MSc System and Network Engineering

Master System and Network Engineering

University of Amsterdam

Bypassing Phishing Filters

Shahrukh Zaidi
shahrukh.zaidi@os3.nl

August 8, 2018

Supervisors: Alex Stavroulakis
Rick van Galen

Abstract

In recent years the threat of phishing attacks over email has seen a dramatic increase. Organisations and end users rely on spam filters for protection against these kind of attacks. The present study is designed to analyse the effectiveness of these filters in protecting end users against phishing emails and determine whether content obfuscation methods exist that may allow bypassing detection. In order to assess which aspects of a phishing email make these messages detectable, a set of phishing emails is analysed using two open source spam filters. The analysis reveals that the presence of certain suspicious words and URLs may block phishing emails from being delivered to the user's inbox. In order to bypass detection, two obfuscation methods have proven to be effective. In many cases the obfuscation of phishing-related words using Unicode transliteration and the replacement of suspicious URLs using URL shortening services appears to be sufficient in order to fool a spam filter. The analysis of these obfuscation methods on a number of popular email service providers shows that no defensive measures against these types of attacks appear to be in place.

Contents

1 Introduction

Over the past decades, email has become one of the major means of communication. Because of the prominent role email has taken in the present society, email security has become a key aspect of the modern Internet. 'Phishing emails' are a common type of email threat. Phishing emails are a special type of spam message that use fraudulent social engineering techniques to elicit sensitive information from unsuspected users[1]. While the main aim of a spam email is to contact a user and inform him about a certain product, a phishing email is specifically designed to appear unsuspicious and to come from a reputable source[5]. As phishing attacks exploit human weaknesses, mitigation of this type of attack is difficult[17]. To minimise the chance that a user becomes victim of a phishing attack, anti-spam tools include phishing detection solutions to prevent a malicious email from reaching the user's inbox.

The main aim of this study is to determine how effective these solutions are in protecting the end user against phishing attacks by analysing how easily certain aspects of a phishing email can be tweaked in order to fool these filters. The main research question of this study is defined as follows:

Which aspects of a phishing email can be modified in order to bypass common phishing filters?

In this study we will limit ourselves to strictly the content-specific aspects that characterise phishing emails. To answer the main research question, a number of sub-questions are defined:

- What are common characteristics of phishing emails?
- What detection techniques are commonly utilised by phishing filters?
- What methods can be deployed to bypass these detection techniques?

The remainder of this study is structured as follows: Section 2 presents a brief overview of the theoretical background concerning phishing emails. A number of common features that characterise phishing emails are described and phishing email detection techniques are discussed. Section 3 discusses various obfuscation techniques that may be used to obscure the contents of an email. Section 4 describes the methods and experiments that are used to answer our research question. A detailed overview of the experiments and results is given in Section 5 and the findings are discussed in Section 6. Finally, we conclude the research in Section 7 and suggest some possible directions for future work.

2 Theoretical framework

2.1 Phishing email characteristics

As phishing emails are likely to have the same ulterior goal, namely tricking users into disclosing confidential information, most phishing emails have similar characteristics in terms of language, layout, and structure of the email[8]. For example, phishing emails are likely to contain specific phrases asking users to visit a certain URL and to enter personal information. The basic characteristics of a phishing email can be broadly divided into four groups of features[27]: structural features, link features, element features, and word-list features. Structural features reflect the body part structure of an email. Link features reflect various properties of links contained in an email. Element features reflect the kind of web technologies that are used. Finally, the word-list features consist of a list of words that hint at the possibility of a phishing email.

Fette *et al.*[10] have defined a number of features that commonly characterise phishing emails. A few of these features are as follows:

- **IP-based URLs:** As phishing attacks may be hosted off of compromised PCs, these machines are not likely to have DNS entries. The simplest way to refer to them is by IP address.
- **'Fresh' linked-to domain names:** To better hide suspicious links, phishers use name-based attacks, in which a phisher will register a similar or otherwise legitimate-sounding domain name. As these malicious domains are reported and blacklisted frequently, these domain names are usually used for a limited time frame. Therefore, often the domains used in phishing URLs will be relatively newly registered.
- **Nonmatching URLs:** Phishers often exploit HTML emails. Using HTML tags attackers are able to hide malicious links in non-harmful looking elements. An example of such a link could look as follows: `<a href="badsite.com"> paypal.com </a>`.
- **'Here' linking to non-modal domain:** To keep the phishing email look authentic, phishers will often include a large number of legitimate links. These might be links to the privacy policy and user agreement of the imitated domain. The domain that is most frequently linked to is called the "modal domain". In phishing emails, often the most predominantly displayed link (often containing a text like "Click here to ...") will link to a domain other than the modal domain.
- **Large number of dots in URLs:** Phishers make use of subdomains to make a link look legitimate. These subdomains make the link have an inordinately large number of dots in the URL.

Almomani *et al.*[2] and Basnet *et al.*[4] have added a number of characteristics to this list. First of all, phishing emails are likely to have a disparity between the domain names in the body of the email and the sender's domain, i.e. the domain name referred to by the "From" field in the SMTP envelope. Secondly, phishing emails contain a number of frequently repeated keywords that urge the user to take some action. These keywords can be split into six different groups:

1. Update, Confirm
2. User, Customer, Client
3. Suspend, Restrict, Hold

4. Verify, Account
5. Login, Username, Password
6. Social Security Number (SSN), Social Security

2.2 Phishing email detection techniques

At present most of the methods applied by email filters are designed to combat spam emails in general, as opposed to phishing emails specifically. However, the identification of phishing emails is different from spam classification[5]. As explained in Section 1, a spam email is used to advertise a certain product, while phishing emails are designed to imitate a certain domain. As a phishing email differs significantly in its properties and characteristics from a spam email, different features and techniques have to be employed in order to detect phishing emails.

Hajgude and Ragha[15] have subdivided common phishing detection techniques into three main groups: blacklisting, whitelisting, and heuristics. When making use of blacklists, a spam filter matches a URL against a list of known phishing websites and DNS addresses of servers that have been reported to send spam in the past. The effectiveness of blacklists depends on the time it takes for a phishing URL to be included. Because there is usually some time before a website is reported and added to the blacklist, the effectiveness of blacklisting is limited.

In contrast to blacklisting techniques, white list approaches compare external links in incoming emails against a list of 'good' URLs[6]. However, as maintaining a list of trustworthy sources can be a time-consuming and labour-intensive task, this approach may not be the most efficient either.

Heuristic or content-based filtering approaches concentrate on detecting phishing attacks by evaluating the contents of an email and trying to identify the presence of various tricks for producing a plausible phishing attack[6]. This evaluation may comprise the detection of typical formulations urging the user to enter confidential information, the identification of spoofed sender addresses and URLs, the detection of invisible content inserted to fool filters, and the analysis of images included in the email.

Many modern phishing detection schemes apply machine learning techniques in order to classify spam. A commonly used filtering framework is the Bayesian classification. Bayesian spam filters attempt to calculate the probability that a message is spam based on previous feature frequencies in spam and legitimate email[32]. Particular words have certain probabilities of occurring in spam mail versus in legitimate mail[19]. The Bayesian filter can be trained to adjust the probability that a word appears in of these mails based on input from the user. By continually training the filter with spam and non-spam emails, the filter can become robust and highly accurate.

Typically a spam filter will combine evidence from many features and assign a 'score' to each feature that may be an indicator of phishing. If the total score exceeds a predefined threshold, the email is classified as phishing.

3 Related work

Spam fighting has often been described as an arms race characterised by an increase in the sophistication adopted by both spammers and spam filters[14]. As spam filters become more efficient in identifying a suspicious email, spammers are forced to find new ways to ensure delivery of their message. As modern-day spam filters are able to analyse the contents of a message to detect suspicious elements, simply not being listed on a blacklist does not ensure that a spammer's email will reach the victim's inbox. To bypass content-based spam filters, obfuscation of email contents has become an important tool for spammers.

Wittel and Wu[32] have grouped the evasion techniques that spammers use into a number of categories. *Tokenization* is used to work against the feature selection of a message by splitting or modifying key message features. *Obfuscation* is applied to obscure the contents of a message from the filter using encoding or misdirection. Statistical evasion approaches are used to attack statistical spam filters, for example the previously discussed Bayesian filter (see Section 2.2). *Weak statistical* evasion techniques attempt to skew the statistics of the message in such a way that the filter has trouble telling whether a message is spam or not. The technique is considered weak when purely randomised data insertion is used to evade detection. A *strong statistical* attack, however, applies a more focused approach. For example by taking advantage of feedback methods to determine which spam messages have reached the user's mailbox.

Lininger and Vines[18] have described a number of approaches to evade content-based filters. The first method uses HTML tricks to break up words in such a way that the final text appears identical to the user, but different to the filter. A common way to achieve this is by inserting fake HTML tags in the middle of a word. If a tag is opened and then closed without any content, nothing will happen to the text. For example, `acc<i></i>ount` and `account` will appear exactly the same to the user, but might not to the filter. A similar technique uses spaces to split words in order to avoid triggering the detection of a specific keyword. As this might look suspicious to the user, HTML properties can be utilised to make the space indiscernible (e.g. `acc<font size="0"> </font>ount`). Another technique that is described aims to hide the message from a filter entirely by sending the email as an image. The message can be a single image or consist of several smaller images stitched together.

One method to obfuscate message contents is by applying encoding tricks. An example of this method is Unicode transliteration[19]. Unicode was designed to represent almost all standard characters in the world. However, this large number of characters brings a new type of problem. As Unicode contains a large number of similar characters, it may be hard to visually differentiate between characters that are technically different. For example, the cyrillic 'a' (U+0430) may replace the latin 'a' (U+0061) and the cyrillic 'i' (U+0456) may replace the latin 'i' (U+0069) to get a technically different, yet visually identical word. Unicode obfuscation can be a valuable tool for spammers as it allows to create identical clones of blacklisted keywords in order to bypass a filter. Fu *et al.*[11] have introduced the UC-Simlist, a list containing similarly looking Unicode characters along with a similarity value between 0.8 and 1, with 1 being visually identical.

Sørensen and Larsen[28] have described two more obfuscation methods: scrambling and misspelling. The scrambling technique distorts words in such a way that the first and last character of a word are left unchanged, but the position of characters in between are modified randomly. The misspelling technique intentionally misspells certain keywords in order for these words not to be recognised by spam filters.

As described in Section 2.1, many characteristics of phishing emails are URL-related. Therefore, URL obfuscation can be an important tool for spammers to avoid detection. One method that phishers have embraced in order to hide malicious URLs, is the use of URL shortening services[23]. URL shorteners are used to obfuscate long URLs by replacing them with short redirected links. Many phishing filters depend on URL blacklists. Even though these may be effective to some extent, the use of URL shortening services may allow the phishers to evade these blacklists and thereby avoid detection[9].

4 Methodology

In order to more accurately assess how spam filters classify phishing emails, we need to test which elements of a phishing email are caught by these filters as being suspicious. As many enterprise email service providers make use of proprietary software, analysing phishing emails using these services may be challenging. For this reason we will limit ourselves to open source solutions for this preliminary analysis. In particular, we will focus on two widely used open source spam filters: SpamAssassin[29] and Rspamd[31].

In order to assess which elements of a phishing email are recognised by a spam filter as being suspicious, a data set containing more than 300 phishing emails will be used. This data set is obtained from Nazario's *PhishingCorpus*[22]. The data set contains the emails in `mbox` format. A Python script is written that iterates over the messages in the mailbox and feeds each email to both SpamAssassin and Rspamd to determine which rules are triggered when the email contents are analysed by the respective filters. The resulting spam reports are saved and can then be analysed in order to find the most frequently triggered rules.

Once the frequently triggered rules are determined, we assess which of the techniques described in Section 3 may be used to obfuscate the email contents in such a way that this trigger is prevented. To analyse the effectiveness of these techniques on popular email providers, we will apply the obfuscation methods to a subset of the test data and determine whether this application avoids detection by the spam filters of these mail providers. The mail service providers that we use for this analysis are the following:

- ProtonMail[24]
- Office 365[20]
- G Suite Gmail[13]
- Amazon WorkMail[3]
- RackSpace Email[25]

5 Results

5.1 Analysis of phishing emails

As described in Section 4, we have used a test data set containing more than 300 phishing emails as input to the SpamAssassin and Rspamd spam filters. The spam reports are then inspected in order to determine which rules are most frequently triggered when a phishing email is analysed. Table 1 and Table 2 below give an overview of these rules. Note that both filters are used with the default settings with no additional plugins. Therefore the rules listed in these tables are included in the default installations.

Table 1: SpamAssassin: frequently triggered rules

Rule	*Description*
MIME_HTML_ONLY	Messages that have only HTML part
RDNS_NONE	Delivered to internal network by host with no reverse DNS lookup
TVD_PH_BODY_ACCOUNTS_PRE	The body matches phrases such as 'accounts'
FREEMAIL_FORGED_REPLYTO	Freemail in Reply-To, but not From
SUBJ_ALL_CAPS	All capital letters in subject
HEADER_FROM_DIFFERENT_DOMAINS	From and EnvelopeFrom different
ACCT_PHISHING	Body matches phrases such as 'access', 'suspended', 'verify', 'restore'
URI_WPADMIN	WordPress login/admin URI, possible phishing

Table 2: Rspamd: frequently triggered rules

Rule	*Description*
MIME_HTML_ONLY	Messages that have only HTML part
FROM_NEQ_ENVFROM	From address is different to the envelope
HAS_ATTACHMENT	Contains attachment
HAS_WP_URI	Contains WordPress URIs
FREEMAIL_REPLYTO	Freemail in Reply-To, but not From
PHISHING	Non matching URLs in HTML text and `href`
RSPAMD_URIBL	URL in URIBL.com blacklist
HFILTER_FROMHOST_NORES_A_OR_MX	FROM host no resolve to A or MX

The frequently triggered rules by SpamAssassin and Rspamd can be split into three different groups. The first group contains those rules which can be relatively easily prevented from being triggered. Examples of these rules are `MIME_HTML_ONLY`, `SUBJ_ALL_CAPS`, and `RDNS_NONE/HFILTER_FROMHOST_NORES_A_OR_MX`. Triggering these rules could be prevented by sending the email in both HTML and plain text, using a subject line that is less suspicious and using an email domain that can be resolved.

The second group consists of those rules that indicate some issue with the headers of the email; for example, a discrepancy between the address in the message `FROM` field and the `FROM` field in the SMTP envelope or the use of free email service providers. As the `FROM` field in the SMTP envelope can not be spoofed, mitigating the trigger of this rule is as good as impossible.

The final group of rules are related to the actual contents of an email. These rules are highlighted in the tables as these are the rules we focus on in our bypass attempts. Firstly, the filters appear to analyse the contents of emails on the presence of suspicious phrases, as indicated by the `TVD_PH_BODY_ACCOUNTS_PRE` and `ACCT_PHISHING` rules. Secondly, URL analysis is performed to match URLs against certain URL blacklists. Moreover, WordPress URLs are marked as suspicious as well. Finally, the `PHISHING` rule indicates that non-matching URLs in the HTML text and `href` link are flagged as being suspicious. However, this rule can be easily prevented from being triggered by not putting URLs in plain text, but using phrases instead (e.g. "Click Here"). Therefore, we will disregard this rule in the evasion attempts.

5.2 Obfuscation of email contents

In order to assess the effectiveness of some of the obfuscation techniques described in Section 3, we use a sample phishing email that is classified as spam. As is apparent from Table 2, Rspamd does not appear to scan the contents of an email in order to detect phishing-related phrases. The phishing-related rules triggered are specific to URLs only. As SpamAssassin appears to detect both suspicious URLs and phrases, we strictly focus on this spam filter for this preliminary analysis. Listing 1 below shows a truncated output of the spam report for the phishing email, where `capitalone.txt` contains the HTML contents of the mail (see Appendix A for a brief description of this email).

```
$ spamc -R < tests/capitalone.txt
Content analysis details:   (5.4 points, 5.0 required)

 pts rule name
---- ----------------------
 1.6 SPOOF_COM2COM
 0.0 HTML_MESSAGE
 1.0 HTML_IMAGE_ONLY_16
 1.5 TVD_PH_BODY_ACCOUNTS_PRE
 0.0 T_DKIM_INVALID
 0.1 MISSING_MID
 1.0 ACCT_PHISHING
 0.0 T_REMOTE_IMAGE
```

Listing 1: Spam report of original sample phishing email

It can be seen from this output that SpamAssassin has detected the presence of various suspicious phrases (as indicated by the `TVD_PH_BODY_ACCOUNTS_PRE` and `ACCT_PHISHING` rules), and a malformed URL (`SPOOF_COM2COM`, URI containing '.com' in the middle and end). In order to obfuscate certain phrases in the email contents that can be considered suspicious, we have first attempted to insert fake HTML tags in the middle of any phishing-related word. This is done by using an opening tag immediately followed by a closing one, e.g. `account` becomes `acc<i></i>ount`. It appears that SpamAssassin is resilient to this kind of attack as applying this technique triggers the `HTML_OBFUSCATE_05_10` rule and the phishing-related phrases are still recognised, as shown in Listing 2 below:

```
$ spamc -R < tests/capitalone_obf_tag.txt
Content analysis details:   (5.4 points, 5.0 required)

 pts rule name
---- ----------------------
 1.6 SPOOF_COM2COM
 0.0 HTML_OBFUSCATE_05_10
 0.0 HTML_MESSAGE
 1.0 HTML_IMAGE_ONLY_16
 1.5 TVD_PH_BODY_ACCOUNTS_PRE
 0.0 T_DKIM_INVALID
 0.1 MISSING_MID
 1.0 ACCT_PHISHING
 0.0 T_REMOTE_IMAGE
```

Listing 2: Spam report of sample phishing email with fake HTML tag insertion applied

The second experiment involves the obfuscation of the aforementioned phrases using the Unicode obfuscation technique described in Section 3. For each phishing-related word we replace all vowels that have an identical Unicode character. After applying the obfuscation, it appears that the filter is not able to recognise the phrases as being phishing-related and the corresponding rules are not triggered:

```
$ spamc -R < tests/capitalone_obf_unicode.txt
Content analysis details:   (2.8 points, 5.0 required)

 pts rule name
---- ----------------------
 1.6 SPOOF_COM2COM
 0.0 HTML_MESSAGE
 1.0 HTML_IMAGE_ONLY_16
 0.0 T_DKIM_INVALID
 0.1 MISSING_MID
 0.0 T_REMOTE_IMAGE
```

Listing 3: Spam report of sample phishing email with Unicode obfuscation applied

To hide the presence of possible malicious URLs as well, we replace each `href` attribute in the email with a shortened URL using the Bitly[7] URL shortener. After replacing these URLs we observe that any URL-related rule is no longer triggered:

```
$ spamc -R < tests/capitalone_obf_unicode_url.txt
Content analysis details:   (1.1 points, 5.0 required)

 pts rule name
---- ----------------------
 0.0 HTML_MESSAGE
```

```
1.0 HTML_IMAGE_ONLY_16
0.0 T_DKIM_INVALID
0.1 MISSING_MID
0.0 T_REMOTE_IMAGE
```

Listing 4: Spam report of sample phishing email with Unicode obfuscation and URL shortening applied

As the Unicode obfuscation and URL shortening techniques appear to be sufficient in order to fool the phishing filter, we have written a simple script that applies these obfuscation methods to any phishing email. The script takes a file containing the HTML email as input. A second file containing common phishing words is loaded as well. Appendix B gives an overview of this list of phishing-related words. The list is compiled using a combination of sources (see [16][4]) and custom additions. For some of these words a 'root form' is used as different forms of the word may be present in a phishing email (e.g. restrict, restricted, or restriction). After loading these two files, we iterate over each element in the HTML in order to find all phishing-related words and `href` elements. Each phishing-related word that is encountered is replaced with a Unicode obfuscated version and the link for all `href` elements is shortened. The new email containing the obfuscated contents is finally written to a new file. For a detailed overview of the obfuscation script, we refer to Appendix C.

5.3 Analysis of obfuscation techniques

As mentioned in Section 4, we have analysed the effectiveness of the previously described obfuscation techniques on a total of five different hosted email providers. As four of these mail providers use closed-source enterprise software, it is challenging to assess what classification methods are applied by the spam filters and what rules are triggered. Therefore, for these email applications the only method that can be used for analysis is by determining whether an email is marked as spam or not. To still be able to make reasonable assumptions about the effectiveness of each of the obfuscation techniques, we have split the experiments for the enterprise mail solutions in three parts. Each test set strictly contains emails that are marked as spam by the email provider that is being analysed. In the first experiment we replace all URLs with a short URL and determine whether this obfuscated email is still classified as spam. The second experiment applies Unicode obfuscation as described in Section 5.2, and performs the same check. Finally, we apply the combination of the two techniques and observe whether the filter is bypassed.

For analysis of the obfuscation techniques, we have selected a subset of emails from the test data set. The criteria for selecting emails from this data set are as follows: firstly, the email should contain both URLs and possibly suspicious phrases. Secondly, the email should be classified as spam by the spam filter that is being analysed. As these criteria may not hold true for all phishing emails in the data set, we have manually selected a relatively small subset of test emails which meet the aforementioned requirements. Important to note is that the same email will not in all cases be recognised as spam by all spam filters. Therefore, the experiments for each analysed email provider may not necessarily contain the same emails. A short description of each email that is used for analysis may be found in Appendix A.

5.3.1 ProtonMail

As mentioned in Section 5.1, the analysis performed on the test data set of phishing emails was done using the default configuration of both SpamAssassin and Rspamd with no additional plugins. To assess whether we observe a difference in the detection of the obfuscation techniques between an out-of-the-box filter configuration and a more tweaked and hardened SpamAssassin implementation, we perform the initial

analysis on the ProtonMail email service. ProtonMail[24] is an open source, privacy-focused email service provider. To filter unwanted spam emails ProtonMail makes use of the SpamAssassin spam filter. One advantage of this open source approach is that we can easily analyse which spam rules were triggered and why a certain email was labeled as spam. Table 3 below shows the outcome of the spam analysis on a set of sample phishing emails:

Table 3: ProtonMail: effectiveness of obfuscation techniques

Sample phishing email	*Phishing related rules triggered using original phishing email*	*Phishing related rules triggered after obfuscation techniques applied*
bitstamp	URI_WPADMIN (Spam score: 3.0)	~~URI_WPADMIN~~ (Spam score: 0.2)
capitalone	SPOOF_COM2COM TVD_PH_BODY_ACCOUNTS_PRE (Spam score: 3.5)	~~SPOOF_COM2COM~~ ~~TVD_PH_BODY_ACCOUNTS_PRE~~ (Spam score: 1.5)
dhl	URIBL_PH_SURBL_PQS RAZOR2_CHECK (Spam score: 9.8)	~~URIBL_PH_SURBL_PQS~~ ~~RAZOR2_CHECK~~ (Spam score: -0.1)
fedex	URI_WPADMIN TVD_PH_BODY_ACCOUNTS_PRE (Spam score: 4.6)	~~URI_WPADMIN~~ ~~TVD_PH_BODY_ACCOUNTS_PRE~~ (Spam score: 1.8)

Important to note is that we have strictly included the rules that are specifically related to phishing. It is apparent from this table that the application of both the obfuscation techniques is undetected by the ProtonMail implementation of SpamAssassin. Any URL-related or content-specific rule is avoided from being triggered as URLs are replaced with a short URL and any phishing-related word is obfuscated using Unicode transliteration. This avoidance ultimately leads to a lower spam score. Particularly interesting is the third experiment, where the spam score is reduced by almost 10 points.

5.3.2 Office 365

Office 365 is a collection of services offered by Microsoft as part of their Microsoft Office product line[20]. These services include a hosted email infrastructure. Table 4 below shows the outcome of the spam analysis on a set of five sample phishing emails for an Office 365 mailbox. The tick and cross mark indicate whether the applied obfuscation technique was successful or unsuccessful in bypassing the spam filter respectively:

Table 4: Office 365: effectiveness of obfuscation techniques

Sample phishing email	*Short URL*	*Unicode obfuscation*	*Short URL + Unicode obfuscation*
`bitstamp`	✗	✓	✓
`capitalone`	✗	✗	✓
`dhl`	✓	✗	✓
`fedex`	✗	✗	✓
`dropbox`	✗	✗	✗

As can be seen from the table, in four out of the five experiments classification as spam is prevented by applying obfuscation techniques. The results of the analysis show that in some cases applying either of the two techniques may be sufficient to bypass detection (see phishing email `bitstamp` and phishing email `dhl`). In other cases the combination of the techniques is needed to successfully circumvent the phishing filter (see phishing email `capitalone` and phishing email `fedex`). A possible explanation for this might be that a certain predefined threshold score is still exceeded when only one of the methods is applied.

In order to assess whether we observe different results when a more tweaked and modified version of Office 365 is used, we have performed the same analysis on the Office 365 infrastructure at a large volunteer company. As not all emails that are used in the previous experiment are marked as spam by the spam filter of this email environment, the test set is not identical. Table 5 below shows the outcome of this analysis:

Table 5: Office 365 large company: effectiveness of obfuscation techniques

Sample phishing email	*Short URL*	*Unicode obfuscation*	*Short URL + Unicode obfuscation*
`dhl`	✗	✗	✗
`fedex`	✗	✗	✓
`docusign`	✓	✗	✓
`netflix`	✗	✗	✓
`security_alert`	✓	✗	✓

The results resemble the analysis performed on the independent Office 365 mailbox to some extent. In contrast to this previous analysis, in none of the experiments strictly applying Unicode obfuscation appears to be sufficient to bypass detection. However, the outcome of the experiment using phishing emails `fedex` and `netflix` shows that Unicode obfuscation is not detected by the spam filter: applying this technique along with URL shortening makes the email look legitimate to the filter, while strictly applying the latter is not sufficient.

5.3.3 G Suite Gmail

G Suite[13] offers a collection of collaboration and productivity tools with businesses as main target group. A mail infrastructure is provided that uses the Gmail mail service. Table 6 below shows the outcome of the spam analysis on a set of five sample phishing emails for a G suite Gmail inbox:

Table 6: G Suite Gmail: effectiveness of obfuscation techniques

Sample phishing email	*Short URL*	*Unicode obfuscation*	*Short URL + Unicode obfuscation*
`bitstamp`	✗	✗	✓
`acc_terminate`	✓	✗	✓
`docusign`	✓	✗	✓
`dropbox`	✗	✗	✗
`bank_of_america`	✓	✗	✓

It is apparent from the table that Gmail is not resilient against the applied obfuscation techniques either. Interestingly, in most of the cases strictly replacing URLs with a short version appears to be adequate to bypass detection. This is a rather remarkable result as G suite Gmail has an option to identify links behind short URLs[12], which is enabled by default. The outcome of the experiment using phishing email `bitstamp` confirms that Gmail does not offer protection against Unicode obfuscation attacks either.

5.3.4 Amazon WorkMail

Amazon Workmail[3] is the managed business email service offered by Amazon. Table 7 below shows the outcome of the spam analysis on a set of five sample phishing emails using this mail service:

Table 7: Amazon WorkMail: effectiveness of obfuscation techniques

Sample phishing email	*Short URL*	*Unicode obfuscation*	*Short URL + Unicode obfuscation*
`bitstamp`	✓	✓	✓
`capitalone`	✗	✓	✓
`dhl`	✗	✗	✓
`fedex`	✗	✗	✓
`dropbox`	✗	✗	✗

From the table above we can see that detection of phishing emails can be avoided by applying obfuscation techniques in four out of the five experiments. Experiments using phishing emails `dhl` and `fedex` indicate that there is some scoring mechanism used: the combined application of the obfuscation techniques allows to bypass the spam filter, while applying these obfuscation methods separately is unsuccessful.

5.3.5 Rackspace Email

Rackspace[26] is a managed cloud computing company that also offers email hosting for small businesses. Table 8 below shows the outcome of the spam analysis on a set of sample phishing emails for a Rackspace mailbox:

Table 8: Rackspace Email: effectiveness of obfuscation techniques

Sample phishing email	*Short URL*	*Unicode obfuscation*	*Short URL + Unicode obfuscation*
`acc_term`	✓	✗	✓
`blacklist`	✗	✗	✗
`alibaba`	✓	✗	✓

What stands out in this table is that only three test phishing emails have been used for analysis. The reason behind this limitation is the fact that Rackspace's filtering mechanisms seem to be lacking in comparison with the other tested mail services. As we were not able to find more emails that Rackspace classified as spam, the analysis of this mail service was problematic. However, based on the results of this preliminary analysis it appears that Rackspace does not protect against attacks that make use of URL obfuscation using short URLs. Because of the small size of the test data set no definitive statement can be made about the effectiveness of the Unicode obfuscation technique.

6 Discussion

The present study was designed to analyse the effectiveness of spam filters in protecting end users against phishing attacks over email and determine whether content obfuscation methods exist that may allow bypassing detection. In order to answer this question, we have first identified a number of phishing email characteristics and the methods that spam filters typically employ in order to classify phishing emails. To get a deeper insight into the specific aspects of a phishing email that makes these messages detectable, a data set containing a large number of phishing emails is analysed using two open source spam filters in order to find the frequently triggered spam rules. The outcome of this analysis is used in an attempt to bypass the trigger of any content-specific rule.

The results of this analysis indicate that spam filters are likely to evaluate the contents of an incoming email for suspicious phrases and URLs. In order to obscure the occurrence of possibly suspicious words, two different obfuscation methods have been attempted on the SpamAssassin spam filter. The use of fake HTML tag insertion in these words appears to be unsuccessful as the words are still detected and an additional rule indicating the presence of HTML obfuscation is triggered. The second experiment assesses the effect of the application of Unicode obfuscation on the suspicious words. This evasion technique appears to be able to successfully prevent detection. To conceal the presence of malicious URLs, a final experiment is performed that replaces any HTML link with a short URL. The method proves to be sufficient to prevent the trigger of any URL-related spam rule.

To determine the effectiveness of the two successful evasion techniques on a number of popular email service providers, we have written a script that applies the two evasion techniques to any HTML email. A sample set of phishing emails that are classified as spam are obfuscated using this script and the effect of this obfuscation is analysed. The results of this analysis show that none of the analysed mail service providers offer protection against the two evasion techniques. In some cases strictly applying one of the two techniques may be sufficient to bypass detection. In other cases the combination of the techniques is needed in order to avoid detection. The application of the two techniques does not in all cases guarantee that the spam filter will classify a phishing email as legitimate. This may be due to the presence of other elements in the email that yield a high enough score to still mark the message as spam.

The fact that the described evasion techniques go unnoticed by the spam filters of major email service providers is rather remarkable. Especially when one considers that awareness of these obfuscation techniques is not recent and mitigation may be fairly simple. A simple list containing Unicode characters that are visually identical (e.g. the UC-Simlist, see Section 3) may be used to create different clones of suspicious words in order to match the contents of the email against a list that contains these clones. An even simpler approach would be to flag any character that is not common in the English language as being suspicious. The detection of malicious URLs hidden using URL shortening services may be trickier as there may be legitimate reasons to include short URLs in an email. However, simply identifying links behind a short URL may be sufficient to detect obscured links. Interestingly, Google claims to do exactly this[12]. However, in practice we are still able to hide malicious links easily by making use of a URL shortener.

It is important to bear in mind that this study has strictly focused on the obfuscation of email contents in order to bypass phishing detection. In reality there will be many more aspects that may be considered in the classification of incoming email. An organisation might, for example, block a certain email regardless of the contents if this mail is detected to be sent in bulk to the employees. SPF-[30] or DKIM-based[21] authentication errors may raise flags as well. Finally, spam filters that utilise machine learning approaches are likely to recognise the described obfuscation methods as being suspicious as more phishing emails are received that apply these techniques. As these aspects are outside the scope of this study, the findings of this study should be interpreted with caution.

7 Conclusion

In recent years the threat of phishing email attacks has grown significantly. Organisations and end users rely on spam filters for protection against these kind of attacks. The analysis of a set of phishing emails using open source spam filters has shown that spam filters typically employ content-analysis methods in order to find the presence of suspicious phrases and malicious URLs. The results of this research show that these two aspects of an email can be relatively easily obfuscated. In many cases the obfuscation of certain suspicious words using Unicode transliteration and the replacement of malicious URLs using short URLs is sufficient to prevent the email from being classified as phishing. As mitigation methods for these evasion techniques may be fairly simple, we conclude that spam filters have enough to improve on.

7.1 Future work

As this study strictly focused on content-specific aspects of phishing emails, the generalisability of these results is subject to certain limitations. As mentioned in Section 6, there are likely to be more aspects that may be considered in the classification of incoming emails. A further study that takes these aspects into account is recommended. A possible direction for future work could be the evaluation of the effectiveness of the obfuscation techniques discussed in this study when an obfuscated email is sent to an organisation in large numbers.

References

[1] Shivam Aggarwal, Vishal Kumar, and SD Sudarsan. Identification and detection of phishing emails using natural language processing techniques. In *Proceedings of the 7th International Conference on Security of Information and Networks*, page 217. ACM, 2014.

[2] Ammar Almomani, Tat-Chee Wan, Altyeb Altaher, Ahmad Manasrah, Eman ALmomani, Mohammed Anbar, Esraa ALomari, and Sureswaran Ramadass. Evolving fuzzy neural network for phishing emails detection. *Journal of Computer Science*, 8(7):1099, 2012.

[3] Amazon Web Services, Inc. Amazon WorkMail - Managed Email for Business - Amazon AWS. `https://aws.amazon.com/workmail/`, 2018. [Online; accessed 26-June-2018].

[4] Ram Basnet, Srinivas Mukkamala, and Andrew H Sung. Detection of phishing attacks: A machine learning approach. In *Soft Computing Applications in Industry*, pages 373–383. Springer, 2008.

[5] Andre Bergholz, Jeong Ho Chang, Gerhard Paass, Frank Reichartz, and Siehyun Strobel. Improved phishing detection using model-based features. In *The Fifth Conference on Email and Anti-Spam (CEAS)*, 2008.

[6] André Bergholz, Jan De Beer, Sebastian Glahn, Marie-Francine Moens, Gerhard Paaß, and Siehyun Strobel. New filtering approaches for phishing email. *Journal of computer security*, 18(1):7–35, 2010.

[7] Bitly Inc. Bitly — URL Shortener, Custom Branded URLs, API Link Management. `https://bitly.com/`, 2018. [Online; accessed 25-June-2018].

[8] Madhusudhanan Chandrasekaran, Krishnan Narayanan, and Shambhu Upadhyaya. Phishing email detection based on structural properties. In *NYS Cyber Security Conference*, volume 3, 2006.

[9] Sidharth Chhabra, Anupama Aggarwal, Fabricio Benevenuto, and Ponnurangam Kumaraguru. Phi.sh/$ocial: the phishing landscape through short urls. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, pages 92–101. ACM, 2011.

[10] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web*, pages 649–656. ACM, 2007.

[11] Anthony Y Fu, Wan Zhang, Xiaotie Deng, and Liu Wenyin. Safeguard against unicode attacks: generation and applications of uc-simlist. In *Proceedings of the 15th international conference on World Wide Web*, pages 917–918. ACM, 2006.

[12] Google LLC. Apply extra phishing and malware protection. `https://support.google.com/a/answer/7577854?hl=en`, 2018. [Online; accessed 28-June-2018].

[13] Google LLC. Google G Suite. `https://gsuite.google.com`, 2018. [Online; accessed 26-June-2018].

[14] Pedro H Calais Guerra, Dorgival Guedes, J Wagner Meira, Cristine Hoepers, MHPC Chaves, and Klaus Steding-Jessen. Exploring the spam arms race to characterize spam evolution. In *Proceedings of the 7th Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS), Redmond, WA*, 2010.

[15] Jayshree Hajgude and Lata Ragha. Phish mail guard: Phishing mail detection technique by using textual and url analysis. In *Information and Communication Technologies (WICT), 2012 World Congress on*, pages 297–302. IEEE, 2012.

[16] Kevin McCaney. The 20 most common words in phishing attacks. https://gcn.com/articles/2012/09/26/20-most-common-words-phishing-attacks.aspx, 2018. [Online; accessed 29-June-2018].

[17] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.

[18] Rachael Lininger and Russell Dean Vines. *Phishing: Cutting the identity theft line*. John Wiley & Sons, 2005.

[19] Changwei Liu and Sid Stamm. Fighting unicode-obfuscated spam. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 45–59. ACM, 2007.

[20] Microsoft. Microsoft Office 365. https://www.office.com/, 2018. [Online; accessed 26-June-2018].

[21] Mutual Internet Practices Association. DomainKeys Identified Mail (DKIM) . Available at http://dkim.org/ (Accessed on August 7, 2018).

[22] Jose Nazario. The online PhishingCorpus. https://monkey.org/~jose/phishing/, 2018. [Online; accessed 13-June-2018].

[23] New York University. Recent Uptick in Phishing Messages Using URL Shorteners. https://wp.nyu.edu/itsecurity/2018/02/08/recent-uptick-in-phishing-messages-using-url-shorteners/, 2018. [Online; accessed 16-June-2018].

[24] Proton Technologies AG. Secure email: ProtonMail is free encrypted email. https://protonmail.com/, 2018. [Online; accessed 26-June-2018].

[25] Rackspace US, Inc. Rackspace — Affordable Email Hosting for Your Small Business. https://www.rackspace.com/email-hosting/webmail, 2018. [Online; accessed 26-June-2018].

[26] Rackspace US, Inc. Rackspace: Managed Dedicated Cloud Computing Services. https://www.rackspace.com/, 2018. [Online; accessed 28-June-2018].

[27] Jasveer Singh. Detection of phishing e-mail. *International Journal of Computer Science and Technology (IJCST)*, 2(1), 2011.

[28] Lars Tabro Sørensen and Martin Møller Larsen. Wordadjust-a deobfuscation frontend to content-aware anti-spam tools. 2008.

[29] The Apache Software Foundation. SpamAssassin, 2018. Available at https://github.com/apache/spamassassin Commit: e4ae3e2f8a9919dad5446877f1c9451294f94ed8 (Accessed on June 4, 2018).

[30] The SPF Project. Sender Policy Framework Project Overview. Available at http://www.openspf.org/ (Accessed on August 7, 2018).

[31] Vsevolod Stakhov. Rspamd, 2018. Available at https://github.com/vstakhov/rspamd Commit: ae53944eec8b20ce389712de5d1cc10b5e6c35bf (Accessed on June 4, 2018).

[32] Gregory L Wittel and Shyhtsun Felix Wu. On attacking statistical spam filters. In *First Conference on Email and Anti-Spam (CEAS)*, 2004.

A Description of test emails

acc_terminate A typical phishing email that pretends to come from the mail service provider. The message indicates that a request to terminate the mail account was received. To avoid termination of the account, the user is urged to click on the contained link.

alibaba An email that appears to originate from Alibaba.com. The user is requested to reply on an offer that is received for a certain product by clicking on the "Reply" link.

bank_of_america A phishing email that pretends to come from the Bank of America. The user is informed about a payee that has been added to the user's account. To confirm the user is asked to complete the verification process by logging in by clicking on the provided link.

bitstamp The current email appears to originate from the Bitstamp cryptocurrency market place. The email urges the user to update his account information in order to prevent suspension of the account. The "Update Account" link appears to refer to a Wordpress interface.

blacklist This email informs the user that the email account has been blacklisted because of spam activities. Failure to confirm and update the account within 24 hours will lead to permanent suspension. To confirm the email account, the user is asked to click on a link.

capitalone A phishing email that pretends to come from the Capital One Bank security department. The user is noticed about the temporary suspension of the user's bank account. To unlock the account, the user is requested to log in by clicking the provided link.

dhl In this email the user is requested to complete his tracking process for a DHL delivery. The user is asked to visit a link that appears to contain a reference to a malicious URL, instead of the DHL tracking page.

docusign An email that appears to be received from DocuSign. The user is requested to click the "Activate" button in order to verify the email address and complete the account registration process.

dropbox This emails pretends to come from the Dropbox service team. The user is informed about a "payment confirmation" document that has been uploaded. The file can be viewed by clicking on the included button.

fedex An email that pretends to be sent by the FedEx customer service. The user is noticed about the suspension of all unused accounts. To avoid suspension the user is advised to update his account information. A "Sign in" link is included.

netflix A phishing email that appears to come from the Netflix support team. The user is noticed about the failed validation of the payment information. To verify the billing and payment details, the user is requested to click on the included link. Failure of completing this validation process will lead to suspension of the Netflix membership.

security_alert This is another mail that pretends to come from the email service provider. The mail informs the user that the user's identity can no longer be verified. To avoid loss of the account, the user is requested to verify his identity by clicking the included button.

B List of common phishing words

label	invoice	post	document	postal
calculation	copy	fedex	statement	financ
dhl	usps	notif	irs	ups
deliver	ticket	updat	user	password
account	secur	verif	confirm	customer
client	suspen	restrict	hold	log
safe	request	expir	cancel	immediate
urgent	click	link	spam	junk
require	unlock	bank	access	terminat
money	malicious			

C Obfuscation script

```
import os
import re
import bitly_api
from bs4 import BeautifulSoup as BSHTML

API_KEY = '************************'
TEST_MAIL = 'test_data/netflix'

def replace_letters(word):
        word = word.replace('a', '&#1072;')
        word = word.replace('i', '&#8560;')
        word = word.replace('e', '&#1077;')
        word = word.replace('o', '&#959;')
        word = word.replace('U', '&#5196;')
        word = word.replace('A', '&#1040;')
        word = word.replace('O', '&#927;')
        return word

def check_common(word, common):
        for elem in common:
                if elem in word:
                        return True
        return False

def shorten_url(url):
        try:
                b = bitly_api.Connection(access_token=API_KEY)
                resp = b.shorten(url)
                return resp['url']
        except:
                return url

# Open HTML email
with open(TEST_MAIL + '.txt', 'r') as f:
        data = f.read()

# Open file containing common phishing words
with open('common.txt', 'r') as f:
        common = []
        for line in f:
                common.extend(line.split())

# Find all text in HTML and add to list
```

```
html_bs = BSHTML(data, 'html.parser')
text_data = html_bs.find_all(text=True)
text_words = []

for elem in text_data:
        for word in elem.split():
                text_words.append(word)

try:
    os.remove(TEST_MAIL + '_obf.txt')
except OSError:
    print('File does not exist')

with open(TEST_MAIL + '_obf.txt', 'a') as f:
        for word in data.split():
                uni_word = unicode(word, 'utf-8', errors='ignore')
                if uni_word in text_words and check_common(word.lower(),
                   ↪ common):
                        # Apply unicode obfuscation
                        word = replace_letters(word)

                if 'href=' in word:
                        # Replace links with short URL
                        link = re.search('href="(.*)"', word)
                        short_link = shorten_url(link.group(1))
                        word = word.replace(link.group(1), short_link, 1)

                if 'charset' in word:
                        # Modiy encoding
                        charset = re.search('charset=(.*)"', word)
                        word = word.replace(charset.group(1), 'utf-8')

                f.write(word)
                f.write(" ")
```