UNIVERSITY OF CALIFORNIA
Los Angeles

Phish Muzzle:
This Fish Won't Bite

A thesis submitted in partial satisfaction
of the requirements for the degree Master of Science
in Computer Science

by

Prashant Hari Narayan Rajput

2017

ABSTRACT OF THE THESIS

Phish Muzzle:
This Fish Won't Bite

by

Prashant Hari Narayan Rajput

Master of Science in Computer Science
University of California, Los Angeles, 2017
Professor Rafail Ostrovsky, Chair

Phishing as an attempt to obtain sensitive information from users using deceitful methods. Email Phishing is done using email as the method of delivery for the attack and targeted email phishing or Business Email Compromise (BEC) are more difficult to detect as they cannot be detected by regular spam filters. Difficulty of detection makes targeted email phishing one of the hardest problem to solve. We present a new method to detect email spear phishing attacks using a method that is less taxing on the machine resource as compared to a spam filter which uses machine learning. We have implemented the scheme as a plugin to outlook and report our performance and effectiveness based on CSDMC2010_SPAM, Enron and Spam_Assassin dataset with a total of 25,834 spam emails and 41,320 ham emails.

The thesis of Prashant Hari Narayan Rajput is approved.

Jens Palsberg

Alexander Sherstov

Rafail Ostrovsky, Committee Chair

University of California, Los Angeles

2017

To my mother . . .

who—among so many other things—

saw to it that I never gave up

TABLE OF CONTENTS

1 Introduction **1**

2 Literature Survey **3**

3 Phish Muzzle **5**

3.1 Adversary Model 5

3.2 Design Goals 7

3.3 System Overview 8

3.4 Implementation 10

4 Evaluation **17**

4.1 Challenges We Had To Overcome 17

4.2 Evaluation Results 18

5 Discussion **20**

6 Conclusion **21**

7 Appendix **22**

7.1 Alert Form 22

7.2 Add-in Settings Form 23

7.2.1 Stored Database 23

7.2.2 Search 24

7.2.3 Protection 25

References **26**

LIST OF FIGURES

3.1 System Overview 8

3.2 Initial Execution Flow 11

3.3 Email Received Flow 12

3.4 Email Read Flow 14

7.1 Email Id Alert 22

7.2 Reply-To Alert 23

7.3 Stored Data Settings 24

7.4 Search Settings 25

7.5 Protection Settings 25

LIST OF TABLES

4.1 Evaluation Results 18

4.2 Plugin Performance 19

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Prof. Rafail Ostrovsky of the Henry Samueli School of Engineering and Applied Science at UCLA. Prof. Ostrovsky guided me whenever I ran into a trouble or had a question about my research. He consistently steered me in the right the direction whenever I needed it.

I would also like to thank Prof. Palsberg and Prof. Sherstov who were in my thesis committee for this research project. Without their passionate participation and input, this research could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents and to my girlfriend for providing me with unfailing support and continuous encouragement throughout the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author
Prashant Hari Narayan Rajput

CHAPTER 1

Introduction

There has been a constant increase in the amount of phishing attacks that are launched every year. According to APWG's Phishing Activity Trends Report there were a total of 1,220,523 attacks launched in 2016 which is a 65% increase from 2015 [1]. This shows a clear trend of increase in phishing attacks each year. This includes website phishing attacks, email phishing attacks and targeted phishing/BEC campaigns. These attacks are often delivered using a misleading pretext, such as email about an account renewal or letter from a colleague/friend. Whatever is the pretext, in the body of the email or in the response message, the user is then redirected to a fake website which is created for one purpose only - to steal user information. Attackers try to make this fake website as similar to the original one as possible. Some users fall for these attacks and provide sensitive information which is then retrieved by the attacker and leads to identity theft, ransomware, etc.

Phishing research is a relatively unexplored topic because it is a hard problem to solve [2]. This is because email delivery system still uses SMTP and all these protocols were implemented when internet was still a small group of trusted computers. SMTP relies on the cooperation among various servers and it assumes that all the servers involved in the communication can be trusted. This trust amongst the STMP servers makes the overall system vulnerable. The sender's email id is not verified, and the email is forwarded to the recipient. For this, Microsoft implemented a Sender Id framework to verify the sender's email id before forwarding the email to the recipient.

Our new approach is based on extracting information from all the emails stored in user's inbox. Indeed, we all already have large mailboxes, where it is typically the norm to retain all the previous email that are not spam/junk. This corpus of email metadata is assumed to

be all legitimate emails because users and spam filters generally filters and delete or move all the spam emails into the spam folder.

Our approach is as follows: we compare the already stored corpus of old emails with the new incoming email. Similarities between this knowledge base of email metadata and the new incoming emails are compared. Our software simply checks if the new email is similar but not identical to the one already stored in the inbox. For example, if the email address is slightly misspelled and is just slightly different email from the sender most likely it is a phishing attack. In this case our software warns the user and email spear phishing attack is detected. Moreover, our algorithm also checks for reply-to email ids and if the reply-to email id of the incoming email does not match with the ones stored in the inbox, for that particular sender email id, then an alert is generated for the user. These two heuristics are useful when we consider email spear phishing attacks. The details about our method is explained in the sections that follow.

The remainder of the thesis is organized as follows. Chapter 2 discusses previous approaches used for filtering email phishing attacks and their limitations. Chapter 3 gives detailed explanation about the approach taken by our software plugin to detect targeted email phishing attack. Chapter 4 then discusses the evaluation method and the results obtained from it. Chapter 5 and 6 focus on discussion and the conclusion respectively.

CHAPTER 2

Literature Survey

Spear Email Phishing is a field of computer science research that has not been fully explored. There have been many attempts at tackling this problem with the help of machine learning techniques. One of such technique was put forth by researchers at Carnegie Mellon University (CMU) where they suggested using features such as - age of linked-to domain names, HTML emails, number of links in the email, numbers of domains, number of dots in the URL and finally URLs which contain IP address in them [3]. This approach was suggested because phishing websites are generally not registered with the domain and attackers then use IP address in their URLs to route users to their websites.

Meanwhile, researchers at IBM developed an intuitive pattern based system for automatic identification of spam email messages [4]. This technique did not use any machine learning methods but instead relied on pattern-discovery as their underlying tool. This method tries to discover patterns that appear two or more times in the collection of learning email database. Then all the incoming emails are checked for any match for the patterns with the learned signatures. As the number of pattern matches increases, the chance that the incoming email is a spam also increases.

On the other hand, researchers from SUNY Buffalo have also used structural properties of spam emails to learn and identify incoming emails. This technique uses features such as total number of words, total number of characters, structure of email subject line and structure of greeting provided in the email body. These features along with one-class Support Vector Machine (SVM) are used for filtering out spam emails in this technique. This filter is used between Mail Transfer Agent (MTA) and Mail User Agent (MUA) and hence it is not a client-side filter. It filters out emails before they reach the MUA [5].

Microsoft has also tried to identify spam based on user's identity rather than the content of the email. This is because content of the email is not always an accurate metric to check for spam. Such spam filters can be easily fooled by a dedicated attacker. For instance, if a spam filter uses metrics such as URLs, a fake URL can be easily hidden by using many legitimate URLS. To overcome such drawbacks, Microsoft designed the Sender ID framework. In this framework, the sender's email id is validated and therefore an attacker cannot spoof its email id as its validation will always fail. This framework is implemented for Exchange Server 2016. Unfortunately, even this framework of validation check can be fooled using email spear phishing attack. This is because email spear phishing is a targeted attack and uses valid sender email id [6]. Therefore, the need for protection against spear phishing attacks is still unresolved.

Another email spam filtering system is - Spamato, which is much more than just a filtering technique. It is an open, extendable spam filter system which can be used by the developers to make new filters. This is not a mere single filter but a collection of filters and a system to develop even more new filters. This system enables developers to develop filters for Thunderbird, Microsoft Outlook and Mozilla. It has a Filter Manager which is a mere repository for all filters and the Web Configuration is a simple HTTP server used to share the services of Spamato [7].

Researchers at Symantec Research Lab put forth a solution for email spear phishing attack which takes into consideration social features extracted from LinkedIn profiles and stylometric features extracted from email subject, bodies and attachments. Combined with machine learning algorithms, they were successful in getting an overall maximum accuracy of 97.76% [8].

CHAPTER 3

Phish Muzzle

Phish Muzzle is our solution to the email spear phishing problem and the details about this plugin are provided in following sections.

3.1 Adversary Model

We assume that emails in the user's inbox are all legitimate emails and that the user has filtered them. Therefore, metadata of emails contained in the inbox are considered as the ground truth against which new emails are compared against and suspicious emails are flagged. Because of this ground truth, our method is useful only against email spear phishing attacks. We discuss these attacks below:

1. **Type-I Attacker:** This type of attacker knows the email id of the victim and email ids of some of their close contacts. The attacker then forges a new email account with an email id that is similar to the email id of one of the aforementioned contacts and makes a few minuscule changes to them. For instance, if the contact's email id is lukeskywalker@gmail.com, the attacker would try to send an email from a fake email id such as "lukeskyvalker@gmail.com" or "lukeskyvalkerr@gmail.com".
   In the example above, attacker makes one or two subtle changes to the email id of the contact in order to fool the victim.

2. **Type-II Attacker:** These attackers spoof the sender's email id to be the email id of the contact. Hence the attacker is careful enough to not give away its identity, but to receive a reply from the victim, attacker changes the "reply-to" email id to its own email id. By default, reply-to email id is the same as the email id of the sender. This

information is generally not visible unless the email header is checked. It is only visible when the person is composing a reply to an email.

The email header of a type-II attack may look something as shown below:

```
To: lukeskywalker@live.in
Subject: Are you there???...Yoda
X-PHP-Script: ....
X-PHP-Originating-Script: 1068:m.php
Date: Mon, 10 Apr 2017 13:21:03 +0000
From: "Yoda" yoda@ucla.edu
Reply-To: darthvader.deathstar@gmail.com
Message-ID: ....
```

Here Darthvader is the attacker and is impersonating Yoda while attacking Luke Sky Walker.

3. **Type-III Attacker:** These attackers perform mass email phishing attacks in which they send emails containing links to malicious websites. These malicious websites look very close to a legitimate website. Users are tricked into sharing their personal information on these compromised websites and it is then obtained by the attacker. These attacks are detected by semantic analysis using machine learning techniques.

Enough research has been done to tackle type-III attackers. In general, solutions for these attacks involve semantic analysis of emails and creation of a model to learn from patterns of incoming emails over time. Then, this model is used for detecting spam emails and for flagging them before these reach the end client.

Our plugin focuses on identifying type-I and type-II attackers, which are not easily detected by semantic analysis. This is because these attackers do not have suspicious links in their email, but instead use a legitimate but slightly different email address or a different reply-to address to trick people into believing them.

3.2 Design Goals

To address the issue of email spear phishing efficiently, our Microsoft Outlook plugin is designed with the following main design goals:

1. **Client-Side Solution:** Unlike researchers from SUNY Buffalo, whose email phishing detection method was placed between MTA and MUA, our plugin resides on the client side. This makes Phish Muzzle independent from any central authority and hence eases its deployment. Moreover, Phish Muzzle also preserves privacy of the user by storing the email header meta-data only on the client's machine.

2. **Quick Computation:** One of the main objectives of Phish Muzzle was to create a simple solution for email spear phishing problem. Our method does not rely on machine learning algorithms and hence is not computation intensive. It performs simple similarity check to find out possible suspicious email ids. Moreover, we perform similarity check on only a subset of emails which reduces the computation even further. Our method only performs computation which increases with the inbox size when it is run for the first time and it must build its knowledge base from the emails stored in the inbox. Once the initial knowledge base is built, only new safe entries are added to the knowledge base.

3. **Optimized Resource Usage:** Most of the solutions to this problem use machine learning for identifying patterns from spam emails. But our method relies on simple email header data to identify phishing attack. This simplifies the computation.

4. **Easy to Use:** Phish Muzzle is designed to be extremely simple to use and requires minimum human interaction to function. It detects suspicious emails based on an automated algorithm and warns the user about the possibility of an attack. But the add-in also uses human input for improving its knowledge base.

5. **Learn Simple Truths:** Phish Muzzle gives the user an option to trust email ids when an alert is displayed. If the user interacts with the system and actively gives it

feedback, the number of false positives will be reduced over time. Users also have an option of trusting multiple email ids at a time by going into the plugin settings.

6. **Grow with the data:** As the amount of unique email header metadata increases in the inbox, the knowledge base used for comparison increases as well. This increases the chance of detection of suspicious email spear phishing attacks.

3.3 System Overview

Phish Muzzle has three main components - Alert Generator, Difference Engine and Knowledge Base. The overall system is shown in the Figure 3.1.

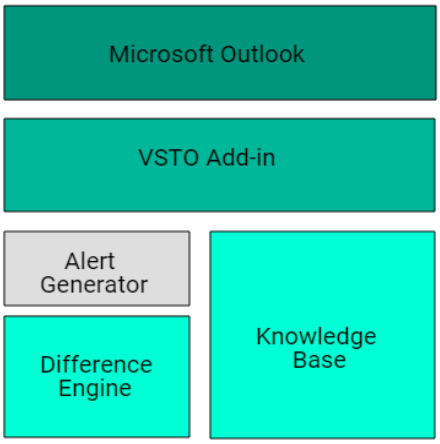

Figure 3.1: System Overview

1. **Difference Engine:** This core component of Phish Muzzle is responsible for calculating so-called edit-distance similarity (also known as "Levenshtein distance") between the email in question and the meta data stored in the knowledge base. Our algorithm uses email ids and reply-to addresses for finding out possible email spear phishing attacks.

   The difference engine finds out possible suspicious email id candidates and compares the email in question with the candidates using Levenshtein distance. Our algorithm

checks for small edit-distance difference between these email id pair. For instance, hansole@gmail.com would be flagged as an attack if the knowledge base contains hansolo@gmail.com.

The difference engine is also responsible for comparing the reply-to address whenever an email is opened. Users don't pay attention to the reply-to email address and fall victim of type-II attacker. Hence, the difference engine stores all the reply-to email ids of emails in inbox and then uses that knowledge for comparing it with the reply-to address of a newly received email.

For type-II attackers, the difference engine doesn't look for small edit-distance character differences but instead expects an exact match from the reply-to address stored in the knowledge base corresponding to the email id. This is a strict metric for detecting targeted email phishing attack and hence it can lead to false positives.

2. **Knowledge Base:** When the plugin is run for the first time, it builds its initial knowledge base from the emails present in the inbox folder for the user. Multiple inboxes are used if a user has multiple accounts logged in Outlook. Email ids and reply-to addresses are extracted from the email headers of all the emails in the inbox. This is then stored in the knowledge base for future use. Moreover, users have an option to trust a particular email id which will exclude it from the check implemented for detecting type-I attacker. This reduces false positives. This additional information is also stored in the knowledge base so that it can be used effectively during plugin operation.

3. **Alert Generator:** This subsystem is used to provide users with a visual alert regarding a suspicious email id. This system is critical as the user only interacts with this part of the add-in. The user can directly trust an email id using the alert generated for a suspicious email id. This makes it easier for the user to provide feedback to the plugin. Moreover, user is also provided with an option to trust reply-to email id based on the type of attack that has been detected.

So, overall the plugin first builds the knowledge base and then this knowledge base is used for

identifying suspicious email ids (those that have small edit distance in the email or reply-to email to legitimate emails already in the inbox). The difference engine is used for detecting suspicious email ids and then this information is sent to the alert generator. Alert is displayed to the user and hence user is warned about the attack. If the user choses to provide feedback to the plugin, this information is then incorporated in the updated knowledge base either to block the email or to add this as a legitimate email to the knowledge base.

3.4 Implementation

Phish Muzzle was programmed in C# using Microsoft Visual Studio, as a VSTO add-in for Microsoft Outlook 2016. It has the following main flows that are mentioned below:

1. **Add-in initial execution:** When the add-in is run for the first time, it iterates through inbox of all the users logged into Outlook and checks if the item is a mailitem as shown in Figure 3.2. The plugin then extracts email addresses and reply-to address from these emails and stores them in a MySQL table. Sometimes it's possible that the reply-to address may contain multiple addresses and hence all of them are concatenated and stored in the table. Moreover, reply-to address might be empty and this situation must also be handled properly. this phase is shown in Algorithm 1.

2. **New email received:** When a new email is received for the first time, email id and reply-to address is extracted from that email as shown in Figure 3.3. Using this information, suspicious email ids are calculated from the knowledge base. Specifically, the system checks whether the new email id or reply-to email is close to (according to the Levenstein distance) but not identical to some legitimate email metadata already stored in the knowledgebase. Specifically, given a new email id the system generates all possible email ids that are "close but not identical" to the new email according to the Levenstein distance and checks for all such entries in the knowledge base. If there is such a suspicious match, then an alert is generated. If there is no suspicious match and

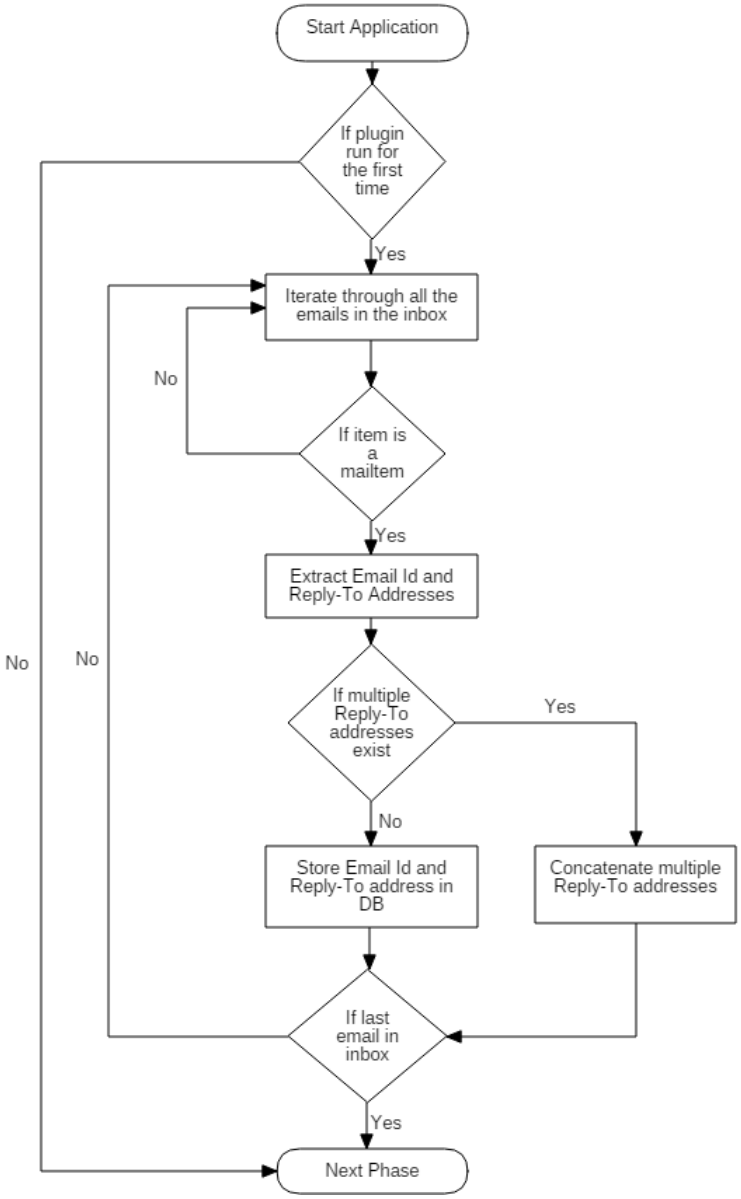

Figure 3.2: Initial Execution Flow

Algorithm 1 Add-In Initial Execution

1: **procedure** INITIALEXECUTION
2: *loop*:
3: **if** $inbox(i) \neq last(inbox)$ **then**
4: **if** $item = mailitem$ **then**
5: $emailidlist \leftarrow ExtractMetadata(mailitem)$.
6: $replytolist \leftarrow ExtractMetadata(mailitem)$.
7: **goto** *loop*.
8: **close**;
9: $knowledgebase \leftarrow$ Store($metadatalist$)
10: *Continue to Outlook*

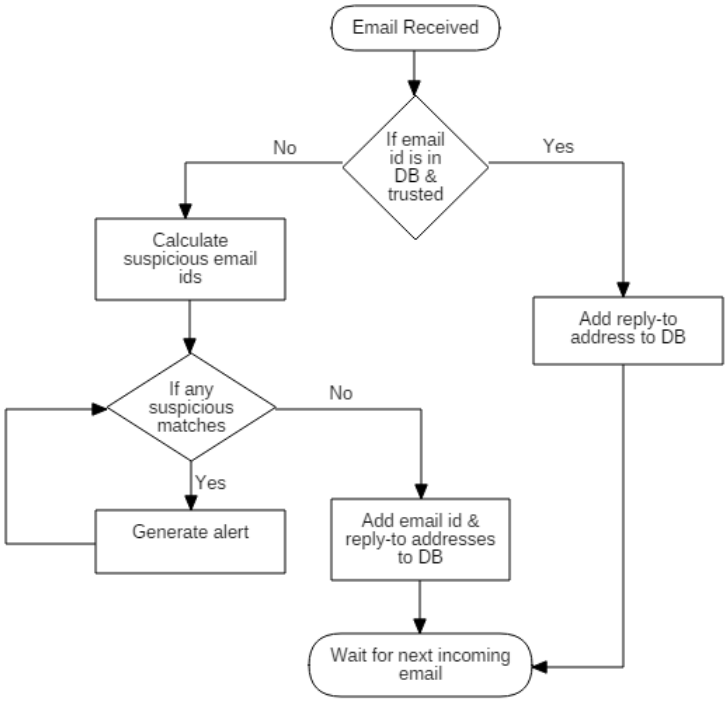

Figure 3.3: Email Received Flow

the user has not untrusted the email, metadata from this email (email id and reply-to address) is added to the knowledge base.

This phase is important for updating the knowledge base as soon as a new email is received. This flow is represented in Algorithm 2.

Algorithm 2 New Email Received

```
1: procedure NewEmail
2:     if emailid not in knowledgebase then
3:         emaillist ← GetSubsetEmailList(knowledgebase).
4: loop:
5:         if emailid in emaillist then
6:             distance ← LevenshteinDistance(emailid)
7:             if distance ≥ 1 and distance ≤ 3 then
8:                 continue
9:             else
10:                updatedknowledgebase ← emailid
11:        else
12:            break
13:        emailid ← incrementlist(emaillist)
14:        goto loop.
15:    else
16:        continue
```

3. **Email is read:** When user reads an email, the difference engine gets activated as shown in Figure 3.4. Using MySQL query, a subset of email ids from the complete list of knowledge base is selected. This subset contains email ids that might be suspicious. Then on this subset we calculate Levenshtein distance to find out possible email spear phishing attack. If any email id is detected from the knowledge base with small character difference, these email ids are put in to the suspicious list. This list is then used

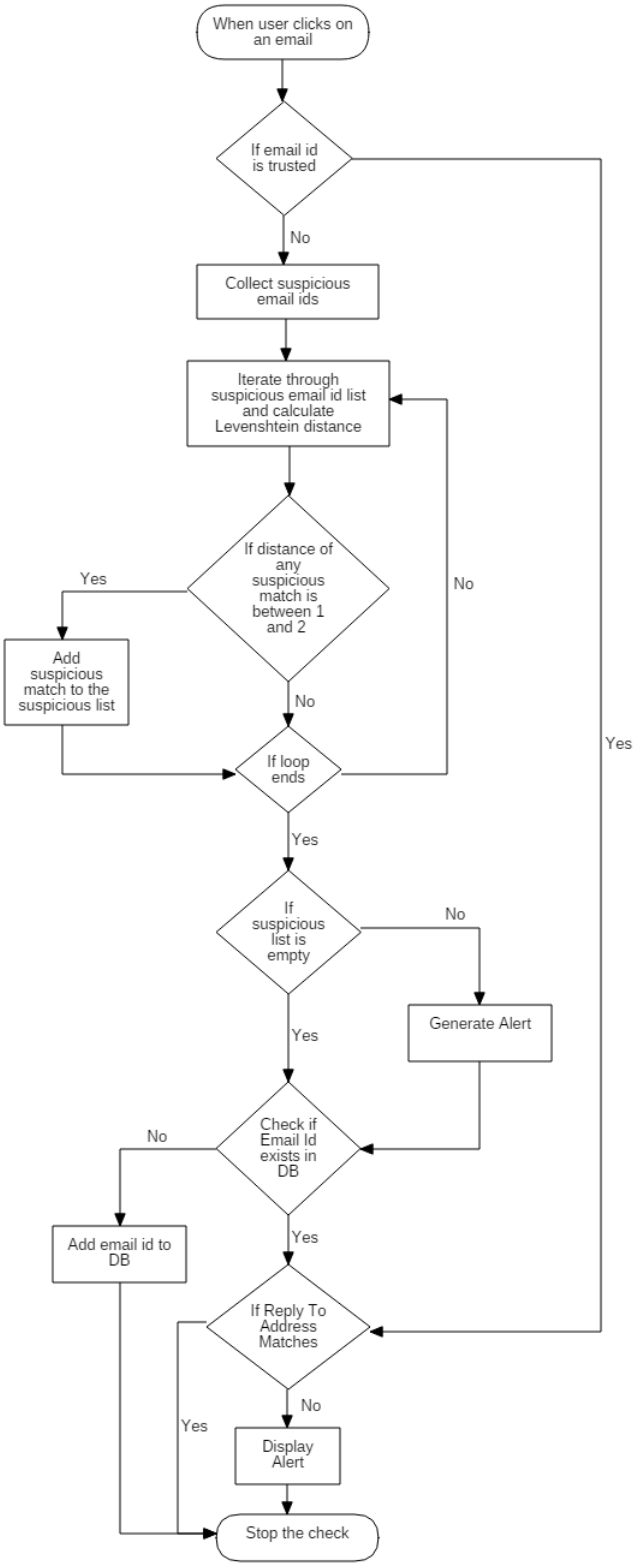

Figure 3.4: Email Read Flow

to alert the user about a possible email spear phishing attacks. If the suspicious email list is empty, then the reply-to address is compared exactly with the reply-to addresses from the knowledge base. Similar to the email id check, the reply-to check also requires prior correspondence between the victim and the spoofed contact. For incoming emails with missing reply-to address, the sender email id is stored as the reply-to address in the knowledge base.

If there are multiple reply-to addresses stored in the knowledge base for an email id, all the stored reply-to addresses are compared with the received reply-to address for a match. If there is no match, then an alert is generated.

Moreover, users have an option to trust an email id. When the user trusts an email id - it is then considered as trusted, but the reply-to address check for type-II attacker is performed regardless of the user trusting an email id. This is because the attacker can change the sender email id field and send an email. But for receiving a reply from the person being attacked, the attacker must change the reply-to email address. Hence, type-II attacker check is performed even when an email id is trusted. For usability, we have allowed users to trust reply-to addresses from the alert that is generated for type-II attackers and the reply to check can be turned off from the plugin setting. In Algorithm 3, we set a threshold for Levenstein distance to be 2, but this is a parameter that can be set arbitrary.

Algorithm 3 Email Read

```
1: procedure EmailRead
2:     if emailid is not trusted then
3:         emaillist ← GetSubsetEmailList(knowledgebase).
4: loop:
5:         if emailid in emaillist then
6:             distance ← LevenshteinDistance(emailid)
7:             if distance ≥ 1 and distance ≤ 3 then
8:                 suspiciouslist ← emailid
9:         else
10:            break
11:        emailid ← incrementlist(emaillist)
12:        goto loop.
13:        close;
14:        length ← ListLength(suspiciouslist)
15:        if length ≥ 1 then
16:            Display Alert
17:    if replytoaddress not in FetchReplyAddress(emailid) then
18:        Display Alert
```

CHAPTER 4

Evaluation

Evaluations for Phish Muzzle were done on 3 openly available email phishing datasets. These datasets were selected because unlike other datasets these had email meta data as well. The datasets which were chosen for evaluation are mentioned below:

1. **CSDMC2010_SPAM:**
   Number of spam emails: 2,949
   Number of ham emails: 1,378
   Total number of emails: 4,327

2. **Enron Email Dataset:**
   Number of spam emails: 19,088
   Number of ham emails: 32,988
   Total number of emails: 52,076

3. **Spam-Assassin Dataset:**
   Number of spam emails: 6,954
   Number of ham emails: 3,797
   Total number of emails: 10,751

4.1 Challenges We Had To Overcome

We faced many challenges while evaluating our add-in and we have mentioned a few of them below:

1. These email datasets were predominantly designed for email body semantic analysis.

Hence most of the datasets we looked at did not have email header information. Since we were trying to prevent email spear phishing attacks, we needed contextual relationship between emails. But these datasets were more focused on general spam analysis and hence did not have any contextual relationships.

2. The ideal dataset we needed required metadata for not only the attackers email but also the metadata from the contact that was being impersonated. Hence, we required a dataset that had complete email addresses from the user's inbox and address of the attacker. All the current datasets are a random sample of spam and ham emails collected into a single database. Moreover, these datasets were not designed specifically for email spear phishing attacks.

3. Since these datasets were focused on general phishing attack, most of the emails did not have reply-to email addresses. This was because, these email datasets were designed for semantic analysis and hence most of the emails were missing reply-to addresses.

4.2 Evaluation Results

The results for the three datasets are presented in Table 4.1 below:

Table 4.1: Evaluation Results

Dataset	**Sensitivity**	**Precision**	**False Negative Ratio**
CSDMC2010_SPAM	0.008	0.32	0.017
Enron Email Dataset	0.07	0.4	0.94
Spam-Assassin	0.05	0.89	0.95

The performance result for Phish Muzzle are shown in Table 4.2 with a corpus of 1,241 emails.

Table 4.2: Plugin Performance

Operation	**Memory Usage (MB)**	**CPU Usage (%)**	**Time (s)**
Initial Database Ingestion	-	-	2.6
Suspicious Email Calculation	-	17	0.063
Suspicious Reply-To Calculation	-	17	0.075
Normal Outlook Startup	106	15	3
Outlook Startup with plugin	197	27	6.5
Outlook operation with plugin	197	13	-

CHAPTER 5

Discussion

Results for the three datasets are shown in Table 4.1. We believe that the datasets were not focused primarily on email spear phishing attacks but instead were created primarily for training machine learning model after semantic analysis. Hence, they focus more on the body of the emails than the header.

Since there are no open datasets available for targeted phishing attacks, we were unable to get accurate test dataset. Hence our results are not well showcased with the presented datasets. But, we decided to use these open datasets anyway.

On the attacks described (i.e. small variation from a legitimate email), the algorithm was 100% successful in identifying such attacks.

From the plugin performance metrics, it's evident that using the plugin has an extra overhead over the normal operation of Outlook. There is a delay of 3.5 seconds when starting outlook with Phish Muzzle enabled. The main cause for this delay is the initial database ingestion which came out to be 2.6 seconds for a corpus of 1,241 emails. We also observed an increase in database ingestion time for increased corpus size. For instance, a corpus size of 2,435 emails took 9.3 seconds. Moreover, first run of the plugin uses about 27% CPU because the plugin creates its initial knowledge base. However, after the first initialization, the overhead is unnoticeable, even if inbox has emails in tens of thousands. This is evident by the amount of time taken for the calculation of suspicious matches as shown in the result above.

CHAPTER 6

Conclusion

Spear-phishing is a real problem as recognized in the literature [1]. Our Phish Muzzle method is easy to deploy and can act in addition to standard methods. We found it easy to use, efficient even on large inboxes with 10 of thousands of emails and effective. The parameter of how far of the edit distance to calculate was a tunable parameter, and we tried it on distance two and distance three.

CHAPTER 7

Appendix

7.1 Alert Form

Figure 7.1 displays the Alert form whenever a suspicious email id is detected. This form is displayed whenever user clicks on a potential suspicious email. The user has an option to trust the email id and once trusted that email id won't be detected for any further checks. This can be undone by going into the settings and untrusting the email id manually.

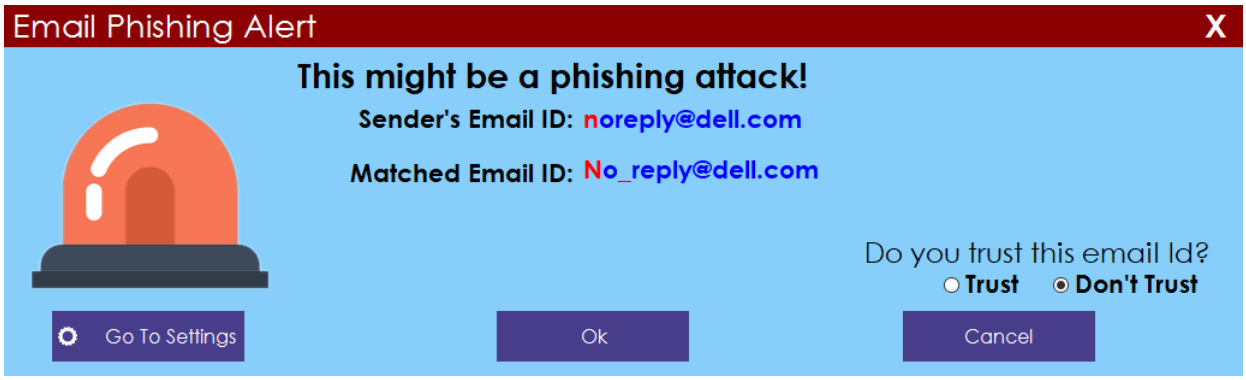

Figure 7.1: Email Id Alert

Figure 7.2 displays the Alert form whenever a reply-to address is not found in the knowledge base for the corresponding email id. Additional to the email id feedback, reply-to address alerts also have the functionality to trust certain reply-to addresses. Once trusted, these reply-to addresses are added to the knowledge base and it won't be detected as suspicious again.

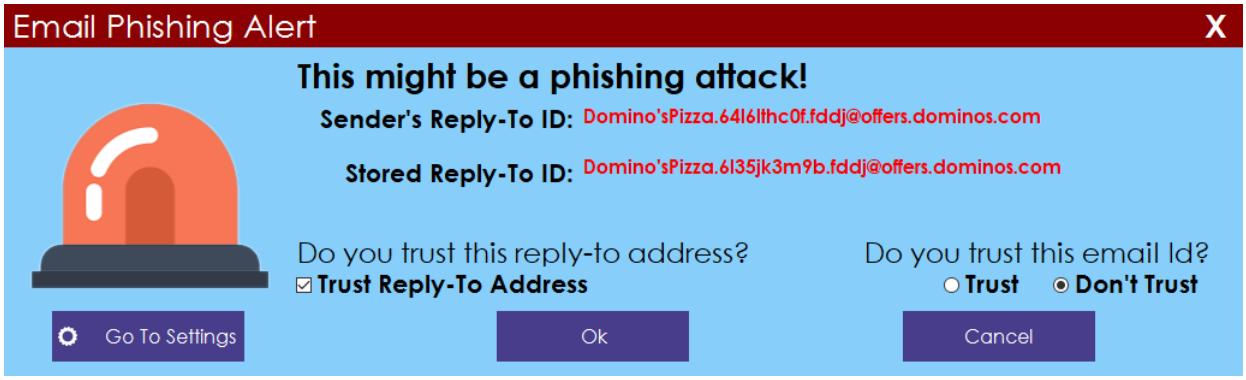

Figure 7.2: Reply-To Alert

Both feedback mechanisms allow the user to make the system faster by incorporating user feedback into the knowledge base.

7.2 Add-in Settings Form

7.2.1 Stored Database

Figure 7.3 displays Stored Data page in the settings window displays the knowledge base and the user can choose to delete a particular entry from the knowledge base. User also has the option to trust or untrust all the email ids in the knowledge base. Moreover, individual email ids can be trusted as well.

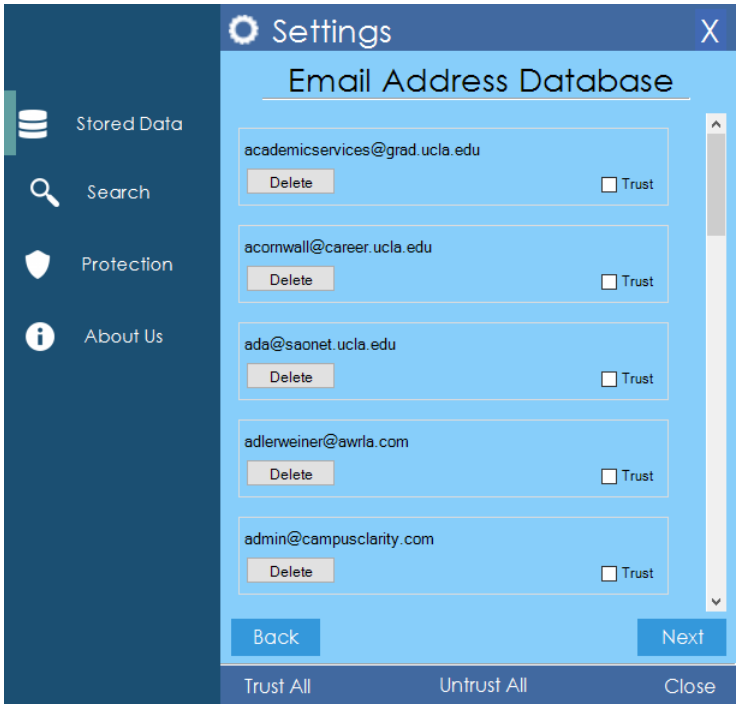

Figure 7.3: Stored Data Settings

7.2.2 Search

Figure 7.4 displays the search page in the settings window allows user to search for a particular email id in the knowledge base. This option is provided for easy navigation through the knowledge base. The search page also accepts regular expressions accepted by MySQL.

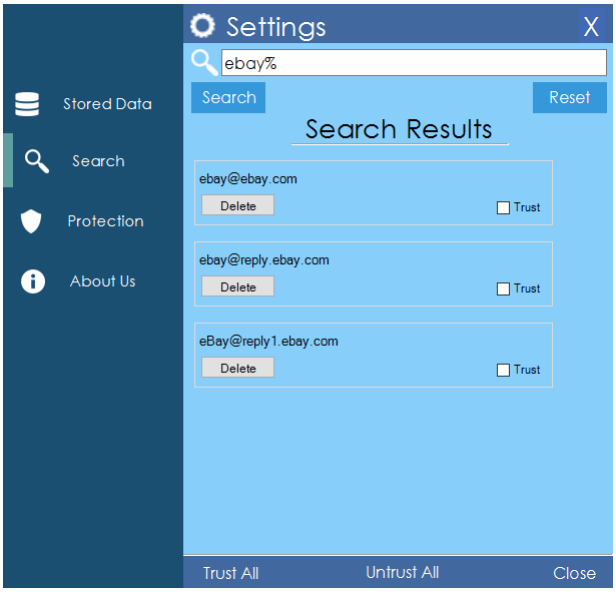

Figure 7.4: Search Settings

7.2.3 Protection

Figure 7.5 displays the protection page in settings window which allows users to selectively enable or disable email id protection and reply-to address protection.

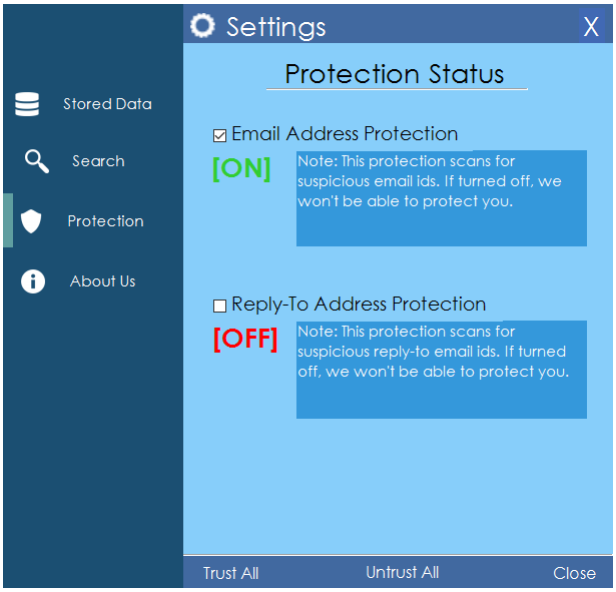

Figure 7.5: Protection Settings

REFERENCES

[1] Phishing Activity Trends Report, APWG, 4th Quarter 2016.

[2] C. Lorrie, E. Serge, H. Jason, Z. Yue, *"Phinding Phish: An Evaluation of Anti-Phishing Toolbars"*, CMU CyLab, November 13, 2006.

[3] F. Ian, S. Norman, T. Anthony, *"Learning to Detect Phishing Emails"*, WWW Conference, May 8-12, 2007.

[4] R. Isidore, H. Tien, *"Chung-Kwei: a Pattern-discovery-based System for the Automatic Identification of Unsolicited E-mail Messages (SPAM)"*, IBM.

[5] C. Madhusudhanan, N. Krishnan, U. Shambhu, *"Phishing E-mail Detection Based on Structural Properties"*, SUNY Buffalo.

[6] Microsoft Sender ID,
https://technet.microsoft.com/en-us/library/aa996295(v=exchg.160).aspx

[7] A. Keno, B. Nicolas, W. Roger, *"Spamato An Extendable Spam Filter System"*, Computer Engineering and Networks Laboratory, ETH Zurich

[8] D. Prateek, K. Anand, K. Ponnurangam, *"Analyzing Social and Stylometric Features to Identify Spear Phishing Emails"*, Symantec Research Lab.