Mining Phishing Campaigns using FP Trees

by

Anirudh Karanth

B.Eng., Visvesvaraya Technological University, 2016

A Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Engineering

in the Department of Electrical and Computer Engineering

SUPERVISORY COMMITTEE

Mining Phishing Campaigns using FP Trees

by

Anirudh Karanth

B.Eng., Visvesvaraya Technological University, 2016

Supervisory Committee

Dr. Issa Traore, Department of Electrical and Computer Engineering
Supervisor

Dr. Imen Bourguiba, Department of Electrical and Computer Engineering

Departmental Member

ABSTRACT

Phishing is a fraudulent online activity conducted by hackers to obtain sensitive information such as credit card number, social security number, or passwords of a user by disguising themselves as legitimate entity via emails, text messages or phone calls. It has been reported that in 2019 nearly 4% of all emails were phishing emails, which correspond to about 3.4 billion emails. Analyzing those phishing emails is an important step towards understanding the motivation and methods of phishers. However, analyzing manually that amount of astronomical data is impossible and ineffective considering that phishers are always finding unique and novel methods to evade detection. One way to keep up with the huge amount of data and the growing sophistication in evasion tactics is to focus the analysis around phishing campaigns. A phishing campaign is the collection of phishing emails built from the same template.

This report adapts and extends previous work on spam campaigns for mining phishing campaigns. The phishing campaigns are mined using Frequent Pattern Tree (FP Tree). The campaigns are identified by investigating the contribution of different email features. Experiments are conducted using a dataset consisting of over 17,342 phishing messages, yielding 231 different campaigns in the best case. The campaigns found for given set parameters are found to be very stable with an error percentage of around 1.5%.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr. Issa Traore, for his constant guidance, support and motivation for my project and throughout my program.

I would also like to thank Dr. Marcelo Luiz Brocardo, for his continuous support and assistance all through my project.

DEDICATION

Dedicated to my mom ***Mrs. Shantha Karanth***, my dad ***Mr. Murali Karanth***, my sister ***Miss Niveditha Karanth*** for their endless support, motivation and guidance.

Chapter 1 : Introduction

1.1 Background

Spam has long been used by criminals as a medium to carry out illegal actvities such as stealing confidential information, identity theft, spread malware, etc, on the internet. The Symantec 2018 Internet Security Threat Report highlights that 55% of the emails are spam [1] which is 2% higher than the preceding two years. Adversaries are taking advantage of the increase in technology to send out spam emails faster than ever to carry out vicious activities. Spam accounts for more than 15 billion messages per day [2] and the sheer amount of email traffic on the Internet makes it next to impossible to manually analyse each of them and their origin along with its intent.

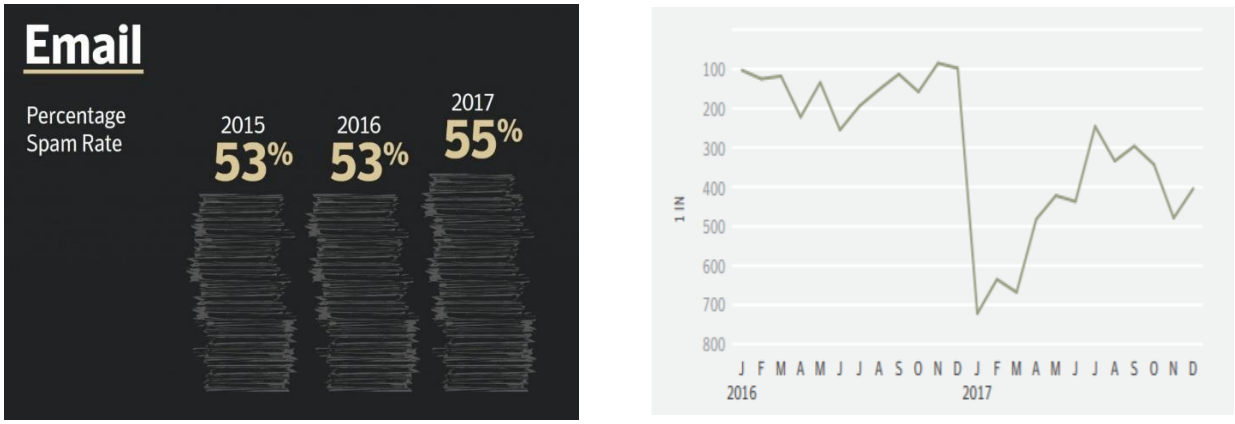

Figure 1.1 *Percentage of spam email and email malware rate 2016-17* [1]

Phishing is particular category of spam which is intended to steal sensitive information from users. Most of the studies use static dataset to analyze the phishing emails that are collected during a specific time frame. However, to avoid evade detection phishers are evolving their craft from using simple programs to send out phishing emails to current point wherein they use highly sophisticated software which generates email templates and send them to a wide variety of users.

To understand how phishers abuse the network resources and to get in-depth knowledge of their target and the motive behind sending the phishing emails, it is important to group the messages into campaigns. A phishing campaign is the collection of phishing emails built from the same template.This grouping of phishing messages helps to identify the behavior and patterns of phishers which is otherwise left unknown due to the volume of the messages. Grouping of phishing messages is also helpful by helping divide the huge amount of phishing data into smaller chunks that can be analyzed seperately.

1.2 Objective and approach

The main objective of this project is to group phishing emails into campaigns by building a Frequent Pattern tree. Once the tree is built, we asccend the tree and find campaigns based on the conditions that we present in the later part of this paper. We run the phishing campaign detection process regulary to detect new campaigns. We adapt and extend an existing FP tree based algorithm developed for spam camapaign identification.

1.3 Report Outline

The remaining chapters in this report are structured as follows:

In Chapter 2, the dataset and the features model are presented.

Chapters 3 gives an overview of the construction of an FP-Tree and the phishing campaign detection algorithm.

In Chapter 4, we present the experiment conducted to explore different parameter settings for the camapaign detection algorithm and the evaluation of the algorithm.

Chapter 5 contains the concluding remarks and an outline of future works.

Chapter 2 : Dataset and Data Model

In this chapter, we present the dataset used in our experiments and give an overview of the feature used to build the FP-Tree. Also, we give an outline on the flow of the phishing campaign detection process.

2.1 Data Source

The dataset consists of a combination of phishing and spam messages collected privately and through online public repositories such as Kaggle. The dataset consists of 17,342 messages, each originally in .eml format. The messages are parsed by extracting the features through a web service (as depicted in Figure 2.1), which are then stored in JSON files, one file per message.

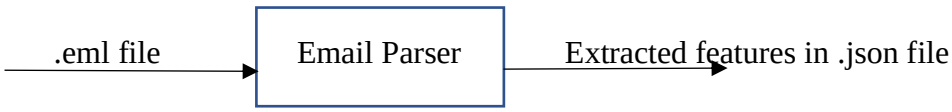

Figure 2.1 *Parsing web service*

Note that the dataset does not contain any benign messages, since the goal is to extract phishing and spam campaigns.

The input dataset to the campaign detection algorithm is a list of transactions, where each transaction consists of features extracted from an email. From the initial set of features, we carefully select the features that are required to build the tree which we use to group them into campaigns later.

Figure 2.2 shows a typical example of phishing email before parsing in .eml format.

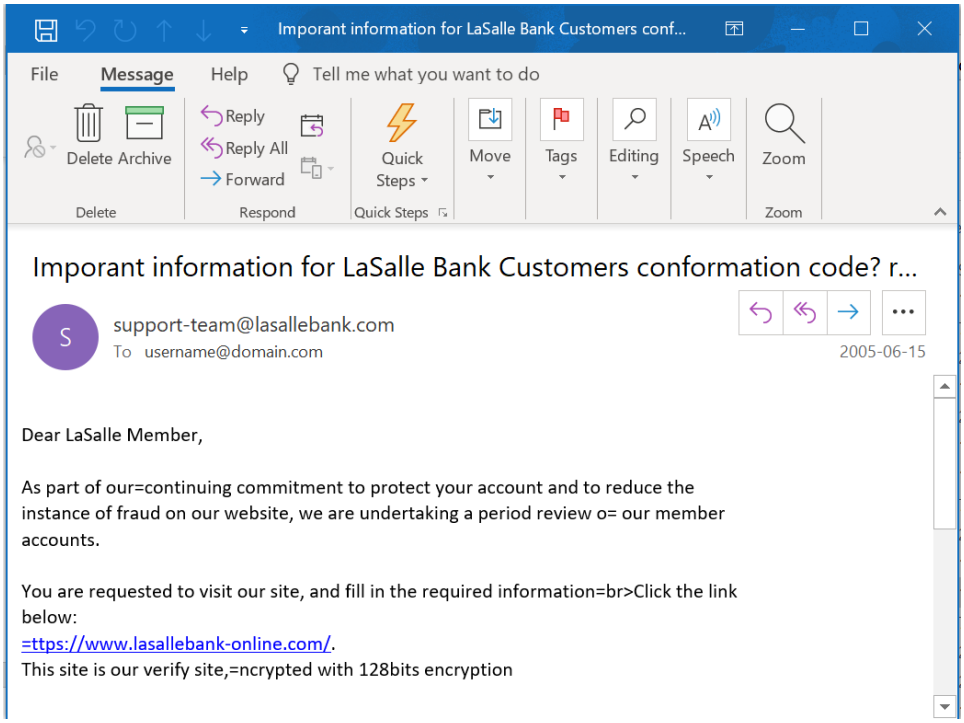

Figure 2.2 *Phishing email before parsing*

Figure 2.3 shows the output of the parser for the same .eml file depicted in Figure 2.2.

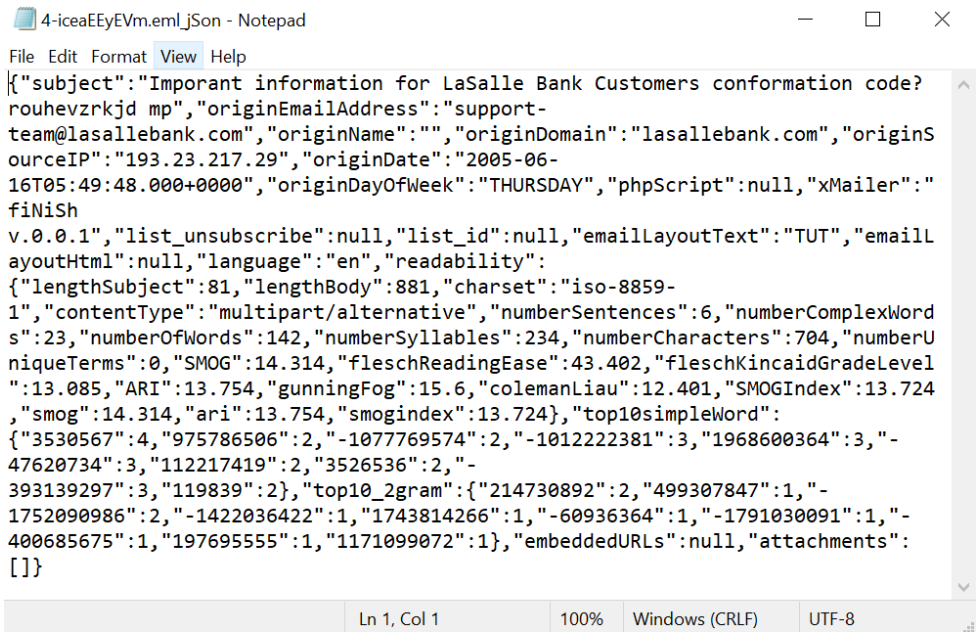

Figure 2.3.*Features extracted by parsing the sample message shown in Figure 2.2*

2.2 Campaign Features

As depicted in Figure 2.3, the parsed email file consists of serveral features such as email subject, origin email address, sent date and time, hashed top 10 words in the body of the email, content type of the email, layout of the email, embedded URLs, attachment information, etc. Out of all these features, we select a few that we think can be unique to a campaign. We include these features in the feature vector for every transaction. Specifically, the following features are used to form the feature vector for every transaction:

Language

This feature represents the language the email is sent in. It is expected that messages in a campaign are written in the same language [3]. This feature is rarely obfuscated by phishers.

Content type

This feature represents the MIME type of the email. This can be found in the content type header of the email. Examples of MIME types include text/plain, multipart/alternative, text/html, multipart/mixed, etc.

Character set

This feature allows the email application (e.g. Outlook or Gmail) to interpret the text characters in the body and subject of an email. It represents the type of encoding that has been done on the email. Character set can be extracted from the content header of an email. This feature is rarely obfuscated by the spammers due to the fact that the email must be successfully decoded for the reader to be able to read the content of the email.

Subject

The whole subject line in the email is considered as a feature. In most cases, phishers use the subject line as a bait for the reader to create interest or motivation to open the email and eventually click on the links embedded in it. Phishers may slightly alter the subject line [4] within the campaign but when interpreted manually, they have the same meaning.

Email Layout

Email layout maps the formatting properties of an email to a sequence of characters T, N, U which stand for Text, New line and URL, respectively, in case of text/plain content type. For text/html, the email layout is obtained by having the top 3 levels of the document object model (DOM) tree. We use the tree structure of the body of the email to get the layout of emails having multipart content. This feature is also largely invariant in a campaign although phishers may introduce some random text in between to obfuscate this feature [5].

URL Tokens

Any embedded URLs present in the spam emails are split into three tokens – hostname, path, and the parameter. Each token is considered as a separate feature. This feature is one of the most obfuscated feature in a spam campaign. Spammers vary the path or the parameter token in the embedded URL by inserting random strings in between them to spy on the receiver. They use this obfuscation technique to confuse the spam filter or redirect the user to some other website.

Origin Domain

This feature represents the domain name of the email. This feature remains invariant inside a campaign .

Origin Email address

This feature represents the sender email address. This can be either an email from a personal account or an email from a work account of a domain.

Origin Source IP address

This feature represents the IP address from which the email is sent. This feature could be helpful in determining what part of the world the email originates from.

Origin Date

This feature represents the time and date the email was sent. Some phishing campaigns send out emails during the weekdays as this increases the chance of user reading the email and clicking on any of the URLs present in it.

2.3 Process flow

Figure 2.4 shows the steps involved in finding the phishing campaigns starting from the arrival of the phishing email to storing the items of a campaign.

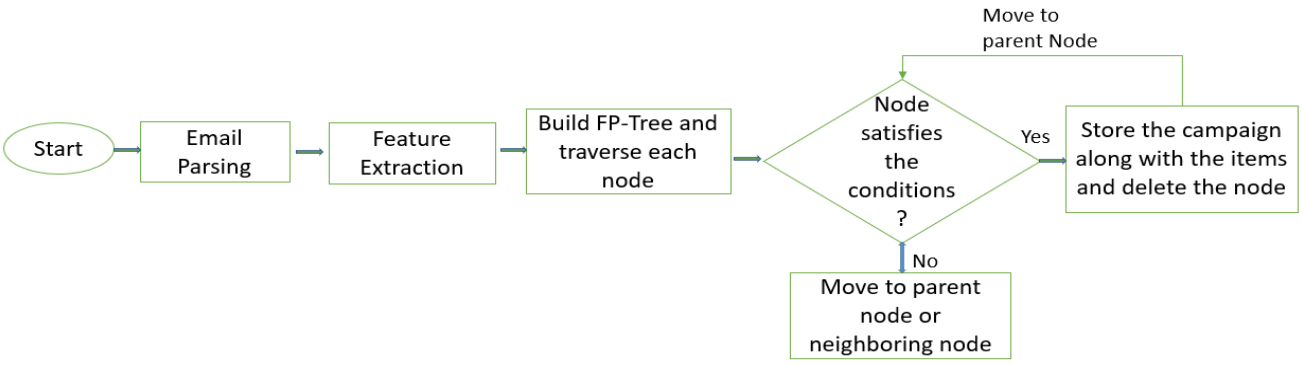

Figure 2.4 *Process flow of Phishing campaign detection*

The different steps of the process flow are described in the following.

Email Parsing: This step involves getting the phishing email, extracting and storing the initial features of the email into a JSON file.

Feature extraction: This step involves extracting the campaign features from the json file. We extract the relevant features that are discussed in this chapter and create a feature vector for each email. This step also involves converting the feature vectors of all the emails into a list of lists that will be used to build the tree.

FP-Tree construction: This step involves constructing the FP-Tree using the minimum support count. Once the tree is built, we ascend the tree and check each node against five conditions discussed in Chapter 3. If a node satisfies all the conditions, we store the node item as campaign along with all the items present in its subtree. We then delete the node and move to another node to check for those conditions. If a node doesn't satisfy all the conditions, we ignore that node and climb to another node.

Chapter 3 : Phishing Campaign Identification using Frequent Pattern growth algorithm

3.1 Frequent Pattern Tree

Fp-growth [6] algorithm uses a compact Tree Data structure to store the data. An item can appear more than once in the tree unlike a search tree. An example of FP- Tree is shown below in Figure 3-1.

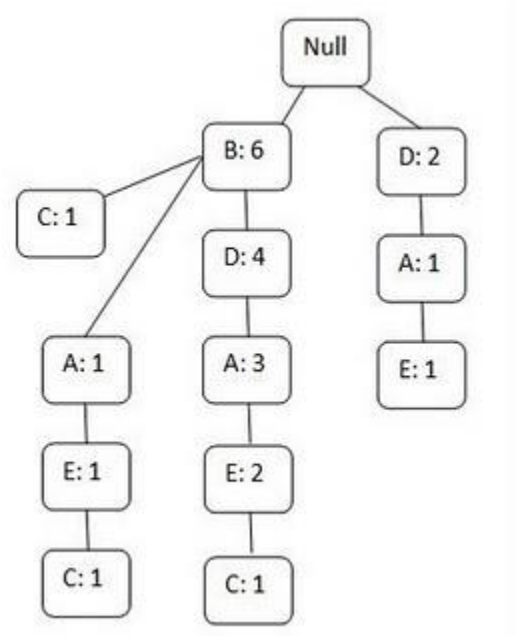

Figure 3.1 – *Frequent pattern Tree* [7]

Before we deep-dive into FP-tree, we will look at some common terminologies.

Root Node: The top most node present in a tree is called a Root node or Null node. It does not have any items or values stored in it. In the example shown in figure 3-1, Null is the root node.

Node: A tree is built using multiple nodes. Each node contains a value or an item. In our work, each node has an item and a corresponding count of the item. In the above example all the blocks in the tree are nodes. Each node has a key and a corresponding value.

Child Node and Parent Node: If a node is connected to another node that is present below it, the node that it is connected to is called child node and the node is called Parent node. Each node can have multiple child nodes (children) but any given node will have only one parent node. The root node doesn't have any parent node since it is present at the top of the tree. Nodes that are connected to the same parent node are called Siblings. In the above example (figure 3-1), B:6 and D:2 are the children of Null. B:6 is the parent node for C:1, A:1, and D:4 and hence they are Siblings.

Leaf Node: In a tree, a leaf node is a node without any children. They are present at the bottom of the tree. In the above example, C:1 and E:1 are leaf nodes (leaves) in the tree.

Subtree: As the name suggests, a subtree is a subset of a tree that has all the properties of the tree such as nodes, root, leaves, etc. All the nodes except the leaf nodes will have a subtree. In the above example , the subtree of node D:2 has 2 nodes, 1 leaf node and a root which is the D:2 node itself.

Items: Each tree can be defined as a dictionary, in which the items are the keys of the dictionary. Each node stores an item in it.

Transaction: A transaction is a list of items. In our work, each transaction consists of the features of a phishing email.

3.2 Building an FP- Tree

One of the advantages of using FP-growth to build an FP-Tree is the fact that this approach requires only two scans of the dataset to build the tree. First scan of the dataset is used to count the number of occurrences of each item and eliminate infrequent items. The second scan is used to build the tree from the frequent items list we created during the first scan. Let us illustrate how FP-growth algorithm works by considering the simple transaction list shown in Table 3.1.

Transaction ID (TID)	Items in transaction
001	r, z, h, j, p
002	z, y, x, w, v, u, t, s
003	z
004	r, x, n, o, s
005	y, r, x, z, q, t, p

Table 3.1 *Transactions with Transaction ID*

In the table 3.1, there are 5 transactions with unique transaction ID and some items in each transaction. The output of the first scan of the algorithm would be to identify the number of all the unique items present in all the transactions along with their count. We then introduce the minimum support count which means that if an item count is less than that of the minimum support count, we will not consider that item in building the tree. Once the infrequent items are eliminated, we now build the tree by considering one transaction at a time. The items in the transaction are sorted in descending order based on the frequency of their occurrence. Let us assume that the minimum

support count is 2 in our case. Table 3.2 lists the items for which the number of occurrence is above the minimum support count after the initial scan.

Item	Z	R	X	Y	S	T
Number of occurrence	4	3	3	3	2	2

Table 3.2 *Items with occurrence number greater than the minimum support*

Table 3.3. lists the sorted transactions after the second scan.

TID	Items in transaction	Sorted transaction
001	r, z, h, j, p	z,r
002	z, y, x, w, v, u, t, s	z,y,x,t,s
003	z	z
004	r, x, n, o, s	r,x,s
005	y, r, x, z, q, t, p	z,r,y,x,t

Table 3.3 *Sorted transactions after the second scan*

The items in the sorted transactions are added into the tree starting from the root. If the item is already present, the count of the node is incremented and we proceed to the next item. If the child node doesn't have the same type of item present in the transaction, we add a new branch to this node along with the rest of the items in the transaction.

3.3 Phishing campaign identification

In [8], an FP-tree based algorithm was proposed to identify spam campaigns. We adopted in this project the proposed algorithm for the identification of phishing campaigns. Although the

core algorithm remains the same, we did some modifications to the properties of the nodes and added a condition to determine if a node is a campaign or not. Furthemore, our implementation of the algorithm relies on an extended feature space. Specifically, we leveraged the features proposed in [8] and also added five new features.

The rationale for using FP -Tree in phishing campaign identification is the fact that the more common a feature is, the more it is shared among the phishing messages. Due to the fact that common features among the emails are grouped together in an FP-Tree, it has smaller dataset than the original dataset and this reduces the amount of data that needs to be processed.

Figures 3.2 and 3.3 depict the spam campaign detection algorithms proposed in [8]. Two data structures are in these algorithms: FPTree and FPNode. The FP-Tree has one root and one property:

- The root of an FPNode is Null. The root node does not hold any value in it.
- The add_transaction property accepts each feature vector fv consisting of a sorted transaction as its parameter and add the items of the transaction to the tree.

After each feature vector is added to the tree, we have to upate the route of all the nodes of the tree. After inserting all the feature vectors, we will have a tree with each node representing a feature from the email along with the number of times it appeared at that point in the tree.

The FPNode has nine properties and four methods:

- The tree property indicates the FPTree of this node [8]
- The item property of the node represents the feature of the email stored in a node
- The count property returns the number of times the item in the node has appeared.

- The root property is boolean. It returns true if the given node is root, false otherwise. This property allows us to use any given node as root and the subtree of that node .
- The leaf property is boolean. It returns true if the given node is a leaf, false otherwise.
- The parent property of a node returns the parent node of the given node.
- The children property indicates the children nodes of the given node.
- The siblings property indicates the neighboring nodes of the given node.
- The add(node) method adds the given FPNode node as a child of its parent node and updates the information in the tree.
- The search(nodeitem) method searches the tree and check if the given node item has a child or not.

In addition of the above elements, we have defined the following property and methods:

- The num_leaves property returns the number of leaves the given node's subtree has.
- The remove(node) method removes the given node from the tree along with all the items in its subtree.
- The get_items method returns the items of the node's subtree if the node satisfies all the conditions.

```
Algorithm 1: FP-Tree Construction
   Input: List of Transactions
   Output: FP-Tree
 1 Create Transactions list                          // empty list structure
 2 Create items dictionary                           // empty dictionary
 3 for each spam email do
 4 |   create a list fv                               // empty list structure
 5 |   for each feature do
 6 |   |   add feature to fv
 7 |   |   items[feature] ← items[feature] + 1
 8 |   end for
 9 |   add tid to the end of fv
10 |   add fv to list of transactions
11 end for
12 Initialize the FP-Tree
13 for each transaction in transactions do
14 |   if count(transactions) > 2 then
15 |   |   Sort the items in transaction[1 : —1] based on their counts in
   |   |     descending order in items dictionary  // tid and contenttype are
   |   |     left out
16 |   |   call FPTree.add(fv)
17 |   end if
18 end for
```

Figure 3.2 *Algorithm for FP-Tree construction*

```
 1 function add(fv)
 2     initialize root node
 3 for each each feature in fv do
 4 |   search nextpoint of current node
 5 |   if nextpoint is None or neq to feature then
 6 |   |   instantiate FPNode class
 7 |   |   Add feature as a child of current node
 8 |   end if
 9 |   if nextpoint eq feature then
10 |   |   Merge feature with child of current node
11 |   |   Increment count of nextpoint
12 |   end if move to nextpoint
13 end for
14 end function
```

Figure 3.3 *add transaction algorithm*

By executing the algorithm, we will have at the end the FP-Tree where each node in the tree represents the features we extracted from the phishing emails. The path from each leaf node to the root represents the feature vector of the corresponding phishing email. Two phishing emails

sharing common features will share the same portion of the tree. We use this characteristic to group the similar phishing emails with several common features into campaigns.

Once the FP-Tree is built, we then traverse the tree bottom up using depth first search method to group the messages. While visiting each node, we implement the following four conditions and check if the node satisfies these conditions:

- The number of children of that node must be greater than the threshold min_num_children parameter [8].
- The avaerage frequency of the children of that node must be greater than the threshhold_frequency parameter [8].
- The path from that node to the root must have at least one node that does not have its feature type in the n_obf_features list [8].
- The number of leaves starting from that node must be greater than the threshold min_num_messages [8].
- The node must have its feature type in the features list.

If a node satisfies all the above conditions, we consider that node as a campaign and store all the items under that node's subtree. We then delete the node and move on to other nodes until every node in the tree is visited. If a node doesn't satisfiy any one of the above conditions, we ignore that node and move to its parent node. If we reach the top most node of a branch in the tree, we move to the neighboring branch and start the process from the bottom of that branch.

Chapter 4 : Experiments and Implementation

In this chapter, we conduct some experiments to explore different parameter settings and discuss the model evaluation results. We also give an outline of the algorithm implementation.

4.1 Varying Campaign Detection Settings

We run the phishing campaign detection algorithm on the full original dataset consisting of 17,342 phishing messages by varying the following setting paraemeters:

- Number of children paramater: min_num_children
- Minimum threshold frequency parameter: threshold_frequency

We vary the above parameters while keeping the required full campaign feature list constant and the minimum number of leaves starting from the node's subtree min_num_leaves = 4.

The required full campaign feature list is [Language, Origin Domain, Origin Name, Origin SourceIP, Hostname, Path, Origin email address, Subject, Charascter set, Content type].

Table 4.1 presents the number of campaigns found by varying the abovementioned parameters.

Min_num_children	Threshold_frequency	Number of campaigns found
3	3	231
4	4	110
5	5	57
6	6	36
7	7	24
8	8	18

9	9	16
10	10	13

Table 4.1 *Campaigns found for different input parameters*

The settings with the most campaigns are (min_num_children=3, threshold_frequency=3, min_num_leaves = 4) yielding 231 campaigns.

Below are a couple of sample campaigns generated using the algorithm.

1. **Campaign Name** : Southtrust.com

   Campaign type: Origin Domain

   IP addresses - 24.60.82.125, 85.250.93.239, 24.22.2.133, 12.218.231.160

   Some common origin email address - support_refnum_05609508@southtrust.com, support_num_874@southtrust.com, support_num_5713484242@southtrust.com.

   Some common Origin Names: SouthTrust Bank, SouthTrust

   Some common subject lines: 0fficiaI Information To SouthTrust Bank CIients [Thu, 30 Jun 2005 17:39:50 -0300], Important account notification [Fri, 08 Jul 2005 00:04:48 -0200], SOUTHTRUST BANK - URGENT SECURITY NOTICE, SouthTrust Bank: Please Validate Your Account [Mon, 11 Jul 2005 01:51:45 -0200], SOUTHTRUST BANK: SPECIAL ANNOUNCE

2. **Campaign Name:** sendgreatoffers.com

   Campaign type: Origin Email address

   IP addresses: 209.216.124.213, 66.111.250.61, 66.111.219.130

   Some common Origin email address: greatoffers@sendgreatoffers.com, ldistancesgo@SendGreatOffers.com, steakssgo37@SendGreatOffers.com.

Some common Origin Names: Lowest Long Distance, Omaha Steaks, Great Offers, Shannon, Your Weight-Loss Partner, Classic Wines, Benefits Department.

Some common subject lines: Best Long Distance On the Net - 3.9 Cents With No Monthly Fees, Get 12 free Burgers + 1/2 Price Omaha Steaks!, Best Long Distance On the Net - 3.9 Cents With No Monthly Fees, Introducing Chase Platinum for Students with a 0% Introductory APR, You have been hand selected... Free Info!, Shipment Status, *Do Your Butt, Hips And Thighs Embarrass You?*.

4.2 Stability of the phishing campaign detection model

Since we do not have a reference labeled campaign dataset, we test the stability of the campaigns found using the phishing campaign identification algorithm in the following way.

First, we run the campaign detection algorithm using the original phishing email dataset and identify the corresponding campaigns.

Next, we select randomly 1% of the abovementioned dataset to be used as test data. We then add the test data to the same dataset and built the FP-Tree again using both the test and old dataset. We run again the campaign algorithm to see the number of campaign found. Ideally, the number of campaigns found should not change since the test data that we are using is already a part of the original dataset that we used to build the original tree. To assess this assertion, we calculate the stability of the algorithm as follows.

Let x be the total number of messages used in the first run to build the initial FP-Tree. Let Δx be the number of test messages selected randomly from the initial dataset. We now build a FP-

Tree using x+ Δx. We compare the number of campaigns found using both the FP-Trees. Let C_x be the number of campaigns found using *x* number of messages and $C_{x+\Delta x}$ be the number of campaigns found using x+Δx number of messages. The stability of the tree can be defined as:

$$\text{Stability} = 1 - \left| \frac{C_{(x+\Delta x)} - C_x}{C_x} \right| \times 100$$

We executed the above test scenario through 5 iterations, where the test data is selected randomaly each time, and computed each time the above stability measure. Table 4.10 lists the obtained results.

Iteration	**C_x**	**$C_{x+\Delta x}$**	**Stability**
1	231	233	99.1%
2	232	239	97%
3	239	242	98.7%
4	242	240	99.1%
5	240	246	97.5%

Table 4.2 *Stability of the model after each iteration*

The number of campaigns found after each iteration varies by about 1.5% on an average. This is due to the fact that the parameters that we are using to determine if a node is a campaign or not are constant for a given tree, i,e they will not change for a given dataset except for the threshold_frequency parameter. By this time, we know that messages sharing a common feature share the same path in the tree until they differ. So after each iteration min_num_children

parameter and min_num_leaves parameter remain the same since we are adding the messages from the same dataset. But the count of the nodes change after each iteration and hence threshold_frequency parameter changes with the change in count of the nodes.

4.3 Implementation

We have implemented the phishing campaign detection algorithm by developing a web service and a watchdog module depicted by Figure 4.1. We have built a web service that receives push notification whenever there is a new phishing email reported by an existing phishing detection system. The service listens for the notification and stores the new parsed email in a folder. We then use watchdog library to determine if the amount of new emails exceed our pre-determined threshold value. If and when it does, we move these new emails into the folder consisting of older emails and run the algorithm on the updated dataset and find out if there are any new campaigns found.

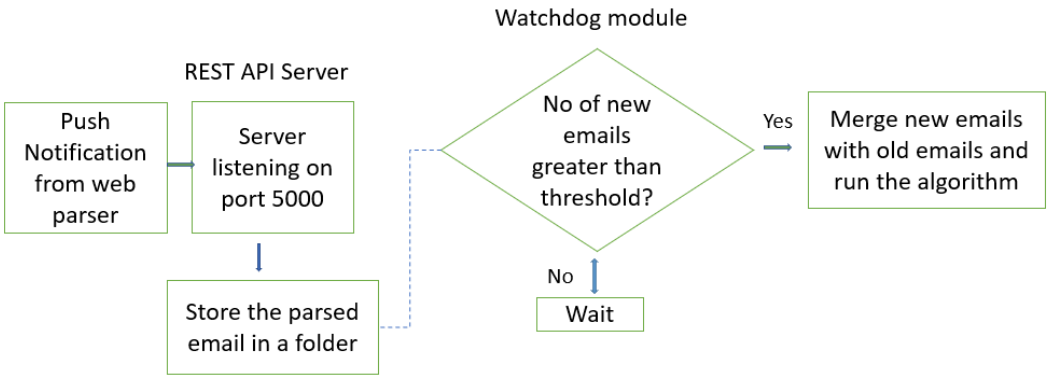

Figure 4.1 *Implementation of Watchdog module*

Chapter 5 : Conclusion and Future work

5.1 Summary

With the rise in technology, the phishers keep finding unique ways to send the phishing emails undetected by the email filters. It is very important to get more information on the phishing emails to do detailed analysis on their origin and the intent behind sending out those emails. By grouping similar types of phishing emails into campaigns it would be helpful to continuously improve spam filters to detect them and not let them pass through. But grouping the emails into campaigns manually is impossible due to the astronomical number of phishing emails that are being sent out everyday on the Internet. Another problem would be the obfuscation techniques used by the phishers to change some of the features of emails to bypass the spam filters [9]. One of the approaches to tackle these problems is identifying campaings by building a Frequent Pattern tree using the features extracted from phishing emails.

FP-Tree requires only two scans of the database to build and store the features of emails in it. This increases the speed of parsing the data and getting the campaign information from it. This solves the first problem of keeping up with the huge amount of phishing emails that are generated everyday. FP-Tree also reveals the obfuscation techniques used by the phishers in a very efficient way. Two similar emails varying by some features which are obfusacted would be grouped in the same campaign. By returning the items of each campaign, we can analyze the campaign data and correlate some of the campaigns with its origin and the intent.

In this report, we have adopted and implemented the campaign detection algorithm proposed in [5]. We studied the stability of the proposed alorithm implementation. The campaigns found for given set parameters are found to be very stable with an error percentage of around 1.5%.

5.2 Future Work

As pointed out by the designers of FP-Tree, no algorithm works in all situations [10] and this holds good for our model too. Some of the shortcomings of FP-growth algorithm includes dificulty in building the tree, high memory utilization for large datasets. Our future work will include using a different algorithm to build the tree and comparing its performance against the FP-Tree.

Our future work will also involve finding more efficient conditions to determine if a node is a campaign or not. The campaigns found using this model can further be fed to machine learning algortihms to determine if a new incoming phishing email belongs to an existing campaign. We plan to study more in depth how certain campaigns abuse certain types of network infrastructure. Correlation is also an important part of our future work. Associating a phishing campaign with a location or a range of IP addresses or threat type by using geo-location, WHOIS information [11] and malware database increases the chance of understanding the motive behind the phishing email. Employing a scoring procedure for each phishing campaign based on certain criteria would be helpful to assign priority to a campaign and concentrate and analyze those campaigns with higher level of threat in detail.

Visualization plays an important role in finding and reporting the top most used features in each of the phishing campaigns found. Visualizing features such as range of source IP addresses, Origin Domains, helps us better understand the origin and motive of the emails. Phishing campaigns act as a pivotal instrument to detect several crime-based activities and visualizing the results based on the threat type and scoring mechanisms can help narrow down our focus on what campaigns to look out for and designing filters accordingly.

References

[1] phishingbox.com, "Symantec Internet Security Threat Report 2018 ". Available: "https://www.phishingbox.com/assets/files/images/Symantec-Internet-Security-Threat-Report-2018.pdf". Retrieved July 10, 2020.

[2] David E. Sorkin, "Spam statistics and Facts". Available :"https://www.spamlaws.com/spam-stats.html". Retrieved July 10, 2020.

[3] Pedro H. Calais, Douglas E. V. Pires, Dorgival Olavo Guedes, Wagner Meira Jr. , Cristine Hoepers, Klaus Steding-Jessen, "A Campaign-based Characterization of Spamming Strategies", in CEAS'08. pp. 1–6(2008).

[4] Sophos.com, "The Spam Economy: The Convergent Spam and Virus Threats. August 2004.," 2004. [Online]. Available: http://www.sophos.com/whitepapers/Sophos_spam-economy_wpus.pdf. Retrieved July 08, 2020.

[5] Gianluca Stringhini, Thorsten Holz, Brett Stone-Gross, Christopher Kruegel, Giovanni Vigna, "BotMagnifier: Locating Spambots on the Internet," *USENIX Security Symposium,* 2011.

[6] Han, J., Pei, J., Yin, Y. et al. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery 8, 53–87 (2004).

[7] Hareendra Lakshan Perera , " How to identify frequent patterns from FP tree", 2011. Available: "http://hareenlaks.blogspot.com/2011/10/how-to-identify-frequent-patterns-from.html,". Retrieved July 13, 2020.

[8] Son Dinh, Taher Azeb , Francis Fortin , Djedjiga Mouheb, "Spam campaign detection, analysis, and investigation," *Digital Investigation 12, Supplement 1 (2015), S12 – S21*.

[9] Calton Pu, Steve Webb, "Observed Trends in Spam Construction Techniques: A Case Study of Spam Evolution", In CEAS 2006 - Third Conference onEmail and Anti-Spam, July 2006. Retrieved July 09, 2020.

.[10] William Cheung, Osmar R. Zaïane, "Incremental Mining of Frequent Patterns Without Candidate generation or support constraint, in Proceedings of the7th IEEE International Database Engineering & Applications Symposium(IDEAS'03), 2003, pp. 111–116. doi:10.1109/IDEAS.2003.1214917 ".

[11] Harrenstien K, Stahl M, Feinler E., "WHOIS protocol specification," *Internet RFC 1985;954. ISSN: 2070-1721,* 1985.

[12] V. J. R. Saiyed Kashif Shaukat, "RansomWall: A Layered Defense System against Cryptographic Ransomware Attacks using Machine Learning," in *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, 2018.