine the dynamics of cloaking over five months, identifying when distinct results were provided to search engine crawlers and browsers. They used text-based method and tag-based method to detect Cloaking page. [5] use summarize previous work and compare text, tag, link based approaches. However, JavaScript redirects were not able be handled by their crawler.

However, non of them take into consideration the data collection part. Nor do they take into account the efficiency of the algorithms.

In order to detect cloaking, we introduce Simhash-based Website Model (SWM) and design and implement a much more efficient algorithm based on SWM, which has comparable false positive rate and true positive rate. We take into account the efficiency of the algorithm and implement a plugin which introduce negligible overhead

to user. With crowdsourcing, we can solve cloaking at scale.

3 Methodology

As mentioned in §2, there are two challenges in cloaking detection: reveal the content and handle dynamics of a webpage.

In order to model the dynamics of a webpage, we propose Simhash-based Website Model, that is, use clusters learned from fuzzy hashs to model average and variance of each change. This is based on the assumption that, the content and layout of a website delivered to different users each time, is consistent despite of the dynamic part on the page, e.g. advertisements. In order to reveal the uncloaked content, we design and implement simhash algorithm as browser plugin, enabling crowdsourcing from user side. Besides, we demonstrate the complexity of simhash algorithm, and show that this plugin introduces negligible overhead to user browser.

3.1 Simhash-based Website Model

3.1.1 Distance Approximation

Simhash [3] is a dimensionality reduction technique. It is a fuzzy hash function family that maps a high dimension dataset into fixed bits and preserves the following properties: (A) The fingerprint of a dataset is a hash of its features, and (B) The hamming distance between two hash values is an estimation of the distance between the original datasets. This is different from cryptographic hash functions like SHA-1 or MD5, because they will hash two documents which differs by single byte into two completely different hash-values and the hamming distance between them is meaningless. However, simhash will hash them into similar hash-values.

In modeling the content and layout change of a website, we want to do learning and comparing on a per url basis, therefore, we are assuming that the small hamming distance between

Since we are modeling website on a per url basis and parameters in url may change every time of visit, it is necessary to define the granurity of comparison. We took a simplified approach, we simply strip all the parameter values and keep all the parameter names, and throw away the scheme field, i.e. *http://www.gatech.edu/?user=1234* is simplified to *//www.gatech.edu/?user=*

3.1.2 Computation

In terms of simhash computation, we use the same settings described in [11], which turns out to be effective given the corpus of 5 billion websites over the Internet.

The computation of simhash start from a set of features. Given a set of features extracted from a document and their corresponding weights, we use simhash to generate an f-bit fingerprint as follows. We maintain an f-dimensional vector V, each of whose dimensions is initialized to zero. A feature is hashed into an f-bit hash value. These f bits (unique to the feature) increment/decrement the f components of the vector by the weight of that feature as follows: if the i-th bit of the hash value is 1, the i-th com- ponent of V is incremented by the weight of that feature; if the i-th bit of the hash value is 0, the i-th component of V is decremented by the weight of that feature. When all features have been processed, some components of V are positive while others are negative. The signs of components determine the corresponding bits of the final fingerprint. In this work, f is set to 64.

3.1.3 Insights On Feature Set Selection

From the computation of the simhash, we can draw two insights: (A) The order of the features in the original dataset does not matter, because the algorithm randomly projects each feature to f-dimensional vector and count the weighted presence in each dimension. (B) Size of the feature set should be relatively large. Simhash is known to perform badly on near duplicate detection when there are feature words. The reason is that, when the number of features are small, each feature votes too much on the vectors, difference in one feature may result in many bits of difference.

Insight (A) is telling us, if the order of the original feature matters, the feature set should cover this. Insight (B) means, simhash works on relatively large number of datasets. The two insights are used in designing the text and dom simhash.

3.1.4 Text Simhash and Dom Simhash

In order to detect cloaking, we need to capture the bahavior and similarity that a same website maintains. That is to say, we not only need to look at the text-based simhash, but also dom-based simhash. We implemented the text-simhash algorithm same as the simhash algorithm used in spam detection by Google. This algorithm extract words, bi-gram, tri-gram set (repeated elements only recorded once) from a website and compute simhash using simhash algorithm described in [3]. Because there are usually large number of words on a website, insight (B) suffices. Besides, bi-gram, tri-gram is essentially the structure of the document, thus insight (A) suffices. This approach and has great benefit in our use scenario, because this requires a single pass of the document and no extra information. In contrast, [11] implement extract weighted features using standard IR techniques like tokenization,

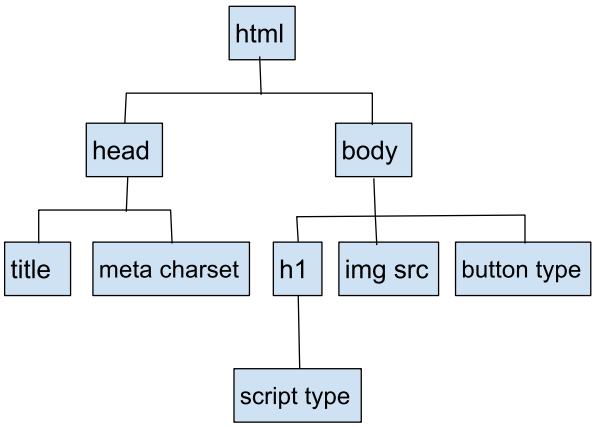

Figure 1: DOM tree example

case folding, stop-word removal, stemming and phrase detection, and may introduce more overhead and need extra knowledge.

The content of a web page, Document Object Model (DOM), is organized in the form of a tree. Figure Figure 1 shows a typical dom tree. There is no current algorithm for generating dom simhash. Therefore, we design an algorithm based on drawn insights.

For each dom tree, we record presence of each node, as well as presence of each child parent pair. The node set tells us information about what tag is present in this page, and child parent pair tells us how these tags are organized, suffices insight (B). And the number of features extracted in this fashion is relatively large, suffices insight (A).

It is pretty straightforward from Figure 3 that, text simhash changes rapidly, indicating dynamic nature of this website, and dom simhash changes relatively slow and less.

Till now, we have demonstrated the algorithm we are using to generate text-simhash and dom-simhash out of a website. Based on our observation, the text-simhash may change rapidly, while dom-simhash relatively remain the same.

3.1.5 Aggregating / Clustering

Assume we are monitoring the same website over a period of time. This website have dynamic changes all the time. But there can be another kind of change - whole page change. In this case, it is reasonable to first separate them apart and look at each of them.

Different from [11], we not only want to know whether two pages are duplicate, we also want to know the patterns of these simhash. In this work, we employ hierarchical clustering to do this job.

Since simhash maps a hight dimensional features set

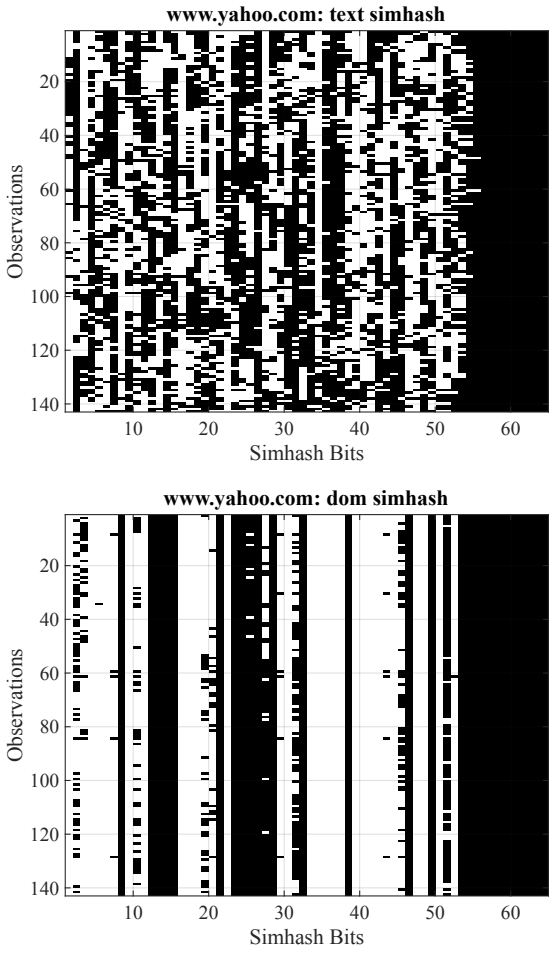

Figure 2: Yahoo simhash changes over 7x24 period Feb.1 - 7, 2015

into fixed number of bits, where hamming distance represents the similarity between the original feature set. We leverage the fact that hamming distance is euclidean distance (L1 norm on 64 dimension). On each dimension, the original value is either 0 or 1. But it can be averaged and centriod can be computed.

In this work, we use hierarchical clustering [9] to cluster collected simhash. The distance metric used is hamming distance, the linkage method used is average. Criterion used is inconsistent coefficient.

These choice is actually straightforward. **Average linkage:** Consider the following use case, spider collect multiple copies of a website and compute corresponding simhash $S_{spider} = s_i, i \in (1,n)$ and the user return observation s_j for this website. In order to compare whether s_j with S_{spider}, and take into consideration of the all the collected simhash, the distance from s_j to centroid of S_{spider} seems a reasonable measure.

Inconsistency coefficient: this coefficient character-

izes each link in a cluster tree by comparing its height with the average height of neighboring links below it in the tree. The higher the value of this coefficient, the less similar the objects connected by the link. By using threshold of inconsistency coefficient as stop scenario, we could get several clusters for each website.

This is reasonable, because for collected copies of a website, there can be two kind of changes, minor change and major chagne. Minor change refers to the change that happens daily or even every time, for example, advertisements are different on each visit. Major chagne refers to new version of a website, e.g., one domain expired and is re-sold to another entity, which follows completely different dynamics. By doing clustering, we could handle differentiate minor and major change of a website, and build separate model accordingly. In the clustering phase, let d denote the distance from s_j to

$$\frac{d-\mu}{\sigma} < T_{learn} \tag{1}$$

3.2 Cloaking Detection

In the above section, for each website, we have learned clusters, which is Simhash-based Website Model. In order to use SWM, let's first look at how the comparison is going to be done. In order to detect SEO and SEM cloaking, this work first search and click results with normal user agent, then visit landing urls collected with google bot user agent. For a specific website, we denote the spider copies as $S_{spider} = s_i, i \in (1,n)$, n is the number observations collected by spiders. Denote the user observation as s_j. In order to compare s_j with S_{spider}, S_{spider} is first lerned through hierarchical clustering, and this results in $C_{spider,i}, i \in (1,c)$, c is the number of clusters learned.

For each cluster $C_{spider,i}$, we have link heights and centroid recorded. When comparing s_j with $C_{spider,i}$, distance from s_j to centroid of $C_{spider,i}$, i.e. $d_{j,i}$ can be computed. In order to reject s_j, we use inconsistent coefficient as well. If

$$\frac{d_{j,i}-\mu}{\sigma} > T_{detect} \tag{2}$$

we reject this observation. An observation is marked as cloaking only when all the clusters reject it.

However, in reality, there can be consistent differences between what spider sees and what user sees and this is totally leagal. An example is, a website may present advertisements to normal user, but non-advertisement version to spider. Besides, there are websites that rarely changes at all. Therefore, σ is zero, thus Equation 2 doesn't work. To fix the two problem, we introduce a minimum radius R_{detect}. The modified formula is like this:

$$\frac{d_{j,i}-R_{detect}-\mu}{\sigma} > T_{detect} \tag{3}$$

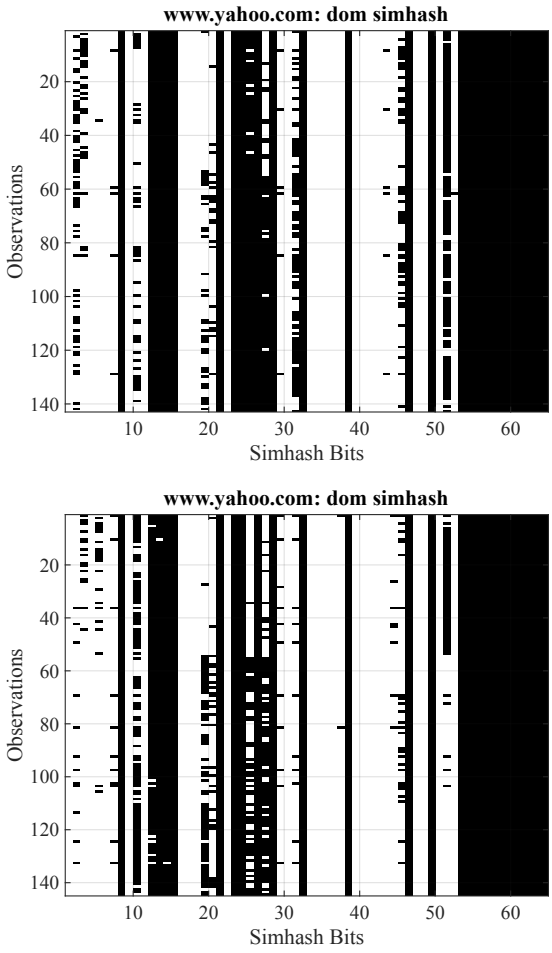

Figure 3: Comparison of user and google seen dom simhash

4 Evaluation

In the above section §3, we propose the SWM and explains how it can be used to do cloaking detection (outlier detection). There are three parameters to be learned, the upper bound of inconsistent coefficient in learning phase T_{learn}, the lower bound of inconsistent coefficient in the detection phase T_{detect}, the fix parameter minimum radius R_{detect}. In this section, we describe our dataset and groundtruth and train and test and performance of the proposed model.

4.1 Dataset

Because we want to measure cloaking in SEO and SEM, we have compiled a list of cloaking words, from the policies specified by Adwords, inspired by the words collection process in [15]. We have looked at the policies, and collected XXX: N words, from ad network abuse,

adult abuse, alcohol abusive, dangerous behavior abuse, dishonest behavior abuse, health abuse, gambling abuse.

We have collected four datasets, spammy search, $D_{spam,search}$, hot search, $D_{hot,search}$, spammy ads $D_{spam,ad}$, hot ads, $D_{hot,ad}$. $D_{spam,search}$, spammy search words collected manually, XXX: N words. $D_{hot,search}$, hot search monthly words from Jan 2013, Dec, 2014, 24 month in total, XXX: N words. From $D_{hot,search}$, we evaluate the monetizability of each word through Google Keyword Planner [1] and simply remove the words that have no bid price. This results in our spammy advertisements words set, $D_{spam,ad}$, we have collected advertisements from 573 words. $D_{hot,ad}$, we need as much words as possible, i.e. as much ads as possible, therefore first download hots words list from Google Trend, for hot words in each category, we download weekly, monthly, and yearly from 2004 to 2014, and then we find the useful keywords with Keyword Planner [1]. We have collected advertisements from 4108 words.

For the above keywords, we do the following: Step 1: For each word, with chrome user agent, we click and visit search results and advertisements on first twenty pages. Step 2: For the landing pages collected in step 1, with google bot user agent, we visit them 6 times at an interval of around 20 miniutes.

6 copy is collected because we need to model the distribution of websites. It is a tradeoff between space and precision. If we want to model the website better, we may want to collect as many examples as possible.

4.2 Groundtruth

Similar to [10], we first remove duplicates (same simhash from user side and Google side), then label $D_{spam,search}$ and $D_{hot,search}$. We also manually add the examples that we observe when we do case study. We manually labeled 1195 cloaking examples. And we randomly 5308 samples from non-cloaking dataset. This composes our dataset 6503 for algorithm learning.

4.3 Detection and Evaluation

4.3.1 Selection of T_{learn} and T_{detect}

Because R_{detect} is a parameter to allow the system to handle consistent difference between spider and user copies, therefore, we first set detect R_{detect} to be zero, do 5 fold stratified cross validation [13]. on the result. In the learning phase, Our objective function is to first minimize the total number of errors in classification $E = FP + FN$, and if E is the same, minimize $d = T_{detect} - T_{learn}$. This is reasonable because d is the area that we cannot judge, and the smaller the area, indicates that the learned model is more compact. The two objective function are widely used

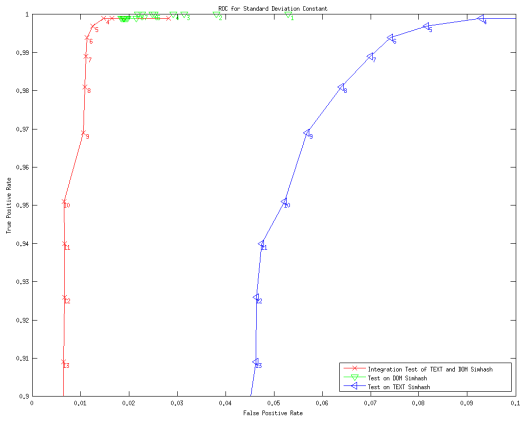

Figure 4: ROC for DOM, TEXT, DOM & TEXT

metrics in machine learning parameter selection XXX: cite.

By applying five-fold cross validation on the groundtruth, and the described objective function, dom simhash returned $T_{detect,dom} = 1.8$ and $T_{learn,dom} = 0.7$, and text simhash yields $T_{detect,text} = 2.1$ and $T_{learn,text} = 0.7$.

4.3.2 Radius Selection R_{detect}

From the above section, we have selected $T_{detect,dom} = 1.8$ and $T_{learn,dom} = 0.7$, $T_{detect,text} = 2.1$ and $T_{learn,text} = 0.7$. Now, we want to decide, $R_{detect,text}$ and $R_{detect,dom}$ separately. In this section, we conduct three experiments: cloaking detection using (1) dom simhash (2) text simhash (3) both dom simhash and text simhash. Again, we use five-fold cross validation to learn and test the dataset. The selected parameter for dom simhash is $R_{detect,dom} = 17$, text simhash is $R_{detect,text} = 16$. If we consider great difference in both text and dom as cloaking (intersect the detected results), the result is $R_{detect,dom} = 13$, $R_{detect,text} = 17$, get $FPR = 0.1\%, TPR = 94\%$. Figure Figure 4 gives an illustration on how FPR and TPR changes as threshold for DOM and TEXT changes.

With the learned model, in Section §5 detect on the four dataset that we obtained and manually label the results. In our detection, we want a low false positive rate, therefore, we use the FPR = 0.1%, TRP = 92% in Figure 4. This yields to $R_{detect,dom} = 16$, $R_{detect,text} = 20$.

5 Measurement

With the model built in §4, we detect cloaking in the four collected datasets, spammy search, $D_{spam,search}$, hot search, $D_{hot,search}$, spammy ads $D_{spam,ad}$, hot ads, $D_{hot,ad}$. $D_{spam,search}$.

5.1 Cloaking in SEO

SEO: How severe is cloaking? How many categories and percentages of various cloaking? For each type of cloaking, what are their incentives?

XXX: Plot a log scale pie or histogram for categories in SEO Out of the 20k distinct urls, we detect 1600 urls. There are mainly composed of 60% phishing sites, 30%illegal services, 10% malicous domain. Phishing sites includes rogue pharmacy sites. Illegal services include gambling, essay writing. Malicious domain are verified manually, we manually visit these URLs and are redirected to malware download.

Compared to [15], we detected a relatively small percentage of cloaking, this is probably because search engine is taking active action now, or the cloaking methods have evolved.

5.2 Cloaking in SEM

How severe is cloaking? How many categories and percentages of various cloaking? For each type of cloaking, what are their incentives?

XXX: Plot a log scale pie or histogram for categories in SEM In SEM, we have detected 100 claoking examples out of 10k ads. This percentage is lower compared to SEO. However, ads are much more important because they matters, clicks in ads equals money. Most of the detected cloaking are providing illegal services.

We argue that, previous methods cannot be used to detect SEM cloaking, simply because performing clicks from search engine side, is ad fraud. In contrast, SWM simply collects the fuzzy signatures of websites. With privacy guarantee provided by RAPPOR [6], and onewayness of simhash, crowdsourcing is an achievable and elegant way to detect cloaking.

5.3 Cross Domain Spam Detection

In our detection result, we have observed many cloaking cases, where URL are completely different, while the content are similar or even the same. We argue that, the foundamental reason for them to do this is the low cost of getting a new URL and lack of efficient way to detect spammy content. Based on this observation, we propose content based blacklist to raise the bar for reusing spammy content. This approach leverages the most popular use of simhash - near duplicate detection. For spammy pages, in order to evade our detection, they not only need to change URL rapidly, but also update their content everytime, which can be expensive in their current mode if they want every copy their website to be different and have meaningful and stay attractive to user.

6 Discussion

This section discusses the potential deployment of the system.

6.1 Server based

Pros: practical, easy deployment. cons: IP cloaking, SEM, hard to decide crawling periods. Solution: IP cloaking, Buy 100k IP. SEM, Deduce visits by search engine.

6.2 Crowdsource based

Employ RAPPOR [6] to provide user privacy gurantee.

pros: 1.privacy 2.Low traffic. 3.SEM 4.Distributed computation 5. Remove the need to do redirect cloaking detection, leveraging the feature that the end goal of attackers is to reach user 6. could decide crawl period passively based on user clicks, data received are based on real users clicks, say, website traffic

cons: user incentives. Solution: Plugin to detect suspicious websites. API

The workflow Figure 5 is to collect page contents simhash on the user side, and compare them to simhash of the same link from ad serving company to find cloaking. When the differences of the simhashes are significantly large, the page is marked cloaking. We generate two simhash for page content and structure respectively. Intuition behind this is, simhash difference between different sites are larger than different visits of the same site. We build a two-phase system to detect cloaking: cluster learning phase, and cloaking detection phase. In the cluster learning phase, an ad company visit urls and generate simhash from its content with its owned IP, and learn pattern and distribution of the simhashes, i.e. simhash-based website model. In the cloaking detection phase, the ad company collects simhash from its users. Compare them with learned patterns, return cloaking score or mismatch

7 Conclusion

Current solution, multiple pass of data, doesnt work, and hard to handle various types of cloaking. Our model works because

References

[1] Google keyword planner. https://adwords.google.com/KeywordPlanner.

[2] Search engine marketing. http://en.wikipedia.org/wiki/Search_engine_marketing.

[3] CHARIKAR, M. S. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing* (2002), ACM, pp. 380–388.

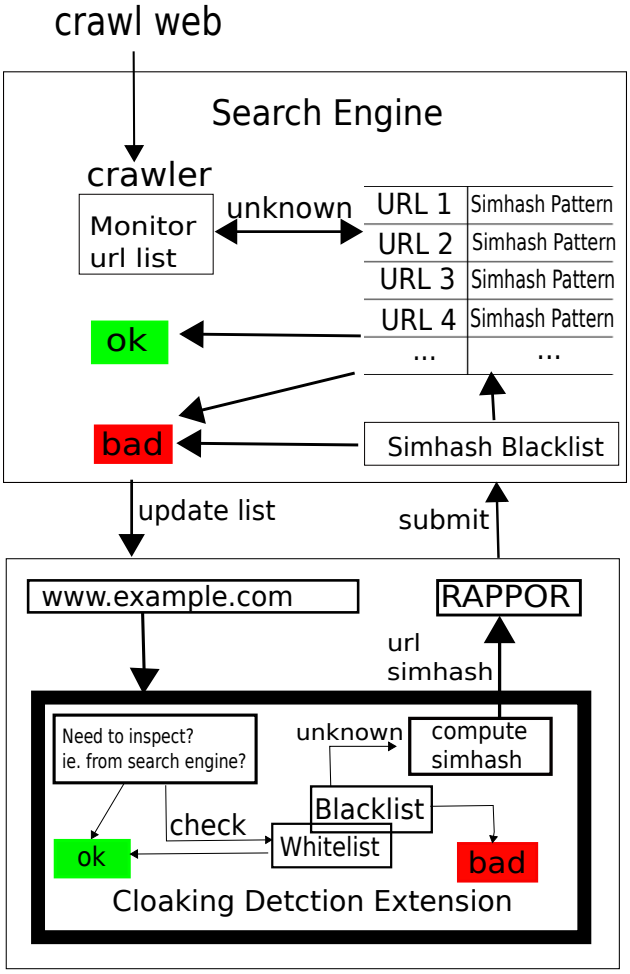

Figure 5: Workflow of crowdsource cloaking detection sysetem

[4] CHELLAPILLA, K., AND CHICKERING, D. M. Improving cloaking detection using search query popularity and monetizability. In *AIRWeb* (2006), pp. 17–23.

[5] DENG, J., CHEN, H., AND SUN, J. Uncovering cloaking web pages with hybrid detection approaches. In *Computational and Business Intelligence (ISCBI), 2013 International Symposium on* (2013), IEEE, pp. 291–296.

[6] ERLINGSSON, Ú., PIHUR, V., AND KOROLOVA, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 1054–1067.

[7] HENZINGER, M. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), ACM, pp. 284–291.

[8] HENZINGER, M. R., MOTWANI, R., AND SILVERSTEIN, C. Challenges in web search engines. In *ACM SIGIR Forum* (2002), no. 2, ACM, pp. 11–22.

[9] JONES, E., OLIPHANT, T., AND PETERSON, P. {SciPy}: Open source scientific tools for {Python}.

[10] LIN, J.-L. Detection of cloaked web spam by using tag-based methods. *Expert Systems with Applications 36*, 4 (2009), 7493–7499.

[11] MANKU, G. S., JAIN, A., AND DAS SARMA, A. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 141–150.

[12] NAJORK, M. A. System and method for identifying cloaked web servers, 2005. US Patent 6,910,077.

[13] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[14] TAO, C. Android app update flaw affects china-based users, Apr. 2014. http://www.blackhatworld.com/blackhat-seo/cloaking-content-generators/654307-fb-cloaker-recommendations-blackjack-wpcloaker-polarbacon.html.

[15] WANG, D. Y., SAVAGE, S., AND VOELKER, G. M. Cloak and dagger: dynamics of web search cloaking. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), ACM, pp. 477–490.

[16] WANG, Y.-M., AND MA, M. Detecting stealth web pages that use click-through cloaking. In *Microsoft Research Technical Report, MSR-TR* (2006).

[17] WU, B., AND DAVISON, B. D. Detecting semantic cloaking on the web. In *Proceedings of the 15th international conference on World Wide Web* (2006), ACM, pp. 819–828.