9/10/2021

Matthew Tan

Security Tools Lab 2

Project 4 Email Spam Filtering

# Introduction

Phishing is a type of online scam where criminals impersonate legitimate organisations via email in order to steal sensitive information about a user (Webroot, 2021). There are several ways for a non-technical user to identify a phishing email. Firstly, they can check if the message is sent from a public email domain. Most organisations tend to use their own email domain name when sending an email `sleepy@google.com` instead of `sleepy@gmail.com`. Secondly, they can check to see if the domain name has been misspelt. Some hackers will buy domain names that look very similar to legitimate ones in order to trick users. One example of this would be `sleepy@techmedia.com` and `sleepy@techrnedia.com`. The first instance uses `m-e-d-i-a` while the second instance uses `r-n-e-d-i-a`. At a glance, the two emails look exactly the same, however upon closer inspection one is a legitimate email while the other has a misspelt domain name (Irwin, 2021).

Educating non-technical users on how to identify phishing email is an important way of dealing with the issue, however, it would be better if we could stop these emails from even reaching their inbox. Enter spam filters, these are programs that are used to detect unsolicited emails and prevent them from entering a user’s inbox. These filters include features such as scanning and analysis of an emails contents, authentication of sender, policy validation and statistical analysis. This will provide us a rough idea on the legitimacy of an incoming email allowing us to handle it appropriately.

For this study, I wanted to examine spam filters and asses what criteria they used to determine if an incoming email is classified as spam. Due to difficulties of experimenting with proprietary solutions, I decided to focus mainly on Apache SpamAssassin and Rspamd. When carrying out the experiment, I decided to focus on using the email headers in order to determine the legitimacy of an email. At the end of the study, I wanted to be able to craft a phishing email that would by-pass the spam filter of known email service providers like Office365 of Gmail.

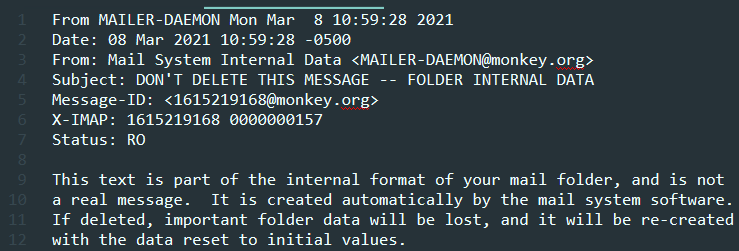
# Method

## Email Gathering

In order to test out SpamAssassin and Rspamd, I needed to gather a large data set of phishing emails that I could feed into the spam filter. I chose to use a dataset created by Jose Nazario that was created for the purpose of tackling phishing emails (Nazario, 2020). Jose had organised his emails based on the year they were received and the dataset ranged from 2005 to 2020. I decided to focus on emails from 2015 to 2020 as it was the most recent dataset available. I felt that spam filters have evolved over time and I would only get the most accurate results by testing the latest phishing emails.

## Email Parsing

After gathering all the emails, I noticed that it was formatted as a huge text file. This made it hard for me to analyse and process each individual email. Therefore, I wrote a script that would go through the entire file and extract out each email header placing it in its own text file. This also gave me the opportunity to remove the body of the email since we will be ignoring that aspect for this study. When looking through the huge email file, I noticed that there were no empty lines inside the email header at all.

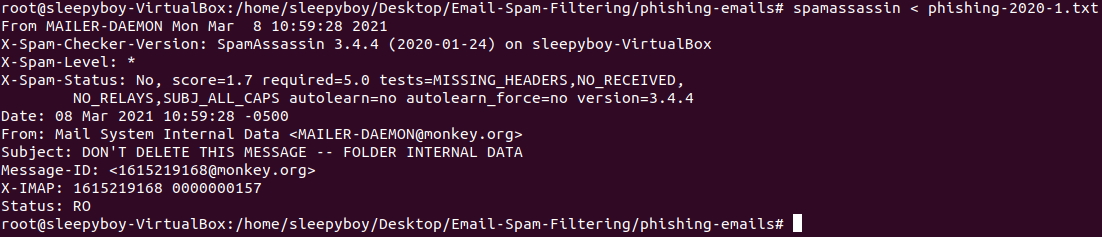


This means that in order to capture an email header, I can go through the file line by line, start recording when I reach a FROM statement and stop recording when I reach an empty line. At the end of this process, we ended up with 1,611 unique phishing emails that I could feed into our spam filter program.

## SpamAssassin

Apache Spam Assassin claims to be the number one open-source anti-spam platform giving system administrators a filter to classify emails and block unsolicited bulk emails. It uses a robust scoring framework to integrate a wide range of advanced heuristic and statistical analysis tests on email headers and bodies including text analysis and DNS block lists (Spamassassin, 2021).

I installed the latest stand-alone version of SpamAssassin from their website and installed it onto my Ubuntu Virtual Machine (VM). I then wrote a python script that made use of the python subprocess library that would loop through the directory of mail.txt files and run the Spamassassin command in order to check our entire dataset of emails.

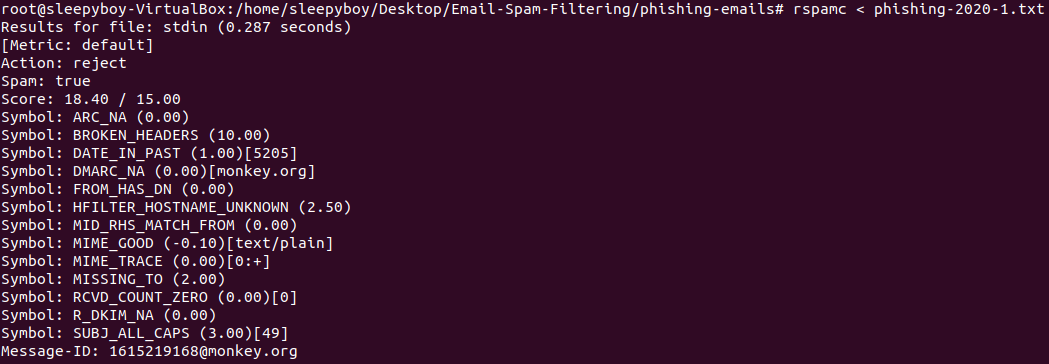


Once I obtained the output of the command, I parsed it and extracted the X-Spam-Level & X-Spam-Status.

## Rspamd

Rspamd is an advanced spam filtering system and email processing framework that allows evaluation of messages by a number of rules including regular expressions, statistical analysis and customer services such as URL black lists (Stakhov, 2021).

I installed the latest stand-alone version of Rspamd on my Ubuntu VM. I then wrote a python script that made us of the python subprocess library to loop through the directory of mail.txt and run the rspamc command in order to check our entire dataset of emails.

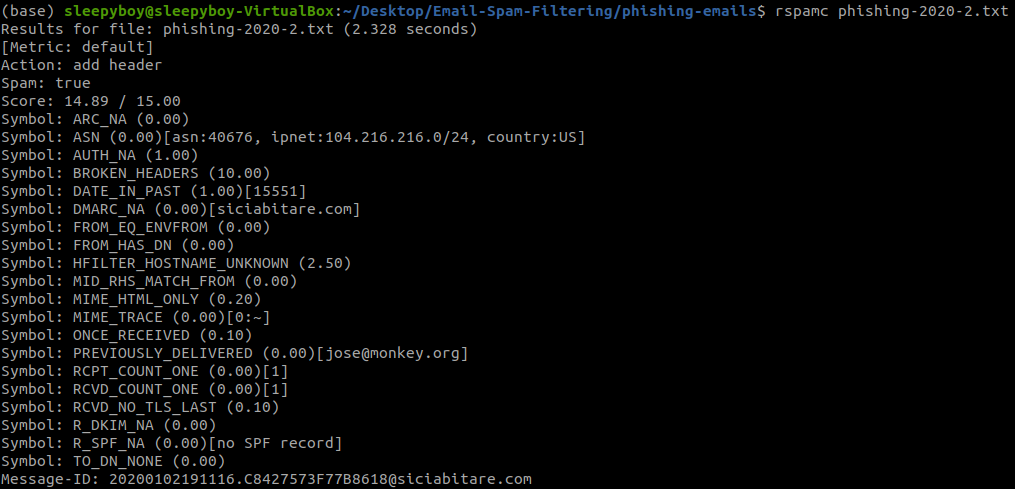


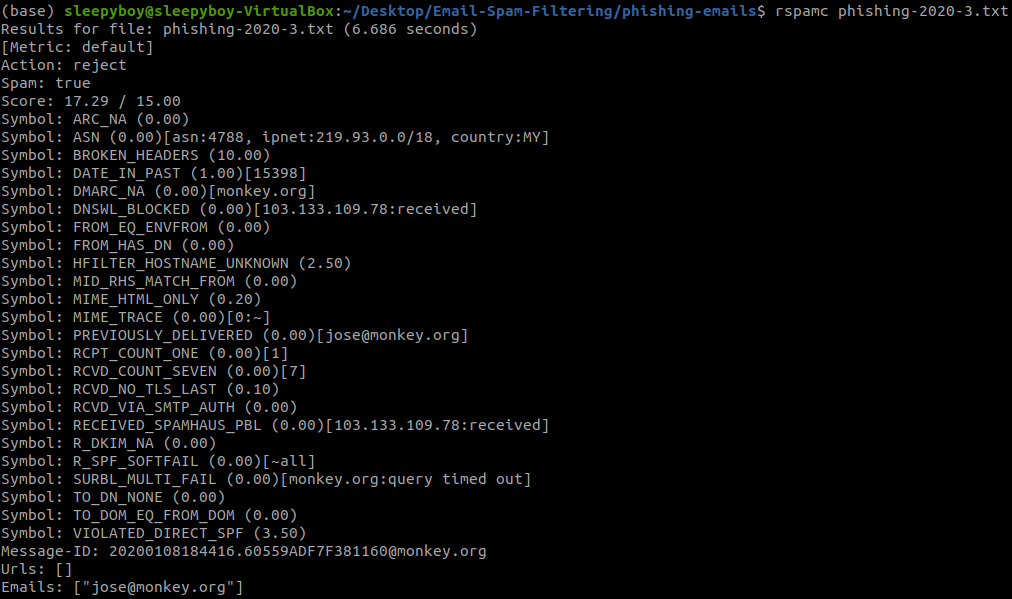
Once I obtained the output of the command, I parsed it and extracted out the crucial fields that impacted the score.

# Issues Faced

## Output Fields in Rspamc

One of the issues I faced when using Rspamc was that the results of each email had its own unique fields. For example, the result of phishing-2020-2.txt has several different fields as compared to the results of phishing-2020-3.txt





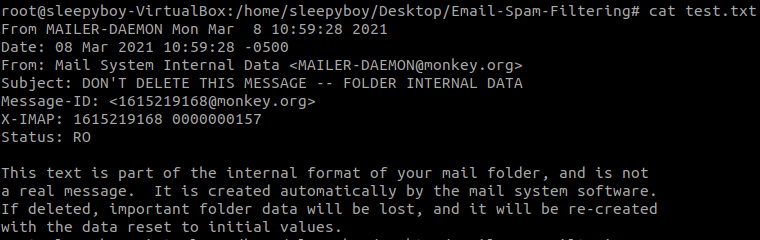
Some of these differing fields include `VIOLATED\_DIRECT\_SPF`, `R\_SPF\_SOFTFAIL`, `DNSWL\_BLOCKED` just to name a few. This made processing the data a little bit more complicated as I had to factor in many different fields into my calculation. Listing down all the possible fields would have made the processing extremely complicated. So, I decided to focus specifically on fields that affected the score provided by rspamc. For this particular example, that would mean I ignored `R\_SPF\_SOFTFAIL` and `DNSWL\_BLOCKED` as they did not impact the score at all.

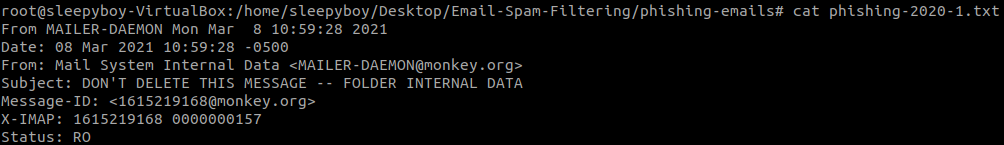
My decision may have resulted in some key information that is not being logged in my csv, however, my csv file still contains the score and I can use that to easily identify if I have left out any crucial fields in my logging process.

## Removing Email Body

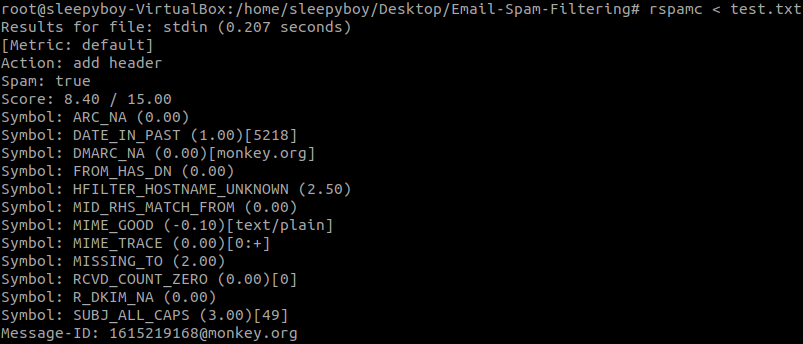
When running the rspamc test on all the different emails, I noticed that all the result sets had the `BROKEN\_HEADER` field which would award a massive 10 points to all the emails. I was curious if this was because I removed the body of the email so I decided to do a test.

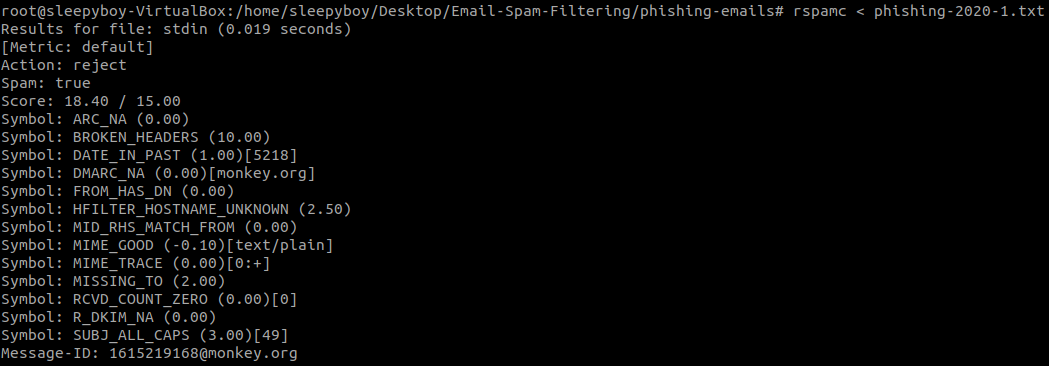
I decided to test two identical emails one with an email body and one without. I wanted to see if removing the body resulted in the `BROKEN\_HEADER` field being added.





This was the result of my experiment.





As you can see, the BROKEN\_HEADERS field only appeared on the email with no body. I also noted that the score difference between the two emails was 10 points which means that the only difference between the two emails is that one field.

This makes things easier as I can simply factor `BROKEN\_HEADERS` into my calculation and remove 10 points from all my findings in order to obtain and accurate result.

We noticed the exact same results with fields like `MIME\_MA\_MISSING\_HTML` and `MIME\_MA\_MISSING\_TEXT` and have taken measures to account for these fields.

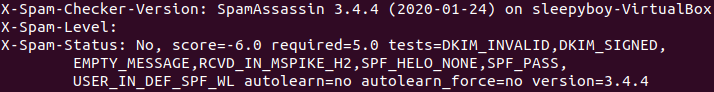
# Results

After taking a look at the results of the email dataset, this was my findings.

## SpamAssassin

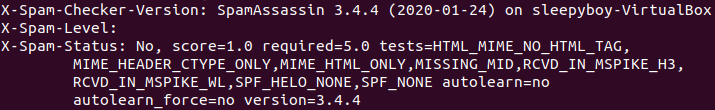
### Lowest Scores

The lowest spam score came from phishing-2020-112.txt with a spam score of -6. When looking through some of the values in the X-Spam-Status, we noticed that the email had an invalid dkim signature, but had an average reputation on mail spike. Lastly, the sender matches the SPF record.

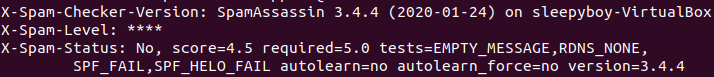


Looking at the entries with the lowest score, I can see that most of these mails have an invalid dkim signature and the sender matches the SPF record. We can see that SpamAssassin likely puts a lot of emphasis on these criteria.

Analysing phishing-2015-95.txt, I can see that it has a spam score of 1. When looking through some of the values in the X-Spam-Status, we noticed the MIME\_HEADER\_CTYPE\_ONLY which means that the ‘Content-Type’ is found without the required MIME headers. They also mentioned that the Message-ID is missing and the sender has a good reputation on mail spike. I also noted that this email scored relatively well even though the sender did not publish an SPF record.



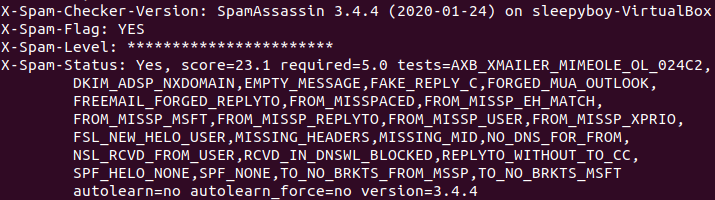
Analysing phishing-2016-18.txt, I can see that it has a spam score of 4.5. When looking through some of the values in the X-Spam-Status, we noticed that the email is delivered to the internal network by a host with no rDNS, the sender does not match the SPF record and HELO did not publish an SPF record. It is quite surprising that simply failing the spf check and HELO not publishing an SPF record could affect the score so drastically.



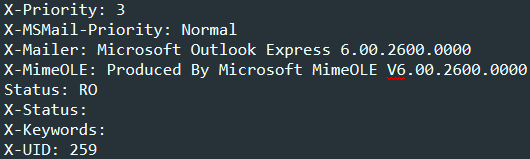
### Highest Scores

Looking through the results of SpamAssassin, we noticed that a lot of the results does not really show what went wrong during the test. Therefore, we decided to look at the emails with the highest scores in order to gain a deeper insight into what went wrong.

The highest spam score came from phishing-2015-259.txt with a spam score of 23. Taking a look at the results, it mentioned `AXB\_XMAILER\_MIMEOLE\_OL\_024C2`

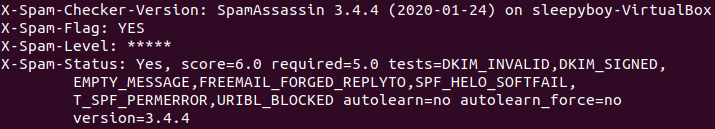


Taking a closer look at the email header, we noticed that the X-Mailer field is `Microsoft Outlook Express 6.00.2600.0000` and the X-MimeOLE field is `Produced By Microsoft MimeOLE V6.00.2600.0000`.



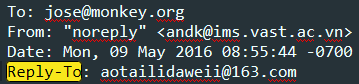
After doing a little research, I found a forum post that mentioned those are insecure OE versions and should be upgraded (AXB, 2012). Checking the data of the email, I also realised that it was from 2015 which was one of the older mails in my dataset and could have been using an outdated OE version.

Analysing phishing-2016-208.txt, I can see that it has a spam score of 6.0. When looking through some of the values in the X-Spam-Status, we noticed that the email had an invalid dkim signature

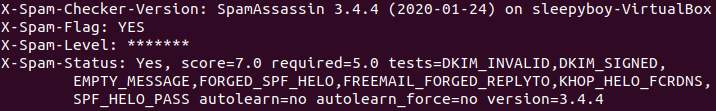


Looking through the actual email, we notice the reply-to section has been forged. The Return-Path differing from the Reply-To may indicate that this email is forged. Especially since the domain of the emails do not match.





Analysing phishing-2016-384.txt, I can see that it has a spam score of 7.0. When looking through some of the values in the X-Spam-Status, we noticed that the email had an invalid dkim signature, a forged spf helo, forged replyto and the Relay HELO differs from its IP’s reverse DNS.



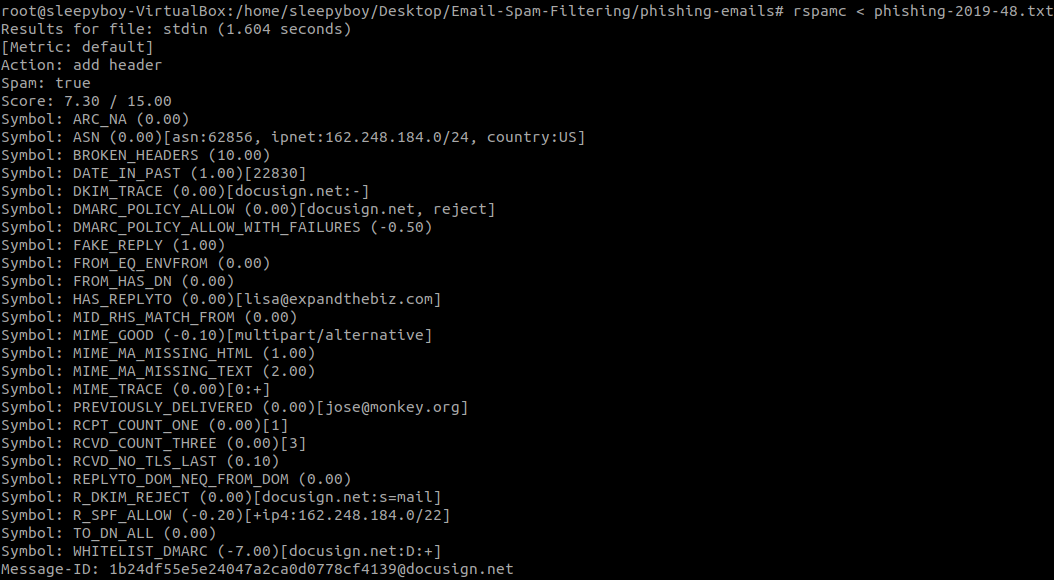
### Conclusion

In conclusion, I can determine that having an invalid dkim-signature does not really affect the spam score given. Also, the sender not having a published SPF record can be overlooked. However, things like having an outdate X-Mailer or forging the reply-to section of an email can be crucial features for identifying an email as spam.

## Rspamd

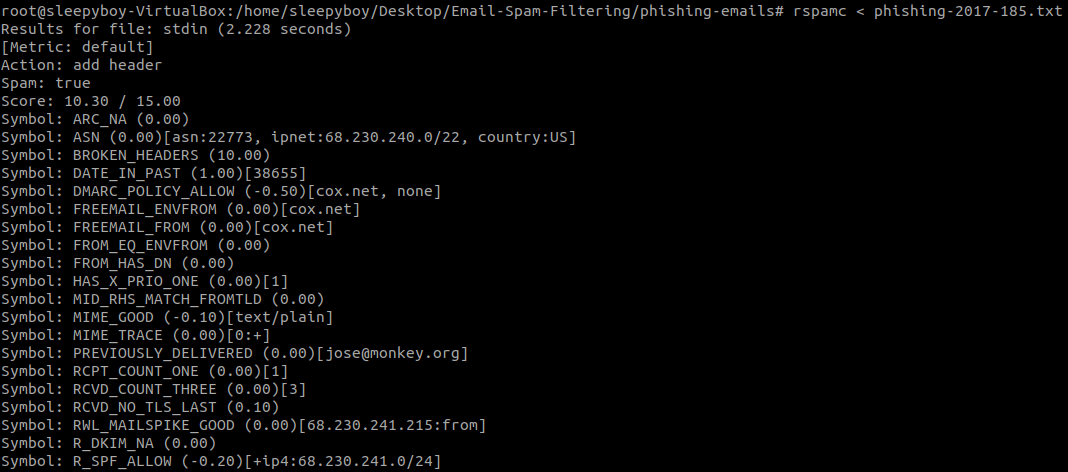
### Lowest Scores

The lowest spam score came from phishing-2019-48.txt with a spam score of -6.3. This value is very interesting as I did not expect a negative value coming from Rspamd. Taking a closer look, I noticed that there was a field WHITELIST\_DMARC which removed 7 points from its score.



This shows that having your email originate from an address that is published on a DMARC TXT record, can significantly increase the chance of your email not being detected as spam. However, this criterion is quite unreasonable as it hard for a malicious party to accomplish this feat.

The second lowest spam score came from phishing-2017-185.txt with a spam score of 0.30. Taking a closer look, we noticed that it had fields like `DMARC\_POLICY\_ALLOW` and `R\_SPF\_ALLOW` had deducted a significant number of points from the final score.

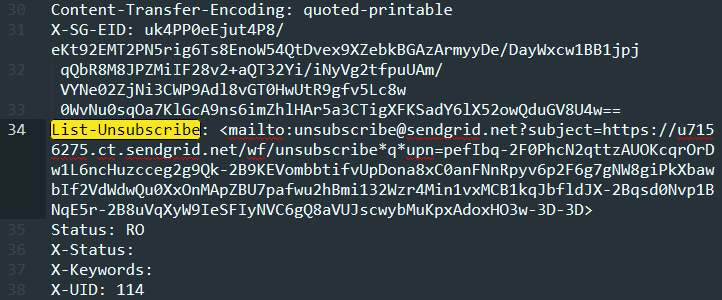


### Overall

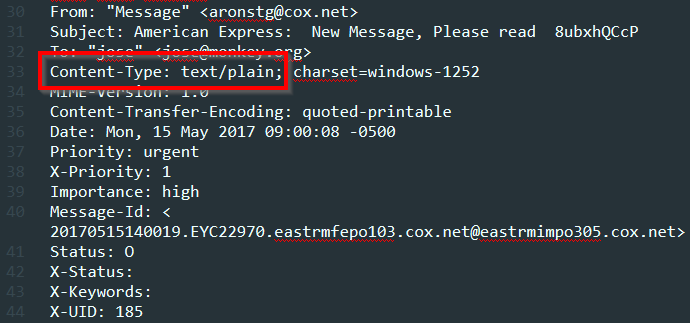
Looking at the broad picture, we noticed that were some fields that were contributing points to the spam score that could easily be rectified.

Firstly, there was a field called `DATE\_IN\_PAST` which added 1 point if the current date is much greater than the received date. This is an issue with the emails in our dataset being over a year old and this issue can easily be rectified by creating and sending a new email.

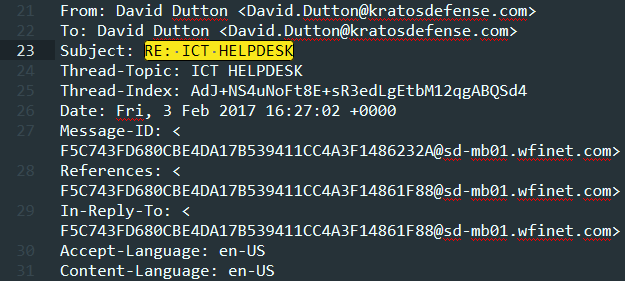
Secondly, there was a field called `HAS\_LIST\_UNSUB` which deducted 0.01 points if there was a List-Unsubscribe field in the header. I could try using any of the List-Unsubscribe fields in my test email to see if it would reduce the score.

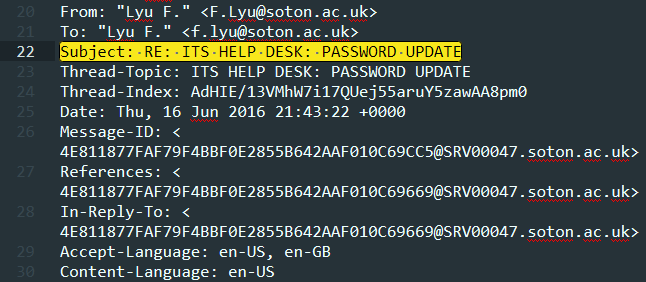


Thirdly, there was a field called `MIME\_GOOD` which deducted scores based on the Content-Type specified in the email header. Content-Types like `text/plain` or `multipart/alternative` deducted 0.10 points while Content-Types like `text/html` received nothing.

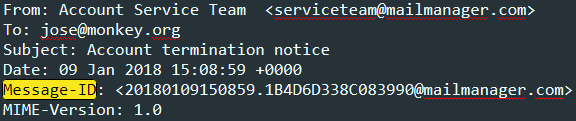


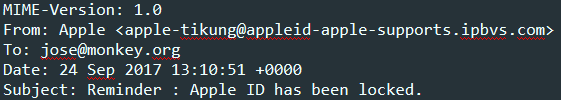
Fourthly, there was a field called `SUBJ\_ALL\_CAPS` which added points if a subject was completely capitalised. The points added ranged from 0.90 to 2.25 and could easily be avoided by only capitalising the first character of every word.



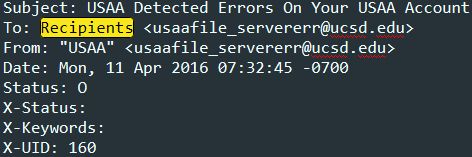


Fifthly, there was a field called `MISSING\_MID` which added 2.50 points if the Message-ID field was missing in the email header. This is quite a huge point increment therefore it is important that we try to add a Message-ID onto our spoofed emails. For reference, I have attached two email headers, one with a Message-ID and one without.

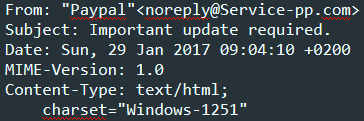




Sixthly, there was a field called `TO\_DN\_RECIPIENTS` which added 2.0 points if the header display name for `To` is Recipients. This is quite a huge point increment therefore it is important that we address our spam email to a specific target.



Lastly, there was a field called `MISSING\_TO` which added 2.0 points if there was a missing To field in the email header. This is also quite a huge point increment that could be easily avoided by focusing our spam email on a specific target.



### Conclusion

I would be able to significantly reduce the score of our email by following some simple steps

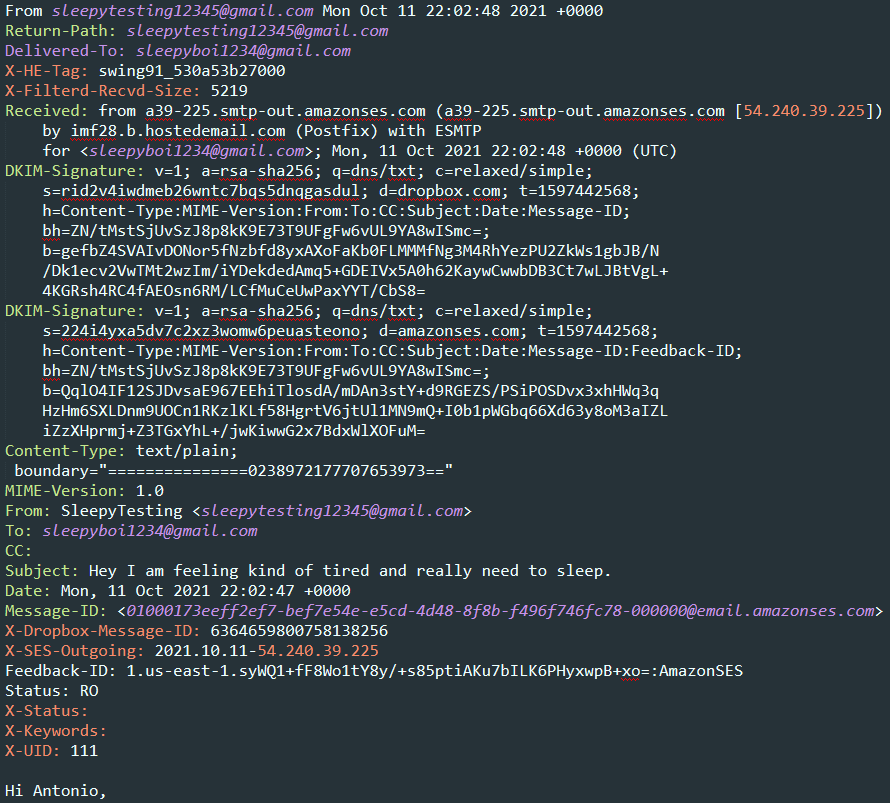
* Send a recent email
* Include a List-Unsubscribe field in the header
* Using a recognised content-type like `text/plain` in the email
* Ensure that the subject is not entirely capitalised
* Ensure that there is a Message-ID
* Ensure that the email is targeted at a specific person

Ultimately, looking at two of the lowest scoring emails, we can determine that the best way to get an email to bypass spam filters would be to generate a DMARC record at your DNS provider. This means that you would no longer be able to spoof your domain to impersonate another organisation, however, it would greatly increase the likelihood that your email would not be classified as spam.

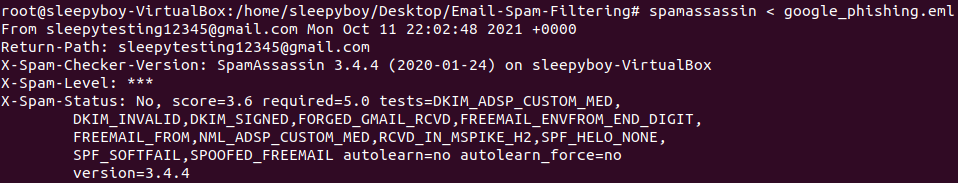
# Discussion

After researching all the different criteria SpamAssassin and Rspamc takes into account when judging if an email is spam, I wanted to try and create an email that would be able to fool their detection system.

When trying to create my email, I could not figure out how to create a forged dkim signature. I also felt that there were a lot of details that would be hard to replicate. Therefore, I decided use one of the low scoring email headers from my data set and edit some of the information. Firstly, I changed the from and to address to align with my use case. I also removed the reference to SPF as earlier in the study, I discovered that not having an SPF Record is better than failing the SPF test.



Before doing a live test, I decided to run the email files I created against Spamassassin and Rspamc in order to see how it would perform.



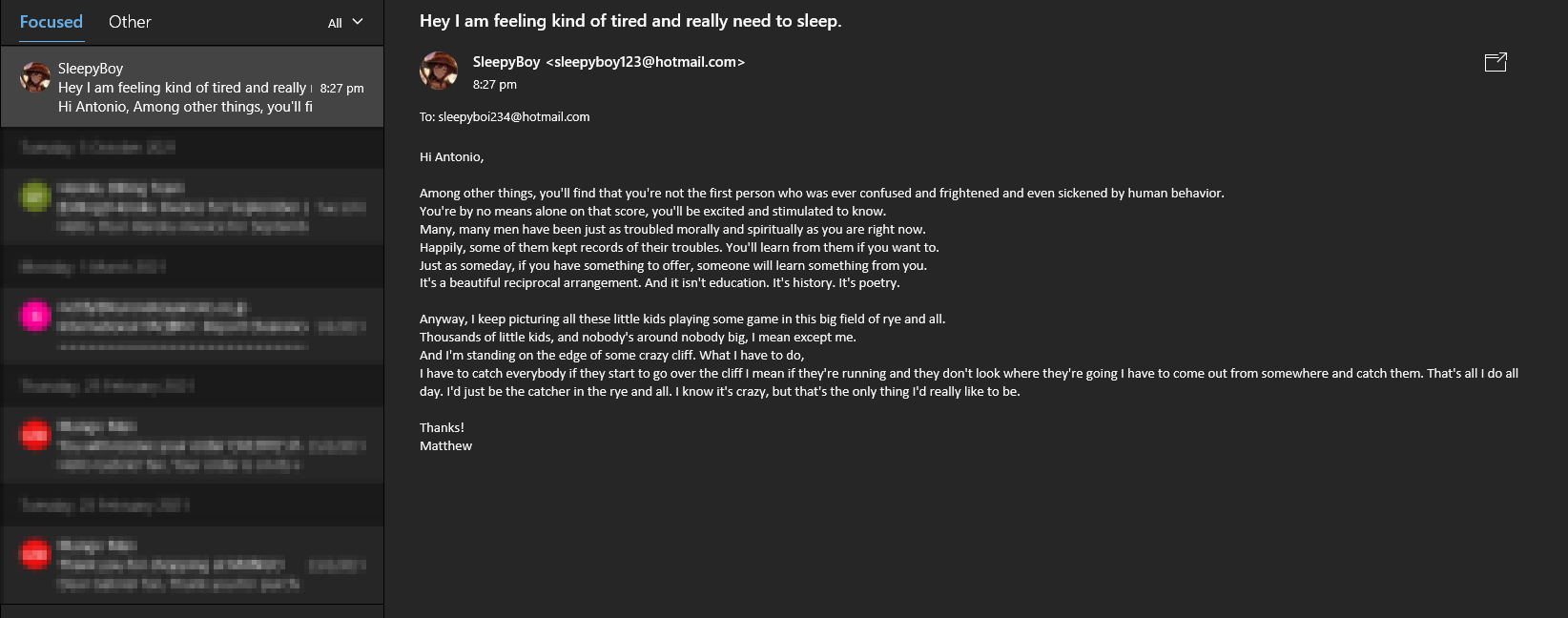
Looking at the results of the SpamAssassin test, my findings were correct. Even with an invalid dkim signature and a spf soft fail, I was still able to get a low enough score to avoid the spam filter.

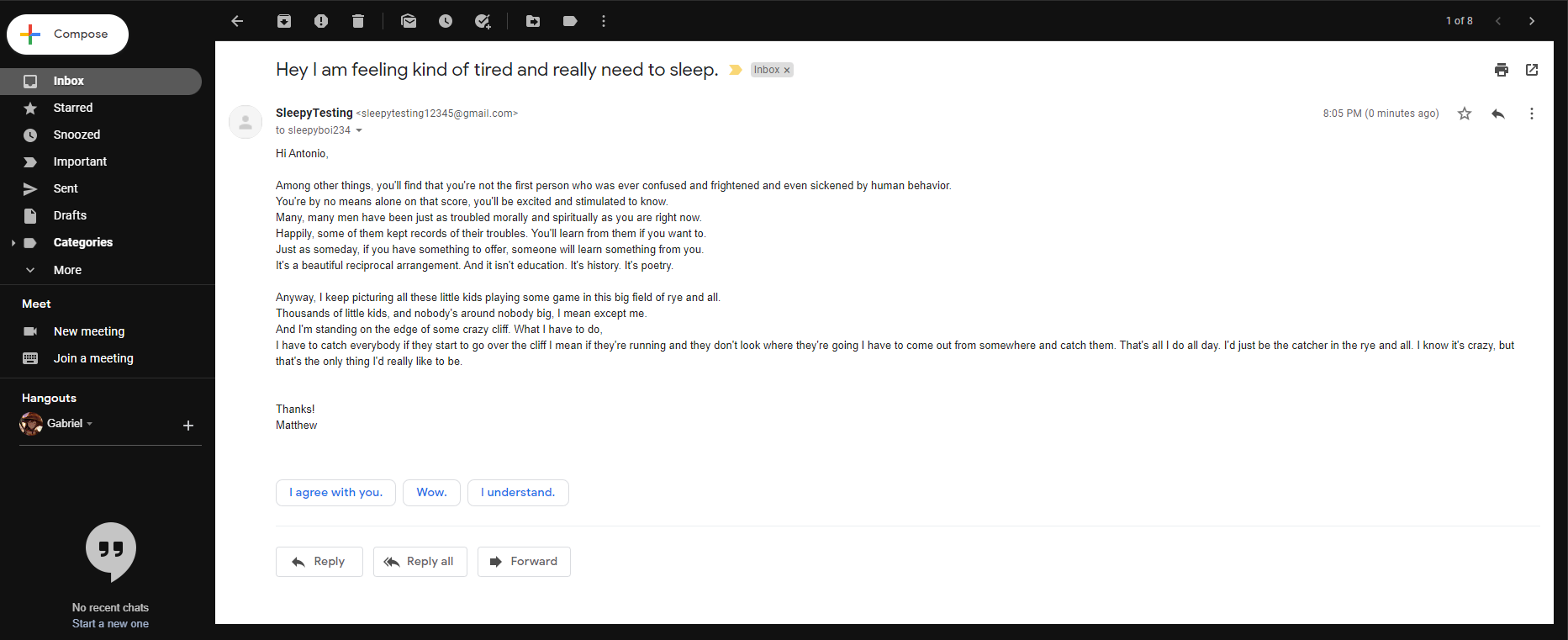


Looking at the Rspamc test, my findings were correct.

* I included an email body to avoid having `BROKEN\_HEADERS`
* I specified the Content-Type to be text/plain in order to get `MIME\_GOOD`
* I included a Message-ID in order to avoid getting `MISSING\_MID`
* I only used two capital letters in my subject in order to avoid ` SUBJ\_ALL\_CAPS`

Now moving onto the actual test, I wrote a python script that would read the .eml file and send an email to both my hotmail and gmail account. As you can see below, my message managed to avoid their spam filtering system and ended up in the inbox.





# Bibliography

Apache SpamAssassin: Welcome. (2021). Retrieved 12 October 2021, from https://spamassassin.apache.org/

Email Phishing, Vishing & Other Types of. (2021). Retrieved 12 October 2021, from <https://www.webroot.com/us/en/resources/tips-articles/what-is-phishing>

Irwin, L. (2021). 5 Ways to Detect a Phishing Email: With Examples. Retrieved 12 October 2021, from https://www.itgovernance.co.uk/blog/5-ways-to-detect-a-phishing-email

Nazario, J. (2021). Index of /~jose/phishing. Retrieved 12 October 2021, from <https://monkey.org/~jose/phishing/>

Stakhov, V. (2021). GitHub - rspamd/rspamd: Rapid spam filtering system. Retrieved 12 October 2021, from https://github.com/rspamd/rspamd

6844 – Almost-overlapping rules for "bad" OE version causing FP. (2021). Retrieved 12 October 2021, from https://bz.apache.org/SpamAssassin/show\_bug.cgi?id=6844