Department of Computing
Imperial College London

BEng Individual Project

Phishing Website Identification through Visual Clustering

Author:
Nic Prettejohn

Supervisor:
Dr. Ben Glocker

Second Marker:
Dr. Bernhard Kainz

June, 2016

Abstract

Phishing is a form of online fraud, where criminals exploit the implicit trust placed in an organisation's website branding to acquire sensitive information from the organisation's customers. Major browser vendors race through hundreds of thousands of spam emails a day, assessing each URL to find Phishing websites to add to their blacklists. The quantity of suspect sites is so large that automated classification is essential.

Creators of Phishing sites are aware of the popular text-based classifiers and implement a variety of countermeasures to disguise the site from software while remaining recognisable to humans. Consequently researchers are exploring using computer vision to better recognise Phishing sites.

We present **Distinctive Region Classifier** - a novel approach to identify the targets of potential Phishing attacks from screenshots. It is a multi-class classifier that combines computer vision techniques with density-based clustering to identify a page by elements that are indicative of a single class in our training set. Our system performs comparably with published approaches in this space.

Additionally, we present resilient distributed implementation of a visual classifier and blacklist. We find our cluster accelerates classification to a rate equal to commercial anti-Phishing measures.

Acknowledgements

I would like to thank my supervisor, Dr. Ben Glocker, for agreeing to supervise this project, and for sharing his insight into computer vision. Secondly I would like to thank Netcraft for sharing some of their Phishing data with me. It was indispensable throughout the project.

To Mum and Dad, thank you for your support, love and encouragement. And finally, thank you Evie, Tony, David, and Luke. I've learnt a great deal over the last three years, and I couldn't have done it without you.

Contents

Introduction

> *Why does phishing work? Basically because con artists are really good at persuading people to do really dumb things.*
>
> – Richard Clayton, Cambridge Computer Laboratory [1]

A career as a cyber-criminal has never been so attractive. The payouts are high, convictions are difficult, and the required technical expertise to get started is surprisingly low. Cyber-criminals exploit the fully globalised nature of the Internet and enjoy the protection offered by a fragmented global justice system and poor extradition policies.

In Iain Softley's 1995 cult classic *Hackers*, Jonny Lee Miller and his co-stars portray hyper-intelligent cyber-vigilantes: using their expert skills and knowledge to compromise corporate firewalls as if they were arcade games. Although *Hackers* takes the hacker subculture to its extreme, the myth that cyber-criminals are extraordinary in technical ability perpetuates to this day. The unfortunate reality is the tools of the trade are readily available to anyone with laptop and an Internet connection. The tools are easy-to-use and sophisticated — the criminal's success is almost invariant to their technical ability.

One of the easiest ways to start one's career in cyber-crime is *Phishing*. Phishing is a form of online fraud, where criminals exploit the implicit trust placed in an organisation's branding to acquire sensitive information from the organisation's customers. This sensitive information can be used for identity theft, bank fraud, or as part of a larger campaign to compromise the victim, their employer, or their national government.

The World Wide Web is a popular platform for Phishing. A *Phishing website* is

a website that masquerades as a trusted brand, but is actually controlled by a criminal. All data sent to the website, such as user names, passwords, and credit card numbers, can then be utilised by the criminal or sold on the black market.

Cloning a website is difficult, but anyone can download a *Phishing Kit* that imitates a particular organisation. The kit is developed by a technical individual and is sold to would-be Phishers. The kit normally contains a clone of the organisation's login page, and is configured to email the Phisher the credentials obtained. Hosting for the website can be paid for anonymously through Bitcoin[1], or stolen credit card numbers obtained on the black market; or through a variety of software packages that discover vulnerable websites and assist the criminal in uploading their Phishing site to the server[2]. Links to Phishing sites can then be distributed via email, social media, or web advertising.

The Internet community has responded to the threat users face from Phishing. All major desktop and mobile web browsers (Microsoft Internet Explorer, Microsoft Edge, Google Chrome, Mozilla Firefox, Apple Safari, Opera) use blacklists to display warnings to their users when they attempt to access a known Phishing site.

It is fundamental to the effectiveness of Phishing blacklists that they are updated regularly. Phishing sites are created at a rate of over 70,000 websites a month [3]. Security researchers find Phishing websites through email honeypots[2] and spam filters. This creates a classification problem, where each website linked within a spam email must be assessed to discern whether it is indeed a Phishing website and should be blocked.

The scale of Phishing activity has lead to the automation of classification of Phish-

[1]Bitcoin is a digital currency that offers anonymity to its users, a popular feature for criminals. For more information, please see `http://bitcoin.org`

[2]A honeypot, in the context of spam collection, is a computer or mail server that deliberately tries to the collect as much spam as possible for analysis.

ing websites. Due to the widespread use of the web browsers (Google Chrome has over a billion users[4]) the impact of the blacklists is massive, and, transitively, so is the impact of the classification process. High precision[3] is essential.

1.1 Motivation

Users depend on the appearance of a website as an indication of its authenticity [5]. While factors such as SSL certificates and a sensible URL field can provide additional signals to inform a careful user, if the page didn't *look* like a trusted brand the user wouldn't be fooled into submitting their sensitive information.

We can make this an image classification problem. For each candidate page, take a screenshot of the page and compare its similarity to screenshots of known legitimate sites. If the candidate looks similar to one of the legitimate pages, and the URL of the candidate does not match, we can consider it Phishing.

A webpage is made up of many components - images, tables, buttons and input fields - but not all of them are equally indicative of the organisation responsible for the website. Like a human, an automated classifier should evaluate how distinctive a component is to the organisation's own pages. For example, a logo is very distinctive as it won't appear on any other legitimate sites, whereas a login button is not.

We could carefully construct a training set of a number of key components, such as extracting the logos for each of the sites we want to protect, and then matching these. However, our hypothesis is we should be able to generalise this. Additionally, it is possible that an automated approach would find recognition that a human might not think to extract - such as icons and buttons.

[3] In the statistical sense

The quantity of URLs that require assessment from spam emails is substantial. Computer vision algorithms are substantially more computationally intensive than text based approaches, limiting the throughput that can be achieved on a single computer. For the security community to make use of these intelligent approaches, the classification system must be distributed across many computers.

1.2 Objectives

The goal of this project is to develop a multi-class classifier that can generalise recognisable elements of a legitimate website (e.g. the company's logo) and use these elements to identify the target of Phishing websites. We aim to create a classifier which can perform at 100% precision and maximise its recall.

Additionally, we have a secondary goal of developing an end-to-end distributed classification system, capable of identifying the target of the screenshots submitted to it through an Application Programming Interface (API), and, if the webpage is indeed Phishing, adding it to our own blacklist.

1.3 Contributions

We present a novel approach to Phishing classification: a **Distinctive Region Classifier**. DRC considers dense clusters of high-contrast keypoints as *regions* of a Web page. DRC is trained with a small set of screenshots of labelled legitimate websites and Phishing sites. It identifies the target of a candidate Phishing site by comparing regions extracted from a screenshot against its training set and establishes an estimate for the region's robustness as an indicator for a single class. If there are enough distinctive regions matching a single brand, the candidate is considered a match.

We also examine a variety of the published work in this space. We discuss their evaluation techniques, performance, and reason which attributes would make a good classifier.

Additionally, we present a scalable distributed classification cluster utilising Apache Storm - a distributed streaming processing system. We evaluate its performance running Distinctive Region Classifier running on 16 instances at Amazon Elastic Compute Cloud. We achieve throughput of up to two classifications a second.

Finally we evaluate the effect of a including Phishing sites in addition to legitimate pages in a brand's ground truth, and show it improves both precision and recall substantially.

Background

In this chapter we introduce techniques used by the security community to protect users, and techniques used by criminals to avoid Phishing countermeasures. We provide background on tools we evaluated for use in our classifier. We review a number of metrics used in reviewing the performance of academic Phishing classifiers. Finally we discuss related work which uses visual similarity to identify Phishing sites.

2.1 Phishing

A *Phishing site*, *Phishing page* or - colloquially - *Phish* is a website that seeks to mislead a victim into giving sensitive information to the site's controller - a *Phisher*. Phishing websites are designed to be similar to their *target* - a legitimate website, often a financial institution or social network, where the credentials have value to the Phisher either personally on the black market.

A victim of Phishing typically received the link to the website in an email. A popular template for Phishing emails is to imitate an official email from a popular institution, and claim that something has gone wrong with the user's account. The email will claim the user must log in soon or verify their details at a link included in the email.

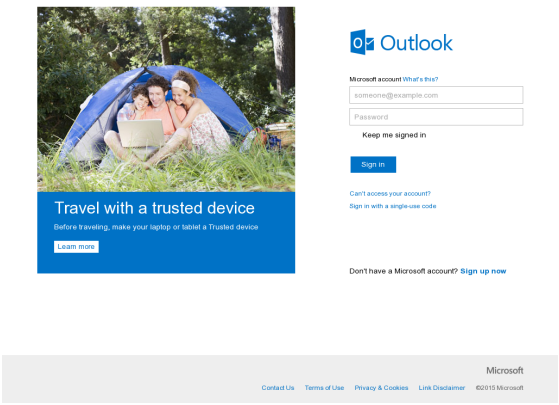


Figure 2.1: A Phishing website *targeting* Microsoft.

2.1.1 Phishing Countermeasures

Browser Plugins and Extensions

A number of clients have been developed to protect users from navigating to fraudulent websites on their computers. In mid 2000s, Anti-Phishing toolbars were incredibly popular. Cranor et al.[6] evaluate the performance of 10 different toolbars. Most made use of a small number of heuristics for detecting Phish in addition to a blacklist of known Phishing sites. Heuristics might include checking whether the page is served over a non-standard port or whether the page has an SSL certificate.

However, in 2016, very few people use anti-Phishing toolbars as the functionality has been integrated into the browsers. Delaying page loads to analyse the content first and cluttering the interface are not popular today. Additionally, none of the most popular mobile browsers support extensions - putting the responsibility into the hands of the browser developers.

Blacklists

Blacklists have been found to be a far more precise mechanism as candidate URLs can undergo much more thorough analysis than can be completed before a page load. Most popular blacklists update every five minutes, so the delay is not significant enough to substantially impact the blacklist's effectiveness.

As blacklists are now the *de-facto* standard for Phishing protection, we aim to develop an approach that would fit as a reliable feed into a blacklist, as opposed to developing a native application or browser extension.

Phishing Takedown

Microsoft estimated the annual cost of Phishing to the world economy at $5 billion[7]. The huge cost to society is most felt by financial institutions - in many countries, including the UK and the US, the banks must reimburse their customers in the event of fraud except in cases where the customer has been unduly careless.

Consequently, a growing number of institutions pay to have Phishing websites defrauding their customers taken down from the Internet. A *Phishing takedown* is ususally very simple to perform. The majority of web site owners hosting Phishing sites do not know the page is on their site [2] as it was placed there by a criminal who compromised the server. By making contact with the web site owner or hosting company over email or telephone, institutions can have the Phishing page off the Internet within hours.

Nevertheless, these companies are not altruistic. Institutions are only interested in paying for takedowns against attacks that target their own brand. Consequently, they require a brand-aware filter to highlight the attacks targeting their institution.

2.1.2 Phishing Classification

Candidate URL Collection

There are a variety of measures used within the security community to find possible Phishing sites. One of the most popular methods is extracting URLs from spam emails. Whittaker et al.[8] developed the Phishing classifier that maintains the Google safe-search blacklist and are in the enviable position of being able to extract URLs from every spam email sent to a GMail account.

Netcraft[9] relies on a combination of reports from its community[10], and additionally purchases feeds of spam URLs and emails from individuals and vendors maintaining email spam *honeypots*.

Binary Classification Vs Multi-class Classification

Anti-Phishing research has attempted to classify Phishing sites with both binary classification and multi-class classification techniques.

Binary Classification Produce a *single* model that can assess a candidate and return 0 if the site is benign, or 1 if the site is phishing.

Multi-class Classification Produce a model (or collection of models) that can assess a candidate's similarity to a brand and cluster it. One can arrive at a binary classification once clustered by verifying whether the site is running on the matched institutions domain or webservers.

Binary classification approaches often extract features that are associated with Phishing sites to train a machine learning classifier. Whittaker et al. [8] utilise a

page's PageRank[1], the number of nested sub-domains in the URL[2], whether the page has a password field and more.

Binary classification limits the response we can make once we have found a Phishing site. If we wish to be able to provide a filter for organisations who perform Phishing takedowns, then we must identify the target of the Phishing site to know which organisation to inform about the attack.

Text Based Classification

The majority of methods employed in the anti-Phishing community extract features from the source code of the website. Whether hashing the source code of the page and the files linked to it[11] or training machine learning classifiers on text features [12], these approaches are susceptible to a number of defences from the the Phisher.

2.1.3 Countermeasures to Anti-Phishing Techniques

Since automated Phishing classification was introduced, Phishers have responded creatively to disguise the websites from classifiers, while still succeeding at imitating their target. There exists an arms race between the Phishing and the Anti-Phishing communities.

To avoid detection from classifiers looking for key words, Phishers will replace key content with images[13]. The technique can be taken to the extreme as in Figure 2.2 where the page background is a screenshot of the authentic page and the content is only the absolute minimum to position text boxes for the victim's input.

[1]Google's proprietary score of how influential a page can be in Google Search

[2]Phishing sites can register domains like `securitybank.com` and will host a site at `www.paypal.com.online.securitybank.com` to mislead users

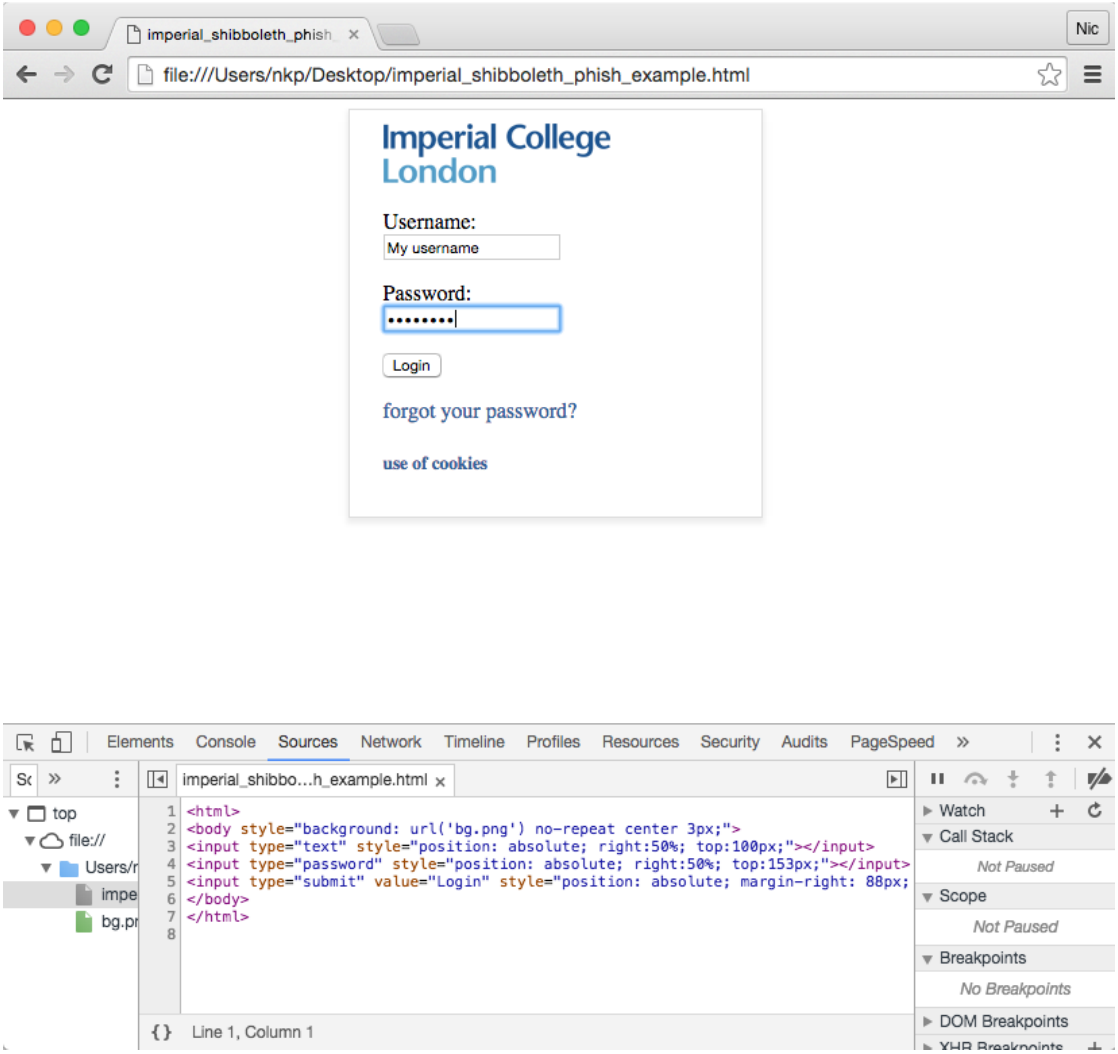
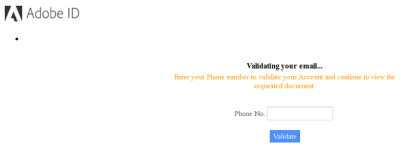


Figure 2.2: An example we created using a screenshot of the Imperial Shibboleth login page as a background image. There is no distinguishing HTML, only the minimum required to layout the text boxes for the victim's details.

Phishers can also obfuscate text from automated classifiers, but maintain an authentic appearance, by replacing some characters with *homoglyphs*. A homoglyph is a character that looks very like another. A simple example would be the character for *zero* and the Latin alphabet letter *O*. However, instead of well-known substitutions, Phishers will make use of alternative alphabets such as Cyrillic to find alternative characters for Latin letters, punctuation, and even whitespace[14].



(a) This Facebook Phishing site requires the user's details to avoid account deletion for "security reasons".

(b) This Adobe Phishing site requires the user's phone number to view a document.

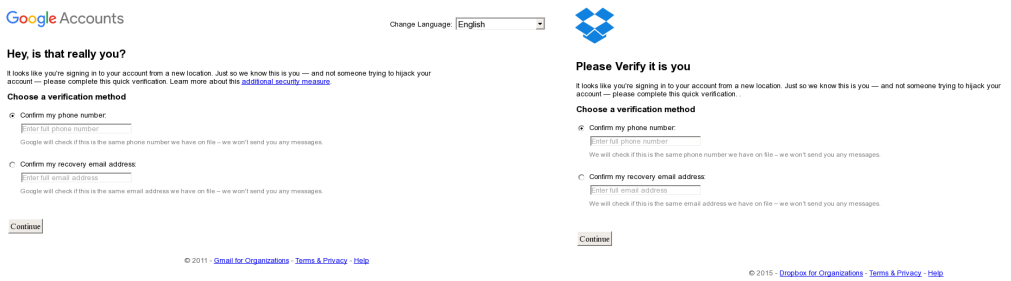
Figure 2.3: Both Phishing pages imitate the branding of their targets, but neither of these pages appear on the legitimate sites

Imitate the brand, not the page

Afroz and Greenstadt [15] address that some Phishers do not attempt to imitate a page that exists on the legitimate site. By cloning components associated with the targeted brand, such as logos and colour scheme, they can achieve a site that is resilient to classifiers that only look for similarity to legitimate pages. A popular choice is a false security alert - such as "confirm your security details" or "your account is at risk of deletion" (See Figure 2.3). This is because potential victims believe it is possible for such pages to exist on the legitimate site, and they have just never seen it before - reasoning away differences in styling.

In Figure 2.4 we present two attacks that imitate their target claiming to have found "irregular activity" on the potential victims account. These pages target two different companies, judging from their logos, but are otherwise identical.

Afroz and Greenstadt found this technique was responsible for 21.05% of the Phishing sites in their dataset from PhishTank. They hypothesised one could improve the classifier by including previously seen examples as in their training set. We



(a) A page seeks the user's details to verify they are not a fraudster.

(b) A similar page, but targeting Dropbox, also seeks to verify the user is not a fraudster.

Figure 2.4: The two pages are possibly from the same Phish Kit, but have been modified to target different brands.

explore the effectiveness of this technique in our experiments and evaluation.

2.2 Computer Vision

Computer Vision is a field of Computer Science aiming to develop algorithms and techniques to enable software to understand images and videos. Typically this includes: distinguishing and recognising objects in scenes; image enhancement; object tracking in video; and pattern recognition.

2.2.1 Template Matching

Template Matching is a technique to find a training image within a larger candidate image. The training image is "dragged" across the larger candidate, comparing all the pixels in the training image to the pixels at the same offset in the candidate image. This is very fast but suffers from being sensitive to small distortions in the candidate image.

Subtle differences can be introduced in the screenshots by errors incurred when the Phishing site was cloned from the original page. This can result in components like input boxes, buttons and logos being translated or scaled in the candidate image. The effect is two images that are very similar to a human, or at least still contain similar branding, can be interpreted as completely different by the technique.

2.2.2 Feature Extraction

Researchers employ a Computer Vision technique called Feature Extraction to sample an image for distinctive elements, such as corners and blobs, and match these samples in techniques that are invariant to luminance, rotation, scale or affine shift. It is made up of two stages: Keypoint detection, which finds the regions of images that the techniques heuristic suggests are recognisable even after significant distortion - such as corners or blobs of high contrast; and extraction, which computes a feature vector called a *descriptor* to represent the feature. The number of dimensions in the descriptor depends on the technique and its parameters.

The keypoints found and what information about the local area the descriptor should contain depends on the feature extraction technique being employed. Two popular techniques are SIFT and SURF.

2.2.3 SIFT

SIFT[16] (Scale Invariant Feature Transform) is a Computer Vision technique developed by David Lowe to extract features that are intolerant to scale (Resizing) and rotation. Additionally, SIFT feature descriptors are partially invariant to illumination changes and geometric distortion.

SIFT finds keypoints through an iterative process. The algorithm first performs Difference of Gaussian filtering - a technique that increases the visibility of edges

and details - on the image at multiple scales. The pixels that have maximum contrast to their immediate neighbours at all scales are selected as keypoint candidates. The DoG process can yield points that are sensitive to noise, so the candidates are further filtered to remove low contrast keypoints and noisy keypoints along edges. SIFT additionally obtains an orientation for each keypoint by evaluating the gradient direction of the pixels around it.

SIFT descriptors have 128 features and are made up of histograms of the orientation of the pixels neighbouring a feature, providing a descriptor that is invariant to location, scale and rotation.

2.2.4 SURF

Speeded Up Robust Features (SURF) is a variation on SIFT that offers improved speed at similar performance[17]. It achieves this by performing filtering using a square shaped filter - in place of the Gaussian filter used by SIFT - when finding high contrast pixels. SURF then makes use of a blob detector to find points of interest. The operations are all on integers which speeds the detection process over SIFT at the cost of accuracy at extremes.

SURF descriptors consist of 64 features. The features are calculated from the intensity distribution of the local area around a keypoint, with additional orientation detection to provide rotation invariance. There is a variant named Upright-SURF (U-SURF) that forgoes rotation invariance in favour of extraction speed. U-SURF can be four times faster than SIFT[17].

2.2.5 Feature Matching

Given a collection of feature descriptors for a candidate image, and a collection of feature descriptors for another image, the next step is to match the keypoints.

Each of the descriptors in the candidate image is compared to every descriptor in the training image. A descriptor in the candidate image *matches* a descriptor in the training image for which the distance between the two descriptors is minimised. For both SIFT and SURF, the distance is calculated as the Euclidean distance between the two feature vectors. A short euclidean distance indicates the neighbouring pixels around the two keypoints are similar.

Feature matching is exhaustive - every feature in the candidate image is matched to a feature in the training image - which can introduce some matches where the distance between the two descriptors is relatively large. Consequently, pruning poorly matching keypoints can improve the performance of algorithms that utilise the matches.

2.3 Clustering Analysis

Cluster Analysis is the action of grouping points or objects into clusters by comparing some or all of their attributes.

2.3.1 K-Means Clustering

K-Means clustering is a technique which groups vectors into K clusters, such that the squared distances from each vector to the center of the cluster (centroid) is minimised. Finding the optimal value is NP-hard, so approximation is required. A popular approach is Lloyd's Algorithm[18].

The algorithm is as follows: Given K initial centroids (Normally chosen through randomisation), assign every point to its nearest centroid. Now iteratively calculate new means for the centre of a cluster and change the centroids to the new value. The cluster is finished when the centroid does not move more than ϵ in an iteration.

K-Means can suffer from requiring the number of clusters to be specified in advance. Additionally, clusters are often similar sizes spatially due to the algorithm minimising the width within a cluster.

2.3.2 DBSCAN

DBSCAN - Density-Based Spatial Clustering of Applications with Noise - is an approach that groups vectors into clusters where, unlike K-Means clustering, DBSCAN will create as many clusters as needed.

Clusters are formed by selecting a point at random and examining its radius ϵ. Any points within a radius of ϵ will be added to the cluster. In turn, any points that are within ϵ of the new points will be added to the cluster. Restricting all nodes to be within ϵ of another node maintains a minimum constant density within a region. If a cluster cannot find more neighbours and does not have at least *minimum samples* points, it will be labelled as noise.

2.4 Evaluating Performance of Phishing Classifiers

To evaluate clustering classifiers, researchers typically train with a set of legitimate web pages. Their testing set will consist of Phishing sites targeting these pages, in addition to a selection of other legitimate sites. A threshold or threshold-vector will be defined, and sites that are above the threshold will be considered Phish and those below marked as not. Researchers can then make use of a number of metrics to compare the performance of their classifiers.

2.4.1 Sensitivity and Specificity

True Positive Rate *TPR* (Sensitivity / Recall) (Equation 2.1) and False Positive Rate *FPR* (1-Specificity) (Equation 2.2) are the most common measures of Phish classifier performance.

True Positive Correctly identifying Phishing site's target.

True Negative Correctly classifying a legitimate site as legitimate.

False Positive Incorrectly identifying the target of a Phishing site, or classifying a non-Phishing site as Phishing.

False Negative Incorrectly classifying a Phishing site as not-Phishing.

$$TPR = Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2.1}$$

$$FPR = \frac{False\ Positives}{False\ Positives + True\ Negatives} \tag{2.2}$$

Additionally, researchers compare their system's *Accuracy* (Equation 2.4) - a ratio of the number of correct classifications and all classifications made; and *Precision* (Equation 2.3) - the proportion of positive classifications that are correct.[3]

$$Precision = \frac{True\ Positives}{Positive\ Classifications} \tag{2.3}$$

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Positive\ Classifications + Negative\ Classifications} \tag{2.4}$$

[3]This serves as an answer to the question: "What is the probability that we have correctly identified a Phishing site?"

These measures aren't necessarily all equally important. Research has shown [5] [19] [20] that even a small number of false positives can lead to the user growing frustrated with the system, losing faith, and disabling it. Phish Blacklists are incorporated and kept up to date by every major web browser vendor - false positives would lead to prominent site being inaccessible to all users of that browser.

Additionally, one must bear in mind the scale at which commercial vendors operate. Whittaker et al. wrote Google's Phishing classifier considered 1,500,000 URLs in the first two weeks of August 2009[8]. Even a 1% False Positive Rate would mean tens of thousands of websites would be incorrectly blocked for the majority of the Web's users.

2.4.2 Confusion Matrix

We can assess multi-class classifiers directly by plotting the classification decisions on a confusion matrix (Table 2.1). The columns represent the prediction the classifier has made for the class of a subject, and the rows represent the actual class of the subject.

Table 2.1: A confusion matrix of four classes

	A	B	C	D
A	1	0	0	0
B	2	1	0	0
C	2	1	3	2
D	0	0	1	4

The diagonal axis represents correct classifications. A better classifier would have a higher density of classifications along this axis.

2.4.3 Macro-Average and Micro-Average

Micro and macro averages can be thought of the average performance of the classifier as a whole. Micro averages are weighted by the class distribution of the underlying data, and macro averages are computed from the total performance of the classifier.

The micro-average of recall and precision are defined in equations 2.5 and 2.6.

$$R_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FP_i} \tag{2.5}$$

$$P_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FN_i} \tag{2.6}$$

The macro-average of recall and precision are defined in equations 2.7 and 2.8.

$$R_{macro} = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i} \tag{2.7}$$

$$P_{macro} = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i} \tag{2.8}$$

Where n is the number of classes, and TP_i, FP_i , FN_i are the number of true positives, false positives, and false negatives for class i.

2.4.4 Threshold-based Decision

A multi-class Phishing classifier attempts to identify the class a subject belongs to. However, it is possible the subject is not a Phishing site and does not belong in any of the classes. Researchers arrive at a 'yes/no' decision for whether a site is

a Phishing site or not by introducing a threshold-based decision. If the similarity between the subject and its closest matching class does next exceed a threshold t, then the subject is considered benign.

2.4.5 ROC Curve and PR Curve

A Receiver Operating Characteristic (ROC) Curve is a plot (See Figure 2.5) of the true positive rates of a classifier against its false positive rates for different decision thresholds. The curve allows us to judge the tradeoffs between improving detection (True Positive Rate) at the expense of degrading Precision.

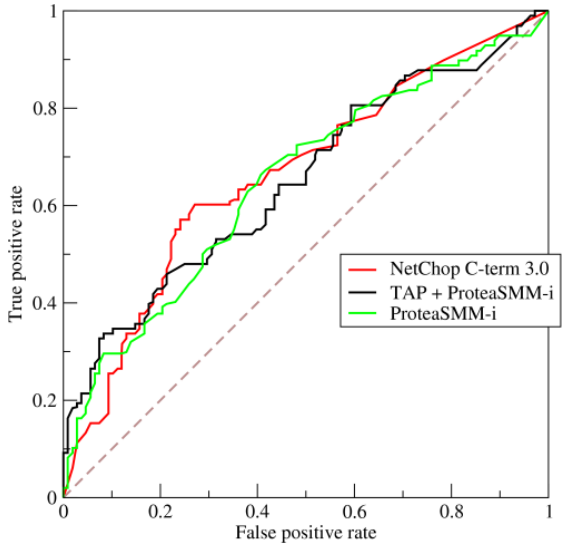


Figure 2.5: Multiple classifiers compared on a ROC Curve.
Source: By BOR at the English language Wikipedia, CC BY-SA 3.0, `"https://commons.wikimedia.org/w/index.php?curid=10714489"`

Precision-Recall (PR) Curves are an alternative to ROC Curves for classifiers operating on datasets where there is a significant skew in the class distribution[21]. A PR Curve is a plot of the Precision of a classifier against its Recall, i.e True Positive Rate, for different thresholds. In a domain that is heavily skewed towards Negatives, a significant change in number of False Positives would not change

the ROC Curve by an equivalent amount as the number of False Positives is compared to the much larger number of True Negatives. A significant change in False Positives would have a much larger effect on Precision, and consequently the graph.

2.5 Visual Similarity Clustering

2.5.1 Classification with Document Object Model

Liu et al. [22] proposed a system for visual clustering of Phishing sites. They compute visual similarity from the webpage's Document Object Model (DOM) as opposed to considering a screenshot of the page. The system compares three degrees of similarity: block similarity, layout similarity, and overall style similarity.

The system first segments both the candidate and training page into *salient blocks* [23] based on the Document Object Model. Nodes which do not have any visual impact on the page are removed, and the remaining ones are clustered according to adjacency and appearance.

Block similarity is calculated by comparing features like font, text size, and colour for text blocks; and width, height, dominant colour and file creation date for image blocks. The similarity score of two blocks is calculated as a weighted average of normalised representations of these features, with the weights set empirically. Through experimentation, the authors found that coloured features (Such as background colour and text colour) were particularly distinctive, so awarded those features higher weights. Blocks with similarities above a threshold are considered *matched* between the two pages. The blocks in the training page are matched one by one with blocks in the candidate page.

Layout similarity is calculated by comparing blocks with identical contents and

comparing their position relative to other blocks on the page using a neighbourhood relationship model[23]. Two blocks are considered matched if, in addition to block similarity, they satisfy the same position constraints (as derived from the neighbourhood relationship model) with corresponding already matched blocks. A similarity score is calculated as the ratio of matching blocks to the total number of all blocks on the page.

Overall style similarity is calculated by comparing histograms of a number of distributions, including page background colour, border size and style, font family and text size. The similarity score is calculated as the normalised correlation coefficient of these histograms.

They evaluate the system against a dataset of 8 Phishing pages, home pages for the 6 different sites targeted, and 314 additional commercial bank home pages. Instead of combining the three metrics - block similarity, layout similarity, and overall style similarity, the authors consider a pair of pages a match if any one of the metrics exceeds a threshold t.

At $t = 0.9$ Liu et al. successfully classified 7 of the 8 Phishing pages (True Positive Rate of 87.5%) and no false positives (False Positive Rate of 0.0%). At $t = 0.7$, the system correctly classifies all 8 Phishing pages, but falsely identifies 4 other pages (True Positive Rate 100.0%, False Positive Rate 1.29%).

The system relies on two key assumptions: The salient block segmentation process will return similar similar blocks for two pages that appear the same, and the block's style attributes are indicative of the block's appearance.

The first assumption is defeated both accidentally and deliberately by Phishers. If a Phisher clones the website indirectly - by developing against it visually, they will could inadvertantly introduce different block structure. Any additional content the phisher includes (adverts) would increase the number of blocks that don't have any

match and reduce the block similarity score considerably. Phishers deliberately replace content with images [13] to avoid detection by string matching algorithms which also dramatically alter the page structure.

The second assumption fails for websites that make heavy use of images. Many parts of a websites distinctive style can come from the images they use. Banners, buttons, borders, icons and more are often implemented with images by the bank's developers. Use of CSS sprites [4] corrupts any attempt at dominant colour detection. Additionally, Phishers can make use of homographs, images in place of text, and other obfuscation techniques to minimise the amount of style information in the Document Object Model.

2.5.2 Classification with Screenshots

The drawbacks in a DOM based classifier are effectively defeated by considering the page at the pixel level. By loading the page in a web browser and taking a screenshot, all the JavaScript has executed, CSS rules have applied, Java Applets, Flash movies, images and others are included in the image data considered by the classifier[5].

Earth Movers Distance

Fu et al. [24] propose a clustering system based on the Earth Mover's Distance between signatures extracted from preprocessed screenshots of the candidate site and a training site. The authors chose this method as it has been shown to have advantages in representing multifeatured signatures. The authors point to

[4] A technique where multiple icons are concatenated to a single image, and the image is carefully positioned and occluded by CSS so only a desired icon is visible. This technique is employed to reduce page load time by reducing the number of requests

[5] Assuming the browser can render the components. We examine where this isn't the case in Implementation Challenges

its success in vision problems in contour matching [25] and as a metric for image retrieval [26].

In the preprocessing step, the images are normalised to 100 * 100 pixels using Lanczo's algorithm. The authors reason there is no best normalised size and chose 100 * 100 pixels empirically - as with all of their variables. Next, the colour space is reduced by downsampling from $2^3 2$ to $(2^8/CDF)^4$ where CDF is the Colour Degrading Factor. Fu et al. use a CDF of 32, giving 4096 different colours.

An image's signature is a collection of features and weights. A feature is a tuple of a degraded colour and its centroid (as a one dimensional pixel index) in the image. The feature's weight is its colour's frequency in the page. To reduce the size of the signature, the authors only include the 20 most heavily weighted features.

A signature is shown in Equation 2.9 where dc is a degraded colour, C_{dc} is its centroid, and F_{dc} is its frequency.

$$S =<<< dc_1, C_{dc_1} >, F_{dc_1} >, ..., << dc_N, C_{dc_N>}, F_{dc_N} >> \tag{2.9}$$

Earth Mover's Distance is an algorithm based on a transportation problem to evaluate the distance between two signatures. Each signature is defined as a set of features and weights. The first signature P represents *producers* and the second signature C represents *consumers*. There is an additional third input: a distance matrix D, representing a map of the distance from any of the producers to any of the consumers where each cell $D_{i,j}$ is the normalised distance from P_i to C_i. Transporting products from P to C carries a fee proportional to distance and product weight.

The task is to move as much product from P to C as possible and minimise the total transportation fee. This is a linear programming problem and, once solved, the

final distance is the ratio of total transportation cost to total product transported.

Fu et al. compute the Earth Mover's Distance for each of their training images against a candidate image. They convert this dissimilarity score d to a similarity score s with the formula $s = 1 - \sqrt{d}$. The square root function serves to improve the distribution of distances in the range (0, 1).

The authors propose a threshold vector for classifying images. Each target has its own threshold found through classifying a training set consisting of 1,000 benign websites and the 9 known Phishing sites. The system is evaluated on 9,272 different benign websites and the same 9 known Phishing sites. Their system arrives at 88.88% True Positive Rate, and 0.13% False Positive Rate.

Intuitively a threshold vector is a significant improvement over a single scalar for the entire classifier. Some Phishing targets have greater variety in their own web pages, leading to a variety in Phishing pages targeting them. Additionally, due to the prevelance of Phishing kits, mistakes made during the cloning process are likely to occur multiple times, with a small amount of variation introduced by Phishers who extend the Phish kit to include adverts, or other variations.

A significant drawback of the authors' evaluation is their limited data set of just 9 Phishing sites, and only one of the targets has more than one Phishing sites targeting it. The authors use the same Phishing sites in training their threshold vector as in the evaluation of its performance. This leads to overfitting, as in real use the classifier won't necessarily have seen the exact image before.

Gestalt Theory and Compression

Chen et al. [27] propose a clustering approach based on Gestalt theory. Gestalt theory is a psychological theory that human perception is based on the whole object, and is not just its parts. Relating to Phishing classification, Chen et al

propose that a user's perception of a website, and consequently their recognition of the site's origin, is based more than just the individual components. The authors reason that the Gestalt process transforms the visual representation of the web page and produces a *supersignal*. The supersignal is the collapsed form of all the features on a page into a single impression.

Chen et al. reason that one can approximate a supersignal with algorithmic information theory, specifically Kolmogorov complexity [28]. Kolmogorov complexity is viewed as the ultimate compressor - producing, for any arbitrary string, the minimum description of that string, given a description language. Cilibrasi and Vitanyi [29] demonstrated that Kolmogorov complexity can be approximated by modern compression techniques and developed *Normalised Compression Distance* (NCD) for clustering data. NCD is a non-negative number representing the difference between the two inputs. It is defined in Equation 2.10 where C is defined such that $C(X)$ is the length of a compressed string X, for any arbitrary compression function.

$$NCD(A,B) = \frac{C(A+B) - min(C(A), C(B))}{max(C(A), C(B))} \tag{2.10}$$

One can intuitively reason that $NCD(x,x) \approx 0$ if one assumes, for an optimal compression algorithm, that $C(x+x) \approx C(x)$ and $C(x+x) \geq C(x)$. Using the interpretation of similarity as the inverse of distance, the more similar two images are the smaller the distance between them.

Chen et al. apply NCD to Phishing clustering by calculating the NCD between a candidate screenshot and each of the images in their training set. The authors recognise that they can only approximate a perfect compression algorithm, and consequently NCD is not commutative. Consequently, the authors define the distance between two images as the arithmetic mean of both orderings of the two

images (Equation 2.11).

$$\frac{NCD(x,y) + NCD(y,x)}{2} \tag{2.11}$$

Chen et al. evaluate their system with a data set of screenshots of home pages of 16 popular targets of Phishing attacks, and 20 screenshots of Phishing sites that target these sites for a total of 320 Phishing sites. The authors consider two one-dimensional compression libraries, Blocksort[30] and LZMA[31], justifying the use of a one-dimensional compression technique over two-dimensional techniques like JPEG by pointing to the success of one-dimensional compression in other image clustering research, and the difficulty of defining what it means to 'concatenate' two images, including their header information.

The authors formed two groups for a z-test for sample means: the first containing each of the pairings of their legitimate sites (120 unique pairs); the second comparing each legitimate site to its known Phishing pages. Their hypothesis is legitimate sites will have a lower distance to their corresponding Phishing sites than to other legitimate sites. In both cases the authors reject the null hypothesis with p ¡ 0.05 and conclude their hypothesis - that distance values in group two are significantly less than group one - is supported.

The authors evaluate their algorithm as a classifier using a threshold-based decision rule: They compare each candidate against each site in their training set. If a pair's NCD is below a threshold, the candidate is judged to be targeting the corresponding training site. The authors evaluated a number of different thresholds such that they were able to compute a ROC curve for false positive rates from 0% to 100%. At the corner of the curve - their best possible result - they achieved a true positive rate of 95.6% and a false positive rate of 0.8% for LZMA.

Intuitively, one would improve their algorithm by calculating a distinct threshold

for each target given some past classifications - there could be more variety in the styles of Phishing sites targeting Paypal than Bank of America. Once the distance has been calculated for a candidate against all images in the training set, the closest matching target and its respective threshold can be compared to the shortest distance found. One could arrive at a threshold by classifying a small training set for each target and choosing a threshold for which true positives and false positives are acceptable.

2.5.3 Using Computer Vision Feature Selectors and Extractors

Afroz and Greenstadt [15] proposed PhishZoo, a system that uses a number of features - both text based, image based - come up with a comprehensive profile of each target. Their most successful features involved a hybrid of text matching, screenshot matching, and logo matching. For screenshot and logo matching they used SIFT[16] for feature detection and matching.

Before arriving at SIFT, the authors evaluated other techniques including template matching and OCR. They found template matching failed due to small variations in scale and other small distortions. They found some success with OCR on logos where the text was clear, such as Paypal, but was less successful for logos mixing fonts, sizes and images such as Bank of America and Ebay.

SIFT was chosen as it is a more sophisticated approach used in the computer vision community for object recognition and image matching. The authors considered using variants of SIFT, but surmised that the SIFT was a reasonable choice for initial exploration.

To arrive at a score for logo matching, the authors extracted features for each of the logos in their training set. They would then extract all the images hyperlinked in the candidate web page individually and find matching keypoints between the

two images, then calculate a match score (Figure 2.12).

$$Match\ score = \frac{Number\ of\ keypoints\ matched}{Total\ keypoints\ in\ the\ original\ logo} \tag{2.12}$$

Afroz and Greenstadt also score the similarity of screenshots of the pages. They would extract the features in full resolution screenshots of their training page and candidate page, then find all the keypoints in the training image that matched the candidate image. The match score would be calculated in the same way as the logo's match score.

The authors evaluated their classifier using a data set of 1000 Phishing sites, in addition to 200 popular legitimate sites. They evaluated each of their features individually and found screenshots alone were not particularly effective (81.% True Positive Rate, 30.3% False Positive Rate). In their evaluation they argue scene analysis or image segmentation would be necessary for screenshots to be a descriptive feature.

With logo matching the authors had more success, particularly improving on the false positive rate with an 82.7% True Positive Rate and 2.5% False Positive Rate. They summarise that their success in logo matching supports their argument that with proper image segmentation, screenshots of pages would be a more descriptive feature.

SURF Selectors and K-Means clustering

Kuan-Ta Chen et al. [32] proposed a system that couples feature extraction with clustering to introduce relative spacial locality into the classification. They hypothesise that features can be clustered into recognisable components - such as features identifying corners of text boxes being clustered into a web form.

Keypoints are found by a Harris-Laplacian corner detector[33] and descriptors are extracted using the authors' own descriptor - named Lightweight Contrast Context Histogram (L-CCH). L-CCH is based on Contrast Context Histogram (CCH)[34], but is adapted to not be rotation invariance - justifying that such transformations are rarely seen in Phishing sites - creating a simpler and more efficient descriptor. The descriptor calculates difference in contrast between each pixel within a 9 pixel radius of the keypoint and the keypoint itself, grouped into histogram bins of 24.

When matching keypoints in the training image to keypoints in the candidate image, they compare the distance to the closest match to the distance to the second-closest match. If the ratio between the two distances is greater than 0.6, they exclude the match. This results in excluding keypoints in the training image that can match equally well in more than one place on the candidate image. This is based on the Lowe Ratio Test[35], a technique to find descriptors that match a single object in a database of images much better than any others.

The system then geographically clusters keypoints that have 'good' matches with the k-means algorithm[18]. The authors empirically evaluated their classifier and arrived at $k = 4$ as yielding the best results. For pages where the number of 'good' matches is lower than 4 (making k-means clustering impossible) the process system can short circuit as the candidate is dissimilar to all the pages in the training set.

Clusters in a training image are then matched to clusters in a candidate image through voting: A cluster A in one image *matches* a cluster B in the other image if the majority of the keypoints in A match keypoints in B. A keypoint is labelled *geographically matched* if its cluster matches its corresponding keypoint's cluster.

The two pages' similarity score is the ratio between the number of geographically

matched keypoints and the number of all keypoints identified in the two pages.

$$Similarity = \frac{|T_g| + |C_g|}{|T| + |C|} \tag{2.13}$$

where: T, C are sets of all keypoints in the training and candidate image; and T_g, C_g are sets of geographically matched keypoints in the training and candidate image.

Kuan-Ta Chen et al. evaluated their system with 1,638 Phishing sites targeting 5 popular sites, in addition to 300 legitimate pages from 74 other websites. Through empirical study they arrived at a similarity threshold of 0.6 - pages with higher scores are designated Phishing. Their system has 98%-100% True Positive Rate and 0.1% across the five targets they considered.

2.6 Distributed Processing

To perform classifications quickly on large data sets, we must consider distributed processing across many computers.

2.6.1 Amazon Web Services

Amazon Web Services[36] is a suite of computer infrastructure-for-hire products offered by Amazon. From load balancers to databases, Amazon offers more than 50 products for rent.

We consider Amazon's Elastic Compute Cloud (EC2). It offers Virtual Private Servers (VPS) with wide range of configuration options. Computation is charged by the hour.

2.6.2 Hadoop

Hadoop is a distributed batch processing framework. Its origin began when Google released a paper describing the Google File System[37] and a second paper describing MapReduce[38]. Google used the technology to perform massive data processing tasks such as ranking every web page on the Web.

Developers at the Yahoo started Hadoop in response one year later. It is made up of a distributed file system HDFS (Hadoop Distributed File System) and the MapReduce system for distributed processing of large amounts of data. Hadoop was open sourced in 2009 and adopted by the Apache Software Foundation who oversee its development to this day.

A *job* is split into a number of *tasks* by the *JobTracker* running on a master node. These tasks are then distributed to worker nodes in the cluster. On each worker a *TaskTracker* performs the work described by the task, and then informs the JobTracker once its finished.

As Hadoop is a batch processing framework, it requires all the data it needs to process at the beginning. It splits, distributes, and replicates all the data to machines in the cluster before performing the desired processing.

Use in Phishing Classification

Shrestha et al. [39] developed a Phishing classifier to run on Apache Hadoop. They evaluated it against a massive testing set of 80,965 Phishing websites. They tested evaluated their approach on the University of Alabama at Birmingham's computer cluster, and on Amazon's Elastic Map Reduce service. They were able to perform all their classifications within 6 minutes across 20 machines at Amazon - one hundred times faster than the same code on their machine. A significant

factor in the speed improvement, in addition to the quantity of computers, was the availability of more powerful machines at Amazon than their own.

2.6.3 Storm

Storm[40] is a distributed stream processing framework. Storm was open sourced by Twitter in 2014.

Storm considers data processing as a pipeline of transformations. Storm clusters are arranged by the application developer into a *topology*. Nodes within the topology are either *spouts* or *bolts*. Spouts provide input into the topology from an external source, and bolts perform data processing.

Data is passed around within the topology as *tuples*, representing a single unit of data as input to the bolts. The application developer specifies the quantity of each of the types of spouts and bolts they want, and Storm's master node named *nimbus* will distribute the applications across a cluster of computers - evaluating which computers are resource constrained to distribute the work in a way that maximises performance from the cluster.

Each tuple is given a unique id. When a bolt has finished processing a tuple, it *acks* the id so the sender of the tuple knows it has been successfully processed. If the tuple is not acked within a configurable timeout, the tuple is resent - though may be delivered to a different bolt of the same type - and the processing is repeated. This improves the robustness of the cluster.

A Phishing site classification system must be real-time. Phishing sites are at their most lucrative in the first few hours, so blacklisting the page quickly is vital.

Experiments

In this chapter we evaluate the clustering approach discussed in chapter 2, in addition to a histogram correlation classifier. We use our dataset to validate their results and to provide a benchmark for our own approach. In chapter 4 we propose our own approach and evaluate it on the same data.

3.1 Test Data Collection

We have obtained data from Netcraft[9] for our evaluation. Netcraft is a world-leading supplier of Phishing countermeasures including a Phishing blacklist, Phishing takedowns, and more.

Candidates that cannot be classified by Netcraft's proprietary Phishing classifier are reviewed manually. To guard against the potential for drive-by malware being inadvertently installed by reviewers during the review process, all manually classified candidates are submitted to the Netcraft's sand-boxed screenshot service. Instead of visiting the sites themselves, reviewers consider the screenshots in addition to meta information from the site.

We downloaded a small subset of Netcraft's screenshots to create a training set and testing set. We then picked the 30 most commonly represented targets from the data we downloaded. Additionally, we created a *truth set* by navigating to the login pages of these common targets and capturing a screenshot of the page.

Our training set consists of 10 randomly selected labelled pages per class for a total of 300 pages. Our full testing set consists of 1600 Phishing sites with an skewed distribution across the classes.

Table 3.1: Size of training and full testing sets for each class

Target	Train	Test	Target	Train	Test	Target	Train	Test
Adobe	10	23	Comcast	10	18	Microsoft	10	190
Apple	10	255	DHL	10	16	Paypal	10	168
B. de Credito	10	16	Dropbox	10	20	Ricardoeletro	10	18
B. do Brasil	10	48	Ebay	10	116	Taobao	10	39
B. Santander	10	17	Facebook	10	79	Tencent	10	32
Barclays	10	35	Freefr	10	20	Tesco	10	13
BNP Paribas	10	16	Google	10	59	USAA	10	29
Capital One	10	13	HMRC	10	25	Wells Fargo	10	47
Cartasi	10	171	ICBC	10	16	Yahoo	10	27
Chase	10	21	L.B. Postale	10	17	Zhejiang	10	36

3.2 Evaluation Harness

Our experiments were implemented in Python, using OpenCV and NumPy, from the original papers with some small variations were required. For each classification we recorded the predicted class, the correct class and the similarity score for the closest match. We then consider the impact of introducing a threshold-based decision classifier at different thresholds.

OpenCV (Open Source Computer Vision) is a substantial library offering hunderds of methods for image processing, computer vision and machine learning. It is written in C++, but has official supported C and Python bindings.

Table 3.2: Performance: Histogram Correlation Classifier - Single Training Page

Target	TP	FP	FN	Precision	Recall
Total	835	769	769		
Micro-averaged				0.52	0.52
Macro-averaged				0.71	0.44

3.3 Evaluation

3.3.1 Hue Saturation Histogram Correlation

We developed a system to classify screenshots by their colour histogram. The system converts each screenshot to Hue Saturation Value (HSV) colour space and compute a 2D histogram from each pixel's hue and saturation. It then compares two screenshots by calculating the correlation coefficient between their histograms. The correlation coefficient acts as our similarity score.

We evaluated the impact of two different kinds of training data: screenshots of the legitimate site, and our training sample of Phish. We experiment with multiple training sizes to establish the impact of introducing screenshots of known Phishing Sites into our training image set.

In Figure 3.1 we evaluate the performance of the Histogram classifier with a single screenshot of the target's website. We calculate the number of True Positives, False Positives, and False Negatives, in addition to Precision and Recall, for each target (Table 3.2). We calculate both micro and macro averages of Precision and Recall to compare the classifier as a whole.

Figure 3.1 shows that sites across our testing set are frequently misidentified as Apple. We inspected the legitimate Apple login page (Figure 3.2) and found the image is almost exclusively greyscale. The HSV colour space defines the hue and

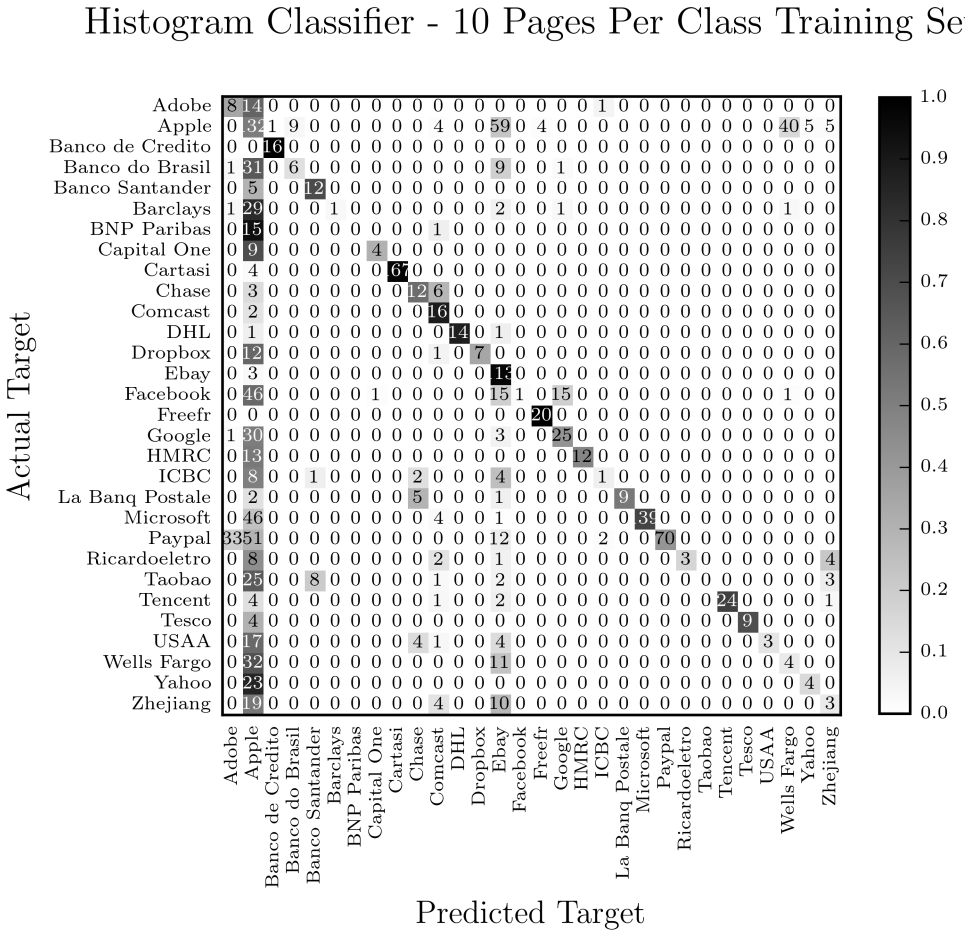


Figure 3.1: A Confusion Matrix of our small test set with a singe legitimate page as a training.

saturation of greyscale colours to be 0. As all greyscale values have a hue and saturation of 0, there is a large peak at the origin of the histogram for Apple. A significant number of pages make use of white, black or grey coloured background - correlating with the Apple page. If the candidate image's other colours don't correlate with the ground truth of their target, the candidate can be misclassified as Apple.

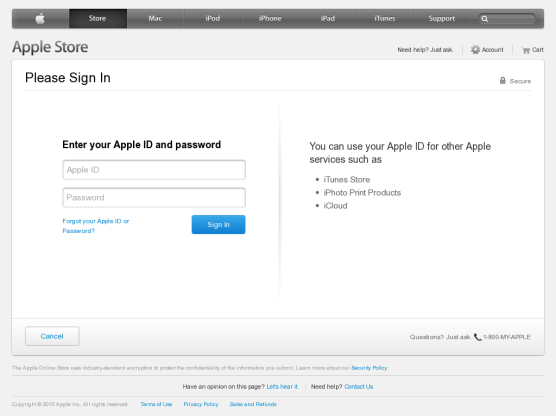
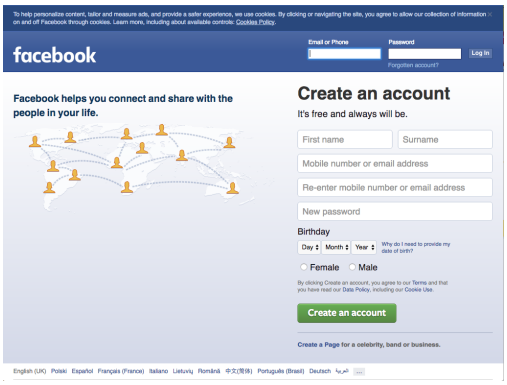


Figure 3.2: The ground truth Apple Store Login form.

Figure 3.1 also demonstrates the limitations of only training with a single screenshot of the legitimate site. No Phishing sites targeting Barclays, BNP Paribas, Facebook or Taobao could be identified. We examined the legitimate pages and Phish for these targets and found Phishing sites that did not look like the legitimate site at all, but are clearly imitating the brand.



(a) The legitimate Facebook home page

(b) An imitation Facebook Security Alert

Figure 3.3: Two completely different pages that target the same brand.

As discussed in subsection 2.1.3, Afroz et al. [15] found that fabricating branded websites that do not have corresponding legitimate pages was responsible for 21.05% of the Phish they observed. Consequently we expanded the ground truth of the Histogram Classifier to include our smaller training set. It is clear from the strong correlation in Figure 3.4 and overall performance (Table 3.3) that even just a small training sample of 10 Phishing pages per target dramatically improves the performance of the classifier compared to training on just a single legitimate page for each class.

Table 3.3: Histogram Correlation - 10 Page Per Class Training Set

	TP	FP	FN	Precision	Recall
Total	1276	328	328		
Micro-averaged				0.80	0.80
Macro-averaged				0.85	0.82

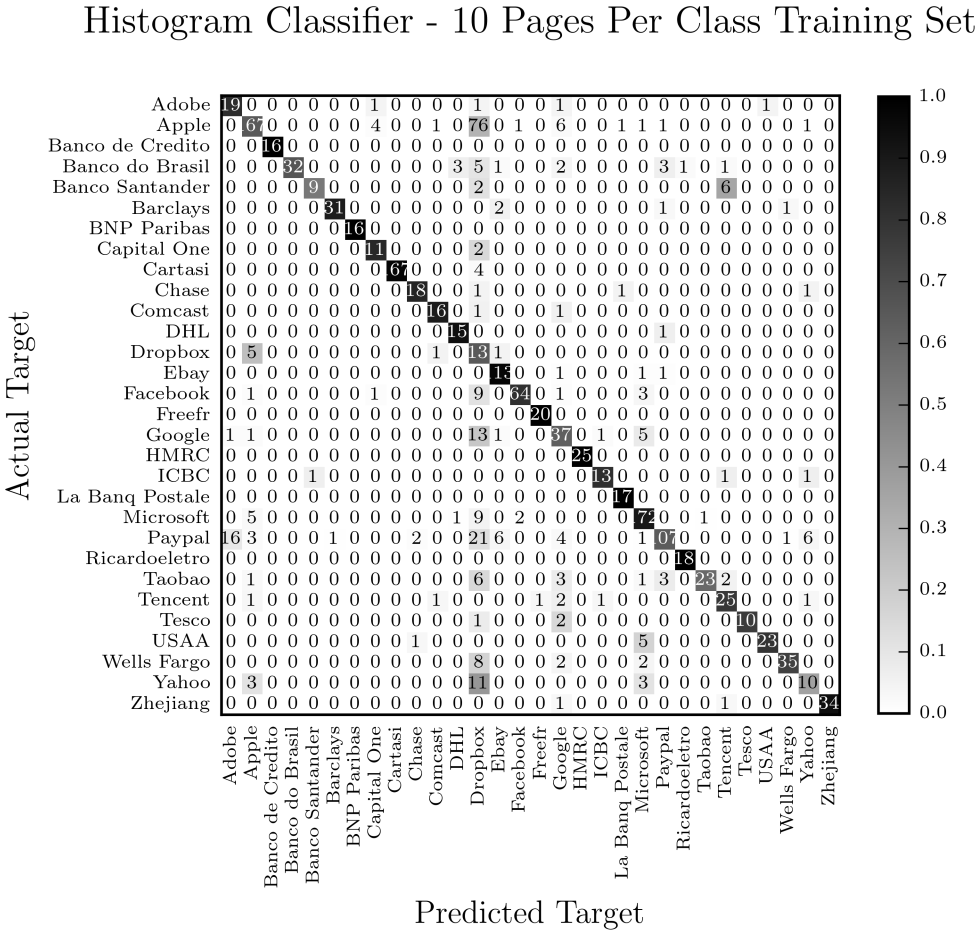


Figure 3.4: Confusion matrix for Histogram Correlation Classifier.

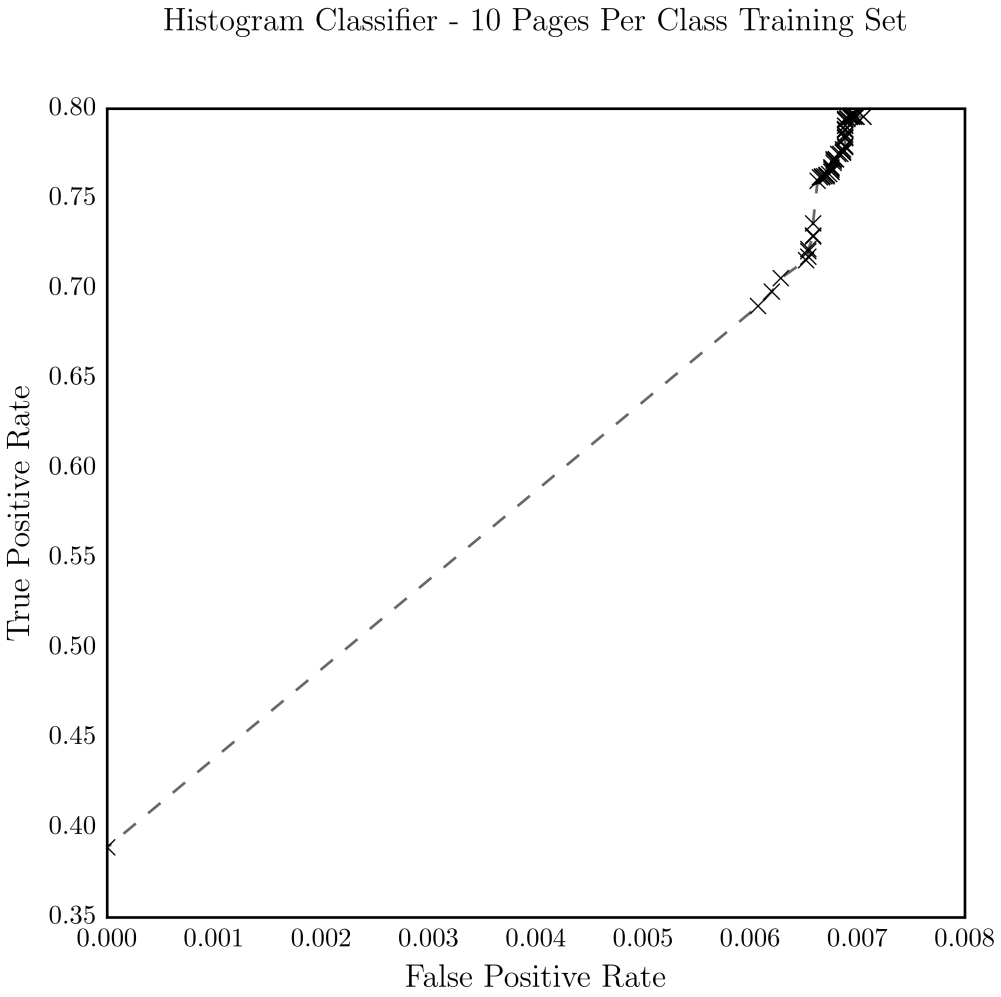


Figure 3.5: A ROC Curve shows the classifier's performance at different thresholds.

Table 3.4: Histogram Correlation - 10 Page per Class Training Set with threshold of 1

	TP	FP	FP	Precision	Recall
Total	624	0	980		
Micro-averaged				1.00	0.39
Macro-averaged				1.00	0.41

We evaluated the performance of the classifier when using a threshold-based decision rule. If the similarity score (the correlation coefficient between the two histograms) is greater than a threshold t, the website is considered Phishing. We evaluated thresholds within the range 0.01 to 1.00 and plotted the performance of the classifier as a ROC Curve (Figure 3.5). At a threshold of 1.00, the classifier performs with a precision of 1 - that is, every page it believes is Phishing *is* Phishing. 41% of the Phish in our test set could be classified with this classifier (Table 3.4) at this threshold.

As we have removed byte-level identical files from our training set, these pages must be otherwise identical except for variations in metadata in the file brought about by the screenshot process. This approach performs well for Phishing campaigns that exist on multiple URLs. This technique could be used in conjunction with another classifier to remove low-hanging fruit before assessing the remainder with a classifier with a more intelligent approach to variations.

3.3.2 Keypoint Clustering with SURF and K-Means - Kuan-Ta Chen et al.

We looked to evaluate the performance of an approach that makes use of spatial clustering. Intuitively, individual keypoints can be combined to form recognisable regions of a legitimate site - such as a logo, navigation bar, and structure of the login section.

Table 3.5: K-Means SURF Clustering - Single Page Per Class Training Set

	TP	FP	FN	Precision	Recall
Total	973	577	627		
Micro-averaged				0.63	0.61
Macro-averaged				0.75	0.61

Table 3.6: K-Means SURF Clustering - 10 Page Per Class Training Set

	TP	FP	FN	Precision	Recall
Total	1343	247	257		
Micro-averaged				0.84	0.84
Macro-averaged				0.86	0.86

Ideally we would have liked to have evaluated the approach proposed by Kuan-Ta Chen et al. however their Context Colour Histogram descriptor is only available as a Windows binary, and not in source code form. The authors have shown that their approach has similar performance to SIFT[34], suggesting it could be used as an alternative. However, we found that the speed of SIFT was too slow to perform a larger evaluation. Consequently we chose to substitute SURF, as it has a very similar design and is far faster.

The authors first propose using a single screenshot of the legitimate site in the training set. There is a clear correlation in the confusion matrix (Figure 3.6) but we see particularly poor results for Apple and Taobao. None of the sites targeting Apple were correctly classified, and only one for Taobao. Apple and Taobao both have a variety of commonly Phished pages that do not look similar, such as the Apple Store and iCloud. To try to catch these, we increased the number of pages per target in our training set to 10.

Both precision and recall improve with the larger training set. We evaluated the approach as a threshold-based decision classifier. Kuan-Ta Chen et al. recommend

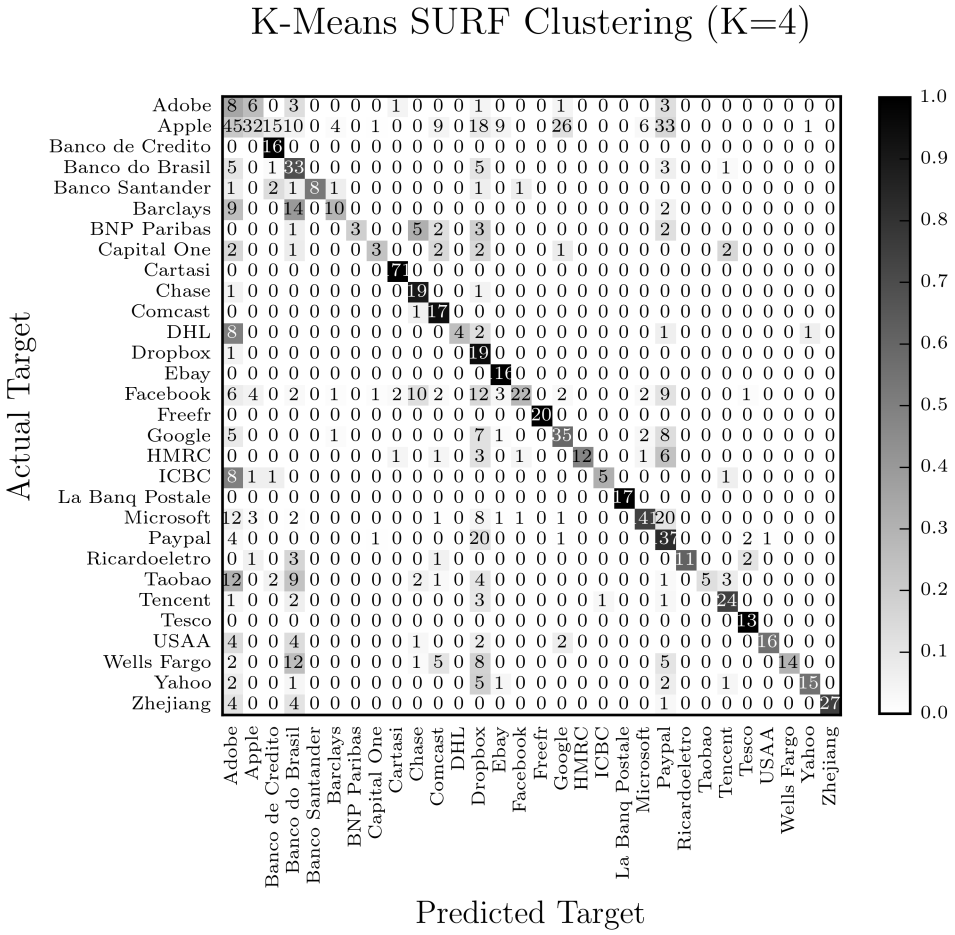


Figure 3.6: Confusion matrix for K-Means SURF Clustering with a single ground truth image.

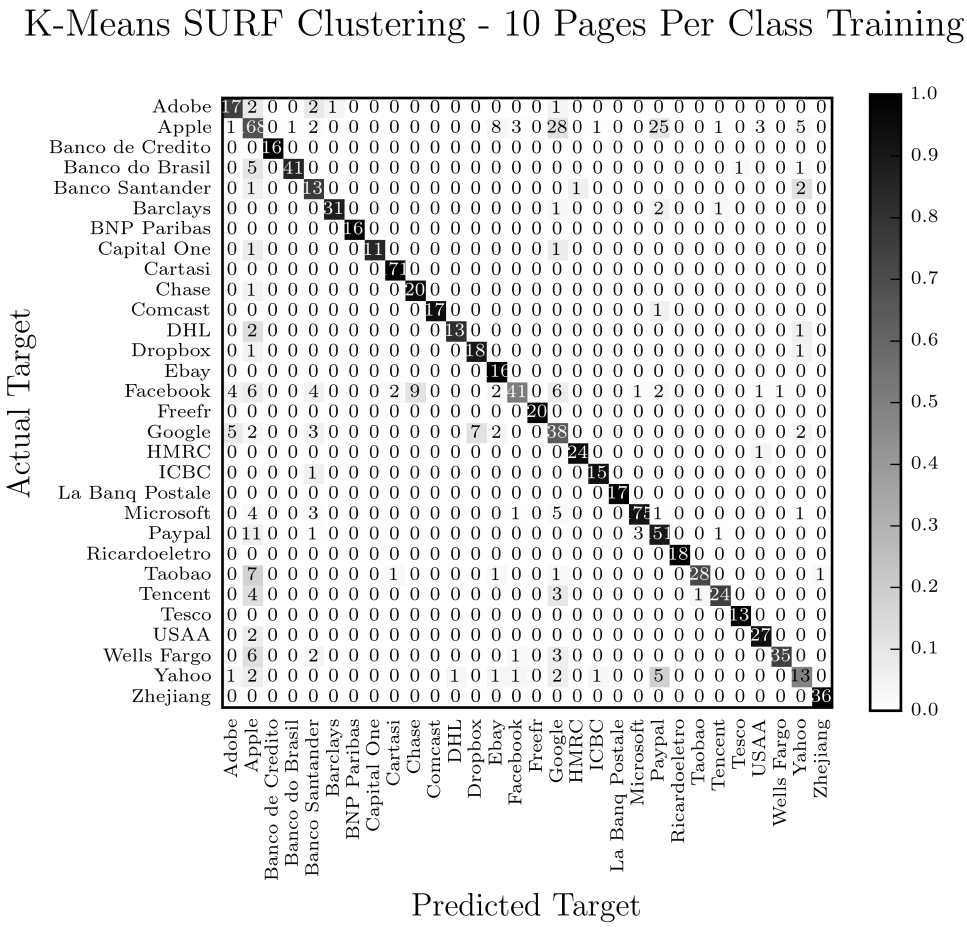


Figure 3.7: Confusion matrix for K-Means SURF Clustering with a 10 page per class training set.

Table 3.7: K-Means SURF Clustering (K=4) - t=0.6

	TP	FP	FN	Precision	Recall
Total	846	1	754		
Micro-averaged				1.00	0.53
Macro-averaged				1.00	0.56

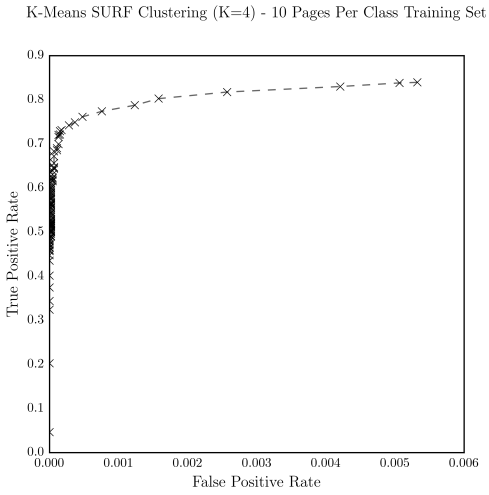


Figure 3.8: A ROC Curve shows the K-Means SURF Clustering classifier's performance at different thresholds.

a threshold of 0.6 to arrive at a binary decision on whether a site is Phishing or not for their classifier. At this threshold (Table 3.7), we found the classifier had a precision of nearly 1 (only one false positive), but a recall of only 0.56.

We evaluated the impact of thresholds t from 0.01 to 1.00. We found (Figure 3.8) that the threshold can be lowered to 0.35, gaining true positive classifications, without incurring additional false positives.

Table 3.8: K-Means SURF Clustering (K=4) - t=0.35

	TP	FP	FN	Precision	Recall
Total	961	1	639		
Micro-averaged				1.00	0.60
Macro-averaged				1.00	0.63

Proposal

4.1 Observations

For a user to identify the *brand* of a website, there must exist components or features on the page that *only* appear on pages associated with that brand. These could be logos, icons, the style of the login form, menus or other aspects of the page. Many regions are common across different targets, such as individual form fields, and we should not base a classification decision on matching such regions.

Feature detectors and extractors used in computer vision, such as SURF and SIFT, find many small keypoints, often with a radius of just a few pixels. A Phishing classifier should match larger elements of a webpage that are recognisable to a human.

We cannot use a K-Means clustering algorithm to find regions that indicate the identity of the page. Variety in styling and layout means there is not a common number of K regions across Phish targeting a single site, let alone across all Phish. Any grouping of keypoints must be relative to the number of components on the page.

A target cannot be represented by a single legitimate page in our ground truth. Many targets have multiple sign-in pages with vastly different styles for the different products they offer. Additionally, Phishing websites may be completely fabricated and not have a corresponding legitimate page.

4.2 Training

The classifier considers a set of labelled training images. We maintain a set of HSV histograms representing each training image the classifier has been trained on. When we train on a new image, we calculate the correlation coefficient between it and the histograms of all the other images we've seen. If a new training image perfectly correlates with a page the classifier has trained on, the new training image is ignored. This avoids cluttering the model with features that offer no additional classification performance.

Next, we find keypoints in the image using FAST - a high speed feature detector. We geospatially cluster these keypoints using DBSCAN. DBSCAN is a density-based clustering algorithm that groups points which are close together into clusters. Keypoints which are not close to others are considered outliers and are not included in any clusters. We name each cluster a *region*.

We exclude the outliers from our set of features. We extract descriptors for the remaining keypoints using BRIEF - a binary descriptor extractor. We chose BRIEF due to its significant matching speed compared to SURF[41], while retaining recognition performance. All keypoints and their corresponding descriptors are grouped by their cluster into regions. Each region is recorded as part of a model representing the target. We repeat this for every page in our training set, adding regions that we find on pages to our model of a particular target. This effectively gives us a model for all the regions associated with our target in our training set.

4.3 Classification

Our hypothesis for page matching is simple: Regions that make good indicators that a page is imitating a target T do not appear on pages that do not imitate T.

e.g. The Paypal logo is a good indicator a Phishing site is targeting Paypal, as it does not appear on

We extract keypoints from the candidate image using FAST and cluster them with DBSCAN, removing any outliers. We then calculate BRIEF descriptors for the remaining features.

Regions that appear in our candidate image are compared to all the regions we've trained on. Regions are compared by matching keypoints within each region. The *distance* between two regions is defined as the mean euclidean distance of their matching keypoints.

For each region in a candidate image, we examine the two closest matching regions in our model - where the second-closest matching region is from a different target to the closest. We then apply a ratio test to establish the distinctiveness of the closest matching region. We calculate the ratio of the distances from the candidate to each of the two regions. If the ratio is less than a threshold r, then the closer region is suitably *distinctive* to use in classification decision making. We tag the candidate region with the identity of the region it most closely matches.

Once we have processed all the regions in the candidate image, we identify its target to be the target its regions have matched the most. The similarity score is the ratio of regions that match that target and the total number of regions in the image.

4.4 Discussion

We recognised the parallels between our region matching requirement and image stitching - a technique used to join two images of the same scene with overlapping regions taken from two different perspectives, often to create a panorama. We

evaluate why we do not use the technique in our evaluation subsection 6.1.4.

Implementation

5.1 Architecture

The classifier consists of three components, a screenshots submission server and queue (section 5.3), an image classifier (subsection 5.4.1), and a blacklist server (section 5.5). The image classifier is part of a storm cluster. The classifier is hosted on Amazon Web Services, a cloud computing provider, to allow horizontal scaling of classification as necessary.

We used Python throughout the system. The language was chosen for the image classifier due to its OpenCV bindings and ease of prototyping when compared to C++ or C. We reduce complexity of the system by using the same language throughout - made easy by Python's great standard library and large choice of additional libraries.

5.2 Amazon Web Services

We deploy our classifier on Amazon Web Service's Elastic Compute Cloud platform. We can speed up deployment of the cluster by configuring a virtual machine once with our classifier and dependencies, and then using Amazon's virtual machine snapshot tool to store the configuration for later. A can set up a new machine using the previous snapshot as a template, saving the substantial time spent installing software.

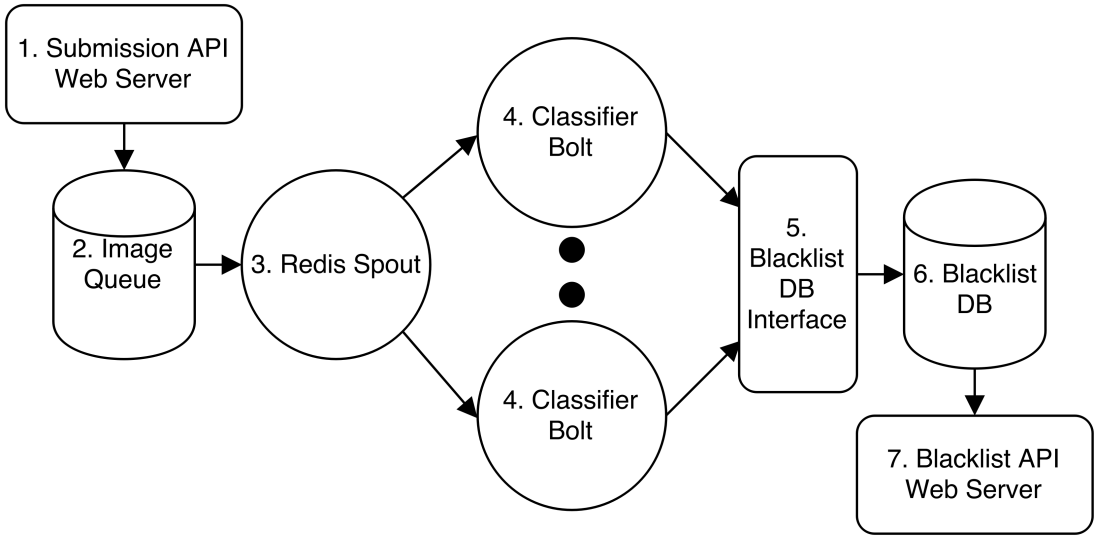


Figure 5.1: Overview of the implementation architecture

point

5.3 Image Submission

We produced an image submission API to receive screenshots to our classification system. Screenshots are submitted in addition to the URL the image was fetched from for identification.

The web server is written in Python using the Tornado[42] web framework to parse the requests. On receiving an image, the submission system Base64 encodes the image and inserts it into a Redis[43] queue through its popular Python client[44]. Each element in queue is a tuple of the URL and the Base64 encoded image.

We chose Redis as the backend of our queueing system as it can recover from disk in the event of a power outage, is incredibly memory efficient, and the data is inherently ephemeral. We do not require a queueing system which supports automatic retries as that is handled out of the box by Storm spouts.

Table 5.1: Redis Tuple

Index	Description
0	URL of the site
1	screenshot, base 64 encoded
2	Submission DateTime

5.4 Storm

We created an Apache Storm cluster for the screenshot and classification process. An Apache Storm cluster is made up of *nodes* divided into *spouts* and *bolts*. Bolts can take input from spouts or other bolts, but spouts take no input from the cluster. Spouts read from a source, in this case the Redis submission queue, and submit tuples to connected bolts.

We use streamparse [45] to communicate with Storm from our Python code. Streamparse is a Python interface to the underlying Storm framework. It supports easy deployment to the cluster through the command `sparse submit`, which packages the source code and uploads it to the cluster for execution.

The submission spout continually polls the submission queue for new images. When one is returned, it is placed on a pending buffer to be delivered to one of the classifier bolts connected to it. When the classifier bolt is available for another image, it will ask for one from the submission spout and will receive a *tuple* containing the image and a unique identifier for the tuple.

The spout keeps track of tuples it has sent and acknowledgements it has received. If the spout does not receive an acknowledgement with the tuple's identifier within a timeout, it will retransmit the tuple. We define the timeout to be a generous one minute. The timeout must be longer than the time it takes to classify a page.

As classification time is proportional to the number of training images, we would need to adjust the timeout if the number of training images was increased from our current number of 10. We configured Storm to drop URLs which have been resubmitted twice so as to not clog up the classifier if the page takes unexpectedly long to classify.

When the classifier receives a tuple, it decodes the image back to a native image format. It executes the classifier against the image to identify its target. The result and its similarity score are submitted to the Blacklist database interface with an HTTP request.

We had to perform extensive configuration tuning to achieve results from Storm. The software has many parameters around timeouts. Storm's ideal use case is a large number of very simple and very fast data transformations arranged in a topology to perform a complex transformation. Our classifier can take up to a minute to classify a site in its worst case. By default, Storm would require an acknowledgement from classifier bolt within a few seconds.

5.4.1 Image Classifier

We implemented our image classifier in Python. We chose Python due to its great library support and ease of prototyping. Although a dynamic 'slow' language, much of the computationally intensive operations in the classifier were performed in C by libraries such as OpenCV 3.1[46], NumPy[47], and scikit-learn[48]. We use OpenCV for constructing histograms, for finding keypoints, extracting descriptors and then matching them. We use NumPy for its array slicing and performant vector operations. And we use scikit-learn for its clustering libraries.

The result of classification can either be one of the labels for the training data used to construct the classifier, or - if the candidate cannot be matched with any

label - the sentinel value *None*. We then submit labeled results to the Blacklist DB Interface over HTTP.

5.5 Blacklist

5.5.1 DB Interface

The Blacklist DB Interface provides a simple web API for the submitting the results of the classification. We choose to go via a web application instead of going straight to the database to avoid directly pinning the database schema to the classification code. The schema is currently very simple, but were the blacklist page given more functionality it may require refactoring. The DB interface API limits the refactoring to the just the database and the API, and not every single classifier running in the cluster.

The interface interacts with the database with SQLAlchemy - a Python Object Relational Mapper for databases.

5.5.2 Database

The database itself runs PostgreSQL[49]. The data has no characteristics that make PostgreSQL a better choice than any other database, it was chosen as I am familiar with it and SQLAlchemy's supports it.

Finally, we prepared a proof-of-concept read-only API to the blacklist that allows developers of products who require a Phishing blacklist, such as email clients and web browsers, to query for new a list of Phishing sites.

Table 5.2: Classification Log

Column	Type
url	Arbitrary length string
submitted_at	DateTime
classified_at	DateTime
target	Arbitrary length string

Evaluation

In this chapter we evaluate the performance of the Distinctive Region Classifier, and examine some case studies of it performing both well and poorly. We also evaluate the performance and cost of the distributed classification cluster.

6.1 Distinctive Region Classifier

6.1.1 Performance

We evaluated the performance of the classifier trained with our training set of 10 images for each target, and tested against our remaining set. From the confusion matrix in Figure 6.1 we can see the classifier performs well across all clusters with a clear strong correlation.

In Table 6.1 we summarise the performance of the classifier in terms of precision and recall. A precision of 0.9 overall or 0.85 weighted by classifier , but not 100%. From the ROC and PR curves in Figure 6.2, we see precision can be improved to 1 with the introduction of a threshold. When we introduce a threshold of $t = 0.73$, the recall is reduced to 0.57 for the classifier as a whole (macro-average).

We found these results to be a promising indicator for the technique, though there

Table 6.1: Distinctive Region Classifier Performance

Target	TP	FP	FN	Precision	Recall
Total	1307	143	293		
Micro-averaged				0.90	0.82
Macro-averaged				0.85	0.85

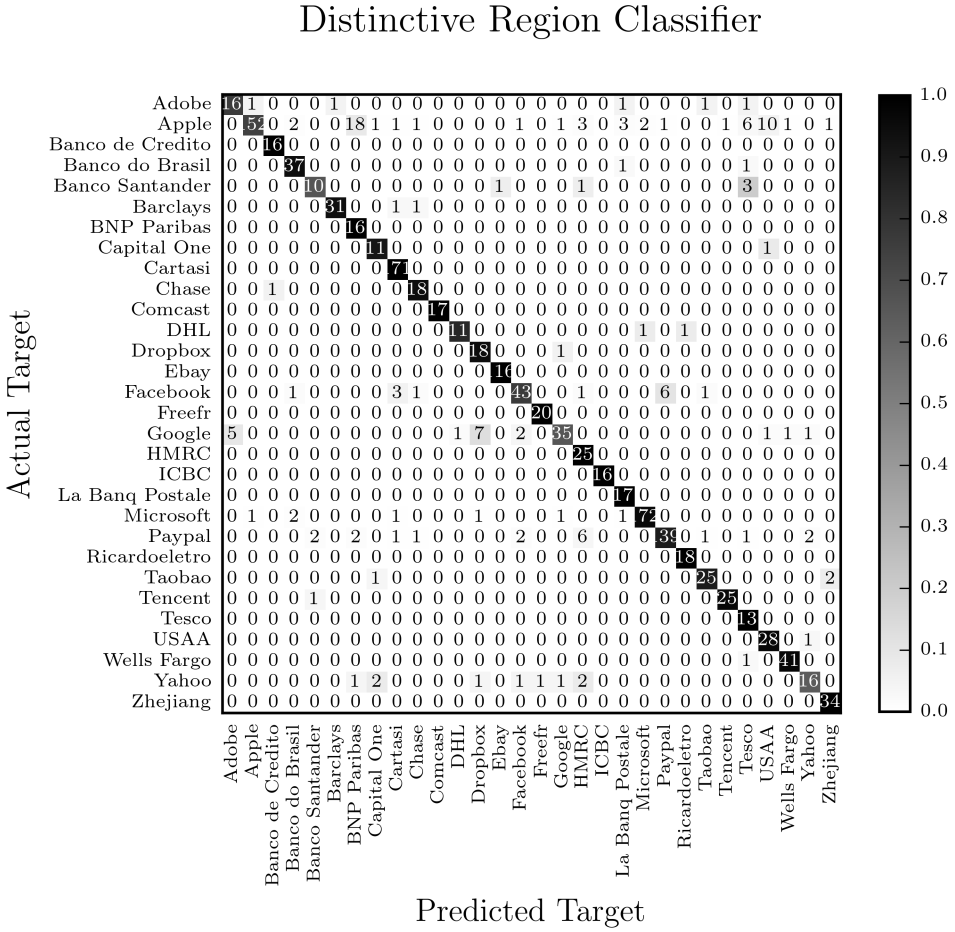
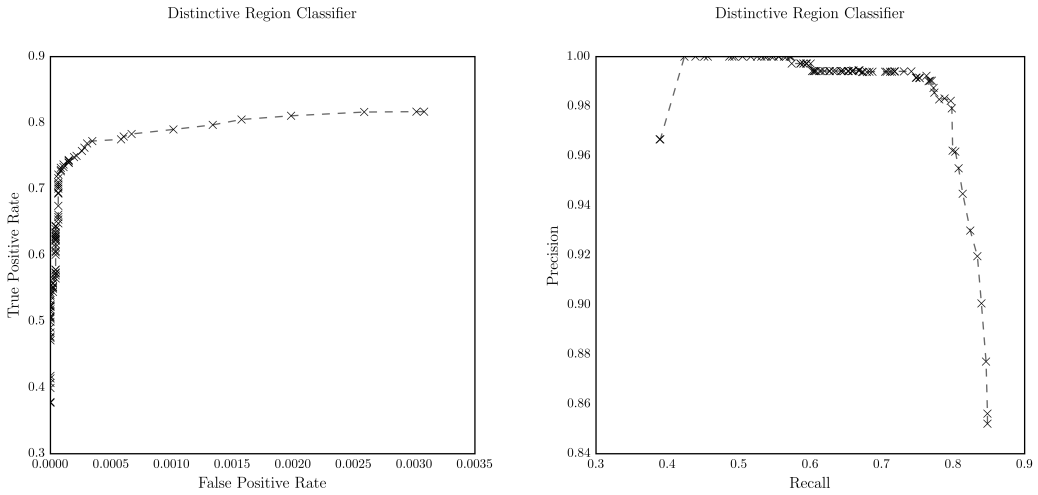


Figure 6.1: Confusion matrix for Distinctive Region Classifier with no threshold

Table 6.2: Distinctive Region Classifier - t=0.73

Target	TP	FP	FN	Precision	Recall
Total	869	0	731		
Micro-averaged				1.00	0.54
Macro-averaged				1.00	0.57



(a) The ROC curve (b) The PR curve

Figure 6.2: ROC and PR curves for the Distinctive Region Classifier

is still some way to go. The high number of false negatives (Table 6.1) is due to classifier being unable to find any regions that pass the 'distinctiveness' ratio test between a site and our training set at all. We believe this is due to calculating the distance between two regions as the mean distance of all the matching keypoints. The distance calculation between matching keypoints does not take into account any change in spatial layout of the points. This result is keypoints can be regarded as very similar, but actually appear scattered around such that it is impossible the regions they are part of are similar (For an example, see Figure 6.5).

If two dissimilar regions are calculated a low distance, when that distance is compared to the distance between two similar regions - the ratio test will fail. The similar region will not be considered distinctive enough to be used for identification.

Figure 6.3: The classifier identifies the Paypal logo and matches the two pages without requiring a separate logo training set.

6.1.2 Case Studies

Paypal Logo Detection

In Figure 6.3 we can see the approach performing as we hoped. The image on the left is a candidate image for classification, and the image on the right is a training image for Paypal. Without any supervision, the classifier has recognised that the region of keypoints which overlap the logo is distinctive to just one target - Paypal. Despite pages actually looking completely different, the classifier correctly identifies the Phishing site.

Matching text

Text on webpages normally has high contrast with its background. Consequently, corners of letters and numbers are frequently selected as keypoints by the keypoint detector. In Figure 6.4 the two pages match by the placeholder text in their password fields. The Santander logo does actually match a region in our training set of the Santander logo. However, there are only two regions on the candidate image that match any other target. This case study exposes that our algorithm

Figure 6.4: The classifier matches the word "Password" between the Santander page and the Ebay page.

needs to make an intelligent decision in the case of a tie in the number of distinctive regions attributed to a target.

Additionally, descriptors that match corners of a letter can also match corners of completely different letters - creating false matches. In Figure 6.5 the labels for the address fields in Capital One are rearranged to match the footer in the Wells Fargo page. We could improve our distance calculation in matching regions to take into account the relative spatial distribution of matching points, possibly through comparing the vector between two keypoints in the candidate region to the vector between the corresponding keypoints in the training region.

6.1.3 Noisy Training Pages

One risk of including Phishing sites in the training set for the Distinctive Region Classifier is if the Phishing sites are 'fake pages' - that is, the page is not imitating any page on the legitimate site - or if the Phishing site includes substantial additional graphics such as adverts, is those regions will be associated with the brand but are not distinctive to Phishing webpages targeting just that brand.

Figure 6.5: The labels for the address fields in the Capital One page are rearranged to match the footer in the Wells Fargo page

A case where this can occur is customisable Phishing kits. As we saw in Figure 2.4, as Phishing kit could provide a site that is completely generic but the only difference between the two pages could be the logo. If we only had the one in our training set, even if we had seen the logo before, it's possible the generic content would outweigh the logo when evaluating the target of the site.

Nevertheless, we could not find any examples of this in our results. Additionally, we believe the effect would be reduced still further for larger training sets.

6.1.4 Homography Detection

When matching regions of two images, we must calculate a distance that represents how dissimilar regions are. At present we use mean distance of the matching points in both regions. We considered the parallels between this process and image stitching - commonly used to create panoramas.

If one region contains a component like a logo that exists or partially exists in the other region, then a homography will exist that transforms the region such that the common points will align. Brown and Lowe[50] proposed a probabilistic

model for assessing whether a homography is correct. Their probability measure would provide a score of the similarity between the two regions. Unfortunately, our implementation proved prohibitively slow - despite using the OpenCV implementation. With potentially 30 regions per page and 300 pages in our training set, the operation is too computationally expensive to perform without an index to reduce the number of regions compared.

6.2 Scalable Classification Cluster

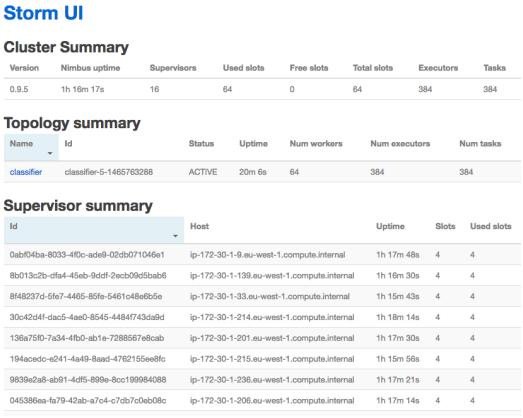
6.2.1 Classification Throughput

We initiated our cluster to have 16 AWS virtual machines - or *instances* - with the `c4.4xlarge` configuration. This configuration was chosen due to offering 16 cores at the lowest price.

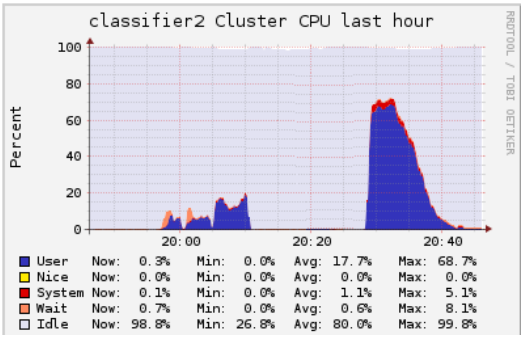
Processor	No. Cores	Memory
Intel Xeon E5-2666 v3	16	30

We initially constructed our Storm topology such that each computer hosted one classifier bolt. Once we started our testing, we found this only used a quarter of the available CPU time. OpenCV and NumPy both make use of multi-processing for some of their algorithms when executed on large datasets. We expect the CPU was not being fully saturated by the process due to time spent waiting for other parallel computations to finish. As the number of cores on the machine is large, the amount of time spent performing housekeeping of parallel process like collecting results bottlenecks the CPU.

We redeployed the cluster, this time with 64 classifiers - 4 classifier bolts per computer. The CPU utilisation of the cluster went up significantly - from 18% to



(a) The Storm User Interface



(b) CPU utilisaiton of the whole cluster. The second peak is after deploying an additional 48 bolts onto the cluster.

Figure 6.6

over 70%. (Figure 6.6).

We tested the system by submitting our 1600 test screenshots. To avoid creating a bottleneck where the computing cluster was classifying screenshots faster than we could upload new ones, we first copied the files to the API server as a batch and then submitted each screenshot automatically to localhost. This tested the submission process, while avoiding slowing down the classification process due to a slow source.

We found the cluster operated at an average throughput of 1.44 classifications a second, and the average classification time approximately 45 seconds.

6.2.2 Cost Analysis

Determining the cost of Amazon EC2 services is straight forward as their prices are public. c4.4xlarge instances cost $0.838[51] an hour. 16 instances would cost $13.408 an hour. At our rate of classification that corresponds to 271 classifications per dollar.

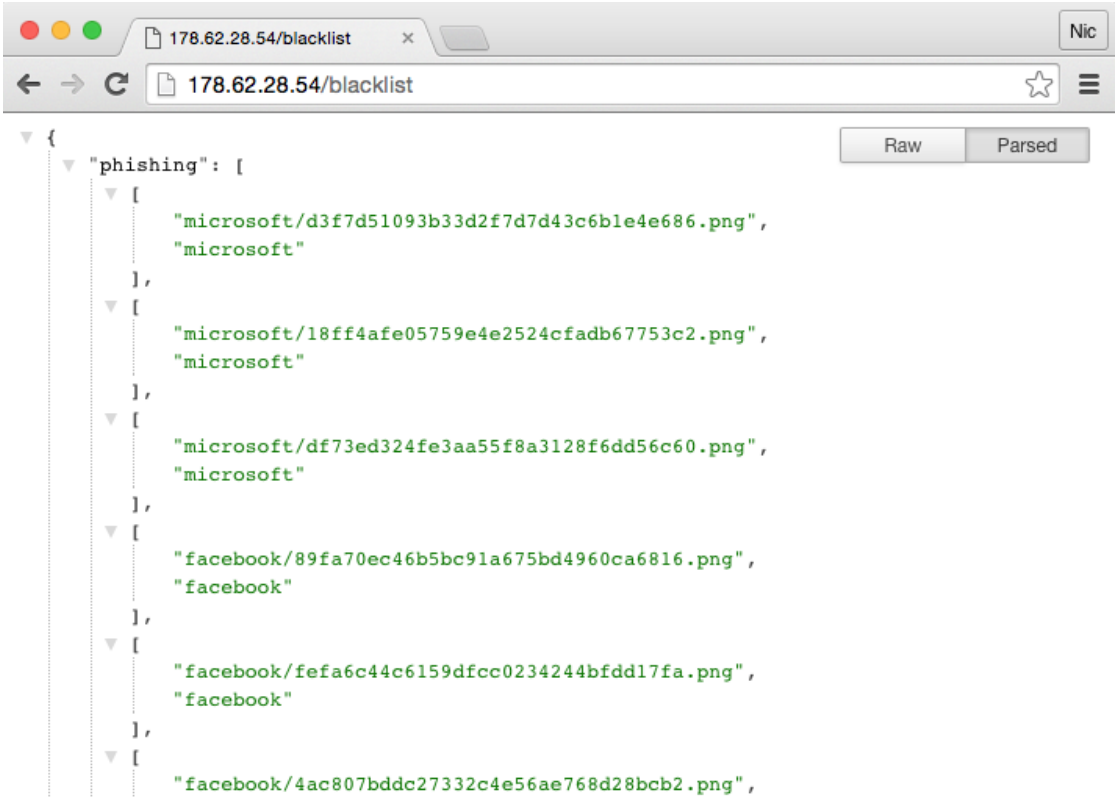


Figure 6.7: A short blacklist from our classifier's decisions. Note the screenshots are identified as file names in place of URLs as we do not have the URLs for these files, and the filename provides the screenshot's location on our file system.

6.2.3 Blacklist

Once classified, reports are available in an application-friendly schema.

6.2.4 Cannot retrain cluster without shutting it down and redeploying

A significant drawback of the cluster is it cannot retrain nodes in the cluster without shutting down the cluster, replacing the training set, and starting it up again. To ensure that the training set stays current, ideally we would like to add functionality for the classifiers to retrain on-the-fly.

We hypothesise the classifier could routinely check a shared file system for new training images. In the event new ones are added, each classifier should wait a random amount of time before beginning training. As a classifier cannot perform classifications while training, the random wait prevents the classifiers from all pausing at the same time.

Conclusions

In this chapter summarize the achievements of the project and suggest some work for the future.

7.1 Objectives

7.1.1 Classifier

We have succeeded in developing an effective classifier that identifies Phishing sites from distinctive features that only occur in sites targeting the brand. We evaluated the performance of our classifier on a large data set of Phishing sites and found it can classify 57% of Phishing sites it examines with 0 false positives. 0 false positives means the system could be deployed in classifying Phishing attacks for influential blacklists such as the Google Safe Browsing blacklist.

We evaluated a wide range of published Phishing classification systems for their merit, and incorporated techniques into our approach such as clustering keypoints found by feature detectors used in computer vision.

Additionally we succeeded in demonstrating that including labelled Phishing sites in the training set for a Phishing classifier will improve the performance of the classifier. We showed this was true with three different classification algorithms, Histogram Correlation, K-Means SURF Clustering, and Distinctive Region Classifier. We believe we are the first to do demonstrate this improvement for exclusively image-based Phishing classifiers.

7.1.2 Distributed Classification Cluster

We also succeeded in developing a scalable classification system, and evaluated it on Amazon's Elastic Compute Cloud. The system uses popular industry distributed processing framework Apache Storm, which can handle resubmission of classifications that if a classifier goes down. The system has developer-friendly APIs for screenshot submission.

In the real-world, classification systems need to be capable of classifying millions of websites a month. Our cluster is already prepared to classify at this frequency.

The distributed classifier proved fast enough to classify far in excess of the 70,000 unique Phishing sites a month recorded by the Anti-Phishing Working Group [3].

Additionally, we evaluated the cost of the cluster in terms of how many classifications it can perform per dollar spent.

7.2 Further Work

Over the period of this project, we established a great many improvements and ideas that we would like to have investigated if more time could be available for the project.

7.2.1 Speed Improvements

At present our implementation of identifying distinctive regions involves comparing to every single region in the training set. We would like to explore the use of fast data structures for spacial search such as FLANN (Fast Library for Approximate Nearest Neighbours). Although we attempted to use its implementation in OpenCV to identify a subset of regions to compare, bugs within the OpenCV

library prevented us from completing it.

7.2.2 An improved algorithm for calculating similarity between regions

In our evaluation we addressed how the lack of a distance metric that was aware of the expected location of two matching points increases our false negative rate. We would like to evaluate more techniques for computing similarity, particularly approaches that incorporate the spatial distribution of matching keypoints on each region. We believe this would reduce the false negative rate of the classifier.

7.2.3 Alternative Feature Descriptors

At present we describe keypoints in a screenshot with BRIEF feature descriptors. BRIEF feature descriptors, like many other descriptors, only consider the intensity of the neighbouring pixels. This affords tolerance to lighting changes which is useful for object recognition in photographs. However, within a webpage, the colour and saturation of a keypoint are particularly useful indicators for matching. Colour carries an important association with branding, so it is unlikely the Phisher will dramatically change the colour scheme of a logo they have put on their website as this would be a huge indicator for potential victims that the page was not legitimate.

Bibliography

[1] Richard Clayton. *Security Protocols: 13th International Workshop, Cambridge, UK, April 20-22, 2005, Revised Selected Papers*, chapter Insecure Real-World Authentication Protocols (or Why Phishing Is So Profitable), pages 89–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[2] Tyler Moore and Richard Clayton. *Financial Cryptography and Data Security: 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, chapter Evil Searching: Compromise and Recompromise of Internet Hosts for Phishing, pages 256–272. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[3] Anti-Phishing Working Group. Phishing activity trends report q1-q3 2015. `http://docs.apwg.org/reports/apwg_trends_report_q1-q3_2015.pdf`, December 2015. [Online; accessed 06-May-2016].

[4] Venture Beat. Google chrome now has over 1 billion users. `http://venturebeat.com/2015/05/28/google-chrome-now-has-over-1-billion-users/`, May 2015. [Online; accessed 06-May-2016].

[5] Rachna Dhamija, J Doug Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM, 2006.

[6] Lorrie Faith Cranor, Serge Egelman, Jason I Hong, and Yue Zhang. Phinding phish: An evaluation of anti-phishing toolbars.

[7] `http://news.biharprabha.com/2014/02/20-indians-are-victims-of-online-phishing-` [Online; accessed 04-June-2016].

[8] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *NDSS*, volume 10, 2010.

[9] `http://netcraft.com`. [Online; accessed 03-06-2016].

[10] `http://toolbar.netcraft.com/stats/reporters`. [Online; accessed 03-06-2016].

[11] Brad Wardman and Gary Warner. Automating phishing website identification through deep md5 matching. In *eCrime Researchers Summit, 2008*, pages 1–7. IEEE, 2008.

[12] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proceedings of the Anti-phishing Working Groups 2Nd Annual eCrime Researchers Summit*, eCrime '07, pages 60–69, New York, NY, USA, 2007. ACM.

[13] `http://news.netcraft.com/archives/2005/05/12/fraudsters_seek_to_make_phishing_sites_undetectable_by_content_filters.html`. [Online; accessed 04-June-2016].

[14] Evgeniy Gabrilovich and Alex Gontmakher. The homograph attack. *Communications of the ACM*, 45(2):128, 2002.

[15] S. Afroz and R. Greenstadt. Phishzoo: Detecting phishing websites by looking at them. In *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, pages 368–375, Sept 2011.

[16] David G. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, 1999.

[17] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[18] Stuart P Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[19] Dinei Florencio and Cormac Herley. Stopping a phishing attack, even when the victims ignore warnings. *Microsoft Research MSR-TR-2005*, 142, 2005.

[20] Wen-tau Yih, Joshua Goodman, and Geoff Hulten. Learning at low false positive rates. In *CEAS*, 2006.

[21] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[22] Wenyin Liu, Xiaotie Deng, Guanglin Huang, and A. Y. Fu. An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing*, 10(2):58–65, March 2006.

[23] Yin Liu, Wenyin Liu, and Changjun Jiang. *Advances in Web-Age Information Management: 5th International Conference, WAIM 2004, Dalian, China, July 15-17, 2004*, chapter User Interest Detection on Web Pages for Building Personalized Information Agent, pages 280–290. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[24] A.Y. Fu, Liu Wenyin, and Xiaotie Deng. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd). *Dependable and Secure Computing, IEEE Transactions on*, 3(4):301–311, Oct 2006.

[25] Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–220. IEEE, 2004.

[26] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[27] Teh-Chung Chen, Scott Dick, and James Miller. Detecting visually similar web pages: Application to phishing detection. *ACM Trans. Internet Technol.*, 10(2):5:1–5:38, June 2010.

[28] Ming Li and Paul Vitnyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag New York, 1997.

[29] R. Cilibrasi and P. M. B. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, April 2005.

[30] Michael Burrows and David Wheeler. A block-sorting lossless data compression algorithm. In *DIGITAL SRC RESEARCH REPORT*. Citeseer, 1994.

[31] Igor Pavlov. 7-zip compression utility, 2009.

[32] Kuan-Ta Chen, Chun-Rong Huang, Chu-Song Chen, and Jau-Yuan Chen. Fighting phishing with descriminative keypoint features. [Online; accessed 03-June-2016], 2009.

[33] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE, 2001.

[34] Chun-Rong Huang, Chu-Song Chen, and Pau-Choo Chung. Contrast context histograman efficient discriminating local descriptor for object recognition and image matching. *Pattern Recognition*, 41(10):3071–3077, 2008.

[35] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[36] `https://aws.amazon.com/`.

[37] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *ACM SIGOPS operating systems review*, volume 37, pages 29–43. ACM, 2003.

[38] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[39] N. Shrestha, R.K. Kharel, J. Britt, and R. Hasan. High-performance classification of phishing urls using a multi-modal approach with mapreduce. In *Services (SERVICES), 2015 IEEE World Congress on*, pages 206–212, June 2015.

[40] `https://storm.apache.org/`.

[41] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, pages 778–792, 2010.

[42] `http://www.tornadoweb.org/en/stable/`.

[43] `http://redis.io/`.

[44] `https://github.com/andymccurdy/redis-py`.

[45] `https://github.com/Parsely/streamparse/`.

[46] `http://opencv.org/`.

[47] `http://www.numpy.org/`.

[48] `http://scikit-learn.org/stable/`.

[49] `https://www.postgresql.org/`.

[50] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.

[51] `https://aws.amazon.com/ec2/pricing/`. [Online; accessed 08-June-16].