École Polytechnique Fédérale de Lausanne

# ActiveStrike: an automated email campaign launcher to uncover weaknesses in email filters

by Baptiste Jacquemot

## Master Thesis

Approved by the Examining Committee:

Prof. Dr. sc. ETH Mathias Payer
Thesis Advisor

EPFL IC IINFCOM HEXHIVE
BC 160 (Bâtiment BC)
Station 14
CH-1015 Lausanne

August 26, 2022

However, unless email is freed from dependence on the networks, I predict it will be supplanted by telefax for most uses in spite of its many advantages over telefax

— John McCarthy, Communications of the ACM, 32(12), pp. 1389-1390, December 1989

Dedicated to you, winner of this month lottery!

Congratulation! You were selected as a winner of this month. In order to claim your price, click this link here

# Acknowledgments

I would like to thank my colleagues at xorlab specially Allan Elleuch and Remi Meier

*Zürich, August 26, 2022* Baptiste Jacquemot

# Abstract

Email filters try to protect users from malicious email such as scam, spam, and malware. Over the years, techniques to detect such malicious email became more and more sophisticated such as using Bayesian theorem to detect spam, static and dynamic analysis to detect malware, OCR to detect scams. However, blocking malicious email is still a challenge today. To defend against attacks, one must know ones own weaknesses, and one way to find out is to make a pen testing tool. In this paper, we made ActiveStrike, a pen testing tool to automatize sending malicious email campaigns with a high variety of obfuscation choice. We will use this tool to test multiple email filters and compare them in to find obfuscations that bypass them. We will show that emails with a message and an attachment are the way to go for an attacker that want to deliver scam. We will also show that encryption and nested attachment are the way to go for an attacker who wants to deliver a virus. In this paper we will also design a metric to evaluate email packing danger.

# Contents

# Chapter 1

# Introduction

## 1.1 Email and Spam History

Despite being often associated with the internet, the history of email begin even before the internet. The first email was sent in 1971 between two computers of ARPANET, at that time TCP/IP was not even a standard in ARPANET.

Nonetheless, the history of spam is quite recent. The first commercial email spam message was sent in 1994, by Canter and Siegel, two Arizona lawyers looking for clients interested in obtaining a U.S. Green Card. After that, it didn't take long for people to understand the economic incentives of email and spam.

The first spam filtering solution begin to appear at the beginning the years 2000, with SpamAssassin (SA) being released by Justin Mason April 20, 2001. In 2002, Paul Graham published an influential paper, "A plan for spam", describing a spam-filtering technique using improved Bayesian filtering. This is still used today in well known mail filter such as Spam Assassin and Rspamd.

To bypass malware in email, static analysis was used as early as 2002, with the creating of ClamAV, a virus scanner. This malware scanner detects if the file is virus by comparing its binary signature with known virus signatures.

Malicious emails can be classified in multiple categories. For this thesis, we will classify them as such [10]:

- Malware email: email distributing malware using an attachment or a link
- Spam: email selling a product or service
- Scam: email deceiving users mostly with financial motives. This category can be split into

phishing, extortion and impersonation. Phishing steals credential, usually using a link. Extortion threatens the receiver. A common example of this scam is a hacker telling that he has access to your computer and wants to be paid in bitcoin in exchange for not releasing information. Impersonations use the identity of another person or company to get access to confidential information.

We will give some definition of the term we use in this thesis: in this paper, *unwanted email* and *malicious email* are similar. When talking about *campaigns* or *email campaign*, we refer to a sending a load of emails from a sender to a receiver.

## 1.2 Email Background

The Simple Mail Transfer Protocol (SMTP) is an internet standard communication protocol for electronic mail transmission. Modern SMTP was designed in 1995. At first, This protocol was only base on ASCII text. Standards such as *Multipurpose Internet Mail Extension* (MIME) was developed to allow exchanging binary files though SMTP. SMTP only defines the mail protocol but does not enforce the email header to be the same than the protocol. Hence, the sender envelope defined by the protocol can be different from the sender header defined in the email.

When sending a email over the network, the sender sends the email to a Mail Transfer Agent (MTA) via SMTP which relay the mail to a Mail Delivery Agent (MDA) via SMTP. The receiver can retrieve and manage emails on the MDA using protocol such as IMAP or POP3.

Common email security mechanics related to transport are SPF, DKIM, and DMARK. These are used in popular mail services such as Google Gmail and Microsoft Outlook 365.

- SPF (Sender Policy Framework) specifies the servers and domains that are authorized to send email on behalf of the sending organization. This is done by adding DNS record containing the email server IP address.
- DKIM: Add a digital signature to every outgoing message, which lets receiving servers verify that the message actually came from the sending organization.
- DMARK: Create policies for email that does not pass SPF and DKIM.

A well-known opensource MTA is Postfix. Postfix can be configured to include email filtering solutions. The most poplar freely available email filter are Rspamd and SpamAssassin. To also detect malware using static analysis, a freely available tool is ClamAV.

## 1.3 Thesis

*Thesis Question:* An attacker wants to deliver a malicious payload. What are the effective ways to package the payload in an email to bypass a mail filter?

To answer the question, we need to be more precise on the terms. The different mail filter that will be evaluated are Rspamd, SpamAssassin, Rspamd with ClamAV, SpamAssassin with ClamAV and ActiveGuard (AG). ActiveGuard is the filtering solution of xorlab AG, the company where I did this thesis. The tested features are described in the evaluation at chapter 5.

For this Thesis, we will test those mail filters with their basic configuration. The exact configuration can be found on Github [5][4][7][6].

We create an attacker profile with the following capabilities:

- The attacker use a domain that has SPF enforced. There is one exception with some impersonations. Suppose we want to impersonate domain $D$. Some of the impersonation email send by the attacker will contains $D$ as sender (in the header "envelope from"). Since the attacker does not have authorization to send email from $D$ then this email will be able to pass the SPF check.
- The attacker use SPF *young domain* which has no *reputation*. Let *young domain* be domains that where created less than three months before the attack. *reputation* means this domains was barley used before, and is not associated with any well know organization.
- The attacker does not use DKIM or DMARK.
- The attacker only use one IP address.

This paper brings multiple challenges: creating the tool for xorlab AG and make it usable. For more detail see section 2.2. Balancing the requirement between xorlab and the thesis, setting up an architecture for testing with xorlab AG infrastructure that can provide value to xorlab AG. Also finding the best way to generate email.

# Chapter 2

# Background

## 2.1 xorlab AG

To conclude my master degree at EPFL, I am doing my master thesis at xorlab AG. xorlab is a Zurich startup funded in 2015 by Antonio Baresi and Matthias Ganz.

"xorlab monitors email communication within your environment to understand the relationships that connect your organization with the outside world. By making this communication network explicit and learning legitimate human behavior, xorlab can identify threats that others miss."[1]

xorlab AG main product is ActiveGuard. ActiveGuard keep track of the relationship between organization in order to construct behavioral relationship and assess risk of every email against the context of the relationship with the email sender.

ActiveGuard use multiple spam mitigation techniques such as dynamics analysis, static analysis, OCR, link extraction, spam filtering techniques and more.

## 2.2 Motivation

My thesis project, ActiveStrike has multiple motivations:

- ActiveStrike can be used as a testing tool to find email attacks bypassing ActiveGuard
- ActiveStrike can also be used to test customer defense. This is a way to verify if ActiveGuard is configured correctly on customer side.
- Finally, The tool can be used to establish data to convince the prospect to buy. Say that a

prospect is interested in ActiveGuard. With their consent, we can send an email campaign using ActiveStrike to their current email solution. We can check what email have been blocked and then create a security from it. Then we can compare the detection performance report to what ActiveGuard would have obtained with the same email campaign to convince the prospect to adopt ActiveGuard

- From my point of view, the motivation for creating ActiveStrike is to answer the thesis question

# Chapter 3

# Design

## 3.1 Requirements

To design ActiveStrike, a set of requirements was given. Those requirements had the shape of user stories. I selected some user stories to present here. The following user stories give an idea of what ActiveStrike should look like. We define AU as the ActiveStrike User.

1. As an AU, I want the tool to be simple to use.
2. As an AU, I want to simulate state-of-the-art attacks against a prospect, so that I can accurately quantify the prospect risk of breach.
3. As an AU, I want the simulator to generate realistic emails.
4. As an AU, I want every generated email part to be configurable.

### 3.1.1 Simple to use

Simple to use means that a semi-technical user can launch effective simulations fast. A command-line interface and well-structured configuration files are fine.

To make it simple, we make ActiveStrike requires only a minimal set of input parameters:

- Name of the prospect organization, e.g. Coop
- Address of the recipient of the simulated attacks, e.g. kurt.rufer@coop.ch
- Selection of tactics to be used in the simulation

- Names of high-ranking members of prospect organization and business partners
- Names of potential business partner organizations
- Names of popular brands known in the context of the prospect

That requirement has issues. In particular it goes against another requirement which is the highly configurable requirement. To implement these two requirements, we designed a system with two levels of usability: the top level is easy to use and can be configured using the former parameters. At a lower level, a user can define in detail how each email is generated. We call the user that interact with this level an ActiveStrike Advanced User (AAU). The user interacting with the upper layer is the AU.

### 3.1.2 State-of-the-art attacks

State-of-the-art means attacks that use novel tactics (unknown file hashes, URLs not blacklisted, email delivery infrastructure not blacklisted).

Tactics to be covered:

1. Spoof display name to impersonate a high-ranking member of the prospect organization and business partners
2. Use of lookalike domains to impersonate prospect domain and business partners
3. Embed links for credential phishing
4. Embed links to deliver malicious files
5. Embed crypto tokens to collect the ransom
6. Attach files with static malware
7. Attach files with zero-days and exploits
8. Attach files that request network resources, spawn processes or write files/registry at run-time
9. Encrypt or nest payload to evade detection. The payload can be in any kind of attachment.
10. Use email sender authentication such as SPF.

### 3.1.3 Realistic Email

Realistic means emails that are convincing for an average user to act upon them.

To leave more space for configuration, we decided that the AAU is responsible for the realism of the email sent. Email realism is an important parameter while evaluating the strength of an email filter. A malicious payload in an email with a high realism will be more likely to be accessed by a user. Blocking realistic email has more impact than blocking unrealistic one. This will be discussed more in depth in the evaluation part chapter 5.

## 3.2 Architecture

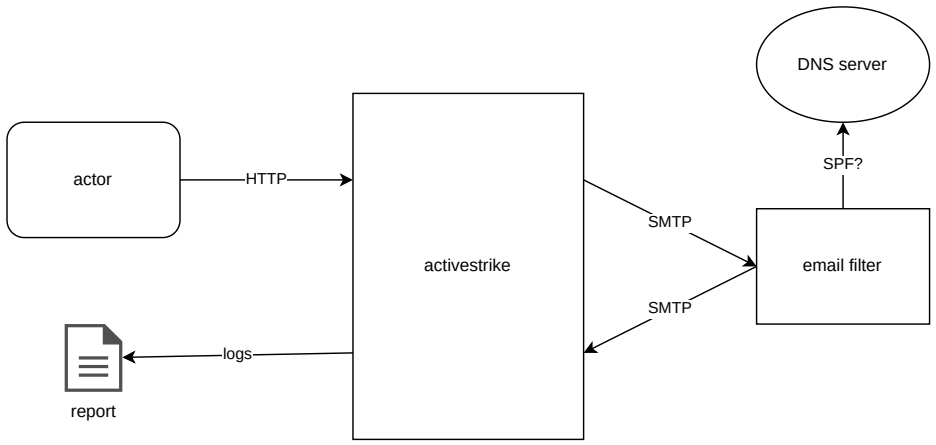


Figure 3.1: Evaluation architecture

The AU sends a campaign sending request to ActiveStrike. ActiveStrike then generates and sends campaign to the email server indicated in the campaign sending request. In this scenario, the email server send back the email campaign to ActiveStrike. ActiveStrike logs the emails and create a detection performance report of the launched campaign. To have email sender authentication, we create a domain with SPF for ActiveStrike.

## 3.3 Email generation

Let's take a look inside ActiveStrike.

To create multiple emails from one payload with different type of packing. We decided to generate a campaign based on two elements: payloads and email template. The payload defines the content of the email. The email template defines what we called the packing. When given

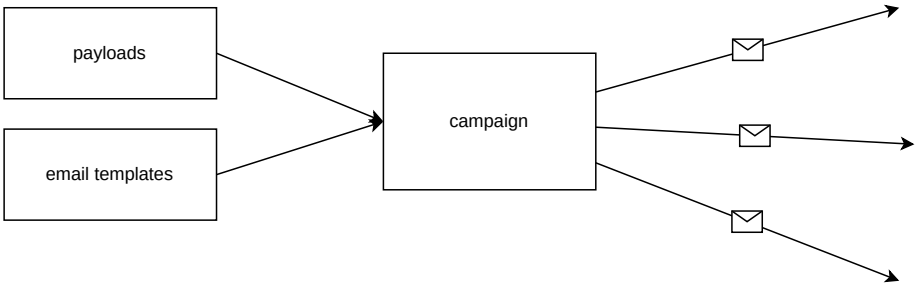


Figure 3.2: email generation

multiple payloads to one email template, ActiveStrike will generate as much email as there is payload.

Once generated, emails are sent according to the information given in the template. This includes the target domain, the target email address, the target port and if we use TLS with SMTP or not.

## 3.4 Email Template

We represent an email template as a tuple: $(payloadType, packing)$

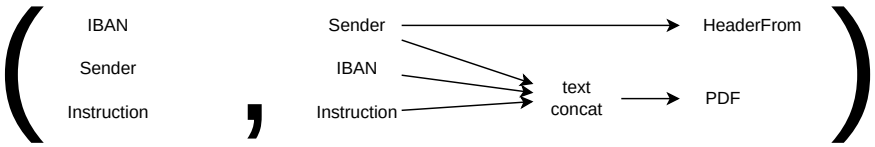


Figure 3.3: Example of an extortion email template

The payload type is the type of payload the email contains. The packing contains the instruction to create the different part of an email.

In this example, this email template has the structure of an extortion email. In an extortion, the attacker asks a target to send money to a given IBAN. In a common extortion attack, the attacker pretend that he has access to some of the victim's private information, and threatens to

release it if the target does not send money. In this example, the threat would be included in the $Instruction$.

In this packing example, the Sender is put in the header *From* of the email. The Sender, IBAN, and instruction are concatenated inside a PDF.

We model emails in such a way: emails has three kinds of content: attachments, headers, and body. The packing is a list of generators, each generator generate any of the three kind of content mentioned before. All generated content are then merged to create an email.

In the image below, you can see a screenshot of an extortion email send by ActiveStrike. The packing is not the same than the one described above. In this packing, the entire payload is put into the body as plain text.

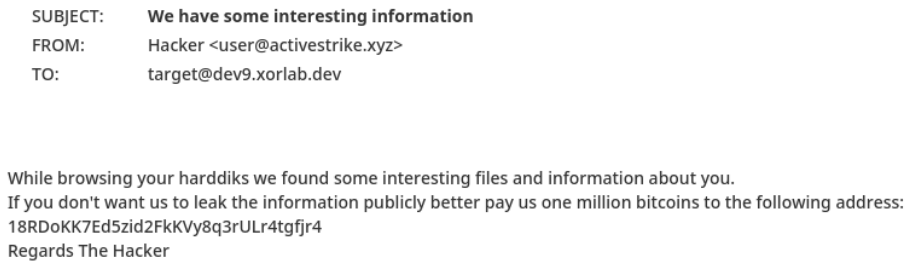


Figure 3.4: Screenshot of an extortion email sent by ActiveStrike

## 3.5 Attachment generation

To deliver state of the art attacks, we decided to generate email attachments in two ways: statically and dynamically.

### 3.5.1 Static generation

Generating attachments statically fulfill requirements 6, 7, 8 of 3.1.2.

To include files inside an email, a file path must be taken by the email template, then this email template will load the file at the given path and include it into the email. Here is an example of loading a DOCX document.

$$FILE\_PATH \xrightarrow[filePathToDocx]{} DOCX$$

Any type of file is supported. The following post processing can be made in the packing once the file loaded:

- Place the file as an attachment
- Place the file in a zip
- Place the file in an encrypted zip. The password is part of the email payload.

### 3.5.2 Dynamic generation

Generating attachments dynamically fulfills the requirement 9 of 3.1.2.

The following gives an example of a dynamic generation.

$$IBAN \xrightarrow[textToPdf]{} PDF$$

Here an IBAN is packed into a PDF. The supported output formats for this type of generation are the following:

- Text to PDF
- Text to encrypted PDF
- Text to DOCX
- HTML to image

For the PDF generation, each line in the text will be a line in the PDF. Moreover to add credibility to the PDF and DOCX we decided to give the possibility of a trailing image at the end of the document.

To create an image from HTML, we use HTML 3.2 with limited functionality: images in the HTML cannot be resolved and script cannot be executed.

# Chapter 4

# Implementation

In this section, we will give a brief overview of the techniques and technologies used to create ActiveGuard.

## 4.1 General view

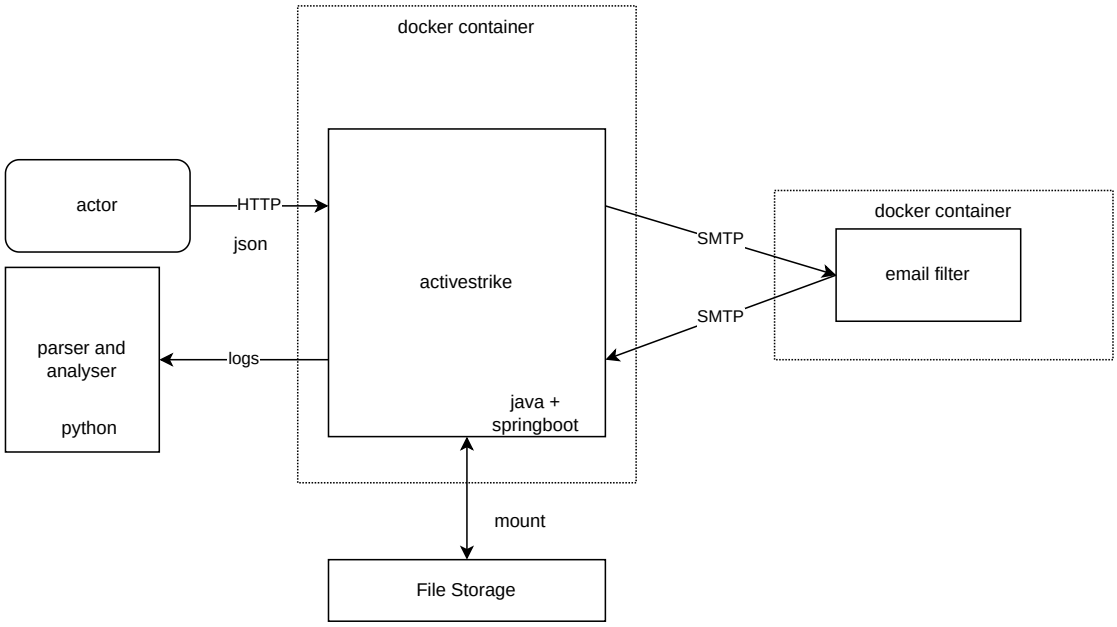


Figure 4.1: Architecture

ActiveStrike was written in Java with Spring Boot. We decided to use those tools to reuse code from ActiveGuard. ActiveGuard code base is also using Java and spring boot. The actor communicates via HTTP to ActiveStrike. The file describing the campaign is a JSON, we call it an

email campaign JSON. This JSON is parsed in ActiveStrike using Jackson. At the beginning of the project, an HTML interface was used instead. Because of the high frequency of redesign of email campaign structure and to the high cost of changing an HTML interface, The interface was made simpler by using email campaign JSON.

All error and results are logged using slf4j, a successor of log4j.

To make ActiveStrike easy to deploy and use, I used the company tools and process. I made three Jenkins jobs: build, deploy and launch:

- build: build ActiveStrike into a jar using gradle, and create a docker image using jib. At the same time pull malware and malicious files from xorlab AG binary repo, place JSON default campaign with those files, zip all this file to be used for deploy.
- deploy: use Ansible to set up a folder on the target instance, run the docker container from ActiveGuard docker image, and finally, unzip the previously mentioned zip into the file storage.
- launch: launch a default campaign against a given target. This is the high level interface used by AU.

Each mail received by ActiveStrike is logged. To analyze those logs and create statistic, a python script fetch the logs, filter the one of interest, get the email verdict or score and create statistic about the campaign.

For each email filtered tested other that ActiveGuard, I created a docker image containing a Postfix relay with the email filter in question. The email filters are Rspamd, Rspamd with ClamAV, SpamAssassin, SpamAssassin with ClamAV. Those docker container have to be deployed manually in ActiveGuard cloud.

ActiveGuard already have a deploy job in Jenkins, and thus can be deployed easily in xorlab AG cloud. However, quarantined emails are not sent to the destination, Not being able to read logs of quarantined email make it difficult to automatically analyze a campaign. Thus, I added rules in ActiveGuard to forward email to ActiveStrike if they are part of an ActiveStrike campaign.

# Chapter 5

# Evaluation

*Thesis Question:* An attacker wants to deliver a malicious payload. What are the effective ways to package the payload in an email to bypass a mail filter?

Here we will choose ActiveGuard, Rspamd and SpamAssassin, Rspamd with ClamAV, and SpamAssassin with ClamAV as email filter we want to bypass. As I discover during my thesis, it turned out that testing open source mail filter faster than testing an email infrastructure with high reputation such as Gmail or Microsoft Outlook 365, since open source mail filter can be tested locally for debug purpose. Moreover attacking Microsoft Outlook 365 or Gmail has risks. For instance the domain used for the attack can be banned. This could have repercussion for xorlab AG. Lastly gathering report of Microsoft Outlook 365 or Gmail require more processing to find what email has been quarantined and which one has been delivered. Nevertheless, given time to adapt to the previous mentioned challenges, ActiveStrike can be used for testing those email filters.

## 5.1 Evaluation Setup

In Figure 5.1 we send a campaign to Rspamd, SpamAssassin and ActiveGuard, The email is returned with a header containing either a *score* (for Rspamd and SpamAssassin) or a *decision*: quarantined or deliver (for activeguard). ActiveStrike will then log every header. A python script is responsible for fetching the log, and create a quality report of the campaign.

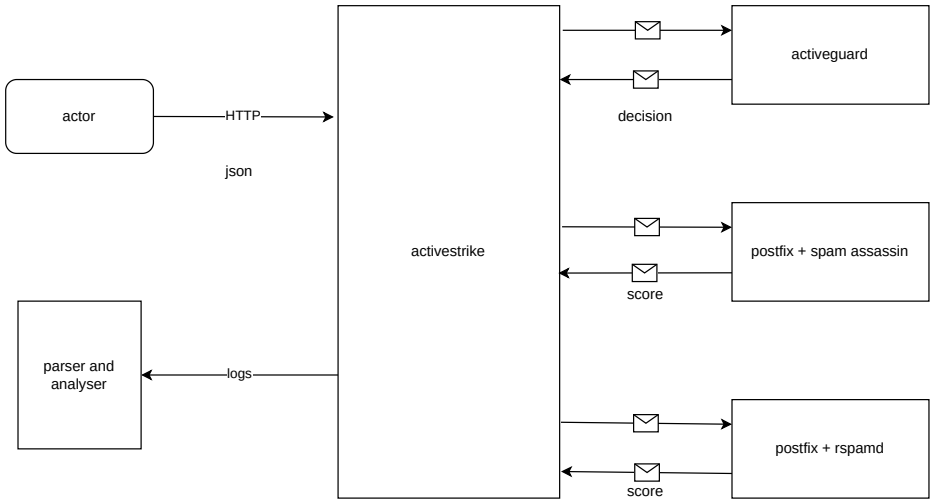


Figure 5.1: Evaluation architecture

## 5.2 Evaluation Methodologies

We define a confusion matrix for a given campaign as the following matrix:

$$\begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix}$$

Where $TP$ is the number of true positive, $TN$ is the number of true negative, $FP$ is the number of false positive, $FN$ is the number of false negative. The row represent the predicted class and the column represent the expected class.

Here we define the two classes of email as ham and spam (or benign email and unwanted email), positive being spam and negative being ham.

For Rspamd and SpamAssassin, the email filter return a score for each email, This score indicate the how likely the email is to be unwanted. An email is classified as as unwanted if it goes beyond a given threshold. The default threshold for Rspamd and SpamAssassin is $5$. To obtain a better threshold, we will optimize it to maximize a given metric. We decided to optimize the threshold so we can compare the two mail filter with working at their peak performance. It exist an high number of metrics related to precision and recall. Among all those metrics we decided to select the following for analysis:

- Accuracy:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$
$$ACC \in [0, 1]$$

- Balanced accuracy:
$$BA = \frac{TPR + TNR}{2}$$
$$BA \in [0, 1]$$
With $TPR = \frac{TP}{TP+FN}$ and $TNR = \frac{TN}{TN+FP}$

- F1 score:
$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$
$$F \in [0, 1]$$
With $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$

- Matthews correlation coefficient:
$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$
$$MCC \in [-1, 1]$$
with $MCC = 0$ when not defined. It measure the correlation of the true class with the predicted class.

We want to find a metric that matches our need for email filtering comparison. The chosen metric will be used to obtain a score threshold.

As a small example, suppose we send 100 emails, 95 spams and 5 benign. We take a filter that always blocks. We want this filter to have a low metric score. Let's take a look at the score of this filter on this campaign according the metrics introduced before. We have the following confusion matrix:
$$\begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix} = \begin{pmatrix} 95 & 0 \\ 5 & 0 \end{pmatrix}$$

- $ACC = \frac{TP+TN}{TP+TN+FP+FN} = 0.95$
The accuracy is close to its maximum: 1. In fact $ACC$ is should not be used for unbalanced data set.

- Balanced accuracy: $BA = \frac{TPR+TNR}{2} = 0.5$
The balanced accuracy is half of the max balanced accuracy.

- F1 score: $F = 2 \cdot \frac{precision \cdot recall}{precision+recall} = 0.97$
The F1 score is close to its maximum: 1. F1 score depends too much on TP.

- $MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} = 0$
A score of 0 means the distribution of output classes is close to random. The correlation

between the true class and the predicted class is nonexistent. The score fit how the classifier work.

The Matthews correlation coefficient (MCC) is known to be more reliable those other metrics[2] From this point we will use the MCC to find the wanted threshold. We design the algorithm 1 to find the threshold.

**Algorithm 1** find best threshold

sort email by score in ascending order
$M \leftarrow$ confusion matrix 2x2 initialized to 0
**for** $email \in emails$ **do**
  **if** email is expected to be a spam **then**
    $ex \leftarrow 1$
  **else**
    $ex \leftarrow 0$
  **end if**
  $M_{1,ex} \leftarrow M_{1,ex} + 1$
**end for**
$best\_MCC = -1$
$best\_threshold = -\infty$
$old\_score = -\infty$
**for** $email \in emails$ **do**
  $score \leftarrow$ email score
  **if** $old\_score \neq score$ **then**
    $new\_MCC = MCC(M)$
    **if** $new\_MCC > best\_MCC$ **then**
      ▷ avoid taking a threshold that has the same score than all emails, or else $best\_MCC$ will represent the MCC after applying the threshold
      $best\_MCC \leftarrow new\_MCC$
      $best\_threshold \leftarrow (old\_score + score)/2$
    **end if**
    $old\_score \leftarrow score$
  **end if**
  **if** email is expected to be a spam **then**
    $ex \leftarrow 1$
  **else**
    $ex \leftarrow 0$
  **end if**
  $M_{1,ex} \leftarrow M_{1,ex} - 1$
  $M_{0,ex} \leftarrow M_{0,ex} + 1$
**end for**
**return** $best\_threshold$

| | malicious | benign |
|---|---|---|
| CSDMC2010 | 1345 | 2936 |
| generated | 677 | 395 |

Table 5.1: Number of email for each campaign for each class

## 5.3 Campaigns

To evaluate our mail filter, we will use the following campaigns:

- A public EML dataset called *CSDMC2010_SPAM*. EML is the file format exchanged in SMTP. This EML dataset can be loaded statically by ActiveStrike and sent as an ActiveStrike campaign. This campaign is composed of spam and ham emails.
- A generated email campaign. This campaign has been generated by ActiveStrike, contains scam, malware and benign emails. The email payload and packing have been designed for the thesis.

Table 5.1 Gives the proportion of unwanted and benign email for each campaign.

In the generated campaign, each email $e$ have a collection of tags $Tags(e)$ in their header. Tags indicate how the email payload has been packed. In Table 5.2, *malicious* indicate the number of unwanted emails which have been packed in this form. *benign* indicate the number of benign emails that have been packed in this form.

Table 5.3 describes in more depth what each list of tags means. The generated campaigns can be spited into two categories: those that contain $FILE$ tag, and those that contain $TEXT$ tags. The payload of those with the $FILE$ tag consist of a static file. In this case, the payload is accessed if the receiver executes or opens the file. The emails with $TEXT$ are either extortion, phishing or impersonation. For extortion and phishing, the dangerous payload is usually an IBAN, bitcoin address or dangerous link. For impersonation, the danger comes from the instruction given by the attacker.

## 5.4 Results

### 5.4.1 General results

General result can be seen at Table 5.4. For each campaign, and each tested mail filters, we computed the $MCC$ and $TPR$. As a reminder $TPR \in [0,1]$ is the True Positive Ratio. It indicates the number of malicious emails that were classified as malicious. Thus higher $TPR$ means that it is harder to bypass the mail filter for an attacker.

| *Tags* | malicious | benign |
|---|---|---|
| FILE, ENCRYPTED, BODY_PLAINTEXT, ZIP, ENCRYPTED, BODY_PLAINTEXT | 3 | 4 |
| FILE, ENCRYPTED, BODY_PLAINTEXT | 3 | 4 |
| FILE, ENCRYPTED, HTML_IMAGE, ZIP, ENCRYPTED, HTML_IMAGE | 3 | 4 |
| FILE, ENCRYPTED, HTML_IMAGE | 3 | 4 |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, BODY_PLAINTEXT | 67 | 19 |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, HTML_IMAGE | 67 | 19 |
| FILE, NOT_ENCRYPTED, ZIP, NOT_ENCRYPTED | 67 | 19 |
| FILE, NOT_ENCRYPTED | 67 | 19 |
| FILE, WEAK_ENCRYPTED, ZIP, NOT_ENCRYPTED | 4 | 4 |
| FILE, WEAK_ENCRYPTED | 4 | 4 |
| TEXT, BODY_HTML | 64 | 49 |
| TEXT, BODY_PLAINTEXT | 69 | 50 |
| TEXT, DOCX_ATTACHMENT | 64 | 49 |
| TEXT, HTML_IMAGE | 64 | 49 |
| TEXT, PDF_ATTACHMENT, ENCRYPTED, BODY_PLAINTEXT | 64 | 49 |
| TEXT, PDF_ATTACHMENT, NOT_ENCRYPTED | 64 | 49 |

Table 5.2: Number of email for each class for each packing

In Table 5.4, we can already see gaps between email filters. For CSDMC2010, Rspamd and SpamAssassin and ActiveGuard have a high performance. The True positive ratio is high (close to 100%) and MCC is also high (close to 1). Adding clamav does not change the results significantly since the CSDMC2010 only contains spam and ham but no malware.

For the generated campaign, Rspamd less than 30% of unwanted emails are blocked. for SpamAssassin less than 10% are blocked. For those two filters, adding malware increase the true positive ratio as well as MCC. The generated campaign contains malware thus, as intended, adding ClamAV antivirus increase the number of unwanted emails that are classified as unwanted.

When a filter has a low MCC (close to 0) and a high TPR, we say that this filter is *aggressive*. A more aggressive filter classifies more benign email as malicious. For both campaigns, ActiveGuard has a high TPR but a low MCC, thus we can say that ActiveGuard is more aggressive than Rspamd and SpamAssassin. ActiveGuard is based on a system of trust, that is to say if an email come from a trusted sender, it will more likely considered as benign. If an email come from an unknown sender, it will be more likely considered malicious. To build trust for a domain, ActiveGuard need multiple email exchange with the domain. The campaign we are testing is based on a model of first contact with the receiver. Thus ActiveGuard will tend to be more aggressive toward those email. Moreover, to compare benign and unwanted email, we used the same type of packing with different payload. The packing by itself can be interpreted as spam. For instance, the body of an email being a single image is hugely suspicious in the first place. ActiveGuard use a system of rules, and can detect the way the email is packaged. Thus explaining the low MCC and high TPR.

| *Tags* | description |
|---|---|
| FILE, ENCRYPTED, BODY_PLAINTEXT, ZIP, ENCRYPTED, BODY_PLAINTEXT | The attached file has been encrypted, the password is in the email body in plaintext. This encrypted file is in an encrypted zip. The password in the email body in plaintext |
| FILE, ENCRYPTED, BODY_PLAINTEXT | The attached file has been encrypted, the password is in the email body in plaintext |
| FILE, ENCRYPTED, HTML_IMAGE, ZIP, ENCRYPTED, HTML_IMAGE | The attached file has been encrypted, the password is in the email body in an image, This encrypted file is in an encrypted zip. The password in the email body in an image |
| FILE, ENCRYPTED, HTML_IMAGE | The attached file has been encrypted, the password is in the email body in an image |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, BODY_PLAINTEXT | The attached file is not encrypted, This file is in an encrypted zip. The password in the email body in plaintext |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, HTML_IMAGE | The attached file is not encrypted, This file is in an encrypted zip. The password in the email body in an image |
| FILE, NOT_ENCRYPTED, ZIP, NOT_ENCRYPTED | The attached file is not encrypted, This file is in a zip |
| FILE, NOT_ENCRYPTED | The attached file is not encrypted |
| FILE, WEAK_ENCRYPTED, ZIP, NOT_ENCRYPTED | The attached file has been encrypted, the password is weak (eg: 12345) and is not in the email. This encrypted file is in a zip |
| FILE, WEAK_ENCRYPTED | The attached file has been encrypted, the password is weak (eg: 12345) and is not in the email |
| TEXT, BODY_HTML | the text (exortion/phishing/spam) is in the body as HTML |
| TEXT, BODY_PLAINTEXT | the text (exortion/phishing/spam) is in the body as plaintext |
| TEXT, DOCX_ATTACHMENT | the text (exortion/phishing/spam) is in a DOCX attachement |
| TEXT, HTML_IMAGE | the text (exortion/phishing/spam) is in a body as an image |
| TEXT, PDF_ATTACHMENT, ENCRYPTED, BODY_PLAINTEXT | the text (exortion/phishing/spam) is in a pdf attachment. The pdf is encrypted. The password is in the email body |
| TEXT, PDF_ATTACHMENT, NOT_ENCRYPTED | the text (exortion/phishing/spam) is in a pdf attachment |

Table 5.3: in depth description of email packing

| | Rspamd | | Rspamd and ClamAV | | SA | | SA and ClamAV | | AG | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MCC | TPR | MCC | TPR | MCC | TPR | MCC | TPR | MCC | TPR |
| CSDMC2010 | 0.48 | 92% | 0.48 | 92% | 0.75 | 82% | 0.73 | 83% | 0.45 | 71% |
| generated | 0.32 | 24% | 0.35 | 27% | 0.12 | 3% | 0.17 | 7% | 0.00 | 80% |

Table 5.4: TPR and MCC evaluated for each campaign and each mail filters

| | Rspamd | | Rspamd and ClamAV | |
|---|---|---|---|---|
| $Tags$ | MCC | TPR | MCC | TPR |
| FILE, ENCRYPTED, BODY_PLAINTEXT, ZIP, ENCRYPTED, BODY_PLAINTEXT | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, ENCRYPTED, BODY_PLAINTEXT | 0.00 | 0.00% | 0.73 | 66.67% |
| FILE, ENCRYPTED, HTML_IMAGE, ZIP, ENCRYPTED, HTML_IMAGE | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, ENCRYPTED, HTML_IMAGE | 0.00 | 0.00% | 0.73 | 66.67% |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, BODY_PLAINTEXT | 0.18 | 13.43% | 0.18 | 13.43% |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, HTML_IMAGE | 0.18 | 13.43% | 0.18 | 13.43% |
| FILE, NOT_ENCRYPTED, ZIP, NOT_ENCRYPTED | 0.18 | 13.43% | 0.18 | 13.43% |
| FILE, NOT_ENCRYPTED | 0.19 | 14.93% | 0.35 | 38.81% |
| FILE, WEAK_ENCRYPTED, ZIP, NOT_ENCRYPTED | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, WEAK_ENCRYPTED | 0.00 | 0.00% | 0.58 | 50.00% |
| TEXT, BODY_HTML | 0.38 | 28.12% | 0.38 | 28.12% |
| TEXT, BODY_PLAINTEXT | 0.41 | 31.88% | 0.41 | 31.88% |
| TEXT, DOCX_ATTACHMENT | 0.38 | 28.12% | 0.38 | 28.12% |
| TEXT, HTML_IMAGE | 0.60 | 56.25% | 0.60 | 56.25% |
| TEXT, PDF_ATTACHMENT, ENCRYPTED, BODY_PLAINTEXT | 0.38 | 28.12% | 0.38 | 28.12% |
| TEXT, PDF_ATTACHMENT, NOT_ENCRYPTED | 0.38 | 28.12% | 0.38 | 28.12% |

Table 5.5: MCC and TPR for Rspamd with/without ClamAV for each packing

### 5.4.2 Packing Result

Table 5.5 Table 5.6 and Table 5.7 are the generated campaign results for each type of packing.

As a disclaimer before analyzing those table, ActiveStrike has a bug that I was not able to reproduce locally before submitting this thesis: when packing a payload into a PDF, encrypted or not. There is a chance that this operation will result in an empty PDF (a PDF with a blank page). Thus, if the email was originally phishing, the phishing link which was supposed to be into the PDF is absent, and the email has not reason to be considered as malicious anymore. This will decrease the TPR for all email filter from what it should have been in the first place. The affected packings are the following:

- TEXT, PDF_ATTACHMENT, ENCRYPTED, BODY_PLAINTEXT
- TEXT, PDF_ATTACHMENT, NOT_ENCRYPTED

| | SpamAssassin | | SpamAssassin and ClamAV | |
|---|---|---|---|---|
| *Tags* | MCC | TPR | MCC | TPR |
| FILE, ENCRYPTED, BODY_PLAINTEXT, ZIP, ENCRYPTED, BODY_PLAINTEXT | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, ENCRYPTED, BODY_PLAINTEXT | 0.00 | 0.00% | 0.73 | 66.67% |
| FILE, ENCRYPTED, HTML_IMAGE, ZIP, ENCRYPTED, HTML_IMAGE | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, ENCRYPTED, HTML_IMAGE | 0.00 | 0.00% | 0.73 | 66.67% |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, BODY_PLAINTEXT | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, HTML_IMAGE | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, NOT_ENCRYPTED, ZIP, NOT_ENCRYPTED | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, NOT_ENCRYPTED | 0.00 | 0.00% | 0.26 | 25.37% |
| FILE, WEAK_ENCRYPTED, ZIP, NOT_ENCRYPTED | 0.00 | 0.00% | 0.00 | 0.00% |
| FILE, WEAK_ENCRYPTED | 0.00 | 0.00% | 0.58 | 50.00% |
| TEXT, BODY_HTML | 0.00 | 0.00% | 0.00 | 0.00% |
| TEXT, BODY_PLAINTEXT | 0.46 | 39.13% | 0.51 | 44.93% |
| TEXT, DOCX_ATTACHMENT | 0.00 | 0.00% | 0.00 | 0.00% |
| TEXT, HTML_IMAGE | 0.00 | 0.00% | 0.00 | 0.00% |
| TEXT, PDF_ATTACHMENT, ENCRYPTED, BODY_PLAINTEXT | 0.00 | 0.00% | 0.00 | 0.00% |
| TEXT, PDF_ATTACHMENT, NOT_ENCRYPTED | 0.00 | 0.00% | 0.00 | 0.00% |

Table 5.6: MCC and TPR for spam assassin with/without clamav for each packing

| | AG | |
|---|---|---|
| *Tags* | MCC | TPR |
| FILE, ENCRYPTED, BODY_PLAINTEXT, ZIP, ENCRYPTED, BODY_PLAINTEXT | 0.00 | 100.00% |
| FILE, ENCRYPTED, BODY_PLAINTEXT | 1.00 | 100.00% |
| FILE, ENCRYPTED, HTML_IMAGE, ZIP, ENCRYPTED, HTML_IMAGE | 0.00 | 100.00% |
| FILE, ENCRYPTED, HTML_IMAGE | 1.00 | 100.00% |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, BODY_PLAINTEXT | 0.00 | 100.00% |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, HTML_IMAGE | 0.00 | 100.00% |
| FILE, NOT_ENCRYPTED, ZIP, NOT_ENCRYPTED | 0.75 | 97.01% |
| FILE, NOT_ENCRYPTED | 0.75 | 97.01% |
| FILE, WEAK_ENCRYPTED, ZIP, NOT_ENCRYPTED | 1.00 | 100.00% |
| FILE, WEAK_ENCRYPTED | 1.00 | 100.00% |
| TEXT, BODY_HTML | -0.35 | 71.88% |
| TEXT, BODY_PLAINTEXT | -0.07 | 92.75% |
| TEXT, DOCX_ATTACHMENT | -0.39 | 67.19% |
| TEXT, HTML_IMAGE | -0.10 | 39.06% |
| TEXT, PDF_ATTACHMENT, ENCRYPTED, BODY_PLAINTEXT | -0.39 | 67.19% |
| TEXT, PDF_ATTACHMENT, NOT_ENCRYPTED | -0.39 | 67.19% |

Table 5.7: MCC and TPR for spam assassin for each packing

As we can see with Table 5.5 and Table 5.6, adding ClamAV increase the detection of malware (packing containing "File" in their $Tags$).

Some malware was detected by Rspamd. Those are the non encrypted files inside a zip, encrypted or not. One hypothesis here would be that Rspamd and ClamAV recognize file extensions and those file extension can be considered malicious (example: exe or bat extension). Moreover the body of the email being empty or containing a password, might contribute to be classified as spam.

If we take a look at SpamAssassin with ClamAV, the malware were only found in the packing with $TPR > 0$:

- FILE, ENCRYPTED, BODY_PLAINTEXT
- FILE, ENCRYPTED, HTML_IMAGE
- FILE, NOT_ENCRYPTED
- FILE, WEAK_ENCRYPTED

The other file packing detected with ClamAV in Rspamd were also detected without ClamAV. We conclude that the best way to bypass ClamAV and Rspamd and SA is to send the malicious file in a ZIP, encrypted or not (all packing type where $TPR < 0$).

Regarding text payload (packings containing "TEXT" in their $Tags$), The best way to package it in order for an attacker to bypass Rspamd is to place it in an image in the email body. For SpamAssassin, the best way to package the payload is either to place it in an image, in the body as HTML, in a DOCX, or in a PDF attachment (encrypted or not). As we can see, the TPR and MCC of those ways of packing are equal. This might be because to a lack of score diversity for SpamAssassin. After looking at the score for payload packed into an HTML body, we can see that some emails had been detected as suspicious. This cannot be seen in Table 5.6 because the threshold that maximize MCC is too high to detect those emails. After further analysis, it turned out that the chosen threshold that maximize MCC is higher than the SpamAssassin default (threshold of $5$). Thus in a normal setup, payload packaged in the body as HTML will also be classified as spam.

In Table 5.7 as said before, we can see that ActiveGuard is aggressive, given the high TPR and low MCC of each packing. Regarding File packing, TPR is always close to or equal to 100%. This comes from a rule of ActiveGuard: first contact email with attachment are qualified as highly suspicious.

We can conclude, for Rspamd and SpamAssassin with ClamAV, email with a message and

attachment is the way to go for an attacker that want to deliver scam. We have also shown that encryption and nested attachment is the way to go for an attacker who wants to deliver a virus.

How can we explain the gap seen between Rspamd, SpamAssassin and ActiveGuard? To bypass those three mails filters, we obfuscated our payload by placing it into different attachment with different kind of encryption and multiple nesting of payload. To mitigate these kinds of attack and reduce the attack surface of our attacker, the mail filter has to implement detection for each of those case. We call this integration extension. The key takeaway of those result is that integration is expensive. Mail filter have to use policies and create metrics to mitigate those case. Maintaining and creating policy are expensive and takes time. This explains why open source email filters are less efficient with the generated campaign than mail filters such as ActiveGuard.

## 5.5 Accessibility Metric

The goal of this part is to create a new metric that measure the importance for a given mail filter to integrate mitigation policies for a given packing method.

The issue with the previous measurement is that it does not take into account the likelihood of the receiver to open an email for a given packing. We introduce email accessibility $w$ as being a metric for measuring how easy it is to access the payload for the email receiver:

$$w(packing) = P\,(\text{receiver access the payload packed with } packing)$$

This is a weight representing the importance value of an email. For spam, this accessibility is the email dangerousness. With this weight, we can create a new confusion matrix.

$$\begin{pmatrix} \tilde{TP} & \tilde{FN} \\ \tilde{FP} & \tilde{TN} \end{pmatrix} = \begin{pmatrix} \sum\limits_{e \in \mathcal{TP}} w(e) & \sum\limits_{e \in \mathcal{FN}} w(e) \\ \sum\limits_{e \in \mathcal{FP}} w(e) & \sum\limits_{e \in \mathcal{TN}} w(e) \end{pmatrix}$$

With this new confusion matrix, we compute the $\tilde{MCC}$ and $\tilde{TPR}$ for each campaign and each mail filter. We use the same threshold than before for Rspamd and SpamAssassin (using MCC, not $\tilde{MCC}$).

Let $X_{packing,filter}$ be a random variable representing the number of successful attacks. This is the number of emails that have their payload accessed by the receiver. Let $X_{e,filter}$ be a Bernoulli variable being $1$ if and only if the email $e$ had its payload accessed after going through $filter$. We have

$$X_{packing,filter} = \sum_{\text{e with packing}} X_{e,filter}$$

We have:

$$\begin{aligned}\mathbb{E}(X_{packing,filter}) &= \sum_{\text{e with packing}} \mathbb{E}(X_{e,filter}) \\ &= \sum_{\text{e with packing}} \mathbb{1}(\text{e bypass filter}) \cdot P(\text{receiver access the payload in e}) \\ &= FN(packing) \cdot w(payload)\end{aligned} \tag{5.1}$$

To evaluate the danger that a packing represent for a given mail filter, we define $d$ as such:

$$\begin{aligned}d(packing, filter) &= \frac{\mathbb{E}(X_{packing,filter})}{TP(packing) + FN(packing)} \\ &= \frac{FN(packing) \cdot w(payload)}{TP(packing) + FN(packing)} \\ &= (1 - TPR(packing)) \cdot w(payload)\end{aligned} \tag{5.2}$$

This represents the likelihood that the receiver opens the dangerous payload for a given filter.

to get a realistic $w$, we pick:

$$w(packing) = \frac{1}{\sum_{t \in Tags(packing)} f(t)} \in [0, 1]$$

Where for all tag $t$, $f(t) > 0$ and represent the number of steps needed to access the payload given the tag $t$. It exists multiple possible ways to evaluate this probability. We decided to take this one because of its simplicity.

To evaluate this new metric, we take $f(t)$ as in Table 5.8. PASSWORD_IN_BODY_PLAINTEXT and PASSWORD_IN_HTML_IMAGE represent the tags given when the payload is encrypted with a password, and the password is put in the body as plain text or in an image respectively. Those tags are always placed after the ENCRYPTED tag. For simplicity reason, we replaced the those two tags (visually) by BODY_PLAINTEXT and HTML_IMAGE respectively. The comment section of Table 5.8 gives an explanation why the we chose the value of $f(t)$.

The accessibility metric is directly correlated to payload obfuscation: The more a payload is obfuscated in an email, the more step the user need to access the payload, thus the less this accessibility metric is.

| tag | f(tag) | comments |
|---|---|---|
| TEXT | 1 | |
| FILE | 1 | |
| BODY_PLAINTEXT | 1 | |
| BODY_HTML | 1 | |
| PDF_ATTACHMENT | 3 | The receiver need to open the pdf |
| NOT_ENCRYPTED | 1 | |
| WEAK_ENCRYPTED | 4 | The receiver need to find the password and decrypt the file |
| ENCRYPTED | 3 | The receiver need to decrypt the file |
| DOCX_ATTACHMENT | 3 | The receiver need to open the pdf |
| HTML_IMAGE | 4 | If it contains a link, the receiver need to copy the link by hand |
| ZIP | 1 | The receiver need to unzip the file |
| PASSWORD_IN_BODY_PLAINTEXT | 1 | The receiver need to copy the password into clipboard |
| PASSWORD_IN_HTML_IMAGE | 5 | The receiver need to copy the password from the image by hand |

Table 5.8: Definition of f(tag)

| | Rspamd | | Rspamd and ClamAV | | SA | | SA and ClamAV | | AG | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\tilde{MCC}$ | $\tilde{TPR}$ | $\tilde{MCC}$ | $\tilde{TPR}$ | $\tilde{MCC}$ | $\tilde{TPR}$ | $\tilde{MCC}$ | $\tilde{TPR}$ | $\tilde{MCC}$ | $\tilde{TPR}$ |
| generated | 0.33 | 25% | 0.37 | 30% | 0.16 | 7% | 0.23 | 14% | 0.00 | 0.82% |

Table 5.9: MCC and TPR computed with the modified confusion matrix

### 5.5.1 Results

As said in section 5.5, payload obfuscation is related to accessibility metric $w$. The accessibility metric is related to $\tilde{TPR}$ and $\tilde{MCC}$. We expect for any mail filter to be able to detect more email that have a high accessibility (i.e. low obfuscation). Thus, we expect for any filter, for a given campaign to have $\tilde{TPR} > TPR$ and $\tilde{MCC} > MCC$. In Table 5.9, for each filter $\tilde{TPR} > TPR$ and $\tilde{MCC} > MCC$. This does reach our expectations.

We will now take a look at the packing danger metric $d$ in Table 5.10.

In Table 5.11 we can deduce that the best way for an attacker to deliver a text payload (spam, scam, phishing) when it goes through Rspamd, SpamAssassin, or ActiveGuard is to package the payload into an HTML body.

In Table 5.12 we can deduce that the best way for an attacker to deliver a file payload (malware or malicious file) when it goes through Rspamd, SpamAssassin, or ActiveGuard is to not encrypt the file and place it as an attachment.

| *Tags* | rspamd | rspamd clamav | SA | SA clamav | AG |
|---|---|---|---|---|---|
| FILE, ENCRYPTED, BODY_PLAINTEXT, ZIP, ENCRYPTED, BODY_PLAINTEXT | 10.00% | 10.00% | 10.00% | 10.00% | 0.00% |
| FILE, ENCRYPTED, BODY_PLAINTEXT | 20.00% | 6.67% | 20.00% | 6.67% | 0.00% |
| FILE, ENCRYPTED, HTML_IMAGE, ZIP, ENCRYPTED, HTML_IMAGE | 5.56% | 5.56% | 5.56% | 5.56% | 0.00% |
| FILE, ENCRYPTED, HTML_IMAGE | 11.11% | 3.70% | 11.11% | 3.70% | 0.00% |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, BODY_PLAINTEXT | 12.37% | 12.37% | 14.29% | 14.29% | 0.00% |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, HTML_IMAGE | 7.87% | 7.87% | 9.09% | 9.09% | 0.00% |
| FILE, NOT_ENCRYPTED, ZIP, NOT_ENCRYPTED | 21.64% | 21.64% | 25.00% | 25.00% | 0.75% |
| FILE, NOT_ENCRYPTED | 42.53% | 30.59% | 50.00% | 37.31% | 1.49% |
| FILE, WEAK_ENCRYPTED, ZIP, NOT_ENCRYPTED | 14.29% | 14.29% | 14.29% | 14.29% | 0.00% |
| FILE, WEAK_ENCRYPTED | 20.00% | 10.00% | 20.00% | 10.00% | 0.00% |
| TEXT, BODY_HTML | 35.94% | 35.94% | 50.00% | 50.00% | 14.06% |
| TEXT, BODY_PLAINTEXT | 34.06% | 34.06% | 30.43% | 27.54% | 3.62% |
| TEXT, DOCX_ATTACHMENT | 17.97% | 17.97% | 25.00% | 25.00% | 8.20% |
| TEXT, HTML_IMAGE | 7.29% | 7.29% | 16.67% | 16.67% | 10.16% |
| TEXT, PDF_ATTACHMENT, ENCRYPTED, BODY_PLAINTEXT | 8.98% | 8.98% | 12.50% | 12.50% | 4.10% |
| TEXT, PDF_ATTACHMENT, NOT_ENCRYPTED | 14.38% | 14.38% | 20.00% | 20.00% | 6.56% |

Table 5.10: Packing danger for each tested email filter

| *Tags* | rspamd | rspamd clamav | SA | SA clamav | AG |
|---|---|---|---|---|---|
| TEXT, BODY_HTML | 1 | 1 | 1 | 1 | 1 |
| TEXT, BODY_PLAINTEXT | 2 | 2 | 2 | 2 | 6 |
| TEXT, DOCX_ATTACHMENT | 3 | 3 | 3 | 3 | 3 |
| TEXT, HTML_IMAGE | 6 | 6 | 5 | 5 | 2 |
| TEXT, PDF_ATTACHMENT, ENCRYPTED, BODY_PLAINTEXT | 5 | 5 | 6 | 6 | 5 |
| TEXT, PDF_ATTACHMENT, NOT_ENCRYPTED | 4 | 4 | 4 | 4 | 4 |

Table 5.11: Ranking of most dangerous packing for a payload consisting of a text

| *Tags* | rspamd | rspamd clamav | SA | SA clamav | AG |
|---|---|---|---|---|---|
| FILE, ENCRYPTED, BODY_PLAINTEXT, ZIP, ENCRYPTED, BODY_PLAINTEXT | 7 | 5 | 6 | 4 | 3 |
| FILE, ENCRYPTED, BODY_PLAINTEXT | 3 | 7 | 3 | 6 | 3 |
| FILE, ENCRYPTED, HTML_IMAGE, ZIP, ENCRYPTED, HTML_IMAGE | 9 | 8 | 8 | 7 | 3 |
| FILE, ENCRYPTED, HTML_IMAGE | 6 | 9 | 5 | 8 | 3 |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, BODY_PLAINTEXT | 5 | 4 | 4 | 3 | 3 |
| FILE, NOT_ENCRYPTED, ZIP, ENCRYPTED, HTML_IMAGE | 8 | 6 | 7 | 5 | 3 |
| FILE, NOT_ENCRYPTED, ZIP, NOT_ENCRYPTED | 2 | 2 | 2 | 2 | 2 |
| FILE, NOT_ENCRYPTED | 1 | 1 | 1 | 1 | 1 |
| FILE, WEAK_ENCRYPTED, ZIP, NOT_ENCRYPTED | 4 | 3 | 4 | 3 | 3 |
| FILE, WEAK_ENCRYPTED | 3 | 5 | 3 | 4 | 3 |

Table 5.12: Ranking of most dangerous packing for a payload consisting of files

Those results are highly different from the previous results. This show that the metric we chose give a high weight to the email accessibility. The accessibility metric and the values of $f(t)$ are arguably debatable. Nonetheless, this gives us a clear overview of what type of packing should be taken first into consideration first when a email filter wants to decrease its attack surface.

In conclusion ActiveStrike can be used as a decision tool for a given mail filter to give a priority list of the package method that one need to mitigate to decrease the mail filter attack surface.

# Chapter 6

# Related Work

The goal of this paper is to perform an end to end benchmark multiple email filter solution. The research needed for this project was minimum.

One end to end testing solution I found was *Automated end-to-end e-mail component testing.* [8]. This paper focus on testing the security in the email protocol.

This paper shows how an attacker can bypass dynamic analysis for virus [3]. This paper shows how an attacker can bypass a Bayesian filter [9].

# Chapter 7

# Conclusion

In this paper, we created a tool, ActiveStrike, to send email campaign against a mail filter. Using this tool, we were able find which way of packing a payload into a email is the way to go. We have shown that email with a message and attachment is the most efficient way for an attacker that want to deliver scam. We have also shown that encryption and nested attachment are the most efficient way for an attacker who wants to deliver a virus. Mail filter implement policies to mitigate those attack. Common mitigations are blocking file extensions, use dynamic analysis and monitor unwanted change after execution, or use signature to detect known viruses.

There was a gap in detection between ActiveGuard, Rspamd and SpamAssassin. To bypass those three mails filters, we obfuscated our payload by placing it into different attachment with different kind of encryption and multiple nesting of payload. To mitigate these kinds of attack and reduce the attack surface of our attacker, the mail filter has to implement detection for each of those case. We call this integration extension. The key takeaway of those result is that integration is expensive. Mail filter have to use policies and create metrics to mitigate those case. For instance creating policies to block some file extension, or use dynamic analysis and monitor unwanted system change after execution or use signature analysis to detect known viruses. Maintaining and creating policy are expensive and takes time. This explains why open source email filters are less efficient with the generated campaign than mail filters such as ActiveGuard, Gmail or Microsoft Outlook 365.

In this paper, we also evaluated our email campaigns on a created metric that take into account the likelihood of the receiver to open an email for a given packing method. We called this metric the danger of a packing method. With this metric, we saw that the best way for an attacker to deliver a text payload (spam, scam, phishing) for all filter is to package the payload into an HTML body. We also saw that the best way for an attacker to deliver a file payload (malware or malicious file) when it goes through Rspamd, SpamAssassin, or ActiveGuard is to not encrypt the file and place it as an attachment. This metric can be used for decision making for mail filter improvements. This metric show what packing method attack need to be mitigated

in order to decrease the most attack surface.

# Bibliography


[1] xorlab AG. *xorlab platform*. https://www.xorlab.com/en/platform. 2022.

[2] Davide Chicco, Niklas Tötsch, and Giuseppe Jurman. "The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation". In: *BioData mining* 14.1 (2021), pp. 1–22.

[3] Saeed Ehteshamifar, Antonio Barresi, Thomas R Gross, and Michael Pradel. "Easy to fool? testing the anti-evasion capabilities of pdf malware scanners". In: *arXiv preprint arXiv:1901.05674* (2019).

[4] Baptiste Jacquemot. *rspamd-clamav-relay*. https://github.com/bajacc/rspamd-clamav-relay. 2022.

[5] Baptiste Jacquemot. *rspamd-relay*. https://github.com/bajacc/rspamd-relay. 2022.

[6] Baptiste Jacquemot. *spamassassin-clamav-relay*. https://github.com/bajacc/spamassassin-clamav-relay. 2022.

[7] Baptiste Jacquemot. *spamassassin-relay*. https://github.com/bajacc/spamassassin-relay. 2022.

[8] Isaac Klop and Kevin Csuka. "Automated end-to-end e-mail component testing". In: (2018).

[9] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, J Doug Tygar, and Kai Xia. "Exploiting machine learning to subvert your spam filter." In: *LEET* 8.1 (2008), p. 9.

[10] Hossein Siadati, Sima Jafarikhah, and Markus Jakobsson. "Traditional countermeasures to unwanted email". In: *Understanding social engineering based scams*. Springer, 2016, pp. 51–62.