University of Amsterdam

Master Security and Network Engineering

MSc Security and Network Engineering

Bypassing phishing protections with email authentication

Adrien Raulot
adrien.raulot@os3.nl

February 4, 2019

Supervisors: Marat Nigmatullin, Alex Stavroulakis

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 many 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

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[7]. 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[21, 26]. In addition to the use of spam filters, large email service providers perform filtering based on the sender's reputation[25].

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 6 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". Certain attacks such as the CEO Fraud attack described by Gupta *et al.*[10] involve the impersonation of an organisation's CEO or other senior executive. By using these methods, 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[6], 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.

CHAPTER 2

Related work

Over the past few years, phishing attacks have become more sophisticated in response to increased countermeasures. Although many techniques exist[7] to prevent phishing emails from reaching end users, phishing persists as a threat to end users and organisations, and is a real challenge for email service providers[12].

A key aspect of phishing emails that is thoroughly examined by spam filters is the email body. Zaidi[27] 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.*[2] 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[11, 2] 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.*[12, 13], 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.*[6] 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[8], 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 phishing emails characteristics 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[22].

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 Zaidi's research[27] and after experimenting with a data set of more than 300 emails in the `mbox` format from Jose Nazario's 2017 phishing email list[19], it appeared that the rules triggered most frequently were the same, excluding rules that were content-specific. After analysing the frequently triggered rules, we will experiment with the *Craigslist* email, extracted from Jose Nazario's most recent phishing emails list (see Appendix A.1 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 with a subset of emails against the spam filters of the following email service providers:

- ProtonMail[23]
- Office 365[17]
- Gmail[9]

CHAPTER 4

Results

4.1 Email processing with spam filters

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 `mbox` format as input, and generates a spam report. The client reads the `mbox` file from the standard input (stdin) and spools it to its connection with the daemon, which is responsible for talking to the corresponding domain in order to validate the email. The client then reads the result back and prints it to the standard output (stdout).

An email stored in the `mbox` format provides a lot of information about the sending end and the intermediate servers. The `Received` headers include hostnames, IP addresses, message identifiers and timestamps, as well as details about the protocol in use and the TLS connection. Although some information such as the use of TLS is locally processed and analysed by the daemon, email validation requires to send several DNS requests. It is important to bear in mind that side effects might occur when processing emails from the past, for instance, querying a domain whose configuration has changed since the time the email was sent, may fail the validation or produce distorted results.

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.

SpamAssassin offers little configuration tuning compared to Rspamd. Both Rspamd and SpamAssassin are composed of several modules (or plugins), with each module corresponding to a feature. Both spam filters allow the user to whitelist or blacklist senders, customise rules and scores as well as actions to be taken. However, we found Rspamd's configuration to be more accessible, better documented, and offer more tuning options. Each Rspamd module has configurable options, on top of the daemon's global configuration. For instance, Rspamd allows for DNS resolver tuning, cache size customisation for SPF and DKIM results, SPF/DKIM/DMARC domain whitelisting, domain mapping with strict multipliers for DKIM violation and DMARC reporting in a Redis database. Finally, Rspamd also supports part of SpamAssassin's rules (glob patterns are not supported so far), which can be imported by Rspamd's module named "Spamassassin rules".

4.2 Analysis of phishing emails

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. Figure 1 shows the ratio of emails detected as spam to emails detected as "ham".

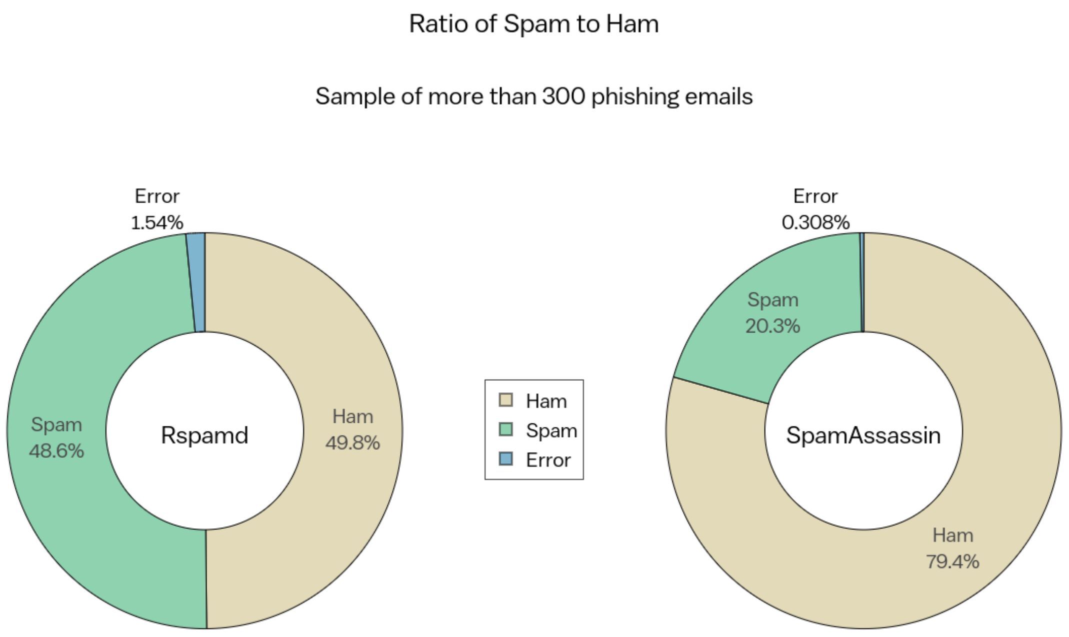


Figure 1: Ratio of Spam to Ham emails caught by the default rules of Rspamd and SpamAssassin

Figure 1 shows an important discrepancy between SpamAssassin's results and Rspamd's. While Rspamd detected 48.6% of emails as spam, only 20.3% were caught by SpamAssassin. Figure 2 and Table 1 below describe the most frequently triggered rules with SpamAssassin. The rules that are most relevant to our study are highlighted in gray.

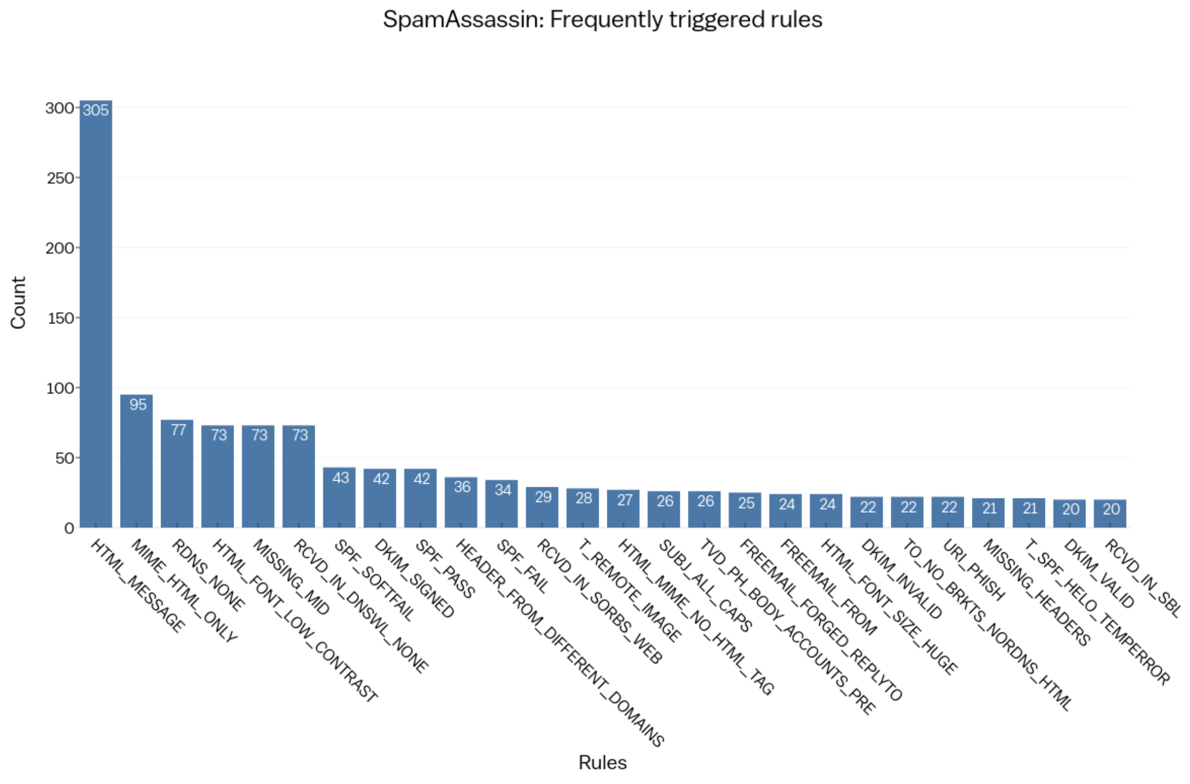


Figure 2: SpamAssassin: Frequently triggered rules

Figure 2 and Table 1 give better insight on the low percentage of emails detected as spam by SpamAssassin. While the most triggered rule has been counted 305 times, we observe an important gap between the first and second most triggered rules in terms of count. Moreover, we notice a difference between the rules that are most frequently triggered and their score. For instance, the first most frequently triggered rule does not seem to increase the score given to the affected emails, since it has a default score of 0.0 (see Table 1). On the contrary, the rules with the highest scores are not triggered very often, apart from the `MIME_HTML_ONLY` and `RDNS_NONE` rules.

Table 1: SpamAssassin: Description of frequently triggered rules

Points	**Rule name**	**Description**
2.6	`RCVD_IN_SBL`	Received via a relay in Spamhaus SBL
2.5	`FREEMAIL_FORGED_REPLYTO`	Freemail in Reply-To, but not From
1.6	`SUBJ_ALL_CAPS`	Subject is all capitals
1.5	`RCVD_IN_SORBS_WEB`	SORBS: To: misformatted and no rDNS and HTML only
1.3	`RDNS_NONE`	Delivered to internal network by a host with no rDNS
1.2	`MISSING_HEADERS`	Missing To: header
1.1	`MIME_HTML_ONLY`	Message only has text/html MIME parts
1.0	`ACCT_PHISHING`	Possible phishing for account information
1.0	`URI_PHISH`	Phishing using web form
1.0	`SPF_SOFTFAIL`	SPF: sender does not match SPF record (softfail)
0.9	`SPF_FAIL`	SPF: sender does not match SPF record (fail)
0.6	`HTML_MIME_NO_HTML_TAG`	HTML-only message, but there is no HTML tag
0.6	`TO_NO_BRKTS_NORDNS_HTML`	SORBS: sender is an abuseable web server
0.1	`DKIM_SIGNED`	Message has a DKIM or DK signature, not necessarily valid
0.1	`MISSING_MID`	Missing Message-Id: header
0.0	`T_DKIM_INVALID`	DKIM-Signature header exists but is not valid
0.0	`T_REMOTE_IMAGE`	Message contains an external image
0.0	`RCVD_IN_DNSWL_NONE`	Sender listed at http://www.dnswl.org/, no trust
0.0	`T_SPF_HELO_TEMPERROR`	SPF: test of HELO record failed (temperror)
0.0	`HTML_MESSAGE`	BODY: HTML included in message
0.0	`HTML_FONT_LOW_CONTRAST`	Attempts to hide message
0.0	`HTML_FONT_SIZE_HUGE`	HTML font size is huge
0.0	`TVD_PH_BODY_ACCOUNTS_PRE`	The body of the mail matches phrases such as “accounts suspended”, “account credited”, “account verification”
0.0	`FREEMAIL_FROM`	Sender email is commonly abused enduser mail provider
-0.0	`HEADER_FROM_DIFFERENT_-DOMAINS`	From and EnvelopeFrom 2nd level mail domains are different
-0.0	`SPF_PASS`	SPF: sender matches SPF record
-0.1	`DKIM_VALID`	Message has at least one valid DKIM or DK signature

Figure 3 and Table 2 below show the most frequently triggered rules with Rspamd. The rules that are most relevant to our study are highlighted in gray.

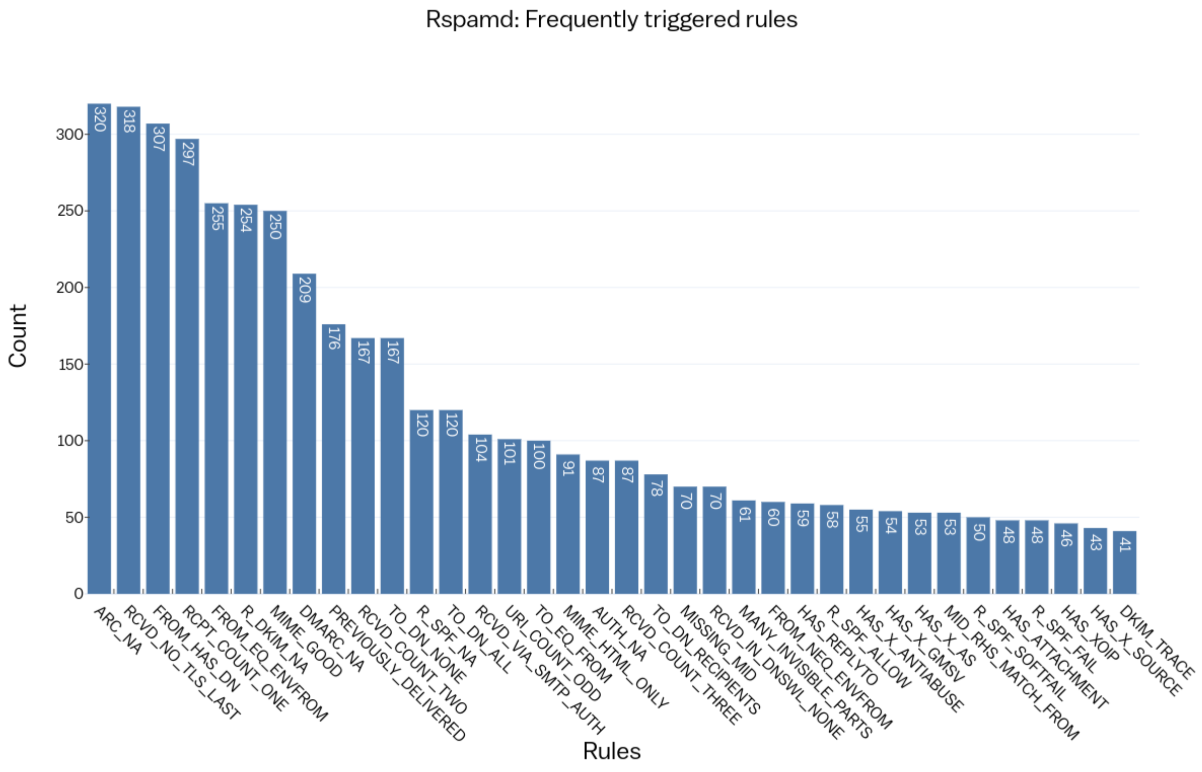


Figure 3: Rspamd: Frequently triggered rules

Figure 3 and Table 2 give a more detailed overview of Rspamd's results. As opposed to SpamAssassin, we observe that the count per rule decreases in a gradual manner. The number of different rules as well as the number of times each rule was triggered are drastically higher than what we observed with SpamAssassin. If we correlate this observation with the high percentage of emails detected as spam, Rspamd appears to be more effective than SpamAssassin.

The first observation we can make from Table 1 and 2 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. After checking SpamAssassin's source code, it appears that DMARC checks are not directly handled by SpamAssassin. Instead, SpamAssassin analyses the `Authentication-Results` header in order to trigger the according DMARC rules. Open-source plugins such as `mail-dmarc`[24] allows SpamAssassin to validate the alignment of incoming messages with the sender's DMARC policy. The spam filters indicate an SPF failure 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, and 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 (`RCVD_NO_TLS_LAST`). SpamAssassin indicates that a sending host does not have reverse DNS by means of the `RDNS_NONE` flag. Rspamd mentions a difference of address between `MAIL FROM` in the SMTP envelope and `From` in the email header by using the `FROM_NEQ_ENVFROM` flag while SpamAssassin uses the `HEADER_FROM_DIFFERENT_DOMAINS` flag. Moreover, the rule `RCVD_VIA_SMTP_AUTH` is inserted when an authenticated hand-off of the message happens. Finally, Rspamd checks for the presence of an ARC signature, which is inserted by intermediate servers to sign the original message's validation results. The ARC signature can be used to authenticate an email when SPF and DKIM validation fail due to the use of a mailing list or forwarder. Rspamd also has an additional rule (`AUTH_NA`) for when the email has no SPF/DKIM/DMARC or ARC authentication possible.

Table 2: Rspamd: Description of frequently triggered rules

Points	Rule name	Description
2.5	`MISSING_MID`	Message ID is missing
2.0	`TO_DN_RECIPIENTS`	To header display name is "Recipients"
1.0	`URI_COUNT_ODD`	Odd number of URIs in multipart/alternative message
1.0	`AUTH_NA`	Authenticating message via SPF/DKIM/DMARC/ARC not possible
1.0	`R_SPF_FAIL`	Sender does not match SPF record
0.4	`MANY_INVISIBLE_PARTS`	Many parts are visually hidden
0.2	`MIME_HTML_ONLY`	Message is HTML only
0.1	`RCVD_NO_TLS_LAST`	Last hop did not use encrypted transport
0.0	`DMARC_NA`	No DMARC available
0.0	`DKIM_TRACE`	Message has a DKIM signature
0.0	`ARC_NA`	ARC signature absent
0.0	`FROM_HAS_DN`	`From` header has a display name
0.0	`RCPT_COUNT_ONE`	One recipient
0.0	`FROM_EQ_ENVFROM`	`From` address is the same as the envelope
0.0	`R_DKIM_NA`	Mail is not signed
0.0	`PREVIOUSLY_DELIVERED`	Message either to a list or was forwarded
0.0	`RCVD_COUNT_TWO`	Two recipients
0.0	`TO_DN_NONE`	None of the recipients have display names
0.0	`R_SPF_NA`	SPF policy not found
0.0	`TO_DN_ALL`	All the recipients have display names
0.0	`RCVD_VIA_SMTP_AUTH`	Authenticated hand-off was seen in `Received` headers
0.0	`TO_EQ_FROM`	`Reply-To` header is identical to `From` header
0.0	`RCVD_COUNT_THREE`	3-5 recipients
0.0	`RCVD_IN_DNSWL_NONE`	Sender listed at `https://www.dnswl.org`, no trust
0.0	`FROM_NEQ_ENVFROM`	`From` address is different to the envelope
0.0	`HAS_REPLYTO`	Has `Reply-To` header
0.0	`HAS_X_ANTIABUSE`	Has `X-AntiAbuse` headers
0.0	`HAS_X_GMSV`	Has `X-Get-Message-Sender-Via:` header
0.0	`HAS_X_AS`	Has `X-Authenticated-Sender` header
0.0	`MID_RHS_MATCH_FROM`	`Message-ID` RHS matches `From` domain
0.0	`R_SPF_SOFTFAIL`	SPF verification soft-failed
0.0	`HAS_ATTACHMENT`	Message contains attachments
0.0	`HAS_XOIP`	Has `X-Originating-IP` header
0.0	`HAS_X_SOURCE`	Has `X-Source` headers
0.0	`R_DKIM_PERMFAIL`	DKIM Signature is not valid
-0.1	`MIME_GOOD`	Known content-type
-0.14	`R_SPF_ALLOW`	SPF verification allows sending

A first conclusion can be drawn by looking at the points attributed to each of the rules. Despite a bigger proportion of phishing emails being marked as legitimate by both spam filters, Rspamd identified more phishing emails as spam than SpamAssassin, with almost half of our data set's phishing emails classified as spam. By default, both spam filters assign a very low number of points to SPF, DKIM and DMARC failures. Moreover, the default threshold of SpamAssassin is 5 while Rspamd scoring system works slightly differently. With Rspamd, an email scoring between 4 and 6 will be greylisted. If the score is over 6, the email will be marked as spam. Finally, the email will be rejected if its score reached 15 or higher. Rspamd seems to have more rules than SpamAssassin, hence more phishing emails in our data set were caught by Rspamd.

4.3 Compliance as a solution

As described in the Section 4.2, 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[6].

Studies[16] 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 allowing phishing emails to 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.

Suppose an attacker wants to impersonate a banking company `A`, and pretend to send emails for domain `a.com`. As seen in Section 4.2, 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 customisable 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[5] 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. However, this solution does not come without limitations. Studies[27, 7] have shown that the use of newly registered domains as well as legitimate-looking subdomains is a common characteristic of phishing emails. While the first issue can be relatively simply avoided by waiting a certain amount of time, the second issue is more difficult to address, while becoming more frequent also due to well-known security issues. For instance, recent attacks such as the one against the online retailer NewEgg[15] in 2018, involved the registration of look-alike domain names in order to steal sensitive data from end users. Therefore, it is not surprising to see domain name registrars proceed with several verification steps in order to spot attackers who are trying to register domain names for phishing purposes. During this study, we encountered many limitations as well as identity confirmation requests from several domain name registrars when trying to register domain names such as `account-google.nl`. Although some registrars might be less concerned than others, it complicates the attacker's task, who then might have to fall back on using subdomains instead.

4.4 Authentication of emails

To assess the effectiveness of the technique described in Section 4.3 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 4.4.1 and 4.4.2 describe the experiment performed with SpamAssassin and Rspamd without and with the use of our proof of concept respectively. Finally, Section 4.4.3 describes the same experiment but this time using our email crafting script.

4.4.1 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

```
$ spamc -R < craigslist.mbox

Content analysis details:   (8.8 points, 5.0 required)

 pts rule name
---- ----------------------
 0.0 URIBL_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 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 URI listed in the DNSBLs spam databases (`URIBL_BLOCKED`). Second, this SpamAssassin report 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` indicate easily identifiable forgery of Microsoft Outlook email headers and body. Finally, SPF is failing because the sending IP address does not belong to the authorized range of addresses allowed to send emails for the domain. As a result, 0.9 points are added to the email score. 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`"). Similarly, SpamAssassin will not mention the absence of DKIM signature if the email has not been digitally signed. Therefore, even though the original domain has implemented DKIM, the absence of DKIM signature in the message leaves no trace in the spam report (see Listing 1).

Listing 2: Rspamd spam report of original phishing email

```
$ rspamc < craigslist.mbox

Results for file: stdin (0.132 seconds)
[Metric: default]
Action: reject
Spam: true
Score: 25.90 / 15.00
Symbol: ARC_NA (0.00)
Symbol: ASN (0.00)
Symbol: DATE_IN_PAST (1.00)
Symbol: DMARC_POLICY_QUARANTINE (1.50)[craigslist.org : No
   valid SPF, No valid DKIM, quarantine]
Symbol: FORGED_MUA_OUTLOOK (3.00)
Symbol: FORGED_OUTLOOK_HTML (5.00)
Symbol: FORGED_OUTLOOK_TAGS (2.10)
Symbol: FROM_DN_EQ_ADDR (1.00)
Symbol: FROM_EQ_ENVFROM (0.00)
Symbol: HAS_REPLYTO (0.00)[NOREPLY@craigslist.org]
Symbol: HAS_X_PRIO_THREE (0.00)[3]
Symbol: MIME_HTML_ONLY (0.20)
Symbol: RCPT_COUNT_ONE (0.00)[1]
Symbol: RCVD_COUNT_FIVE (0.00)[6]
Symbol: RCVD_HELO_USER (3.00)
Symbol: RCVD_NO_TLS_LAST (0.10)
Symbol: REPLYTO_DOM_EQ_FROM_DOM (0.00)
Symbol: R_DKIM_NA (0.00)
Symbol: R_SPF_FAIL (1.00)[-all]
Symbol: R_UNDISC_RCPT (3.00)
Symbol: SPAM_FLAG (5.00)
Symbol: TO_DN_ALL (0.00)
Message - spf: (SPF): spf fail
```

From the Rspamd output outlined in Listing 2, 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 domain's DMARC policy. The `DMARC_POLICY_QUARANTINE` rule adds 1.50 points to the score and 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` is triggered in 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.

4.4.2 With authentication

For the second part of our experiments, we use the authentication part of our proof of concept to send the same *Craigslist* email, authenticating the email with SPF, DKIM and DMARC but leaving the body and subject unchanged. We registered the domain "`users-accounts.com`" and defined a subdomain "`craigslist`". The original phishing email is sent to our mail server at the receiving end using the email address "`no-reply@craigslist.users-accounts.com`", therefore creating a new SMTP Envelope in the process. 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 authentication.

Listing 3: SpamAssassin spam report of authenticated phishing email

```
$ spamc -R < craigslist.mbox

Content analysis details:   (7.8 points, 5.0 required)

 pts rule name
---- ----------------------
 0.0 URIBL_BLOCKED
 0.0 FSL_CTYPE_WIN1251
-0.0 SPF_PASS
```

```
-0.0 SPF_HELO_PASS
 0.0 HTML_MESSAGE
 1.1 MIME_HTML_ONLY
 -0.1 DKIM_VALID
 0.1 DKIM_SIGNED
 -0.1 DKIM_VALID_AU
 0.6 FORGED_OUTLOOK_TAGS
 0.0 FSL_NEW_HELO_USER
 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
```

We observe no significant difference in terms of score between the current and the previous experiment where the email was not authenticated. Authenticating the `Craigslist` email lowered the total score by 1 point. This is because trigger of the `SPF_FAIL` rule has been avoided which reduces the total score by 0.9. Additionally, rules such as `SPF_PASS`, `SPF_HELO_PASS`, `DKIM_SIGNED`, `DKIM_VALID` and `DKIM_VALID_AU` corresponding to the SPF and DKIM validation, are decreasing the score by 0.1. Since the email headers have not been modified, the forgery-specific rules are still present, which in this case does not allow the email to avoid detection.

Listing 4: Rspamd spam report of authenticated phishing email

```
$ rspamc < craigslist.mbox

Results for file: stdin (0.132 seconds)
[Metric: default]
Action: reject
Spam: true
Score: 22.70 / 15.00
Symbol: ARC_NA (0.00)
Symbol: ASN (0.00)
Symbol: DATE_IN_PAST (1.00)
Symbol: DKIM_TRACE (0.00)[craigslist.users-accounts.com:+]
Symbol: DMARC_POLICY_ALLOW (-0.50)[users-accounts.com,
   reject]
Symbol: FORGED_MUA_OUTLOOK (3.00)
Symbol: FORGED_OUTLOOK_HTML (5.00)
Symbol: FORGED_OUTLOOK_TAGS (2.10)
Symbol: FROM_DN_EQ_ADDR (1.00)
Symbol: FROM_EQ_ENVFROM (0.00)
Symbol: HAS_REPLYTO (0.00)[NOREPLY@craigslist.org]
Symbol: HAS_X_PRIO_THREE (0.00)[3]
Symbol: MIME_HTML_ONLY (0.20)
Symbol: RCPT_COUNT_ONE (0.00)[1]
Symbol: RCVD_COUNT_FIVE (0.00)[6]
Symbol: RCVD_HELO_USER (3.00)
Symbol: RCVD_NO_TLS_LAST (0.10)
Symbol: R_DKIM_ALLOW (-0.20)[craigslist.users-accounts.com]
Symbol: R_UNDISC_RCPT (3.00)
Symbol: SPAM_FLAG (5.00)
Symbol: TO_DN_ALL (0.00)
```

The same observation is valid for Rspamd, which gave a score lowered by 3.2 to the authenticated email in comparison to the previous experiment. SPF, DKIM and DMARC are validated, and the corresponding rules are decreasing the email total score by 0.7. This has allowed the `DMARC_POLICY_QUARANTINE` and `R_SPF_FAIL` rules to be bypassed, reducing the total score by an additional 2.50. However, this is not enough to avoid detection by the spam filter since forgery of the email headers is triggering too many rules with higher scores.

4.4.3 Using proof of concept

For the third part of our experiment, we use the entirety of our proof of concept. The email generation script described in Appendix B is used to craft the email, which means that the email headers are modified and a new SMTP Envelope is created during the email delivery process. The email body remains untouched, apart from the included URI whose domain we changed to our own, `craigslist.users-accounts.com`. Once crafted, the email is then sent to our mail server at the receiving end over an encrypted communication channel (STARTTLS), using the email address "`no-reply@craigslist.users-accounts.com`". The received email is given as input to SpamAssassin and Rspamd whose reports are presented in Listings 5 and 6.

Listing 5: SpamAssassin spam report of phishing email using the proof of concept

```
$ spamc -R < craigslist.mbox

Content analysis details:   (0.7 points, 5.0 required)

 pts rule name
---- ----------------------
-0.0 SPF_PASS
-0.0 SPF_HELO_PASS
 0.0 HTML_MESSAGE
-0.1 DKIM_VALID
 0.1 DKIM_SIGNED
-0.1 DKIM_VALID_AU
 0.8 TVD_PH_BODY_ACCOUNTS_POST
 0.0 TVD_PH_BODY_META_ALL
```

The first observation that can be made from Listing 5, 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 Listings 1 and 3 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_HTML_ONLY` has been avoided. Finally, triggering the rule `URIBL_BLOCKED` was avoided, since we replaced the original included URI's domain by our own, which also gives more credibility to the email.

Listing 6: Rspamd spam report of phishing email using the proof of concept

```
$ rspamc < craigslist.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 detected that the DKIM signature is valid (`DKIM_TRACE` and `R_DKIM_ALLOW`). For this reason, Rspamd lowers the email score with a number of points equal to 0.20. It also appears that Rspamd attributes importance to the email's known content-type (`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.

4.5 Analysis of the solution

In Section 4.4, we have assessed the efficiency of a compliant solution that crafts and authenticates phishing emails in order to avoid detection by spam filters. This section describes how we tested our solution with different phishing emails, gradually combining authentication and obfuscation techniques. For this experiment, we used the following techniques:

- **No technique**: the original phishing email is sent, with the original links and no authentication nor obfuscation
- **Authentication**: the original phishing email is sent, with the original links but using our solution to authenticate the email
- **Authentication + replaced URI**: the original links are replaced by links pointing to our domain name, and the email is authenticated
- **Authentication + obfuscation**: the original phishing email is sent, with the original links and the email is obfuscated using Zaidi's obfuscation script
- **Authentication + replaced URI + obfuscation**: the original links are replaced by links pointing to our domain name, and the email is obfuscated and authenticated

With regard to Zaidi's research[27] on bypassing phishing filters, we used Zaidi's script for the tests where obfuscation was applicable. As described in Zaidi's study[27], the script makes use of Unicode transliteration and URI shortening to obfuscate the email content. Since it is important for our experiment to keep the original links, we commented out the script part where link shortening is implemented, leaving the Unicode transliteration part untouched. 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 experimented with different email service providers with the following phishing emails:

- *Craigslist* (see Appendix A.1)
- *JIRA* (see Appendix A.2)
- *Google* (see Appendix A.3)
- *File Transfer* (see Appendix A.4)
- *DHL* (see Appendix A.5)

4.5.1 ProtonMail

ProtonMail[23] 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 first sent our list of five emails using different techniques described in section 4.5. Table 3 presents the results of this experiment.

Table 3: ProtonMail: Effectiveness of spam filter against different phishing emails

Technique	Craigslist	JIRA	Google	File Transfer	DHL
No technique	✓	✓	✓	✓	✓
Authentication	✓	✓	✓	✓	✓
Authentication + replaced URI	✓	✓	✓	✓	✓
Authentication + obfuscation	✓	✓	✓	✓	✓
Authentication + replaced URI + obfuscation	✓	✓	✓	✓	✓

The results shown in Table 3 are surprisingly unanimous: none of the phishing emails were caught by ProtonMail's spam filter. Figure 4 gives an overview of the rules triggered during this experiment and Table 4 describes each triggered rule. The rules that are most relevant to our study are highlighted in gray.

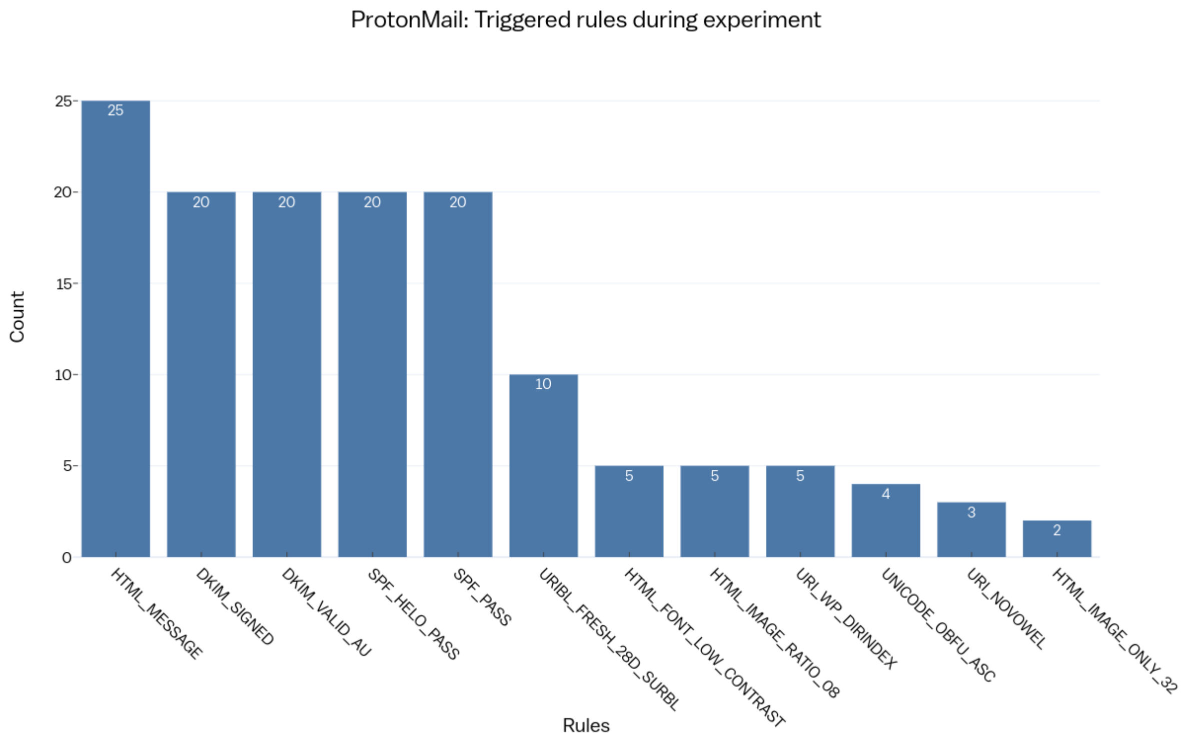


Figure 4: ProtonMail: Triggered rules during experiment

Table 4: ProtonMail: Description of triggered rules during experiment

Rule name	**Description**
DKIM_SIGNED	Message has a DKIM or DK signature, not necessarily valid
DKIM_VALID_AU	Message has a valid DKIM or DK signature from author's domain
HTML_MESSAGE	BODY: HTML included in message
HTML_FONT_LOW_CONTRAST	Attempts to hide message
HTML_IMAGE_RATIO_08	TML has a low ratio of text to image area
HTML_IMAGE_ONLY_32	HTML: images with 2800-3200 bytes of words
SPF_HELO_PASS	SPF: HELO matches SPF record
SPF_PASS	SPF: sender matches SPF record
URIBL_FRESH_28D_SURBL	Contains a domain registered less than 28 days ago
URI_NOVOWEL	URI hostname has long non-vowel sequence
URI_WP_DIRINDEX	URI for compromised WordPress site, possible malware
UNICODE_OBFU_ASC	Obfuscating text with unicode

Although the score of each triggered rule is not available, SpamAssassin inserts the email total score in the `X-Spam-Status` header. 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 experiment with SpamAssassin described in Section 4.2, it is possible that some rules such as `DKIM_VALID` have a negative score and therefore contribute to lower the total score of an email (see Table 1). The lowest total score observed during this experiment is $-0.1/4.0$. This score has been attributed to 92% of the emails sent. The highest score given by SpamAssassin during this experiment is 1.4/4.0, attributed to the *File Transfer* email in the fourth row, second last column of Table 3. The email triggered both the `URI_NOVOWEL` and `UNICODE_OBFU_ASC` rules, because of the use of an uncommon URI hostname as well as Unicode obfuscation respectively. Even though this obfuscation technique was used in 10 emails, SpamAssassin detected it in only 4 cases. Finally, the `URIBL_FRESH_28D_SURBL` rule was triggered in the cases where we replaced the URI by our own. This is not a surprise since the domain we use is recent, but may be a disadvantage because of the recurrent use of this technique and its detection in phishing emails, as described in Zaidi's paper[27].

From this spam analysis, we can conclude that no authentication nor obfuscation techniques were even required to bypass ProtonMail's protections. ProtonMail's SpamAssassin implementation do not seem to be effective enough, hence offering a very poor spam protection to the end user and relying more on user input to filter spam.

4.5.2 Office 365

Office 365[17] is a line of subscription 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[20] where users can make requests to delist an IP address that would have been misclassified as a spam sender.

We first experimented with our five previously selected emails, which we sent to our Office 365 Outlook email address using different techniques described in Section 4.5. The results of this experiment are presented in Table 5. A check mark and a cross indicate whether the email bypassed the spam filter or not respectively. DNS timeouts happening during validation of an authentication method are indicated by the method's acronym followed by a question mark. When timeouts occurred, the corresponding emails were not able to reach the inbox.

Table 5: Office 365: Effectiveness of spam filter against different phishing emails

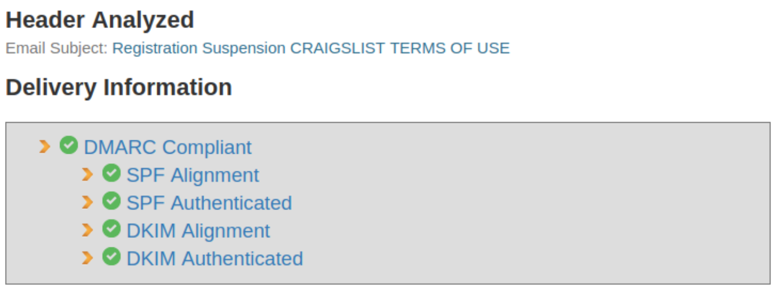
Technique	Craigslist	JIRA	Google	File Transfer	DHL
No technique	✗	✗	✗	✗	✗
Authentication	**✗ DKIM?**	**✗ (SPF/ DKIM/ DMARC)?**	✓	✓	✗
Authentication + replaced URI	✗	**✗ DMARC?**	✓	✓	**✗ (SPF/ DKIM/ DMARC)?**
Authentication + obfuscation	**✗ DMARC?**	✓	✗	✓	**✗(DKIM/ DMARC)?**
Authentication + replaced URI + obfuscation	✗	✓	**✗ DMARC?**	✓	✗

The results of this experiment are of an interesting nature because of their disparity. First, it is important to notice that Office 365 Outlook's spam filter blocked all the original phishing emails, as can be seen in the first row of Table 5. This is not the case with different email service providers. Second, we notice several DNS timeouts occurring during the experiment. They appear to be unpredictable, as some test emails such as *JIRA* and *DHL* suffered from many timeouts, whereas no timeout occurred when experimenting with the *File Transfer* email. Although no evident correlation between the emails and the timeouts could be established, these DNS timeouts will be further examined and discussed in the next experiment below. The *Craigslist* email was not able to go through no matter what technique was used to bypass the spam filter. Although the *JIRA* email suffered from DNS timeouts during the verification of its authenticity, it appears that content obfuscation and authentication is enough to avoid detection. Office 365's spam filter seemed to detect the *Google* email obfuscation, but was easily bypassed using only authentication. The *File Transfer* email easily went through using only authentication as well. Finally, the *DHL* email was blocked whether content obfuscation, authentication or both were used. In conclusion, these results show that authentication has its importance in the anti-spam verification process, as several emails (*Google*, *File Transfer*) were able to bypass Office 365 Outlook's spam filter by using authentication only. However, email content still seems to be in the foreground. Emails such as *DHL*, *Craigslist* were categorically blocked regardless of their authenticity and using obfuscation on the *Google* email was the trigger of Office 365's spam filter detection.

As a second experiment, the *JIRA* email (see Appendix A.2) was sent 3 times to the professional email address of a person working for a large financial organisation. This organisation was using Office 365 Outlook as its email service. A description of the three test emails is given as follows:

- **Email 1**: the original *JIRA* email, the links are replaced by ones from our own domain and the email is authenticated
- **Email 2**: the same email as Email 1, includes a link pointing to the target's domain and the email is authenticated
- **Email 3**: the same email as Email 2, Unicode transliteration is applied using Zaidi's script and the email is authenticated

The message source has then been analysed using MxToolbox[18], an online tool for analysing email headers more easily. Figure 5 below shows an example of the *Craigslist* email headers analysis performed by MxToolbox, when the email is sent to our Office 365 Outlook email address.


Header Analyzed

Email Subject: Registration Suspension CRAIGSLIST TERMS OF USE

Delivery Information

- ✓ DMARC Compliant
  - ✓ SPF Alignment
  - ✓ SPF Authenticated
  - ✓ DKIM Alignment
  - ✓ DKIM Authenticated

Figure 5: MxToolbox: Example of email headers analysis

As shown on Figure 5, SPF, DKIM authentication and alignment as well as the email's DMARC compliance are checked. Figures 6, 7 and 8 show the email headers analysis of Email 1, 2 and 3 respectively.

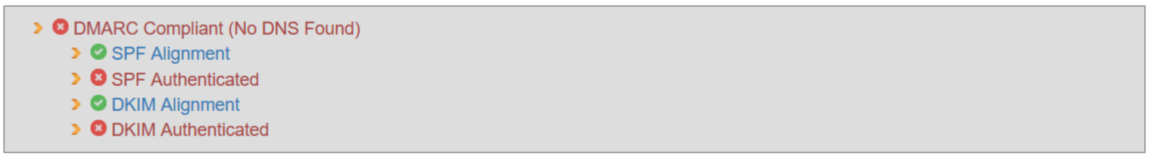

- ✗ DMARC Compliant (No DNS Found)
  - ✓ SPF Alignment
  - ✗ SPF Authenticated
  - ✓ DKIM Alignment
  - ✗ DKIM Authenticated

Figure 6: MxToolbox: Overview of Email 1's headers analysis

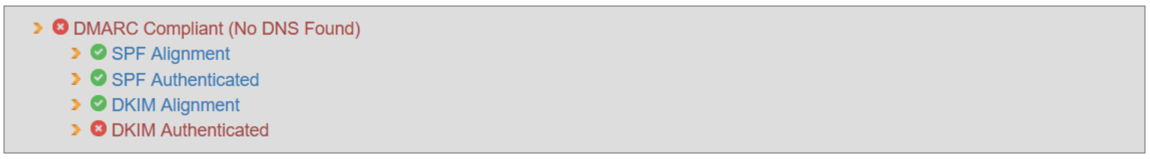

- ✗ DMARC Compliant (No DNS Found)
  - ✓ SPF Alignment
  - ✓ SPF Authenticated
  - ✓ DKIM Alignment
  - ✗ DKIM Authenticated

Figure 7: MxToolbox: Overview of Email 2's headers analysis

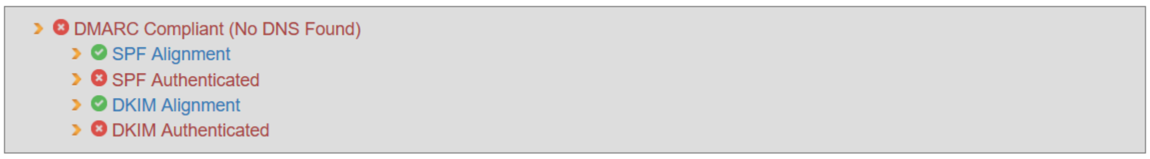

- ✗ DMARC Compliant (No DNS Found)
  - ✓ SPF Alignment
  - ✗ SPF Authenticated
  - ✓ DKIM Alignment
  - ✗ DKIM Authenticated

Figure 8: MxToolbox: Overview of Email 3's headers analysis

The result of this experiment is that 3 out of 3 times, the email was caught by Office 365 Outlook's protections and was delivered to the spam folder. Multiple DNS timeouts have been observed, which explains the "No DNS Found" error and resulted in the emails' authenticity not being fully validated. Listing 7 shows the anonymized `Authentication-Results` header of Email 1.

Listing 7: Office 365 Outlook: DNS timeouts in Authentication-Results header of Email 1

```
Authentication-Results: spf=temperror (sender IP is ****)
 smtp.mailfrom=jira.users-accounts.com; ****;
 dkim=timeout (key query timeout) header.d=jira.users-accounts.com;
    ****;
 dmarc=temperror action=none header.from=jira.users-accounts.com;
    compauth=none reason=405
Received-SPF: TempError (protection.outlook.com: error in processing
   during lookup of jira.users-accounts.com: DNS Timeout)
```

Despite the lack of information, a reason code is included in the header presented in Listing 7. Unfortunately for us, this is an internal code for why a message passed or failed authentication, and therefore this code has no universal meaning. This allows an administrator or even an end user to define how Office 365 should determine that a sender may be spoofed[4]. These DNS timeouts 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's authenticity took too long to be resolved for Outlook, the emails were classified as spam.

4.5.3 Gmail

Gmail[9] 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[25] 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 sender's reputation 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 a first experiment, we sent the five emails listed in section 4.5 to our Gmail email address. Table 6 gives an overview of the results for each technique:

Table 6: Gmail: Effectiveness of spam filter against different phishing emails

Technique	**Craigslist**	**JIRA**	**Google**	**File Transfer**	**DHL**
No technique	✓	✓	✓	✓	✗
Authentication	✓	✓	✓	✓	✗
Authentication + replaced URI	✓	✓	✓	✓	✓
Authentication + obfuscation	✓	✓	✓	✓	✓
Authentication + replaced URI + obfuscation	✓	✓	✓	✓	✓

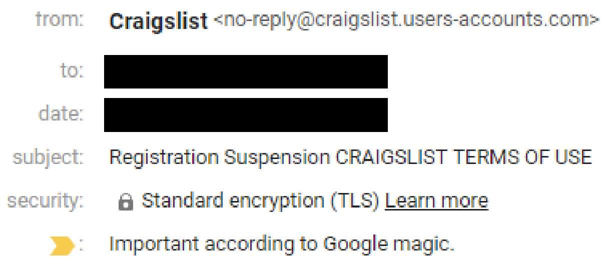
The results presented in Table 6 show that only the unobfuscated version of the DHL email was caught by Google's protection, regardless of whether it was authenticated or not. Since we could successfully avoid detection by only replacing the malicious URIs by our own, we can assume that the links were the anti-spam system trigger. These results must be interpreted with extreme caution because of the use of a reputation-based anti-spam system by Google. This anti-spam system might require more data and user input to prove its effectiveness at protecting the end user against larger phishing campaigns. However, based on the results of our experiments, we observe that this system seems to be very tolerant and does not offer an effective protection against narrowed phishing attacks.

As a second 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 8 shows the anonymized header of one of our authenticated emails.

Listing 8: 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 8 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.



from: Craigslist <no-reply@craigslist.users-accounts.com>
to: [redacted]
date: [redacted]
subject: Registration Suspension CRAIGSLIST TERMS OF USE
security: Standard encryption (TLS) Learn more
: Important according to Google magic.

Figure 9: Gmail: User interface

Gmail's user interface (Figure 9) also indicates to the user by means of a small gray lock icon that "Standard encryption (TLS)" has been used for the email transport. Finally, because a user may often read messages with specific keywords[14], 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 7: Gmail: Effectiveness of reputation-based classification system

Email No.	**Detected as spam by Gmail**	**Further action**
1	✗	N/A
2	✗	N/A
3	✗	N/A
4	✗	N/A
5	✗	N/A
6	✗	Emails 4, 5, 6 marked as *spam* by the user
7	✓	N/A
8	✓	Emails 7, 8 marked as *ham* by the user
9	✗	N/A
10	✗	N/A

As shown in Table 7, 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.

CHAPTER 5

Discussion

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[11] have shown that anti-spoofing protocols are still not widely accepted and 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 is too low and the default threshold of the spam filters is too high to efficiently mitigate spoofing. However, studies[16] 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 4.3).

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 subset of emails from our large data set of phishing emails to experiment with popular email service providers using our solution. Part of the experiment also involved the use of content obfuscation by combining our solution with Zaidi's obfuscation script (see Section 4.5). 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 the majority of these providers. Office 365 has proven to be slightly more resilient but is unreliable in email validation due to many DNS timeouts. In this context, protection of the end user against phishing emails relies mostly on the users' awareness, which is known[5] to be the weakest link of the chain.

Mitigation of content obfuscation techniques have been discussed in Zaidi's paper[27] 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 4.5 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[1] 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 4.5 has shown interesting capacities to block malicious senders based on user input but has not proven to be effective against our more narrowed phishing experiment. Moreover, an attacker may try to build a good reputation for a domain before starting to send phishing emails. A possible countermeasure may be to keep senders' reputation specific to each email account. For this reason, it is important to bear in mind the impact of end users on the effectiveness 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 spam filters are more concerned about email content than sender authenticity. However, we have shown that implementing these authentication protocols with newly registered domains helps lower the email score attributed by spam filters, which in some cases is enough to avoid detection. 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. Further work could also measure and discuss the impact of our solution using different top-level domains (TLD) instead of subdomains. 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

[1] Saeed Abu-Nimeh et al. “A comparison of machine learning techniques for phishing detection”. In: *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*. ACM. 2007, pp. 60–69.

[2] Ammar Almomani et al. “A survey of phishing email filtering techniques”. In: *IEEE communications surveys & tutorials* 15.4 (2013), pp. 2070–2090.

[3] *Apache SpamAssassin*. URL: https://spamassassin.apache.org/.

[4] KC Cross et al. *Anti-spoofing protection in Office 365*. June 2018. URL: https://docs.microsoft.com/en-us/office365/securitycompliance/anti-spoofing-protection.

[5] Estelle Derouet. “Fighting phishing and securing data with email authentication”. In: *Computer Fraud & Security* 2016.10 (2016), pp. 5–8.

[6] Zakir Durumeric et al. “Neither snow nor rain nor MITM...: An empirical analysis of email delivery security”. In: *Proceedings of the 2015 Internet Measurement Conference*. ACM. 2015, pp. 27–39.

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

[8] Ian D Foster et al. “Security by any other name: On the effectiveness of provider based email security”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2015, pp. 450–464.

[9] *G Suite*. URL: https://gsuite.google.com/.

[10] BB Gupta, Nalin AG Arachchilage, and Kostas E Psannis. “Defending against phishing attacks: taxonomy of methods, current issues and future directions”. In: *Telecommunication Systems* 67.2 (2018), pp. 247–267.

[11] Hang Hu, Peng Peng, and Gang Wang. “Towards understanding the adoption of anti-spoofing protocols in email systems”. In: *2018 IEEE Cybersecurity Development (SecDev)*. IEEE. 2018, pp. 94–101.

[12] Hang Hu and Gang Wang. “End-to-end measurements of email spoofing attacks”. In: *27th USENIX Security Symposium (USENIX Security 18)*. 2018, pp. 1095–1112.

[13] Hang Hu and Gang Wang. “Revisiting Email Spoofing Attacks”. In: *arXiv* (2018).

[14] *Importance markers in Gmail - Gmail Help*. URL: https://support.google.com/mail/answer/186543?hl=en.

[15] Yonathan Klijnsma. *Another Victim of the Magecart Assault Emerges: Newegg*. Sept. 2018. URL: https://www.riskiq.com/blog/labs/magecart-newegg/.

[16] A Malatras, I Coisel, and I Sanchez. “Technical recommendations for improving security of email communications”. In: *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2016 39th International Convention on*. IEEE. 2016, pp. 1381–1386.

[17] *Microsoft Office*. URL: https://www.office.com/.

[18] *MxToolbox*. URL: https://mxtoolbox.com/EmailHeaders.aspx.

[19] Jose Nazario. *The online PhishingCorpus*. URL: https://monkey.org/~jose/phishing/.

[20] *Office 365 Anti-Spam IP Delist Portal*. URL: hhttps://sender.office.com/.

[21] *Rspamd features*. URL: https://rspamd.com/features.html.

[22] *Rspamd spam filtering system*. URL: https://rspamd.com/.

[23] *Secure email: ProtonMail is free encrypted email*. URL: https://protonmail.com/.

[24] Matt Simerson. *msimerson/mail-dmarc*. Oct. 2018. URL: https://github.com/msimerson/mail-dmarc.

[25] Bradley Taylor. “Sender Reputation in a Large Webmail Service.” In: *Conference on Email and Anti-Spam (CEAS)*. Vol. 27. 2006, p. 19.

[26] Gregory L Wittel and Shyhtsun Felix Wu. “On Attacking Statistical Spam Filters.” In: *First Conference on Email and Anti-Spam (CEAS)*. 2004.

[27] Shahrukh Zaidi. *Bypassing Phishing Filters*. 2018.

APPENDIX A

Phishing emails

A.1 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. This email has been extracted from Jose Nazario's 2017 phishing email list[19].

```
We reserve the right, at our sole discretion, to change, modify or otherwise alter your account
    at any time.
Therefore your account has been blocked.

Click here to confirm and restore your Craigslist account.
<http://****>

Thank you for using craigslist!
```

A.2 JIRA

This email pretends to be sent by a JIRA administrator who invites the user to log in to their account by clicking one of the two malicious links. This email template is the same as the original JIRA account invitation email of 2018.

```
Dear user,

Your Administrator has set up a JIRA account for you! JIRA is the project tracker for teams
    doing great work.
Log in now to track your issues and tasks, collaborate with your team, and stay on top of
    project activity.

Get started by setting your own password and logging in.

Set my password
<http://****>

This password reset request is valid for the next 24 hours.

Don't worry. You can always ask for a new password using the following link:
<http://****>
```

A.3 Google

This email is very similar to the typical email a Google user may receive when a new device has signed in to their account. To review the recent activity, the user is invited to click a link pointing to a malicious address. This email template is the same as the original Google "Security Alert" email of 2018.

```
New device signed in to your account
Your Google Account was just signed in to from a new Windows device. You're
getting this email to make sure it was you.



Check activity
<http://****>
<http://****>

You received this email to let you know about important changes to your
Google Account and services.
   2018 Google LLC,1600 Amphitheatre Parkway, Mountain View, CA 94043, USA
```

A.4 File Transfer

In this email, the attacker impersonates a company's internal File Transfer system. The email pretends to follow up on a reset password request and invites the user to confirm the request. Another malicious link is included, in case the user realizes no password reset was requested. This email template is the same as the original email sent by the company's File Transfer system in 2018.

```
**** File Transfer

Hello **** File Transfer user,

A password change request for your account was requested. You can use the following link to
    confirm your request and create a new password:
<http://****>

If you did not request a password reset, we need you to confirm your email address using the
    following link:
<http://****>

Please note that these links will expire 24 hours from the time it was sent.
```

A.5 DHL

This email pretends to be DHL and provides a user with a fake tracking number and a link to pick up their parcel. The attacker hopes to fool the user into clicking the malicious link by pretending to be a manager of the company. This email has been extracted from Jose Nazario's 2017 phishing email list[19].

```
Hello,

Your Package has Arrived, Click on the link to Track status and pick up your Package with your
    tracking number below

TRACK YOUR PACKAGE
<http://****>

TRACKING NUMBER
  5464733839842

Best Regards,
Mr. Mark Williams
Shipping Department Manager

DHL Express Ng
```

APPENDIX B

Email generation script

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 whose ID is given as first argument.

```
#!/bin/bash

if [ "$#" -lt 4 ] || [ "$1" = "-h" ] || [ "$1" = "--help" ]; then
    echo -e "Usage: $0 <container ID> <content.txt> <content.html> <recipient>|-f <recipients.
        txt> [-h headers.txt]\n"
    echo -e "\t- content.txt: email body that will be encoded in base64"
    echo -e "\t- content.html: email body that will be encoded in quoted-printable"
    echo -e "\t- recipients.txt: a list of recipients, one recipient per line"
    echo -e "\t- headers.txt (optional): a list of headers to append, one 'HEADER: VALUE' per
        line; double quotes must be escaped\n"
    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 -e "--$boundary\nContent-Type: text/plain; charset=\"UTF-8\"\nContent-Transfer-Encoding
        : base64\n" > body
    base64 < "$1" >> body
    echo -e "\n--$boundary\nContent-Type: text/html; charset=\"UTF-8\"\nContent-Transfer-
        Encoding: quoted-printable\n" >> body

    if ! hash qprint 2>/dev/null; then
        apt-get install -y qprint
    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 \"Content-Type: multipart/alternative; boundary=
            $boundary\" -s \"$subject\" $rcpt < body"
        eval docker exec -i $1 $cmd
    done < "$5"
else
    echo "Sending email '$subject' to <$4>..."
    cmd="mail -r $return_path $headers -a \"Content-Type: multipart/alternative; boundary=
        $boundary\" -s \"$subject\" $4 < body"
    eval docker exec -i $1 $cmd
fi
```