

Knock It Off: Profiling the Online Storefronts of Counterfeit Merchandise

Matthew F. Der, Lawrence K. Saul, Stefan Savage, Geoffrey M. Voelker
Department of Computer Science and Engineering
University of California, San Diego
{mfder, saul, savage, voelker}@cs.ucsd.edu

ABSTRACT

We describe an automated system for the large-scale monitoring of Web sites that serve as online storefronts for spam-advertised goods. Our system is developed from an extensive crawl of black-market Web sites that deal in illegal pharmaceuticals, replica luxury goods, and counterfeit software. The operational goal of the system is to identify the affiliate programs of online merchants behind these Web sites; the system itself is part of a larger effort to improve the tracking and targeting of these affiliate programs. There are two main challenges in this domain. The first is that appearances can be deceiving: Web pages that render very differently are often linked to the same affiliate program of merchants. The second is the difficulty of acquiring training data: the manual labeling of Web pages, though necessary to some degree, is a laborious and time-consuming process. Our approach in this paper is to extract features that reveal when Web pages linked to the same affiliate program share a similar underlying structure. Using these features, which are mined from a small initial seed of labeled data, we are able to profile the Web sites of forty-four distinct affiliate programs that account, collectively, for hundreds of millions of dollars in illicit e-commerce. Our work also highlights several broad challenges that arise in the large-scale, empirical study of malicious activity on the Web.

Categories and Subject Descriptors

I.5.4 [Pattern Recognition]: Applications—*Text processing*; K.4.4 [Computers and Society]: Electronic Commerce—*Security*

General Terms

Security

Keywords

Web page classification; Email spam

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD'14, August 24–27, 2014, New York, NY, USA.

Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM 978-1-4503-2956-9/14/08 ...\$15.00.

<http://dx.doi.org/10.1145/2623330.2623354>.

1. INTRODUCTION

The Web plays host to a broad spectrum of online fraud and abuse—everything from search poisoning [5, 20] and phishing attacks [12] to false advertising [10], DNS profiteering [4, 15], and browser compromise [17]. All of these malicious activities are mediated, in one way or another, by Web pages that lure unsuspecting victims away from their normal browsing to various undesirable ends. Thus, an important question is whether these malicious Web pages can be automatically identified by suspicious commonalities in appearance or syntax [1, 4, 8, 11, 18, 19, 21]. However, more than simply distinguishing “bad” from “good” Web sites, there is further interest in classifying criminal Web sites into groups of common origin: those pages that are likely under the control of a singular organization [3, 9, 13]. Indeed, capturing this “affiliation” property has become critical both for intelligence gathering as well as both criminal and civil interventions. In this paper, we develop an automated system for one version of this problem: the large-scale profiling of Web sites that serve as online storefronts for spam-advertised goods.

While everyone with an e-mail account is familiar with the scourge of spam-based advertising, it is only recently that researchers have come to appreciate the complex business structure behind such messages [14]. In particular, it has become standard that merchants selling illegitimate goods (e.g., such as counterfeit Viagra and Rolex watches) are organized into *affiliate programs* that in turn engage with spammers as independent contractors. Under this model, the affiliate program is responsible for providing the content for online storefronts, contracting for payment services (e.g., to accept credit cards), customer support and product fulfillment—but direct advertising is left to independent affiliates (i.e., spammers). Spammers are paid a 30–40% commission on each customer purchase acquired via their advertising efforts and may operate on behalf of multiple distinct affiliate programs over time. Such activities are big business with large affiliate programs generating millions of dollars in revenue every month [7].

Thus, while there are hundreds of thousands of spam-advertised Web sites and thousands of individual spammers, most of this activity is in service to only several dozen affiliate programs. Recent work has shown that this hierarchy can be used to identify structural bottlenecks, notably in payment processing. In particular, if an affiliate program is unable to process credit cards, then it becomes untenable to sustain their business (no matter the number of Web




GlavMed	Ultimate Rep.	SoftSales
		

Table 1: Screenshots of online storefronts selling counterfeit pharmaceuticals, replicas, and software.

sites they operate or spammers they contract with). Recently, a concerted effort by major brand holders and payment networks to shutdown the merchant bank accounts used by key affiliate programs demonstrated this vulnerability concretely. Over a short period of time, this effort shut down 90% of affiliate programs selling illegal software and severely disabled a range of affiliate programs selling counterfeit pharmaceuticals [13].

The success of this intervention stemmed from a critical insight—namely, that the hundreds of thousands of Web sites harvested from millions of spam emails could be mapped to a few dozen affiliate programs (each with a small number of independent merchant bank accounts). At the heart of this action was therefore a classification problem: how to identify affiliate programs from the Web pages of their online storefronts?

There are two principle challenges to classification in this domain. The first is that appearances can be deceiving: storefront pages that render very differently are often supported by the same affiliate program. The seeming diversity of these pages—a ploy to evade detection—is generated by in-house software with highly customizable templates. The second challenge is the difficulty of acquiring training data. Manually labeling storefront pages, though necessary to some degree, is a laborious and time-consuming process. The researchers in [9] spent hundreds of hours crafting regular expressions that mapped storefront pages to affiliate programs. Practitioners may endure such manual effort once, but it is too laborious to be performed repeatedly over time or to scale to even larger corpora of crawled Web pages.

Our goal is to develop a more automated approach that greatly reduces manual effort while also improving the accuracy of classification. In this paper we focus specifically on spam-advertised storefronts for three categories of products: illegal pharmaceuticals, replica luxury goods, and counterfeit software (Table 1). We use the data set from [9] consisting of 226,000 potential storefront pages winnowed from six million distinct URLs advertised in spam. From the examples in this data, we consider how to learn a classifier that maps storefront pages to the affiliate programs behind them.

We proceed with an operational perspective in mind, focusing on scenarios of real-world interest to practitioners in computer security. Our most important findings are the following. First, we show that the online storefronts of several dozen affiliate programs can be distinguished from automatically extracted features of their Web pages. In particular, we find that a simple nearest neighbor (NN) classifier on HTML and network-based features achieves a nearly perfect accuracy of 99.99%. Second, we show that practitioners need only invest a small amount of manual effort in



Figure 1: Data filtering process. Stage 1 is the entire set of crawled Web pages; stage 2, pages tagged as pharmaceutical, replica, and luxury; stage 3, storefronts of affiliate programs matched by regular expressions.

the labeling of examples: with just one labeled storefront per affiliate program, NN achieves an accuracy of 75%, and with sixteen such examples, it achieves an accuracy of 98%. Third, we show that our classifiers are able to label the affiliate programs of over 3700 additional storefront pages that were missed by the hand-crafted regular expressions of the original study. Finally, we show that even simple clustering methods may be useful in this domain—for example, to identify new affiliate programs or to reveal when known affiliate programs have adopted new software engines.

2. DATA SET

We use the data set of crawled Web pages from the measurement study of the spam ecosystem by Levchenko et al. [9]. Figure 1 depicts how they collected, filtered, and labeled the data, which we summarize in this section; we refer the reader to their paper for a more detailed explanation of their methodology.

2.1 Data Collection

Over a period of three months, Levchenko et al. [9] crawled over six million URLs from various spam feeds, including one feed from a top Web mail provider and others captured as output from live spamming bots. From the combined feeds, they obtained 1,692,129 distinct landing pages after discarding duplicate pages referenced by multiple URLs. For each landing page the crawler saved the raw HTML as well as a screenshot. In addition, a DNS crawler collected network information on the extracted domains. The first level in Figure 1 corresponds to this set of Web pages.

2.2 Data Filtering

The authors of [9] focused on three popular categories of spam-advertised goods—illegal pharmaceuticals, replica luxury goods, and counterfeit software. They used keyword matching to filter out Web pages that were unrelated to these categories of merchandise. Keywords included both

brand names, such as *Viagra* and *Cialis*, in addition to more generic terms, such as *erectile dysfunction* and *drug*. This filtering removed pages whose text did not advertise for sales of pharmaceuticals, replicas, or software—an omission that runs counter to the business interest of any viable storefront. The absence of such text was found to be a reliable screen for non-storefront pages. A total of 226,439 storefront pages emerged from this filter, which were then broadly grouped into the three categories of pharmaceuticals, luxury goods, and software. These categorized pages, shown in the second level of Figure 1, comprise the universe of pages that we consider in this paper.

The filtering by domain-specific keywords may be viewed as a simple operational heuristic to narrow the massive data set of blindly crawled URLs to a subset of pages of interest. Note that the filtering in this step was purposely conservative so that legitimate storefront pages were not winnowed from the data set.

2.3 Data Labeling

Through a combination of manual inspection and heuristic pattern-matching, the authors of [9] managed to link the spam-advertised storefronts for pharmaceuticals, luxury goods, and pirated software to individual affiliate programs. An initial round of manual inspection was necessary because even the cast of affiliate programs was not known a priori. Thus the first pass over crawled Web pages focused on identifying prominent affiliate programs and collecting a moderate sample of storefront pages from each. The authors identified forty-five distinct affiliate programs in this way (although one was later discovered to be a subset of another).

The next and most time-consuming step of this process was to expand the number of storefront pages labeled for each affiliate program. The authors of [9] sought to automate this process through the use of regular expressions. Specifically, for each affiliate program, they devised a set of characteristic regular expressions that matched the HTML content of its online storefronts but not those of other programs. They also fixed an integer threshold for each affiliate program; if the Web page of an online storefront matched this number of regular expressions (or more), then it was labeled as belonging to the program. Using this approach, they were able to identify the affiliate programs of 180,690 online storefronts. The bottom level in Figure 1 represents this final set of storefront pages.

The above approach depended on the meticulous crafting of regular expressions that distinguish the storefront pages of different affiliate programs. To devise such an expression, it was necessary (roughly speaking) to find a pattern that was present in all the pages of one program *and* absent in the pages of all other programs. The painstaking effort required for this approach is best illustrated by example. Here is one regular expression for storefront pages in the GlavMed affiliate program:

```
var\s+SessionType\s*=\s*"URL";\s*\var\s+
SessionPrefix\s*=\s*" [0-9a-fA-F] 32";\s*\var\s+
SessionName\s*=\s*"USID";
```

It can take a couple hours just to hone a single regular expression such as the one above. Much more time, naturally, is required to obtain coverage of multiple affiliate programs. The full effort in [9] involved not only the careful scrutinization of HTML content, but also iterative refinement and ex-

Examples	1st 3 months	2nd 3 months
Labeled	178,281	29,581
Unlabeled	43,442	12,756
Largest class	58,215	13,529
Smallest class	2	0

Table 2: Summary of the data from crawls of consecutive three-month periods.

tensive validation. In total, the authors estimated that they spent over two hundred man-hours designing and validating the regular expressions for all forty-five affiliate programs.

In the present work, we limited our study to forty-four¹ of these affiliate programs. We also discarded pages from the above collection which lacked a screenshot (which we need for manual validation of our results). From the initial three-month Web crawl, this left us with 178,281 labeled storefronts and 43,442 unlabeled storefronts; all of these storefronts were tagged as selling pharmaceuticals, luxury goods, or software, but only the former were successfully mapped to affiliate programs by regular expressions.

We also obtained a subset of data crawled from a period of the following three months. These Web pages were collected and labeled in the same manner as before; in particular, they were labeled without updating the regular expressions from the initial crawl. It is known that storefronts evolve over time, and therefore we do not expect the labels during this time period to be as reliable as the earlier ones. Table 2 summarizes the data from each crawl. Note also the severe class imbalance: the largest affiliate program has 58,215 distinct storefronts, while the smallest has just 2. In addition, there were five affiliate programs whose regular expressions in the first period did not detect any storefront pages in the second period.

3. AN AUTOMATED APPROACH

It should be clear that the approach of Section 2.3 does not scale well with the number of storefront pages or affiliate programs. This approach is also too time-consuming to repeat on a regular basis—updating the regular expressions before they go stale—as would be necessary to maintain a working operational system. In this section we describe a faster and more highly automated approach for the classification of storefront pages by affiliate program. Our data-driven approach consists of two steps. First, we use automatic scripts to extract tens of thousands of potentially informative features from the crawl of each online storefront. Next, we represent these features in a lower-dimensional vector space and use nearest neighbor classification to identify affiliate programs from whatever labeled examples are available. Our hope with this approach is to avoid the painstaking crafting of regular expressions in favor of simple, well-tested methods in data mining and machine learning.

3.1 Feature Extraction

For each online storefront in our data set, we have both the HTML source of its Web page and the DNS record of its domain. Together these provide a wealth of information

¹The affiliate program we exclude was later found to be a proper subset of a different program.

about the storefront that can be extracted by automatic methods. We consider each in turn.

3.1.1 HTML Features

We know from previous work [9] that the storefronts of different affiliate programs can be distinguished by properties of their raw HTML. This is likely the case because each affiliate program uses its own software engine to generate storefront templates; as a result, different storefronts within the same affiliate program often have similar DOM² structures. Indeed, the regular expressions of [9] only work insofar as commonalities in implementation exist among the different storefronts of individual affiliate programs.

We extract HTML features using a bag-of-words approach. This approach ignores the ordering of HTML tags and text within a Web page, but it is simple enough for the quick and automatic extraction of thousands of potentially informative features. In this respect our approach differs considerably from the manual “feature engineering” of regular expressions; we do not seek to find the most informative features ourselves, merely to extract all candidate features for a statistical model of classification.

A key consideration of our approach is how to encode HTML tags as words. Consider the following snippet of HTML code:

```

```

The problem in encoding this snippet is how to achieve the appropriate level of granularity. It is clear that important information is carried by the context in which individual HTML tags appear. However, this information is lost if we encode the tags and tokens in this snippet as single words. On the other hand, it is clearly infeasible to encode this whole snippet, and others like it, as a single “word”; taken as a whole, the HTML element is akin to a full sentence of text. Our compromise between these two extremes is to encode each tag-attribute-value triplet as a word. We also remove whitespace. This solution, for example, parses the above HTML into the following words:

```
img:src=example.jpg
img:alt=Examplepic
img:height=50
img:width=100
```

To parse HTML element content—the text between start and end tags—we treat the text as a single string that we split on certain characters (, ; {}) and whitespace. We then form “words” by stripping non-alphanumeric characters from the resulting substrings and converting all letters to lowercase. We parse HTML comments in the same way. Following convention, we exclude words that are either very common (e.g., stopwords) or very rare (appearing in few Web pages).

The bag-of-words approach has the potential to generate very large feature vectors: the dimensionality of the feature space is equal to the number of extracted words. We limited our vocabulary of words to the HTML of storefronts from the initial three-month crawl. From this period alone, however, we extracted over eight million unique words. Since most of these words were likely to be uninformative, we used sim-

²The document object model (DOM) refers to the representation of a Web page as a tree of HTML elements.

ple heuristics to prune this space. In particular, for each of the forty-four affiliate programs, we only retained words that appeared in the HTML source of 10% (or more) of the program’s online storefronts. Finally, concatenating these words we obtained a total of 34,208 HTML-based features for classification.

3.1.2 Network Features

The authors of [9] also developed a DNS crawler to enumerate all address and name server records³ linked to the domains of spam-advertised URLs. We expect such records to help us further in identifying affiliate programs from their online storefronts. In particular, we expect different affiliate programs to use different naming and hosting infrastructures, and thus we might hope to distinguish them by the name and Web servers that support their domains. Such information may also serve to link online storefronts with different Web pages to the same affiliate program. For example, the DOM trees of two storefronts might be quite different, but if their Web pages were hosted at the same server, then we could predict with confidence that they belong to the same affiliate program.

We mined both the address and name server records of online storefronts for useful features. An address record, or A record, maps a domain name to the IP address of the machine on the Internet where the domain is hosted. For example, the A record of the domain name `ucsd.edu` maps to the IP address `132.239.180.101`. A name server record, or an NS record, identifies the name servers that provide these mappings of names to addresses. The NS record for the domain `ucsd.edu` contains the domain names of four name servers: `ns0.ucsd.edu`, `ns1.nosc.mil`, `ns1.ucsd.edu`, and `ns2.ucsd.edu`. In addition, these name servers have their own IP addresses; e.g., the A record for `ns0.ucsd.edu` maps to the IP address `132.239.1.51`. Note that a storefront’s domain name may be associated with multiple A and NS records. These multiple associations are a counter-defense against security measures such as domain blacklisting.

For each storefront domain, we extracted the IP address from its A record, the IP addresses of its name servers, and the autonomous system numbers (ASNs) of both these IP addresses. (The ASN of an IP address uniquely identifies its Internet Service Provider.) All together, these IPs and ASNs yield four categorical network features for each storefront. We tallied the number of unique IPs and ASNs observed during the initial three-month crawl of storefronts: there were 17,864 unique A record IPs, 8,825 unique NS record IPs, 2,259 unique ASNs of A record IPs, and 1,853 unique ASNs of NS record IPs. We encoded each categorical feature with k uniquely occurring values as a k -dimensional binary vector (i.e., a simple unary encoding). Concatenating the elements of these vectors, we obtained a total of 30,791 binary-valued network features for each online storefront.

3.2 Dimensionality Reduction & Visualization

Table 3 shows the numbers of HTML-based and network-based features, as well as the total number of features when combined. We were interested in the different types of information contained in HTML vs. network features. To explore the value of different features, we experimented with classi-

³Due to a technical issue, however, this information was only recorded during the first three-month crawl of online storefronts.

Features	Count	Density	PCA	Unique Exs.
Bag-of-words	34,208	2.20%	66	12.07%
Network	30,791	0.12%	665	6.09%
Both	64,999	1.21%	72	31.52%

Table 3: Feature counts, density of the data, dimensionality after principal components analysis, and percentage of unique examples.

fiers that used either one set of features or the other; this was in addition to classifiers that worked in their combined feature space.

In all of these cases, the data is very sparse and high dimensional. To reduce the dimensionality of the data we used principal component analysis (PCA) [6]: in particular, after normalizing the feature vectors to have unit length, we projected them onto enough principal components to capture over 99% of the data’s variance. We did this for each of the three feature spaces. The third and fourth columns of Table 3 show, respectively, the small percentages of non-zero features before PCA and the number of principal components needed to capture 99% of the data’s variance.

In the last column of Table 3 we have noted the high percentage of duplicate representations in feature space. Within a single affiliate program, it is often the case that different storefronts have identical representations as feature vectors; this is true even though every Web page in our data set has a distinct DOM tree. These duplicates arise when the extracted features do not contain the information that distinguishes actual Web pages. For example, two storefronts may have the same bag-of-words representation if their Web pages differ only slightly in their HTML (e.g., a single link, image path, or JavaScript variable name) and if, in this case, the differentiating words were too rare to be included as features. Likewise, two storefronts may have equivalent network features if they are hosted on the same server.

In Section 2 we noted that 180,690 *distinct* storefront pages were collected. The many duplicates (and many more near-duplicates) arise from the operational constraints faced by affiliate programs. To maximize their sales, affiliate programs must deploy and support many storefronts across the Internet. This process can only be streamlined by the aggressive use of template engines. Ironically, it is precisely the requirement of these programs to operate at scale that makes it possible to automate the defenses against them.

For illustration, Figure 2 plots the projections of storefront feature vectors from the largest affiliate program (EvaPharmacy) onto the data’s two leading principal components. Two observations are worth noting. First, the distribution is clumpy, with some irregularly-shaped modes, evidence of the variety of storefronts in this single program. Second, some storefronts which render very differently map to nearby points. This proximity is evidence of a similar underlying structure that associates them to the same affiliate program.

3.3 Nearest Neighbor Classification

We use nearest neighbor (NN) classification [2] to identify the affiliate programs of unlabeled storefronts. In particular, we store the feature vectors of the labeled storefronts (after PCA) in memory, and for each unlabeled storefront, we compute its NN in Euclidean distance among the training examples. Finally, we identify the affiliate program from



Figure 2: Projection of storefront feature vectors from the largest affiliate program (EvaPharmacy) onto the data’s two leading principal components.

that of its NN. A common extension is to compute k nearest neighbors and to take the majority vote among them, but we did not find this necessary for our results in this domain.

There are many statistical models of classification—some of them quite sophisticated—but in this domain we were drawn to NN classification for reasons beyond its simplicity. A primary reason is that NN classifiers do not make any strong parametric assumptions about the data: they do not assume, for instance, that the examples in different classes are linearly separable, or that the examples in a single class are drawn from a particular distribution. The two-dimensional projection of EvaPharmacy storefronts in Figure 2 illustrates the problematic nature of such assumptions. Some affiliate programs use multiple template engines to generate a diversity of storefronts, and this in turn yields a complicated distribution in feature space. NN classification copes well with these complexities; to label examples correctly, it is only necessary to find that they are closest to others that are similarly labeled. The resulting decision boundaries can be quite nonlinear.

There are other natural advantages of NN classification for our work. It does not incur any additional complexity when there are large numbers of classes, and in our problem, there are dozens of affiliate programs. Also, NN classifiers do not need to be re-trained when new labeled data becomes available; in our problem, an operational system would need to cope with the inevitable launch of new affiliate programs. A final benefit of NN classification is that it lends itself to manual validation. For this study, we implemented a visual Web tool for validating predictions in the absence of ground truth labels. The tool displays two adjacent storefronts—one unlabeled, the other its nearest neighbor among the labeled examples—along with the distance between them and hyperlinks to their HTML source code. The tool was invaluable for ongoing validation of the classifier, a necessary part of any eventual, real-world deployment.

4. EXPERIMENTS

In this section we evaluate the potential of automated, data-driven methods for identifying the affiliate programs behind online storefronts. We take a bottom-up approach,

beginning with simple diagnostics of the features in the previous section and ending with tests to measure how well a deployed NN classifier would perform in the real world. We begin with a brief overview of our experiments.

First we verify, as a proof of concept, that the online storefronts of different affiliate programs can in fact be distinguished by their HTML-based and network-based features. In these experiments, we attempt only to classify the Web pages of storefronts that belong to our previously identified set of forty-four affiliate programs. This is unrealistic as an operational setting, since in practice a working system would also need to classify Web pages that belong to other affiliate programs (or that do not even correspond to storefronts). However, as a logical first step, it is important to verify that the features we extract can—at least, in principle—separate the online storefronts of different affiliate programs.

Our next experiments remove the assumption of a “closed” universe of Web pages from previously identified affiliate programs. In these experiments we attempt to classify all Web pages that matched the category keywords of pharmaceuticals, replicas, or software (i.e., pages in the second level of Figure 1). A complication arises here in that we do not have “ground truth” labels for Web pages that were not matched and labeled by regular expressions. Many of these pages may indeed belong to other affiliate programs, but many from known programs may have simply gone undetected. (The regular expressions do not provide complete coverage; they can fail to match storefronts with slight, unforeseen variations in their HTML.) In this experiments we therefore follow NN classification with manual validation of its predictions. Here we succeed in detecting many additional storefronts from known affiliate programs that the regular expressions failed to match.

Our next experiments explore the effectiveness of NN classification at the outset of a Web crawl of spam-advertised URLs. At the outset of a crawl, practitioners must operate with very few labeled storefronts per affiliate program. While some manual labeling is unavoidable, the key question for practitioners is a simple one: how much labeling is “enough”? In particular, how many storefronts must be identified in each affiliate program before a NN classifier can take over, thereby automating the rest of the process? To answer this question we measure the accuracy of NN classification as a function of the number of labeled storefronts per affiliate program.

Our final experiments consider the most difficult scenario, one where practitioners lack the prior knowledge to label any storefronts by affiliate program. In these experiments we investigate how well (fully) unsupervised methods are able to cluster storefronts from multiple unidentified affiliate programs. We conduct two experiments to evaluate this potential. First, we cluster all the Web pages tagged as storefronts and measure how well the clusters correlate with known affiliate programs. Second, we investigate whether unsupervised methods can discover new affiliate programs among unlabeled storefronts. We also consider a closely related problem: recognizing when a known affiliate program generates storefront templates from a new software engine.

4.1 Proof of Concept

Our first experiments compare how well different features distinguish storefronts from a previously identified set of affiliate programs. In particular we experimented on the three

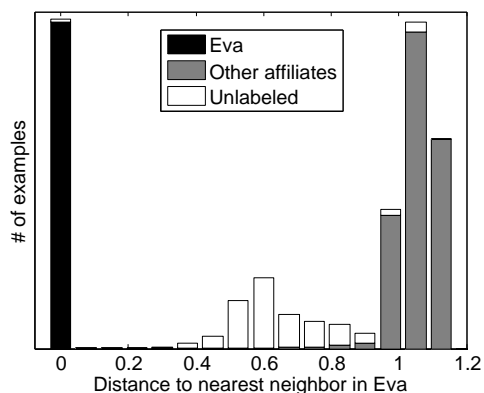


Figure 3: Histogram of three NN distances to EvaPharmacy storefronts: distances to storefronts in the same affiliate program, to those in other programs, and to unlabeled storefronts. Distances were computed using the HTML bag-of-words representation of the Web pages.

different sets of features listed in Table 3. For each of these, we computed the accuracy of NN classification averaged over ten 70/30 training/test splits of labeled storefronts from the first three-month Web crawl. The splits were random except that we did preserve the proportion of affiliate programs across splits; this was necessary to ensure that even the smallest affiliate programs were represented in the training examples. NN classification in these experiments was extremely accurate. The classifier achieved over 99.6% (test) accuracy with just network-based features, and with HTML bag-of-word features it verged on ideal prediction: on average only 8 out of 53,484 test examples were misclassified.

Figure 3 provides a visual explanation for the accuracy of NN classification. The distribution of NN distances illustrates how well-separated one class (EvaPharmacy) is from other classes. The shape of this distribution also reflects the large fraction of duplicates in the data set; these are different storefronts whose Web pages have the same representation in feature space.

These results demonstrate the power of very simple low-level features to distinguish the storefronts of different affiliate programs. Two points are worth emphasizing: first, that these features require no further processing for the high accuracy of NN classification, and second, that they are easy to extract automatically, requiring none of the manual effort of previous approaches.

In the experiments that follow we use only the bag-of-words features for NN classification. It is clear that these features suffice to distinguish the storefronts of different affiliate programs; they are also more readily available, as only the HTML source of a Web page is required. This consideration is important for ease-of-use as a deployed system. Also, as we have already remarked, the network-based features were not available for storefronts crawled during the second three-month period of data collection.

4.2 Labeling More Storefronts

Next we investigated whether our approach can identify the affiliate programs of storefronts whose HTML was not matched by regular expressions. One goal was to expand the

Regex matched	Not matched
 <p>ViaGrow</p>	 <p>ViagPURE</p>
 <p>swissapotheker.net swissapotheker24.com swiss-apotheker.com</p>	 <p>swiss-apotheker.net</p>

Table 4: Examples of storefronts matched by regular expressions (left column), and storefronts not matched but discovered by NN classification (right column).

number of labeled examples for subsequent evaluation of our approach. We generated bag-of-words feature vectors for the unlabeled Web pages using the vocabulary of HTML words in labeled storefronts. For each unlabeled page, we found the NN among labeled storefronts and marked it as a candidate match for that neighbor’s affiliate program. Finally, for each affiliate program, we ranked the candidate pages by the proximity of their NNs and examined the rankings with the Web tool we implemented for manual validation.

In total we labeled 3,785 additional storefronts in the first three-month crawl of spam-advertised URLs; these newly labeled storefronts came from twenty-eight of the forty-four known affiliate programs. For most of these programs the new storefronts were detected as the top-ranked candidates, and a simple distance threshold separated the “hits” from the “misses.” For example, Figure 3 shows that we discovered some new EvaPharmacy storefronts that were close or identical to their NN among labeled storefronts (shown by the small white segment of the first bar), but nothing beyond that. Only two affiliate programs had highly ranked candidate storefronts that belonged to a different but previously identified affiliate program.

Table 4 shows two storefronts detected by NN classification but missed by the regular expressions. In both cases, the discovered storefronts (right column) are very similar to their NN (left column) in both their HTML content and how they render in a browser. These examples expose the brittleness of the regular expressions, evading detection by a slight tweak to the domain name (`swiss-apotheker.net`) or a simple re-branding (ViaGrow → ViagPURE). Of course, a straightforward refinement of the regular expressions would rectify both these misses. However, it is precisely this need for refinement that we wish to avoid. The strength of our system is that it discovered and labeled these new storefronts automatically.

We repeated this same process to label new storefronts from the second three-month crawl of spam-advertised URLs. For this period, we found and labeled 761 new storefronts belonging to eighteen different affiliate programs. Table 5 shows how many additional storefronts we detected in all six months for certain affiliate programs. In the first time period, we detected the most new storefronts (1,467) for the RX-Promotion affiliate program, although this number only represented a 4% increase in the program’s size. However, we detected an eight-fold increase in storefronts for the affiliate program Club-first.

4.3 Classification in the Wild

Our next experiments performed NN classification on all spam-advertised Web pages that were categorized as pharmaceutical, replica, or software Web sites. These are all the pages in the middle panel of Figure 1, not merely those matched by regular expressions. Most of these Web pages—the ones matched by regular expressions, and the ones detected and manually validated in the previous section—belong to the known family of forty-four affiliate programs. The others we assign to a catch-all label of “other,” thereby yielding a 45-way problem in classification. With ground-truth labels for these Web pages, we can now reliably measure the performance of NN classification on this enlarged data set.

We began with experiments on the Web pages crawled during the first three-month period. As before, but now with the inclusion of an “other” label, we measured the accuracy of NN classification on ten 70/30 splits of the data. The average accuracy for 45-way classification was still remarkably high at 99.95%. These results show that NN classification can distinguish the storefronts of different affiliate programs even when many of them are collectively assigned an undifferentiated label of “other.”

Next we investigated how well NN classification holds up over time. For this we performed NN classification using Web pages from the first three-month crawl as training examples and pages from the second three-month crawl as test examples. The accuracy remained high at 87.7%, a level that is still quite operationally useful for forty-five way classification of affiliate programs. But the drop of performance also indicates that a better system should adapt to affiliate programs as they evolve in time. In particular, as we detect and label new storefronts, it would behoove NN classification to augment the set of training examples and extract a larger vocabulary of HTML features.

Table 5 lists precision and recall on this task for some affiliate programs. The most startling result is the 0.25% recall for ED Express; in fact, almost 70% of the incorrect predictions involved misclassified storefronts from this affiliate program. We manually examined these storefronts and concluded that ED Express switched to a new template engine for generating storefronts. As a result the storefronts for ED Express from the second three-month crawl were rather different than those from the first. This is a valuable insight into how affiliate programs⁴ operate, and we return to this case in our final set of experiments.

⁴As an aside, we learned later that these misclassified storefronts belonged to Pharmacy Express, and that Pharmacy Express and ED Express are in fact part of the same aggregate program called Mailien. Some of these errors can therefore be attributed to noisy and/or imperfect labeling of the training examples.

Affiliate program	1st 3 months				2nd 3 months			
	labeled	detected	precision	recall	labeled	detected	precision	recall
EvaPharmacy	58,215	434	99.98	100.00	13,529	98	99.29	99.29
Pharmacy Express	44,017	42	100.00	100.00	2,454	11	99.59	99.55
RX-Promotion	37,245	1,467	100.00	100.00	3,317	198	94.31	93.88
Online Pharmacy	16,546	758	100.00	100.00	3,457	271	92.86	92.86
GlavMed	6,616	263	99.95	100.00	176	49	77.73	76.00
EuroSoft	2,215	15	100.00	100.00	1,681	54	97.00	96.89
Ultimate Replica	79	10	100.00	100.00	121	5	96.03	96.03
ED Express	77	5	96.30	100.00	3,653	0	90.00	0.25
Club-first	14	114	100.00	100.00	6	0	100.00	100.00

Table 5: Data sizes and performance for select affiliate programs. The first five are the largest classes. EuroSoft and Ultimate Replica are representative software and replica programs. ED Express suffers drastic misprediction in the 2nd 3 months since it changed template engines. Club-first is the affiliate program whose size experiences the largest percent gain as a result of our classifier. In total, we profiled the storefronts of 29 pharmaceutical, 10 replica, and 5 software affiliate programs.

4.4 Learning with Few Labels

We next consider how many storefronts must be manually labeled before an automated system can dependably take over. In these experiments we vary the amount of labeled storefronts that are available as training examples for NN classification. In previous sections we studied the regime where there were many labeled storefronts per identified affiliate program. Here we consider the opposite regime where each affiliate program may, for instance, have no more than one labeled storefront.

We make two preliminary observations. First, in these experiments we not only have many fewer training examples; we also have many fewer features because the feature extraction is necessarily limited to counts of HTML words that appear in the training examples. Thus, to be clear, if we have just one labeled storefront per affiliate program, then we can only extract features from those forty-four labeled storefronts. In this regime we do not exclude rare words (though we still exclude stopwords).

Second, in these experiments we handle the “other” label for unidentified Web pages as a special case. At the outset of a Web crawl, we imagine that practitioners are likely to encounter numerous instances of Web pages that belong to unidentified affiliate programs or that do not represent storefronts at all. Thus in all the experiments of this section we assume that 100 Web pages are available from the undifferentiated “other” class that does not map to a previously identified affiliate program.

Figure 4 plots the balanced accuracy of NN classification per affiliate program (averaged over five random training/test splits) versus the number of labeled storefronts. Note that throughout this regime, the median accuracy holds at 100%, meaning that at least half the affiliate programs have none of their storefronts misclassified. Overall the results show that even with few labeled storefronts, NN classification is accurate enough to be operationally useful. With only one labeled storefront per affiliate program, the average 45-way classification accuracy remains nearly 75%; with two storefronts, it jumps to 85%, and with four, eight, and sixteen, it climbs respectively to 93%, 97%, and 98%.

Table 6 shows four correctly classified storefronts that render quite differently than their nearest (labeled) neighbors in this regime. Visually, it is not immediately obvious that

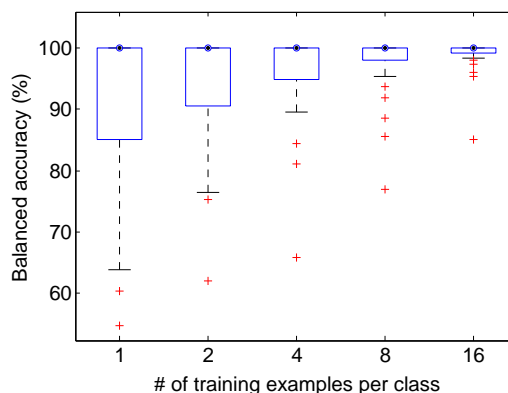


Figure 4: Boxplot showing balanced accuracy for all 45 classes as a function of training size. The top and bottom edges of the box are the 75th and 25th percentiles, respectively. The whiskers show the lowest points not considered outliers, and the outliers are individual plus marks.

these pairs of storefronts belong to the same affiliate program; upon inspection of their HTML source, however, it becomes apparent that their DOM trees share a similar underlying structure.

We found that the affiliate program called RX-Partners suffered the worst performance throughout this regime. Upon closer examination of these errors, however, we observed that the regular expressions for RX-Partners often matched Web pages that were not storefronts at all. This mislabeling created many confusions in NN classification between RX-Partners and the undifferentiated “other” label.

These results suggest an iterative strategy for practitioners that balances the demands for high levels of both accuracy and automation. To begin, practitioners may seed NN classification with just one or two labeled storefronts per affiliate program. Then, using the preliminary predictions from NN classification in this regime, they may attempt to grow the number of labeled storefronts (and extractable features) in alternating rounds of manual validation and retraining. This “bootstrapping” strategy seems the most effective way to develop a practical, working system.

Training Example	Correct Classifications

Table 6: Examples of correctly classified storefronts when there is only one training example per affiliate program. The affiliate programs shown here are 33drugs and RX-Promotion.

4.5 Clustering

Finally, we investigate the capabilities of fully *unsupervised* methods for distinguishing storefronts from different affiliate programs. We do this first for the full set of pharmaceutical, replica, and software Web pages, in the hope that these storefronts might cluster in an organic way by affiliate program. Next we do this just for the set of undifferentiated storefronts marked as “other”, in the hope that we might identify new affiliate programs within this set.

4.5.1 Clustering of Storefronts by Affiliate Program

A preliminary clustering of online storefronts can be useful as a first step before the manual labeling of their affiliate programs. This was the approach taken in [9], and it is the operational setting that we emulate here.

We ran the *k*-means algorithm, one of the simplest unsupervised methods for clustering, on all the Web pages tagged as storefronts for pharmaceuticals, replica luxury goods, and software. These pages included those in the “other” class not identified with a known affiliate program. In general, the cluster labels are highly predictive of the affiliate program identities: of the 42 clusters, 22 are entirely composed of storefronts from a single affiliate program, and 32 very nearly have this property, with less than 1% of their storefronts from a competing program.

A quantitative measure of this correlation is given by the uncertainty coefficient [16], which measures the reduction of uncertainty in a Web page’s class label c (e.g., affiliate program) given its cluster label d . Let C and D denote random variables for these labels. The uncertainty coefficient is computed as:

$$U(C|D) = 1 - \frac{H(C|D)}{H(C)} = 1 - \frac{\sum_{c,d} p(c,d) \log p(c|d)}{\sum_c p(c) \log p(c)},$$

where $H(C)$ is the entropy of C and $H(C|D)$ is the conditional entropy of C given D . The joint probability $p(c,d)$ is obtained by counting the number of Web pages with class label c and cluster label d , then dividing by the total number of Web pages. Note that the uncertainty coefficient is bounded between 0 and 1, where 0 indicates that the two

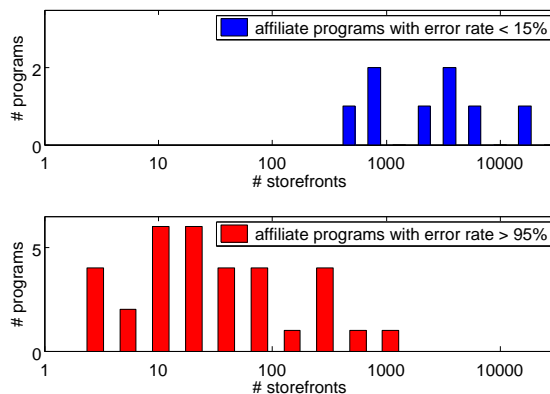


Figure 5: Number of affiliate programs of different sizes with few versus many clustering errors; see text for details. In general the larger programs have low error rates (top), while the smaller programs have very high error rates (bottom).

variables are completely uncorrelated, and 1 indicates that one variable is completely determined by the other. From the results of *k*-means clustering we obtain an uncertainty coefficient of 0.933. This coefficient shows that very few clusters contain Web pages from more than one class.

Another measure of cluster “purity” is obtained by computing the overall percentage of storefronts assigned to a cluster that contains more storefronts from a different affiliate program. This percentage is 3.27%, quite a low confusion rate for a problem in 45-way classification.

It must be emphasized, however, that while *k*-means clustering generally separates Web pages from different affiliate programs, it does so by modeling the larger affiliate programs at the expense of the smaller ones. In particular, only 11 out of 44 affiliate programs are represented as the dominant class of one or more clusters; the other 33 programs are swallowed up by the larger ones. We say a clustering “error” occurs when a storefront is mapped to a cluster that contains more storefronts from one or more different affiliate programs. The top panel of Figure 5 shows that all the largest affiliate programs (with over 2000 storefronts) have error rates less than 15%; conversely, the bottom panel shows that all the smallest affiliate programs (with fewer than 100 storefronts) have error rates greater than 95%.

We conclude that unsupervised methods have high accuracy but poor coverage of affiliate programs. On one hand, by distinguishing storefronts in the largest affiliate programs (which account for over 96% of the Web pages), these methods may greatly reduce the set of pages that require manual labeling. On the other hand, for complete coverage of affiliate programs, it seems imperative to label at least one storefront per program. Once these labels are provided, moreover, we believe that the best approach will be a supervised model, such as NN classification, that exploits the information in these labels.

4.5.2 Detecting New and Evolving Programs

Lastly we investigate the potential of clustering methods to detect new and evolving affiliate programs. In our first experiment, we used the *k*-means algorithm to cluster the storefronts labeled as “other” from the first three-month crawl. We then ranked the storefronts in each clus-

ter by their distance to the cluster centroid and manually examined the top-ranked (i.e., most “central”) ones with our Web visualization tool. Using this approach, we discovered a new affiliate program called RxCashCow; the 430 top-ranked storefronts in one cluster belonged to this program, and only a small fraction of the top-ranked 1,100 did not.

Finally we revisit the interesting case of ED Express. Recall that NN classification failed to identify these storefronts during the second three-month crawl because the affiliate program switched to a new template engine. Our last experiment used the k -means algorithm to cluster all the storefronts from the second three-month crawl. Our hope was that the algorithm might separate out the evolved storefronts of ED Express, and indeed we found a cluster whose 3,580 top-ranked pages were exactly those generated by the new template engine.

5. CONCLUSION

We have described an automated approach for profiling the online storefronts of counterfeit merchandise. We evaluated our approach on hundreds of thousands of storefronts gathered from a Web crawl of spam-advertised URLs. Our methods aim to identify the affiliate programs behind these storefronts, an important first step in tracking and targeting illicit e-commerce. Our first experiments showed that these affiliate programs could be identified, with high accuracy, from simple NN classification on bag-of-words features. With an operational setting in mind, we also showed that the feature extraction and NN classification only required a small seed of labeled storefronts to be highly effective. Our final experiments investigated the potential of unsupervised methods for clustering storefronts by affiliate program. Here we found that k -means clustering could be used to discover affiliate programs with large spam footprints, but that smaller affiliate programs (i.e., those with fewer storefronts) were not cleanly identified. Overall, our work is an encouraging case study in the application of machine learning to an important class of security problems on the Web.

Acknowledgments

We thank Andreas Pitsillidis and Tristan Halvorson for making this data set accessible, as well as Cindy Moore for setting up our computing resources. This work was supported in part by National Science Foundation grant CNS-1237264, by the Office of Naval Research MURI grant N00014-09-1-1081, and by generous research, operational and/or in-kind support from Google, Microsoft, Yahoo, and the UCSD Center for Networked Systems (CNS).

6. REFERENCES

- [1] S. N. Bannur, L. K. Saul, and S. Savage. Judging a site by its content: learning the textual, structural, and visual features of malicious Web Pages. In *Proc. of the 4th ACM Workshop on Security and Artificial Intelligence*, 2011.
- [2] T. Cover and P. Hart. Nearest neighbor pattern classification. In *IEEE Transactions in Information Theory, IT-13*, pages 21–27, 1967.
- [3] J. Drew and T. Moore. Automatic identification of replicated criminal websites using combined clustering. In *International Workshop on Cyber Crime (IWCC), IEEE Security and Privacy Workshops*. IEEE, 2014.
- [4] T. Halvorson, K. Levchenko, S. Savage, and G. M. Voelker. XXXtortion? Inferring Registration Intent in the .XXX TLD. In *Proceedings of the International World Wide Web Conference (WWW)*, Apr. 2014.
- [5] J. P. John, F. Yu, Y. Xie, A. K. and M. Abadi. deSEO: Combating Search-Result Poisoning. In *Proceedings of the 20th USENIX Security Symposium*, August 2011.
- [6] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.
- [7] C. Kanich, N. Weaver, D. McCoy, T. Halvorson, C. Kreibich, K. Levchenko, V. Paxson, G. M. Voelker, and S. Savage. Show Me the Money: Characterizing Spam-advertised Revenue. In *Proc. of the USENIX Security Symposium*, San Francisco, CA, Aug. 2011.
- [8] R. Layton, P. Watters, and R. Dazeley. Automatically determining phishing campaigns using the uscap methodology. In *eCrime Researchers Summit (eCrime), 2010*, pages 1–8, Oct 2010.
- [9] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G. M. Voelker, and S. Savage. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *Proceedings of the IEEE Symposium and Security and Privacy*, Oakland, CA, May 2011.
- [10] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang. Knowing Your Enemy: Understanding and Detecting Malicious Web Advertising. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, October 2012.
- [11] J.-L. Lin. Detection of cloaked web spam by using tag-based methods. *Expert Syst. Appl.*, 36(4):7493–7499, May 2009.
- [12] C. Ludl, S. McAllister, E. Kirida, and C. Kruegel. On the Effectiveness of Techniques to Detect Phishing Sites. In *Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2007.
- [13] D. McCoy, H. Dharmdasani, C. Kreibich, G. M. Voelker, and S. Savage. Priceless: The Role of Payments in Abuse-advertised Goods. In *Proceedings of the ACM Conference on Computer and Communications Security*, Raleigh, NC, Oct. 2012.
- [14] D. McCoy, A. Pitsillidis, G. Jordan, N. Weaver, C. Kreibich, B. Krebs, G. M. Voelker, S. Savage, and K. Levchenko. PharmaLeaks: Understanding the Business of Online Pharmaceutical Affiliate Programs. In *Proceedings of the USENIX Security Symposium*, Aug. 2012.
- [15] T. Moore and B. Edelman. Measuring the Perpetrators and Funders of Typosquatting. In *Proceedings of the 14th International Conference on Financial Cryptography and Data Security*, 2010.
- [16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 2007.
- [17] N. Provos, P. Mavrommatis, M. Rajab, and F. Monrose. All Your iFRAMEs Point to Us. In *Proceedings of the USENIX Security Symposium*, 2008.
- [18] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and Evaluation of a Real-Time URL Spam Filtering Service. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2011.
- [19] T. Urvoy, E. Chauveau, P. Filoche, and T. Lavergne. Tracking Web Spam with HTML Style Similarities. *ACM Trans. Web*, 2(1):3:1–3:28, Mar. 2008.
- [20] D. Wang, S. Savage, and G. M. Voelker. Juice: A Longitudinal Study of an SEO Campaign. In *Proc. of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2013.
- [21] Y. Zhang, J. Hong, and L. Cranor. CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. In *Proceedings of the International World Wide Web Conference*, pages 639–648, 2007.