Abstract

Over the past years, phishing has been an increasing threat to end users and organizations. Spam filters form the first line of defense against phishers, aiming at preventing spoofed emails from reaching the end user’s inbox. The present study is designed to assess the effectiveness of these filters against phishing emails and determine whether techniques exist that may be used to avoid detection. In order to determine what network and authentication aspects of phishing emails allow for detection, a subset of phishing emails has been analysed using two popular open source spam filters. After experimenting with the spam filters, being compliant has proven to be an effective method to bypass detection. In most cases, sending authenticated phishing emails using a newly registered domain appears to be sufficient in order to avoid detection and allow phishing emails to reach the end user’s inbox. The analysis of this method on some popular email service providers reveals that no effective protections against this method appear to be in place. Finally, some mitigation measures that may be used to protect the end user against such attacks are discussed and recommendations for future work are given.
# Contents

1 Introduction 
   1.1 Theoretical framework 
      1.1.1 Phishing email characteristics 
      1.1.2 Phishing email detection techniques 
   1.2 Related work 
3 Methodology 
4 Results 
   4.1 Analysis of phishing emails 
   4.2 Compliance as a solution 
   4.3 Authentication of emails 
      4.3.1 Without authentication 
      4.3.2 With authentication 
   4.4 Analysis of the solution 
      4.4.1 ProtonMail 
      4.4.2 Office 365 
      4.4.3 Gmail 
5 Discussion 
6 Conclusion 
   6.1 Future work 
Appendix A Phishing email 
Appendix B Email generation script
CHAPTER 1

Introduction

Since the early days of the Internet, email has become one of the most used means of communication by people and organisations. Because of the popularity of email in today’s society, email security is essential in order to guarantee integrity, confidentiality and authenticity of messages. Email phishing is currently one of the most problematic threats. A phishing email is an unsolicited email that attempts to remain undetected by exploiting human unawareness in order to steal sensitive information about a user. While spam emails are most frequently seen as unsolicited commercial messages, phishing emails require particular attention as they put users’ privacy at risk. By sending out emails that may be very similar to legitimate ones, attackers aim to harvest sensitive information, making users believe that they are communicating with a trusted entity. Because of the social engineering aspect of this attack, it is very challenging to mitigate phishing attempts[6]. To prevent phishing emails from reaching the end users’ inbox, spam filters are used as protection. These filters include features such as scanning and analysis of emails content, authentication of senders, policy validation and statistical analysis[18,22]. In addition to the use of spam filters, large email service providers perform filtering based on the sender’s reputation[21].

The aim of this study is to assess the efficiency of these filters as protection for the end user against phishing attacks, limiting ourselves to the network and authentication-specific aspects of phishing emails. The main research question of this study is defined as follows:

Which network and authentication aspects of phishing emails can be modified in order to bypass common spam filters?

In order to answer our main research question, a number of sub-questions have been defined:

• What network level protections and authentication mechanisms are commonly used to prevent phishing attacks?
• Which of these protections can be found in spam filters?
• How efficient are these solutions?
• How efficient is reputation-based email filtering?

The remainder of this study is structured as follows: Section[1.1] gives a brief overview of background information about phishing emails and existing detection techniques are discussed. Section[2] discusses various issues related to email authentication that may be exploited by attackers in order to send phishing emails. Section[3] describes the methods and the experiments that are used to answer our research question. Section[4] further discusses the experiments and presents the findings of this research. Finally, we conclude this study in Section[5] and give some recommendations for future work.
1.1 Theoretical framework

1.1.1 Phishing email characteristics

While content is a very important part of phishing emails detection, other aspects have to be considered in order to filter phishing emails more efficiently. Internet mail standards do not prevent users from modifying email headers, whether it is for illegitimate uses or legitimate uses, such as mail listings. The main goal of phishers is to impersonate a trusted entity by falsifying the address of origin of a message, also known as “spoofing”. By using this method, an attacker hopes to fool a target and acquire their trust to click on a link pointing to a malicious website. When the target clicks on the link, their information will be stolen, or some malicious content will be delivered.

1.1.2 Phishing email detection techniques

Early anti-spam solutions used “blacklisting” to detect and block spam emails, by performing lookups to a list of senders known for sending unsolicited emails. Although this method is still used today, nothing prevented senders from lying about their identity and therefore be considered as legitimate and trustworthy. New techniques related to the network and authentication aspects of phishing emails have been implemented in order to guarantee the authenticity of email senders. As Durumeric et al. discussed in their paper[5], we have seen SMTP security extensions like STARTTLS, SPF, DKIM and DMARC emerging in the early years of the 21st century to counter email spoofing. Short descriptions of these techniques can be given as follows:

- **DNSBL**: DNS Blacklist (DNSBL) is a mechanism used to stop email spamming by blacklisting IP addresses most often reputed to send email spam. DNSBLs use a wide array of criteria for listing and delisting IP addresses. Mail servers can then query DNSBLs for a specific domain name or IP address and depending on their configuration, use the answer to reject or flag incoming spam emails.

- **rDNS**: Reverse DNS lookups (rDNS) are used to determine the domain name associated with a given IP address. By means of a PTR record in the DNS zone, organisations can define which IP address is associated with their domain. By querying the DNS, spam filters determine if the PTR record matches the domain and IP address of an incoming email. If that is not the case, it is assumed that the email has been spoofed and spam filters may classify the email as spam.

- **SPF**: Sender Policy Framework (SPF) is a protocol used to validate email senders and help detect and block email spoofing. By means of a TXT record in its DNS zone, an organisation can publish a range of hosts authorized to send emails for its domain as well as an SPF policy. Mail servers will then check the host of an incoming message and send a DNS request to determine whether it belongs to this organisation or not.

- **DKIM**: DomainKeys Identified Mail (DKIM) is a technique used to prove the authenticity of an email by digitally signing some or all of its header fields using a key pair. The public key must be included by the sender in a TXT record of the domain zone. When the recipient will receive the message, their verifier will perform a DNS query for “selector_domainkey.domain”, in order to retrieve the key and validate a message’s digital signature.

- **DMARC**: Domain-based Message Authentication, Reporting and Conformance (DMARC) is used to validate emails in order to prevent email spoofing. It has been built on top of SPF and DKIM and allows the sender of a domain to publish a policy, defining how the receiver should deal with SPF and DKIM failures. It also provides a reporting mechanism when an action is taken under these policies.

- **STARTTLS**: STARTTLS refers to the SMTP command used to upgrade a plaintext connection into an encrypted communication channel, by encapsulating SMTP within a TLS (Transport Layer Security) session, thus providing confidentiality without the need to change ports.
Over the past few years, phishing attacks have become more sophisticated in response to increased countermeasures. Although many techniques exist to prevent phishing emails from reaching end users, phishing persists to be a threat to end users and organisations, and a real challenge for email service providers.

A key aspect of phishing emails that is thoroughly examined by spam filters is the email body. Shahrukh has discussed various content obfuscation techniques that may be used in order to avoid detection and classification of a phishing email as spam. The study revealed that applying basic evasion techniques such as Unicode obfuscation and URL shortening is enough to fool the SpamAssassin and Rspamd spam filters and bypass the protections of a number of popular email service providers.

Spoofing is another important aspect of phishing attacks, and plays a key role in the trustworthiness of phishing emails. Although several anti-spoofing techniques exist (see Section 1.1.2), Almomani et al. have shown that a lot of the email authentication protocols lack efficiency, are costly and too complex to be used in large environments or are simply not used. One of the main issues of these protocols does not come from its design, but rather from its non-adoption. Several studies have stated that these protocols are still not widely accepted by organisations. As these protocols are an important part of email authenticity verification for further classification by spam filters, its non-adoption greatly reduces the effectiveness of spam filters, thus increasing the chance of phishing emails reaching the end user.

The lack of strict DMARC policies as described by Hu et al. in their studies, is also a factor that helps phishing emails reaching the end user’s inbox. By defining a ‘relaxed’ or sometimes non-existent policy, a sender may put a receiving server in a difficult position, as no instruction to reject an email is given when SPF, DKIM or both, are failing.

Furthermore, as mail security heavily relies on DNS to verify email integrity and sender authenticity, attacking DNS may have a large impact on these techniques. Durumeric et al. discuss possible network attacks on emails, which include DNS Hijacking in order to spoof the DNS records of an organisation. Moreover, as described by Foster et al. in their study, the low rate of DNSSEC adoption is putting message integrity at risk and allows attackers to tamper the required DNS records. This would allow an attacker’s email to pass verifications performed by DNS Reverse Lookup, SPF, DKIM, and DMARC, therefore increasing the legitimacy of the attacker’s emails. In their research, Durumeric et al. also discuss network attacks on STARTTLS, and describe how downgrading an existing TLS session may allow for tampering of messages. A practical application of this attack may involve tampering of links included in a legitimate message with malicious links that redirect to a phishing website.
CHAPTER 3
Methodology

In order to assess how emails are classified as spam, we first need to determine which characteristics of phishing emails are examined by spam filters. Furthermore, we need to determine what detection techniques are implemented in spam filters. For this purpose, and due to the difficulties of experimenting with proprietary solutions, we will limit ourselves to experimenting with two largely used open source spam filters, namely Apache SpamAssassin 3.4.2\[3\] and Rspamd 1.8.1[19].

Our setup is composed of two machines, each one containing a mail server. One machine is used to send emails, the other to receive them, where they will later be processed by SpamAssassin and Rspamd. As our study is not focusing on the content of email bodies but rather on email headers, triggering of content-related rules during our experiments will not be discussed in this paper.

Based on Shahrukh’s research[23] and after experimenting with a data set of more than 300 emails in the mbox format from Jose Nazario’s 2017 phishing email list[16], it appeared that the rules triggered most frequently were the same, excluding rules that were content-specific. Once these frequently triggered rules are determined, we will experiment with the Craigslist email, extracted from Jose Nazario’s most recent phishing emails list (see Appendix A for a brief description of this email). The Craigslist email is a great example of typical email phishing for the following reasons:

- the sender’s email address has been spoofed
- the sender is impersonating a large company
- the spoofed domain is still active today
- the spoofed domain has SPF, DKIM and DMARC enabled
- the email body contains a malicious URL
- some of the email headers have been forged

After identifying which spam filter rules, other than those that are content-specific, are triggered by the email, we will assess what network and authentication mechanisms are used by spam filters to classify incoming emails. We will then research how to lower the score given to the email by the two spam filters, in order to avoid detection and classification as spam. Finally, after determining and implementing a solution to bypass the spam filters’ protections, we will test this solution against the spam filters of the following email service providers:

- ProtonMail[20]
- Office 365[14]
- Gmail[8]
4.1 Analysis of phishing emails

SpamAssassin and Rspamd both include two main components: a daemon process and a client binary. The daemon process can work directly with SMTP connections whereas the client binary takes an email in the \texttt{mbox} format as input, and generates a spam report. A typical report from a spam filter client includes a list of the rules triggered by an email as well as a short description and points assigned to each rule. The sum of these points is then calculated, establishing a score for the email. If the score is equal or greater than the threshold defined in the spam filter configuration, then the email is classified as spam. If the score is below the threshold, the email is considered as “ham”, or legitimate email.

As described in Section 3, we have used a data set of more than 300 original phishing emails as input to SpamAssassin and Rspamd, which generated a report for each email. A script has been created to feed each email from the data set to both spam filters to determine the most frequently triggered rules. Note that during our experiments, both spam filters have been used with the default configuration, default rules, and no extra plugins. Table 1 describes the most frequently triggered rules with SpamAssassin whereas Table 2 shows the most frequently triggered rules with Rspamd. The rules that are most relevant to our study are highlighted in gray.

### Table 1: SpamAssassin frequently triggered rules

<table>
<thead>
<tr>
<th>Points</th>
<th>Rule name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6</td>
<td>SPOOF_COM2COM</td>
<td>URI: URI contains \texttt{.com} in middle and end</td>
</tr>
<tr>
<td>1.6</td>
<td>SUBJ_ALL_CAPS</td>
<td>Subject is all capitals</td>
</tr>
<tr>
<td>1.0</td>
<td>ACCT_PHISHING</td>
<td>Possible phishing for account information</td>
</tr>
<tr>
<td>0.9</td>
<td>SPF_FAIL</td>
<td>SPF: sender does not match SPF record (fail)</td>
</tr>
<tr>
<td>0.1</td>
<td>DKIM_SIGNED</td>
<td>Message has a DKIM or DK signature, not necessarily valid</td>
</tr>
<tr>
<td>0.0</td>
<td>T_DKIM_INVALID</td>
<td>DKIM-Signature header exists but is not valid</td>
</tr>
<tr>
<td>0.0</td>
<td>T_REMOTE_IMAGE</td>
<td>Message contains an external image</td>
</tr>
<tr>
<td>0.0</td>
<td>HTML_MESSAGE</td>
<td>BODY: HTML included in message</td>
</tr>
<tr>
<td>1.3</td>
<td>RDNS_NONE</td>
<td>Delivered to internal network by a host with no rDNS</td>
</tr>
</tbody>
</table>

### Table 2: Rspamd frequently triggered rules

<table>
<thead>
<tr>
<th>Points</th>
<th>Rule name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>HFILTER_FROMHOST_NORES_A_OR_MX</td>
<td>From host no resolve to A or MX</td>
</tr>
<tr>
<td>1.22</td>
<td>PHISHING</td>
<td>Non matching URLs in HTML text and href</td>
</tr>
<tr>
<td>1.0</td>
<td>R_SPF_FAIL</td>
<td>Sender does not match SPF record</td>
</tr>
<tr>
<td>0.2</td>
<td>MIME_HTML_ONLY</td>
<td>Message is HTML only</td>
</tr>
<tr>
<td>0.1</td>
<td>RCVD_NO_TLS_LAST</td>
<td>Last hop does not use TLS</td>
</tr>
<tr>
<td>0.0</td>
<td>DMARC NA</td>
<td>No DMARC available</td>
</tr>
<tr>
<td>0.0</td>
<td>DKIM_TRACE</td>
<td>Message has a DKIM signature</td>
</tr>
<tr>
<td>0.0</td>
<td>R_DKIM_PERMFAIL</td>
<td>DKIM Signature is not valid</td>
</tr>
</tbody>
</table>
The first observation is that both spam filters check for SPF and DKIM. Rspamd checks for DMARC whereas SpamAssassin leaves no trace of any DMARC check, not even when run in debug mode. In this case, both spam filters are indicating that SPF is failing with the flags SPF_FAIL and R_SPF_FAIL, meaning the sending IP address is not allowed to send emails for this domain. The flags DKIM_SIGNED and DKIM_TRACE indicate that the message has been signed with DKIM, but the T_DKIM_INVALID and R_DKIM_PERMFAIL flags show that the digital signature is not valid.

A second observation is that as opposed to SpamAssassin, Rspamd makes sure that TLS is enabled between the last hop and the receiving end (RCVDO_NO_TLS_LAST). Rspamd shows us that the sending host does not resolve into any A or MX record by means of the HFILTER_FROMHOST_NORES_A_OR_MX flag whereas SpamAssassin does this by means of the RDNS_NONE flag. Both spam filters notice phishing URLs included in emails with the SPOOF_COM2COM and the PHISHING flags. Finally, SpamAssassin is flagging links pointing to remote images, which is not the case for Rspamd.

A first conclusion can be drawn by looking at the points attributed to each of the rules. By default, both spam filters assign a very low number of points to SPF, DKIM and DMARC failures. Moreover, the default threshold of Rspamd is 15, whereas SpamAssassin has a threshold of 5. While Rspamd assigns more points to its rules compared to SpamAssassin, most phishing emails in our data set scored much lower than the Rspamd threshold, which was not the case with SpamAssassin.

4.2 Compliance as a solution

As described in the Section 4.1, spam filters first perform a DNS lookup of the sending host, then check for the use of TLS, SPF, DKIM and DMARC. Even though the emails in our data set scored poorly according to Table 1 and Table 2 due to a low number of points being assigned to the related rules, it is known that most email providers use these mechanisms to filter spam more efficiently[5].

Studies[13] have shown that using STARTTLS, SPF, DKIM and DMARC can provide effective mitigation against spoofing and prevent spam very effectively. Therefore, it can be a nearly impossible challenge to bypass these protections. In this context, compliance may be an effective option to bypass spam filters. Using a domain that can be resolved, and complying by using the anti-spoofing protocols, appears to be a solution to let phishing emails be trusted by spam filters. Since the main goal of an attacker is to encourage users to click on a link, not only spam filters, but users have to trust the sender as well.

Suppose an attacker wants to impersonate a banking company A, and pretend to send emails for domain a.com. As seen in Section 4.1 SPF will effectively prevent this scenario from happening, and DKIM will not allow emails to be digitally signed and validated. To work around these protections, an attacker may use compliance to his advantage, as described in the following steps:

1. The attacker registers a domain named users-accounts.com, then defines a subdomain a, matching the name of the organisation
2. The attacker implements SPF, DKIM and DMARC for the domain users-accounts.com
3. The attacker enables TLS for the domain users-accounts.com, and always uses STARTTLS when sending out emails
4. Phishing emails are sent from user@a.users-accounts.com, with a username customizable by the attacker. The emails include images hosted by the attacker and phishing links pointing to the attacker’s website a.users-accounts.com

Using a different domain not only allows the attacker to implement SPF, DKIM and DMARC in order to lower the score applied by spam filters to phishing emails, but also allows the use of subdomains, in order to trick the user into believing that the sender is company A. Studies[1] show that organisations tend to rely too often on the weakest link of the chain: the users. This solution exploits this weakest link to bypass the anti-spoofing mechanisms used by spam filters.
4.3 Authentication of emails

To assess the effectiveness of the technique described in Section 4.2 as a solution to bypass spam filters, a highly automated proof of concept has been developed using scripting and containerization. This proof of concept includes three containerized applications: a DNS server to handle the domain and subdomains, a mail server to send emails out, and a web server to serve remote content included in the phishing emails, such as images. Moreover, TLS has been enabled on the system, STARTTLS has been enforced, and SPF, DKIM and DMARC have been implemented. Because of the sensitivity of this proof of concept, only the script responsible for the proper crafting of the emails has been disclosed in this paper (see Appendix B). Section 4.3.1 and 4.3.2 describe the experiments performed with SpamAssassin and Rspamd without and with the use of our proof of concept respectively.

4.3.1 Without authentication

For the first part of our experiments, we use the original Craigslist phishing email and give it as input to the spam filters. Listing 1 and 2 show the output of the reports generated by the spam filters:

Listing 1: SpamAssassin spam report of original phishing email

```bash
$ spamc -R < craigslist.txt
Content analysis details: (8.8 points, 5.0 required)

pts  rule analysis name
----- ----------------------
0.0  URIIBL_BLOCKED
0.0  FSL_CTYPE_WIN1251
0.9  SPF_FAIL
0.0  HTML_MESSAGE
1.1  MIME_HTML_ONLY
0.6  FORGED_OUTLOOK_TAGS
0.0  FSL_NEW_HELO_USER
0.0  FORGED_OUTLOOK_HTML
0.0  FORGED_OUTLOOK_HTML
0.0  TVD_PH_BODY_ACCOUNTS_POST
3.4  MSOE_MID_WRONG_CASE
0.0  AXB_XMAILER_MIMEOLE_OL_024C2
0.0  TVD_PH_BODY_META_ALL
2.8  FORGED_MUA_OUTLOOK
```

First, it can be seen from Listing 1 that the message contains a URL listed in the DNSBLs spam databases (URIBL_BLOCKED). Second, the flag FSL_CTYPE_WIN1251 indicates the presence of a charset type that can be found in advance-fee scam emails. This SpamAssassin report also shows a clear indication of headers and body forgery. The flags FORGED_OUTLOOK_TAGS, FORGED_OUTLOOK_HTML, MSOE_MID_WRONG_CASE, AXB_XMAILER_MIMEOLE_OL_024C2 and FORGED_MUA_OUTLOOK are all indicating easily identifiable forgery of Microsoft Outlook email headers and body. Finally, SPF is failing because the sending IP does not belong to the authorized range of addresses allowed to send emails for the domain. Further testing has been performed to check the behaviour of SpamAssassin when SPF is not available in the domain. After running SpamAssassin in debug mode, it appears that SpamAssassin fails silently when SPF is not implemented in the original domain (“No applicable sender policy available”). The same behaviour has been observed with DKIM signing. No trace of DKIM can be found in Listing 1 because it is not available in the original domain and the email is not digitally signed.
From the Rspamd output above, it can be seen that the forgery-specific flags are matching the ones found in Listing 1. By contrast, no rule related to DNSBLs has been triggered. Moreover, as opposed to SpamAssassin, Rspamd checks for the DMARC policy (DMARC_POLICY_QUARANTINE), which in this case indicates that the email is still accepted but should be treated with increased scrutiny. Rspamd also checks for TLS, and the rule RCVD_NO_TLS_LAST notifies of its absence. Finally, the SPF failure is detected (R_SPF_FAIL) and contrarily to SpamAssassin, Rspamd mentions the absence of DKIM with the R_DKIM_NA flag.

4.3.2 With authentication

For the second part of our experiments, we use our proof of concept based on the solution described in Section 4.2 to send the same Craigslist email, authenticating the email but leaving the body and subject unchanged. We registered the domain “users-accounts.com” and defined a subdomain “craigslist”. Our email generation script described in Appendix B is used to craft the email, which is then sent to our mail server at the receiving end, using the email address “no-reply@craigslist.users-accounts.com”. Once the email is received, it is then given as input to SpamAssassin and Rspamd. Listings 3 and 4 correspond to the spam reports of both spam filters when the email is sent using our solution:

Listing 3: SpamAssassin spam report of phishing email using the solution

```bash
$ spamc -R < mbox

Content analysis details:  (0.7 points, 5.0 required)

<table>
<thead>
<tr>
<th>pts</th>
<th>rule name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>URIBL_BLOCKED</td>
</tr>
<tr>
<td>-0.0</td>
<td>SPF_PASS</td>
</tr>
<tr>
<td>-0.0</td>
<td>SPF_HELO_PASS</td>
</tr>
</tbody>
</table>
```
The first observation that can be made from Listing 3 is that the email has a very low score of 0.7. This happens for several reasons. First, SPF is passing ([SPF_HELO_PASS and SPF_PASS]) and the DKIM signature is valid (DKIM_VALID, DKIM_SIGNED and DKIM_VALID_AU). This increases the trustworthiness of the email to SpamAssassin. Second, because we did not make use of header or body forgery, the rules triggered by the forgery observed in Listing 1 are not applicable in this case. Moreover, we created a script that properly generates and encodes a version of the email in plaintext as well as a version in HTML (see Appendix B for a detailed overview of the script). Therefore, triggering of the rule MIME-HTMLONLY has been avoided. Finally, the rule URIBL_BLOCKED remains active, since the body of the email has not been modified. This can be easily avoided by replacing the original link with a link pointing to the domain “craigslist.users-accounts.com” that we own, which would also give more credibility to the email.

Listing 4: Rspamd spam report of phishing email using the solution

```
$ rspamc < mbox

Results for file: stdin (4.080 seconds)
[Metric: default]
Action: no action
Spam: false
Score: -0.70 / 15.00
Symbol: ARC_NA (0.00)
Symbol: DKIM_TRACE (0.00)[craigslist.users-accounts.com:++]
Symbol: DMARC_POLICY_ALLOW (-0.50)[users-accounts.com, reject]
Symbol: FROM_EQ_ENVFROM (0.00)
Symbol: FROM_HAS_DN (0.00)
Symbol: MID_RHS_MATCH_FROM (0.00)
Symbol: MIME_BASE64_TEXT (0.10)
Symbol: MIME_GOOD (-0.10)
Symbol: PREVIOUSLY_DELIVERED (0.00)
Symbol: RCPT_COUNT_ONE (0.00)[1]
Symbol: RCVD_COUNT_THREE (0.00)[3]
Symbol: R_DKIM_ALLOW (-0.20)[craigslist.users-accounts.com]
Symbol: TO_DN_NONE (0.00)
```

In the case of Rspamd, the email scored below zero. The spam filter noticed that the DKIM signature is valid (DKIM_TRACE and R_DKIM_ALLOW). For this reason, Rspamd lowers the score of the email with a number of points equal to 0.20. It also appears that Rspamd attributes importance to the known content-type of the email (MIME_GOOD). Since our solution implements a DMARC policy, Rspamd’s checks for DMARC are successful (DMARC_POLICY_ALLOW). Finally, as our solution enforces the use of STARTTLS, the TLS-specific rule triggered during our last experiment as seen in Listing 2 and worth 0.10 points, has been removed.

4.4 Analysis of the solution

In Section 4.3, we have assessed the efficiency of a compliant solution that crafts and authenticates phishing emails in order to avoid detection by spam filters. With regard to Shahrukh’s research [23] on bypassing phishing filters, we combined content obfuscation with our solution. The goal was to assess how spam filters react to an authenticated phishing email with obfuscated content. In order to analyse this combination of techniques, we used the Craigslist email, obfuscated its content with Shahrukh’s script, and then used our proof of concept to authenticate and send the email to the different email service providers.
4.4.1 ProtonMail

ProtonMail\[20\] is an open-source, end-to-end encrypted email service focused on privacy. To filter undesirable emails, ProtonMail makes use of SpamAssassin, which allows us to easily analyse the rules that were triggered by the phishing email. To experiment with ProtonMail’s spam filtering system, we have sent the obfuscated and authenticated \textit{Craigslist} email to our ProtonMail email address. Table \ref{table:3} lists the rules triggered by the email. These rules have been extracted from the \textit{X-Spam-Status} message header added by SpamAssassin to the email. The rules most relevant to our study are highlighted in gray.

Table 3: ProtonMail: rules triggered by phishing email using obfuscation and authentication techniques

<table>
<thead>
<tr>
<th>Rule name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DKIM_SIGNED</td>
</tr>
<tr>
<td>DKIM_VALID</td>
</tr>
<tr>
<td>DKIM_VALID_AU</td>
</tr>
<tr>
<td>HTML_MESSAGE</td>
</tr>
<tr>
<td>SPF_HELO_PASS</td>
</tr>
<tr>
<td>SPF_PASS</td>
</tr>
<tr>
<td>URIBL_-FRESH_28D_SURBL</td>
</tr>
</tbody>
</table>

Although the score of each rule is not available, SpamAssassin inserts the email total score in the \textit{X-Spam-Status} header. During this experiment, the email scored 1.9/4.0. A first observation is that ProtonMail lowered the default threshold of SpamAssassin from 5.0 to a more strict threshold of 4.0. Based on our experiments with SpamAssassin described in Section 4.3 and from the Table \ref{table:3} we can safely assume that the only rules that increased the score of the email are HTML\_MESSAGE and URIBL\_-FRESH\_28D\_SURBL. The latter is a rule indicating that the domain is only 28 days old. Although this is not enough to classify the email as spam, it is important to notice that this parameter is taken into consideration during the spam detection process. From this spam analysis, we can conclude that the application of the obfuscation and authentication techniques were enough to successfully bypass the spam filter protections.

4.4.2 Office 365

Office 365\[14\] is a collection of services offered by Microsoft, as part of the Microsoft Office product line. Although this proprietary service can be challenging to analyse, we noticed that Outlook anti-spam solution checks for SPF, DKIM and DMARC. We also noticed that Microsoft has a dedicated portal\[17\] where users can make requests to delist an IP address that would have been misclassified as a spam sender. During our experiments, the \textit{Craigslist} email has been sent 10 times using both content obfuscation and authentication to our Outlook email address. The source of the message has then been analysed using MxToolbox\[15\], an online tool for analysing email headers more easily. Figure 1 below shows an overview of the email headers analysis performed by MxToolbox.

![Figure 1: MxToolbox: overview of email headers analysis](image-url)
The conclusion of our experiments is that 8 out of 10 times, the authenticated email successfully bypassed Outlook’s protections and was delivered to the inbox. However, some DNS timeouts have been observed, which resulted in the authenticity of two emails not being validated and the emails going to the spam folder. This might be a consequence of containerization of the DNS server or the use of a CNAME record for our subdomain. Because of the low reproducibility nature of the timeouts and the time constraints of this study, further investigation and experimentation is recommended in order to verify these hypotheses. Since the multiple DNS queries taking place during the validation of the sender authenticity took too long to be resolved for Outlook, the affected emails were classified as spam.

4.4.3 Gmail

Gmail is a free email service developed by Google. Unlike the aforementioned email providers, it is known that Google is using a reputation-based anti-spam solution. Taylor gives an overview of how Gmail determines a sender’s reputation in order to classify an authenticated domain as “spammy” or “not spammy”. This technique might appear to be more efficient to mitigate our solution because of its learning capabilities. First, the domain we use to send phishing emails is recent and has a neutral reputation from Google’s standpoint. Second, Taylor mentions in his paper that the user input is taken into consideration in order for their system to make better decisions in classifying incoming emails. Therefore, if a sender’s emails happen to be marked as spam by Google’s systems or users, the reputation of the sender would decrease drastically, resulting in the sender’s emails being blocked by Google’s systems categorically, regardless of the content or authenticity of these emails.

As an experiment, the Craigslist email was sent 10 times using our solution combined with content obfuscation to our Gmail email address. Although no trace of spam analysis could be found in the emails’ headers, the Authentication-Results header containing the results of the SPF, DKIM and DMARC validation has been examined. Listing 5 shows the anonymized header of one of our authenticated emails.

Listing 5: Gmail: Authentication-Results email header

```
Authentication-Results: mx.google.com;
    dkim=pass header.i=@craigslist.users-accounts.com header.s=mail header.b=do3vasHt;
    spf=pass (google.com: domain of no-reply@craigslist.users-accounts.com designates as permitted sender) smtp.mailfrom=no-reply@craigslist.users-accounts.com;
    dmarc=pass (p=REJECT sp=REJECT dis=NONE) header.from=users-accounts.com
```

The output of Listing 5 shows that the email is passing the SPF, DKIM and DMARC verification tests. The same result has been observed with all 10 emails. These results indicate that our phishing emails are successfully authenticated and validated by Google’s spam filter. Gmail’s user interface also indicates to the user by means of a small gray lock icon that “Standard encryption (TLS)” has been used for the transport of the email. Finally, because a user may often read messages with specific keywords, we noticed that phishing emails including these keywords may be marked as important by Gmail using a small yellow marker as an indicator. This is surprisingly interesting, as Google’s important marking system might give additional credibility to phishing emails that reach the inbox.
Table 4: Gmail: effectiveness of reputation-based classification system

<table>
<thead>
<tr>
<th>Email No.</th>
<th>Detected as spam by Gmail</th>
<th>Further action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✗</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>✗</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>✗</td>
<td>N/A</td>
</tr>
<tr>
<td>6</td>
<td>✗</td>
<td>Emails 4, 5, 6 marked as spam by the user</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>N/A</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>Emails 7, 8 marked as ham by the user</td>
</tr>
<tr>
<td>9</td>
<td>✗</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>✗</td>
<td>N/A</td>
</tr>
</tbody>
</table>

As shown in Table 4, the results of this experiment indicate that after sending Email 1, 2, and 3, they were successfully validated by Gmail spam filters, thus reaching the user’s inbox. We then sent Email 4, 5 and 6 which after reaching the user’s inbox, were manually marked as spam by the user. Emails 7 and 8 were then sent and automatically marked as spam by Gmail. After manually marking Email 7 and 8 as legitimate (ham), we sent Email 9 and 10, which reached the user’s inbox and therefore were not marked as spam by Gmail. These experiments give a first glimpse of how reputation-based anti-spam systems work. It is important to note that experimenting with such learning systems can be very challenging, since there is no way to reset the reputation of a sender. Further work needs to be done to establish how variables such as the number of recipients or a longer period of time between the emails affect the classification of emails by Gmail.
In the present study we investigated the effectiveness of spam filters in protecting the end users against phishing attacks. We first examined the existing network and authentication mechanisms and determined which of these mechanisms are implemented in spam filters. Then, we researched what methods can be employed to bypass these mechanisms. In order to answer this question, we used a large data set of phishing emails and experimented with two open source spam filters. From these experiments, we identified a number of verifications performed by spam filters in order to validate the authenticity of an email. The first conclusion drawn was that the spam filters do not come with proper default configuration. Studies have shown that anti-spoofing protocols are still not widely accepted. An effect of this can be observed in the authentication-specific rules assigned by the spam filters to spam emails. The default number of points for these rules as well as the default threshold of the spam filters are too low to efficiently mitigate spoofing. However, studies have shown that techniques such as SPF, DKIM and DMARC are effective at preventing spam. When properly configured, spam filters may benefit from these techniques in order to better classify incoming emails. This has led to the conclusion that one method to bypass the spam filters protections is to comply with the rules (see Section 4.2).

As a result of this analysis, a proof of concept has been developed which includes a DNS server, a mail server and a web server. This solution uses containerization and scripting for easy deployment and configuration. This infrastructure shows that it is possible to relatively easily bypass spam filters by sending authenticated phishing emails and impersonate several different organisations using DNS subdomains. We selected a sample email from our large data set of phishing emails to experiment with popular email service providers using our solution combined with obfuscation techniques described in Shahrukh’s paper. The results of these experiments show that the application of evasion techniques combined with email authentication is remarkably enough to bypass the anti-spam solutions of these providers. In this context, protection of the end user against phishing emails relies mostly on the users’ awareness, which is known to be the weakest link of the chain.

Mitigation of content obfuscation techniques have been discussed in Shahrukh’s paper and is outside the scope of this study. Compliant techniques such as the one discussed in this paper may be very challenging to mitigate, since it heavily relies on human weakness. Moreover, a key objective of anti-phishing solutions is to filter malicious emails while keeping a false positive rate as low as possible. However, several aspects may be considered in order to minimize the risk of a phishing email reaching the end user. First, it is important that more organisations adopt anti-spoofing protocols in order for spam filters to be more effective. Second, we observed during our experiments described in Section 4.4 that spam filters are able to determine how old a domain is. This may be an indication of a phishing attack in the case where an attacker starts sending phishing emails relatively soon after registering a domain. Further checking of the sender’s domain and subdomain names should be performed by the filters, in order to identify popular organisation names and detect senders that are using these names as subdomains. Studies on using techniques like machine learning and reputation-based techniques as a solution to detect phishing emails have shown interestingly low error rates. Google’s reputation-based filtering system discussed in Section 4.4 has shown interesting capacities to block malicious senders but requires some improvements. An attacker may try to build a good reputation for a domain before starting to send phishing emails. As a countermeasure, senders’ reputation should be specific to each email account. Finally, these systems also rely on the end user’s input, which is an aspect to bear in mind and integrate in the development of anti-phishing solutions.

While this study has strictly focused on the network and authentication aspects of phishing emails, other aspects need to be considered in order to properly classify incoming emails. For instance, anti-phishing solutions might react differently when phishing emails are sent in bulk to many different users. As this is outside the scope of this study, the results of this study must be approached with caution.
CHAPTER 6

Conclusion

Over the past years, spam filtering has become an arms race between spammers and anti-spammers. Likewise, email phishing is a serious threat for end users as well as organisations, who rely on spam filters for protection against these attacks. This study set out to investigate which network and authentication-specific protections are used by spam filters to classify incoming emails and assess how effectively spam filters use these mechanisms to protect against phishing attacks. The analysis of a set of phishing emails using two open source spam filters revealed that spam filters use techniques such as SPF, DKIM and DMARC to validate the authenticity of an email. The results of this study show that using compliance by implementing these authentication protocols with newly registered domains is enough to bypass these protections. In conclusion, this study suggests minor improvements that spam filters could implement in order to better mitigate phishing attacks.

6.1 Future work

Since this study was limited to the network and authentication aspects of phishing emails, these results should be interpreted with caution. Further research might explore the efficiency of our solution when numerous authenticated phishing emails are sent to an organisation. Although this study provided more insight on how to bypass spam filters using phishing emails authentication, considerably more work will need to be done to determine how to better protect the end user against this technique. In order to improve the detection of phishing emails, further studies regarding the role of DNS subdomains in phishing attacks would be worthwhile. As the evasion technique discussed in this paper exploits human weaknesses, research on the role of user interfaces of email services in mitigation of this type of attack would be a fruitful area for further work. More broadly, further studies could assess the effectiveness of machine learning and reputation-based anti-spam solutions against such attacks.
References

[12] Importance markers in Gmail - Gmail Help. URL: https://support.google.com/mail/answer/186543?hl=en
[16] Jose Nazario. The online PhishingCorpus. URL: https://monkey.org/~jose/phishing/.
[18] Rspamd features. URL: https://rspamd.com/features.html
[20] Secure email: ProtonMail is free encrypted email. URL: https://protonmail.com/
Phishing email

**Craigslist**: This email pretends to be Craigslist and informs a user that the user’s account has been blocked. To confirm and restore the account, the user is invited to click on a link pointing to a malicious address.
The shell script below is part of the proof of concept created during this research to experiment with spam filters. The purpose of this script is to generate compliant emails for further sending using the `mail` command of the GNU Mailutils package. To determine what a compliant email is, we first examined the tests performed by spam filters on the MIME types and the structure of emails. We then identified a number of characteristics that emails should possess in order to comply with the rules of spam filters. This includes different MIME types, known encodings and proper headers structure. To generate the email body, the following script takes a text file and an HTML file as input, which will be encoded in base64 and quoted-printable respectively. A file containing a list of headers to include in the email can optionally be specified. Finally, the script crafts the email and sends it to one or many recipients. The email will be sent from the container which the ID is given as first argument.

```bash
#!/bin/bash
if [ $# -lt 4 ] || [ "$1" = "--help" ]; then
    echo "Usage: $0 <container ID> <content.txt> <content.html> <recipient>|-f <recipients.txt> [-h headers.txt]"
    echo "- content.txt: email body that will be encoded in base64"
    echo "- content.html: email body that will be encoded in quoted-printable"
    echo "- recipients.txt: a list of recipients, one recipient per line"
    echo "- headers.txt (optional): a list of headers to append, one 'HEADER: VALUE' per line; double quotes must be escaped"
    exit 1
fi

parse_headers() {
    echo "Opening headers file $1"
    echo "Parsing headers..."
    while IFS=' ' read -r line || [[ -n "$line" ]]; do
        headers="$headers-a "$line"
    done < "$1"
}

generate_body() {
    echo "Creating the email body..."
    if ! hash pwgen 2>/dev/null; then
        apt-get install -y pwgen
    else
        boundary=$(pwgen 12 1)
    fi

    echo "--$boundary
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: base64" > body
base64 < "$1" > body

    echo "--$boundary
Content-Type: text/html; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable" > body
qprint < body
if ! hash qprint 2>/dev/null; then
    apt-get install -y qprint
```
```bash
else
    qprint -e "$2" >> body
    echo "--$boundary--" >> body
fi

if [ ! -f "$2" ]; then
    echo "Email content file $2 not found."
    exit 2
fi

if [ ! -f "$3" ]; then
    echo "Email content file $3 not found."
    exit 2
fi

return_path=$(sudo docker exec $1 bash -c "echo "$ROOT_ALIAS"")
if [ -z "$return_path" ]; then
    return_path=$(sudo docker exec $1 bash -c "echo $(echo $smtp_user | cut -d: -f1)@"maildomain"")
fi

content=$(basename "$2")
subject=$(echo "$content" | cut -d. -f1)

if [ "$4" = "-f" ]; then
    if [ ! -f "$5" ]; then
        echo "Recipients file $5 not found."
        exit 3
    fi
    if [ "$6" = "-h" ]; then
        if [ ! -f "$7" ]; then
            echo "Headers file $7 not found."
            exit 4
        fi
        parse_headers "$7"
    fi
else
    if [ "$5" = "-h" ]; then
        if [ ! -f "$6" ]; then
            echo "Headers file $6 not found."
            exit 5
        fi
        parse_headers "$6"
    fi
fi

generate_body "$2" "$3"

if [ "$4" = "-f" ]; then
    echo "Opening recipients file $5"
    while IFS= read -r rcpt || [[ -n "$rcpt" ]]; do
        echo "Sending email '$subject' to <$rcpt>..."
        cmd="mail -r $return_path $headers -a "$subject" -s "$rcpt" body"
        eval docker exec -i $1 $cmd
    done < "$5"
else
    echo "Sending email '$subject' to <$4>..."
    cmd="mail -r $return_path $headers -a "$subject" -s "$subject" $4 body"
    eval docker exec -i $1 $cmd
fi
```