2016

Clustering Phish Using The Simple Set Comparison Tool

Jason Robert Britt
University of Alabama at Birmingham

CLUSTERING PHISH USING THE SIMPLE SET COMPARISON TOOL

by

JASON BRITT

ALAN SPRAGUE, COMITTEE CHAIR
MICHAEL BAILEY
TYLER MOORE
NITESH SAXENA
CHENGCUI ZHANG

A DISSERTATION

Submitted to the graduate faculty of The University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

BIRMINGHAM, ALABAMA

2016

CLUSTERING PHISH USING THE SIMPLE SET COMPARISON TOOL

JASON BRITT

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

ABSTRACT

Phishing has been a problem since before the early 2000s and has only become more prevalent and diverse since. Phishing countermeasures have been developed and used to prevent or mitigate phishing attacks. However, each countermeasure has pros and cons and not every countermeasure is effective in every situation. Choosing the best suited phishing countermeasure or combination of phishing countermeasures to use and track their effectiveness requires grouping phish based upon common characteristics and tactics used by phish or phish grouping. To be effective phish grouping needs to produce dependable groupings, quickly produce groups, and analyze large volumes of phish.

This dissertation develops the Simple Set Comparison (SSC) tool. The SSC tool enables existing phish grouping processes to run faster. It also decreases the maximum amount of memory required allowing grouping of a larger number of phish. The SSC tool utilizes a multi-step approach that makes use of parallel processing to improve runtime and reduce the maximum amount of memory required. This dissertation evaluates the efficiency and quality of using the SSC tool with the SLINK style phish grouping algorithm used by Malcovery Security. The SLINK style algorithm using the SSC tool is compared to the SLINK style algorithm without using the SSC tool on the ability to produce a clustering, the quality of the clustering produced, and the runtime to produce a clustering.

Four experiments are run using three different implementations of the SLINK style clustering algorithm over large phishing data sets. The SSC tool improved the runtime of the SLINK style algorithm in each experiment. The SLINK style algorithm algorithm with the SSC tool produces results 37 times faster than without in the first experiment, 404 times faster in the second experiment, 6 times faster in the third experiment, and 10.8 times faster in the fourth experiment. The tool produces results faster, while maintaining equivalent quality. The SSC tool improves the SLINK style algorithm's runtime and reduces the maximum amount of memory required to produce a clustering, allowing larger volumes of phish to be grouped, and produces similar clusterings to the SLINK style algorithm without the tool.

DEDICATION

Dedicated to...

Jenn, Thanks for all the love and support.

It’s been a long road and there is more to come...

ACKNOWLEDGMENT

Many people helped me along my path. I would like to thank Jenn for all the love and support throughout my graduate studies as well as my parents, Sylvia and George, and my two younger sisters, Catherine and Laura.

I would also like to thank Dr. Sprague, Gary, and my committee members for providing assistance and guidance during my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Overview

Phishing is a social engineering attack that attempts to convince victims to provide personal or sensitive information to the criminal by directing the victim to a website that imitates another website. The information secured by the criminal is then used to gain access to systems used by the victim, gain access to the victim's financial accounts, or perpetrate identify theft against the victim.

Phishing has been a threat since before 2003 and continues to be a significant and increasingly frequent threat today. There are many different countermeasures used to combat phishing attacks. Each type of countermeasure has its own advantages and disadvantages. Selecting the appropriate type of countermeasure to counter the type and scale of phishing attack is important to effectively combat the attack.

Phish clustering is the ability to group phish by a shared characteristic. The goal of phishing clustering is to identify related phish and can be used to make determinations about the techniques used and motivations behind a concerted phishing attack or phishing campaign. Quick and accurate phish clustering is needed to enable selection of the appropriate type of countermeasure and quantify a countermeasure's effectiveness at combating phishing attacks. To be effective a phish clustering process needs to meet two goals. The first is to produce dependable high quality phish clusterings. The second is to produce results quickly to be able to follow phishing attacks as close as possible to real time.

Algorithms currently exist to perform phish clustering and a number of existing algorithms used for other purposes can be applied to phishing data to produce phish clusterings. Improvements can be made to reduce runtime to produce results closer to real time and reduce memory consumption to process larger volumes of phish. Runtime can be improved and memory consumption reduced while maintaining clustering quality with a new tool, the Simple Set Comparison tool (SSC tool).

The SSC tool utilizes a multi-step approach with parallel processing to improve runtime

and reduce memory consumption. The SSC tool is evaluated in several experiments against a traditional clustering algorithm, SLINK, on the quality of results produced, the runtime it takes to produce results, and the maximum memory required.

1.1 Contributions

The SSC tool improves existing phish grouping algorithms in two areas. It develops phish clusterings closer to real time by improving runtime and allows larger volumes of phish to be processed by reducing the maximum amount of memory required.

1.2 Outline

The rest of the dissertation defense document is organized in a series of chapters covering details about phishing, related work, the SSC tool, the experiments performed, a detailed discussion about the SSC tool, and conclusions drawn from the work. Future work directions are also included.

The introduction chapter gives an overview of phishing. The subsections address why phishing is a threat, current countermeasures used against phishing, how phish clustering can improve the effectiveness of phishing countermeasures, and metrics to use to evaluate a phish clustering process. The evaluation metrics include a phish clustering process's ability to produce high quality clusters, the runtime taken, and the amount of memory required.

The related work chapter covers current and past phishing detection and clustering research. The next topics covered are current data mining techniques and their uses in other fields. The improvement subsection of the related work section outlines areas for advancement in current data mining techniques when trying to achieve the phish clustering goals described in the introduction.

The algorithm chapter covers the details of the SSC tool. There is an overview, a detailed explanation for each of the four steps in the SSC tool, and its theoretical computational complexity.

The experiments chapter covers four experiments performed to evaluate the effectiveness of a SLINK style algorithm using the SSC tool compared to a SLINK style algorithm without using the SSC tool. The experiments compare the ability to process a data set without running out of memory, the quality of the clusterings produced, and the runtime required to produce a clustering.

The discussion chapter sums up the results from all four experiments. It looks at the quality, runtime, and maximum memory of all four experiments. It also covers the interchangeable similarity metric and clustering algorithm.

The conclusion chapter sums up the argument that the SSC tool is more effective at providing phishing intelligence than traditional clustering and current phishing intelligence processes based upon the goals a phishing intelligence process should meet.

The future work chapter covers possible future directions including the use of different similarity metrics and applicability for use with other clustering algorithms useful in different applications other than phishing.

CHAPTER 2

Introduction

The introduction gives an overview of phishing and a direction to help alleviate phishing attacks. It consists of five sections. The first section introduces the threat of phishing and why it matters. The second section gives a broad classification of types of phishing threats showing the variety of social engineering tactics used. The third section describes various phishing countermeasures and how they can combat phishing. The fourth section discusses phish grouping and its importance to effectively using phishing countermeasures. The fifth section lays out the attributes that should be used to evaluate a phish grouping process.

2.1 Phishing Threat

Phishing has been a problem and organizations such as the Anti-phishing Working Group (APWG) founded in 2003 and PhishTank founded in 2005 have been fighting phishing for years [1, 2]. Today, phishing is still a problem. A 2013 Kaspersky lab report places phishing attacks as one of the three most prevalent external threats facing corporations [3]. Between April and June of 2014, APWG reported observing 128,378 new phishing attacks [4]. This is the second highest phishing attack volume observed in a three month period by APWG [4]. Phishing attack volumes are large and have increased over the years.

2.2 General Classification of Phish

Types of phishing attacks can be categorized in many different ways. Phishing can be classified by the type of organization being attacked, the mechanism used to advertise the phishing website, or the particular tactic used to entice users to the phish. The following broad categorization of phish is based upon the type of tactic used to entice users as it highlights the differences in the breadth of organizations affected by phishing and particular circumstances where the different types of phish are effective. The three broad categorizations of phishing used here are continuous phishing, opportunistic phishing, and targeted or spear phishing.

Continuous phishing is a type of phishing that is constantly occurring and generally targets well-known brands. These brands can vary from financial institutions, email providers, and social media firms, or government entities. Continuous phishing is characterized by its broad targeting. It aims for success by advertising to many people to be effective. One example is a phishing campaign targeting university employees around the country. Around the 10th of November 2014 a phishing campaign targeting university employees at George Mason University, University of Chicago, University of Oregon, Brown University, Michigan State University, Georgetown University, and many others [5]. This particular phishing campaign began by sending a phishing email with the subject "Your Salary Raise Confirmation" and contained a link to a website that mimics the particular university's employee payroll website [5]. The goal appeared to be to gain employee payroll logon credentials and banking information to steal money from the employee's bank account and/or redirect employee pay.

Opportunistic phishing is a type of phishing where phishers use a current topic or fear to help enable their form of social engineering to be more effective. Examples include phishing campaigns directed at possibly affected users after a breach notification such as the Anthem breach or the rash of income tax related phishing seen in many different countries around income tax filing time. Opportunistic phishing is characterized by its use of user awareness to assist in making the social engineering attack effective and like continuous phishing it often indiscriminately targets many people. In February 2015 Anthem health care made a public announcement that customer Social Security information and other personal information had been compromised [6]. The systems that were compromised contained information for some 80 million customers. Shortly after the public announcement a phishing campaign began that was targeting current and former Anthem customers. The campaign was tailored around customer fears of the data breach and attempted to navigate victims to a phishing page using a link button in the phishing email that read "Click Here To Get Your Free Year Of Credit Card Protection".

Spear phishing is a type of phishing targeted at and tailored for a very small population. Spear phishing attacks are often aimed at organizations to gain access to internal financial systems or other resources within private networks. Beginning sometime in 2012 and continuing until June 11th 2015 an international group of cyber criminals defrauded over fifty different companies from all over the world for approximately eight hundred thousand euros and caused an estimated five million euros worth of damage [7]. The criminal group tailored phishing emails to particular

company employees, spear phishing, to direct the employees to a phishing site to try and gain the employees' corporate logon credentials. These logon credentials were then used to gain access to the corporate computer network with the goal of redirecting funds intended to pay suppliers to the criminal group. This criminal network was broken up on June 11th 2015 by an international investigation involving multiple European law enforcement agencies. The result of the investigation was the arrest of forty-nine individuals in the following countries: Spain, Poland, Italy, Belgium, Georgia, and the United Kingdom.

2.3 Phishing Countermeasures

Currently there are many different ways of trying to combat phishing. Each phishing countermeasure has its own pros and cons.

2.3.1 Alerting Users

Alerting users can take many forms including email notifications, posts on organization alert websites, or any other method of communicating to a user group. The goal of the alerting users countermeasure is to give the user base an idea of the specifics of a phishing email with the hope the user will recognize the phishing email and not fall victim to it. Specifics about a phishing email include the subject, the text in the email body, and email sender. The alerting users countermeasure is usually going to be performed by the organization being targeted by the phishing campaign and depends on the targeted organization being aware it is the target of a phishing campaign and knowing specifics about the campaign to use to alert users. The Alerting users countermeasure fails to protect users who are not a part of the organization, but are phishing targets.

2.3.2 User Education

User education involves training users to recognize and avoid phishing websites. It can take a number of forms including online lectures, testing, and gamified training [8]. There are many different online training tutorials for detecting phishing websites. Some tutorials are free and offered by not for profit organizations such as that provided by APWG, the Open Web Application Security Project (OWASP), and others [9, 10]. Other tutorials are purchased as a part of user training packages offered by security companies such as McAfee, Wombat Security Technologies,

and others [11, 12].

An example of gamified training is the online game Anti-Phishing Phil created at Carnegie Mellon University [8]. Anti-Phishing Phil presents users with the challenge of distinguishing between phishing websites and non-phishing websites in a game environment with in-game rewards for correct answers. The game also gives players an introductory tutorial on detecting phishing websites via training messages [8]. The goal of the user education is to give users general tools to recognize and avoid phishing emails and phishing websites. User education is important as it decreases the chance that users will fall victim to phishing, but it does not always prevent educated users from falling victim to phishing [8]. Also, user education will only help users that use the education material or exercise, which will not cover all internet users who can possibly become a phishing victim.

2.3.3 Phishing Takedown

A phishing takedown is the process of detecting and then removing the phishing website, also known as a phishing take down. To perform a phishing takedown the phishing website Universal Resource Locator (URL) must first be identified. Next, information about the website such registration and hosting information needs to be collected. Based on the information gathered the website can be taken down a number of ways including contacting the domain owner, sending an email to the abuse email listed on the whois domain information, contacting the hosting service, or the domain registrar [13, 14]. Phishing takedowns are often performed by security companies such as PhishLabs, Cyveilance, and others [15, 16]. These security companies are contracted by a firm to protect the firm's brand via phishing take downs and other countermeasures. The goal of the phishing takedown countermeasure is to limit victim exposure to phishing websites [14]. This countermeasure has been shown to be useful, but not a solution by itself because each phishing website has to be detected, investigated, and removed by a third party ISP, registrar, or hosting provider [14]. The phishing takedown countermeasure combats individual phishing websites and requires multiple parties to cooperate.

2.3.4 Blacklisting

Blacklists are a widely used technique to prevent users from visiting phishing sites [17]. Blacklisting works by developing a list of bad URLs, domains, or IP addresses and preventing users

from visiting these phish. The bad URL, domain, and/or IP address list can be developed from a number of sources including: manual reporting, web crawler heuristics, passive DNS analysis, and honeypots. The blacklists are then used by companies, internet service providers, web browsers, third party software, and others to prevent users from visiting the URLs, domains, and IPs listed in the blacklist. For blacklists to be effective phish have to be identified and their URL, domain, and/or IP added to the list quickly. It has been shown that blacklists are only effective at blocking phish that are twelve plus hours old mainly due to the time it takes collect and identify phish [13]. Blacklists alone are not enough to protect users from phish.

2.3.5 Web Toolbars

Web browser toolbars are used to detect and alert users they are visiting a malicious website, such as a phishing website so the user can avoid the malicious website [13]. There are a variety of methods used including blacklists, various signature detection mechanisms, and other heuristics to detect malicious websites such as phish [13, 17]. Web toolbars depend on user confidence in the tool bar to prevent them from visiting the phish. If the user does not believe or care about the tool bar supplied warning they can override it and visit the malicious website anyway. An evaluation of security toolbars showed that they do completely prevent phish attacks [18]. All of the security toolbars detection mechanisms depend on timely updates to blacklists, signatures, or heuristic signatures to effectively present users alerts against new novel phishing attacks. Web toolbars are not enough to stop phishing attacks by themselves because they depend on user awareness and timely updates.

2.3.6 Email Filters

Email filters attempt to detect and prevent the delivery of spam including spam containing phish. Preventing the user from receiving the phishing advertisement keeps the victim from visiting the phishing site. Email filters user a variety of mechanisms including blacklists, signature detection, and heuristics to detect malicious emails. Like web toolbars, email filters depend upon timely updates to the detection mechanism to prevent a user from receiving malicious emails. Email filters will only stop email advertisements for phish. Phishing advertisements can also be delivered via other mechanisms such as social media applications, malware redirections, and DNS hijacking. Email filters alone do not stop phishing attacks and to be effective at preventing

spam phishing advertisements depend on timely updates to the underlying detection mechanisms.

2.3.7 Criminal Prosecution

Prosecuting criminal behind phishing attacks attempts to decrease the number of criminals performing phishing attacks and deter future phishers by raising the risk of phishing. Criminal prosecutions are difficult to perform due to the number of parties that must be involved to achieve a successful prosecution. Prosecuting phishing criminals holds promise as another countermeasure that can add to the current countermeasures and present different challenges to phishers [19]. Because of the nature of the internet, phishing can involve many different law enforcement jurisdictions, companies, and individuals spread across the world. Often the main barrier to criminal prosecutions is identifying the scale of the crime [19]. Aggregating phishing activity and identifying likely suspects is a key problem in criminal prosecutions due to various parties having different motivations [19]. Also, coordinating many different law enforcement jurisdictions (often international) can be a problem. Recently there has been some hope for jurisdictional cooperation, identification, and prosecution of phishers as seen with recent arrest and prosecution of phishers [7]. Criminal prosecution of phishers faces many hurdles including timely aggregation of phishers committing crimes. Quickly identifying and aggregating phishers is one cornerstone to enabling effective prosecution of criminals.

2.3.8 Increasing Exploitation Difficulty

Increasing exploitation difficulty involves any technique that makes it harder to convert phished credentials into system access or monetization them. These techniques are mainly within the control of the institution being phished and involve increasing the security around intrusion defense and detection.

Phishers can quickly monetize phished banking credentials on underground black markets. This monetization does not involve interaction with the financial institution. Also, there is a good chance the credential's buyer or another party will attempt to monetize the credentials via exfiltrating money from the account or making fraudulent purchases at some point in time. Locating these black markets and shutting them down because of a violation of terms of service or law enforcement action would help in decreasing incentives for phishers. Phishing attacks against financial institutions often involve a direct path to monetization via fraudulent charges or

exfiltrating money from the account. Phishing attacks against email providers can also result in banking fraud. Financial institutions often require an email account be verified to reset online banking credentials. Once a criminal has phished an email account, the criminal can search the email history for any references to financial institutions and use the compromised account to reset the online banking credentials. Phishing attacks against social media are not generally able to be pivoted to banking fraud, except for instances like Google where email, social media, and Google Wallet are all tied together via one username and password. Increasing exploitation difficulty can be used to combat phishing financial institutions, email accounts, and system logon credentials. There are a variety of techniques to increase exploitation difficulty for phished credentials via increasing authentication effectiveness such as multifactor authentication, out of band one time passwords, and others [20]. The goal is to make accessing the resource more difficult and therefore more difficult to exploit. Some of these same techniques can be used to help protect against exploiting credit card data in card present and card not present fraud. For example having to input information on gas pumps such as the billing addresses zip code to help prevent card present fraud or Verified by Visa that prompts the card user for a password to help prevent card not present fraud. Data Analytics encompasses a number of different technologies to classify transactions as valid or fraudulent based on information gathered from the transaction such as frequency between transactions, the sender account, the receiver account, physical location where the transaction is initiated, and other data [20]. Some of the different technologies utilized in data analytics include statistical models, supervised machine learning, and non-supervised machine learning [20–22]. Increasing exploitation difficulty is a mitigation countermeasure as it attempts to prevent exploitation after credentials have been phished. For the Increasing Exploitation Difficulty countermeasure to be effective it needs information about exploitation techniques used in current phishing attacks.

2.4 Grouping Phish

Categorizing or Grouping phish provides a method to detect phish, enables efficient use of countermeasures, and enables measurement of countermeasures against populations of phish. Binary categorization enables detection and non-binary categorization enables efficient use of countermeasures and measurement of countermeasures.

Identifying a website as a phish or not a phish is a binary categorization. Products and services exist that provide lists of verified phishing attacks [15]. These products and services categorize a website as a phish or not a phish. These products and services are effective at phishing detection.

Non-binary categorization can enable more than detecting phish. Phish can be grouped into non-binary categories using shared or common characteristics. It allows phish to be grouped by possible bad actors behind a phishing campaign, tactics used to advertise the campaign, tactics used to avoid detection, motivations behind the campaign, etc.

Understanding the characteristics of a particular phishing campaign allows the selection of the best countermeasure or combination of countermeasures based upon cost and effectiveness. Once a countermeasure's effectiveness and cost have been benchmarked against phish with a particular characteristic the appropriate countermeasure or combination of countermeasures can be chosen given the makeup of current phish and the willingness to accept the cost associated with the chosen countermeasures. Grouping phish by common characteristics enables informed use of countermeasures.

Grouping phish by common characteristics can be used to measure the effectiveness of phishing countermeasures against a particular group of phish. All countermeasures are not equally as effective at preventing or mitigating all types of phishing attacks. For example, email spam filters can help mitigate spam based advertisements for phishing, but will not help mitigate social media based advertisements for phish. Also, enacting phishing countermeasures is not free of cost. The cost can be financial such as labor, software, and or hardware to implement the countermeasure. The cost can be non-financial such as increased user frustration when a countermeasure increases a user's wait time or workload to complete an action. Grouping phish by common characteristics allows measuring both the effectiveness and cost effectiveness of countermeasures against a particular style of phish.

Criminal prosecution needs phish grouping to function. Grouping phish by suspected criminal is required at several points during criminal prosecution. Law enforcement can only pursue a limited number of investigations and cannot start an investigation for each individual phish. Instead law enforcement need phish aggregated by the criminal suspected to be behind the phish. Aggregating phish by the suspected criminal enables effective law enforcement investigations.

After an effective investigation is underway or completed, prosecutors need to be engaged to move the process forward. Prosecution cannot move the process forward until jurisdictional damage thresholds are met. These thresholds differ by jurisdiction. Enough monetary damage must be incurred in a jurisdiction by a criminal or criminal group before prosecution will occur. Aggregating phishing attacks by the criminal or criminal group and being able to assign the monetary damage by jurisdiction is required. Phishing intelligence is required to enable effective investigation and prosecution.

Malcovery Security currently uses an unsupervised machine learning algorithm to group phish by several different shared characteristics. The result is categorizing websites as phish or not a phish and categorizing phish by the brand it imitates. Non-binary phish classification is needed to properly deploy phishing countermeasures, enable law enforcement actions, and enable criminal prosecution.

2.5 Dissertation Goals

The currently observed phishing attack volumes make grouping phish via manual means uneconomical. Automated methods to group phish need to be used to keep pace with the currently observed volume of phish.

Automated methods that produce phish groupings faster enable quicker responses to phishing attacks. Automated methods that group phish by common characteristics can benefit from improved runtime.Automated methods can also run into hardware memory limitations when trying to group large numbers of phish limiting the number of phish that can be grouped. Automated methods that group phish by common characteristics can benefit from reduced maximum memory requirements. Automated methods to group phish can benefit from improved runtime and reductions in maximum memory consumption.

The goal of this dissertation is to improve the current state of grouping phish by improving an existing method's runtime and reducing the maximum memory requirements, while producing similar clusterings without having to have specialty hardware or alter existing algorithms.

CHAPTER 3

Related Work

The related work chapter is divided into three main sections. The first section covers existing phishing detection and phishing categorization processes. The second section covers data mining algorithms that have been applied to other research areas. The third section discusses areas where data mining algorithms can be improved for use as phish grouping and categorization processes.

3.1 Phishing Detection and Phishing Categorization

Various automated methods or algorithms exist to group phish into binary categories to enable detection and non-binary categories to enable phishing categorization. The existing algorithms are classified based upon the type of phishing related data the algorithm utilizes. The two broad areas of phishing related data used by existing algorithms are URL and phishing website content data. The most recently published algorithms are reviewed.

3.1.1 URL Based

Gyawali et al.. [23] and Ma et al. [24] propose solutions to phishing identification by using features that can be derived from a URL. These researchers demonstrated that URL-based methodologies can identify phishing URLs with high accuracy; however, such techniques can be avoided causing lower detection rates by shortening the phishing URLs or hosting the website in the root directory. Phishing actors used to create domains on the same IP blocks, which Weaver and Collins leveraged in a clustering algorithm using the IP address or hosting network to cluster phish [16]. The researchers in Wardman et al. [25] suggest that domains compromised by the same attack may indicate the same phisher. URL based methods are limited to only detecting or grouping phish using a single type of data and cannot be easily adapted to use other types of phishing related data.

3.1.2 Content Based

Content-based approaches use the content of the website for detection. Dunlop et al. [26] presents a method for determining the visual similarity between screenshots of phishing websites. Other researchers have used components within the source code [27, 28]. Britt et al. [29] and Wardman et al. [30] describe a structural analysis technique (DeepMD5) looking at local domain files to create groups of related phish. Zawoad et al. clusters phish using the similarities in the drop email addresses from kits found with the phish [31]. Content based methods are limited to only detecting or grouping phish using a single type of data and cannot be easily adapted to use other types of phishing related data.

3.2 Data Mining

The data mining algorithms found to be useful for phish grouping fall into one of two categories. The first category is supervised machine learning and the second category is unsupervised machine learning. Supervised machine learning algorithms are briefly reviewed. Unsupervised machine learning algorithms are broken down into several different families of algorithms and each family of algorithms is reviewed.

3.3 Supervised Machine Learning

Supervised machine learning is a broad categorization for machine learning algorithms that require training data to develop models to predict labels for new data. Anecdotally I have heard many researchers say that the worst part and biggest barrier to using supervised machine learning is manually assigning labels for training data sets. Sampling small amounts of data for training can decrease the amount of manual effort required to tag a training data set. However, this introduces the possibility that the sampled data is not representative of the data set being sampled. Unrepresentative training data samples can lead to the model working correctly for the training data, but under performing on the testing data. This effect is known as over fitting. Since the supervised learning develops models based on current data and identifying trends in the current data it can be ineffective when new types of trends are introduced. To avoid this problem retraining is required. Retraining involves manually labeling a new training data set and rerunning the supervised machine learning algorithm. Supervised machine algorithms are also notorious for

having long training run times. Having to retrain models involves more manual effort to tag new training data sets and more run time as a new model is developed. The drawbacks of supervised learning algorithms are the manual effort to tag training data, the susceptibility to over fitting based on training data sampling, and run time issues from frequent retraining.

3.4 Unsupervised Machine Learning

Unsupervised machine learning is a broad categorization for machine learning algorithms that do not require training data to predict labels for data. There are several subcategories of unsupervised machine learning algorithms that are well suited for particular tasks. One example is frequent set generation algorithms that are used to develop frequently seen sets of items in sets of transactions such as market basket analysis. A particular subcategory of unsupervised machine learning algorithms known as clustering algorithms can be used to classify data sets into non-binary categories. Clustering algorithms classify data sets by grouping or clustering data into groups or clusters based upon a distance metric applied to the data points and the particulars of the clustering algorithm. The result is a sorting of a data set's points into groups or clusters. The resulting clusters can be used for categorization when the data points within a cluster share a common label. The following subsections cover many of the different families of clustering algorithms where each family approaches the clustering problem in a different way.

3.4.1 Partition Clustering

There are many different partition clustering algorithms. They all share a common characteristic of clustering data points by repetitively sorting data points into clusters, applying an evaluation metric, and updating the clustering until the evaluation metric meets a certain criterion. One partition clustering algorithm that has relatively good scalability is CLARA (Clustering LARge Applications). CLARA is makes use of data sampling. CLARA tries to choose suitable representative data points out of all data points in the data set. The representative data points are then fed into a standard partitioning algorithm called PAM [32]. Reducing the number of data points being considered allows better scalability with respect to the number of data points. Using representative data points could be a useful idea in reducing the runtime.

3.4.2 Hierachical Clustering

Hierarchical clustering algorithms such as BIRCH [33], CURE [34], and Chameleon [35] use a multi-phase approach to clustering [32]. In the first phase of clustering the algorithms cluster relatively similar objects into small clusters or micro clusters and generate representatives or summary information for each micro cluster. In the second phase one or more levels of clustering are performed on the micro clusters using the representatives or summary information about each micro cluster. The concept of micro-clusters varies between BIRCH, CURE, and Chameleon. BIRCH's micro-clusters summary information consists of the number of points, the sum of the points, and the square sum of points. CURE's micro-clusters summary information consists of representative members of the micro-cluster shrunk towards the cluster centroid. Chameleon does not capture summary information using its micro-clusters. It generates relative interconnectivity between micro-clusters and relative closeness between micro-clusters. However, BIRCH, and CURE micro-clusters can only represent numerical data. The Chameleon micro-cluster concept can represent categorical data, but can have a quadratic runtime in bad cases.

3.4.3 Relational Data Classification and Clustering

Relational data classification and clustering attempts to classify or cluster richly structured data that is in a relational database. In particular, relational data clustering's goal is "to use our knowledge about one object to reach conclusions about other, related objects" [36]. Three such techniques that try to exploit relational data are described in Neville et al. [37], Tasker et al., and Xavier-Junior et al. [38]. Neville et al. performs classification on relational data by using iteratively updated Bayesian classifiers based upon the database's relational structure. In Tasker et al., classification or clustering is performed by applying Bayesian learning to a relational schema or skeleton that describes a database's relations and attributes. A skeleton is developed to describe the relational data. A Bayesian learning algorithm is then given the skeleton and a select amount of relational data. A probability distribution (PRM) is then developed based upon the attribute values over the skeleton using an inductive Bayesian learning algorithm to assign class labels. The PRM is then used to assign class labels to the data not used to develop the PRM. Xavier-Junior et al. describes a method to convert a relational database schema into a hierarchical structure to allow for a distance measure to be computed between individual instances over the largest number of shared attributes possible. The hierarchical structure and resulting distance metric can

be used by standard classification and clustering algorithms, such as k-means and agglomerative hierarchical clustering, as demonstrated by the authors. Relational classification and clustering techniques try to leverage the format of relational databases to improve upon non-relationally aware classification or clustering techniques. Relational classification and clustering techniques must be aware of how each of the relations (tables) relate to each other. Relational data clustering ideas may be useful when trying to develop relations between data from different user defined data groups, different subsets of the same data sources, and different data sources that have fields in common.

3.4.4 Dynamic Clustering

Dynamic clustering algorithms are clustering algorithms dealing with changing or dynamic data. In dynamic data insertions and deletions happen rapidly causing issues with clustering. Stream clustering can be viewed as a sub-genre of dynamic clustering because it "...can be regarded to a certain degree as a degenerate case of an incremental database where the database size is extremely small (the size of a window in a stream), and insertions and deletions arise such that the current "database" content is completely replaced." [39]. There are two general directions followed when trying to solve the dynamic clustering problem [39]. The first direction involves the creation of a new clustering algorithm that can handle dynamic changes. The second direction involves the creation of a data compression or summarization algorithm followed by the application of a standard clustering algorithm that has been minimally modified. Nassar et al. describes the concept of data bubbles to be used to minimize the amount of re-computation to regenerate a hierarchical clustering in a dynamic environment [39]. The technique sorts data into a series of data bubbles and subdivides the data bubbles into regions. These regions are then monitored for activity when additions and deletions occur. When an addition or deletion occurs in a sub-region the affected data bubble is evaluated as to whether it is good or bad. If found to be bad the data bubble goes into a merge and split pool with other bad data bubbles. The merge and split pool of data bubbles is then merged and split to the data bubbles across the data so as to create good data bubbles again. Nassar et al. presents mathematical analysis showing how effective the data bubbles summarization is when applying a merge and split strategy to accommodate non-static data. The data bubble concept could be leveraged when trying to parallelize clustering processes to improve runtime and maximum memory consumption.

3.4.5 High Dimensionality Clustering

There are various high dimensionality data clustering algorithms and flavors of these algorithms can be found to perform clustering on data streams. High dimensionality clustering algorithms are varied in their approaches. All are comparing data objects across many different dimensions. High dimensionality clustering algorithms are optimized to deal with a large number of dimensions, but otherwise are not different than other clustering algorithms. High dimensionality clustering algorithms are applicable for use when clustering high dimensionality data.

3.4.6 Bi-Clustering or Co-Clustering

Usually clustering derives a global model, but biclustering produces local models [40, 41]. Applying normal clustering algorithms to microarray data can be used to globally tie conditions together (columns) or globally tie genes together (rows). Applying biclustering algorithms to microarray data can be used to tie a local subset of conditions (columns) to a local subset of genes (rows). The individual genes (rows) and conditions (columns) can be present in multiple clusters. So bi-clustering/co-clustering is creating local clusters of high dimensionality data. Biclustering/co-clustering algorithms are useful when wanting to generate local clusters of high dimensionality data, but not global clusterings.

3.4.7 Subspace Clustering

Subspace clustering generates clusterings on a subset of high dimensionality data. In [42] a subspace clustering algorithm named SONAR is described. The algorithm is inspired by the way active sonar pings are sent out and interpreted. SONAR "pings" the high dimensionality data set using several runs of the expectation maximization (EM) algorithm to find multiple local minima. The "pings" are then statistically compared to the various data objects and the results are collected in a matrix. The response matrix generated from the pings is used in independent component analysis (ICA) to identify the separation points between the data in various subspaces. These split points are combined to generate a clustering of the data across various subspaces. Subspace clustering algorithms are useful when dealing with high dimensionality data sets and creating clusters across different dimensional subspaces.

3.4.8 Multi-View Clustering

Multi-view clustering attempts to identify multiple clusterings of data unlike traditional clustering that only attempts to find a single clustering [43]. In [43], a complex statistics modeling algorithm, MVGen, is presented to identify clusters in different and sometimes overlapping subspaces. MVGen utilizes a Bayesian framework modeling process to identify the various clusters and various subspaces in which these clusters lie. Multi-view clustering is similar to subspace clustering as both attempt to develop find important subspaces within high dimensionality data. However, multi-view clustering can identify multiple important global subspaces and [43] can identify multiple important local subspaces. Multi-view clustering is useful when trying to develop multiple views of the same data.

3.4.9 Comparing Individual Clusters

Cluster generation and evolution analysis algorithms compare two clusters [44]. Cluster generation algorithms perform cluster comparisons in order to construct a clustering of data, while evolution analysis algorithms compare clusters across a time axis to identify changes. In clustering generation algorithms, a cluster comparison technique is used that implements a distance metric to measure the distance between two clusters in order to generate a clustering of data. There are distance metrics for numerical data like Euclidean distance and distance metrics for categorical data like the Jaccard coefficient. There are various comparison techniques that use distance metrics to determine a distance between two clusters. A few examples are the complete linkage technique that uses the largest distance between elements in the two clusters, the average technique that uses the average distance between all the points in the two clusters, and the centroid technique that measures the distance between cluster centroids. Evolution analysis algorithms compare two clusters to determine the amount of change over time. A particularly interesting evolution analysis algorithm example is presented by Charu Aggarwal [45]. Aggarwal outlines an evolution analysis algorithm that identifies data density changes as well as the magnitude of the changes. Two clusters' data points are plotted and density is computed at several corresponding locations. The density distributions are then compared to identify changes in density, magnitude, and direction. Cluster comparison techniques are useful when trying to compare two clusters.

3.4.10 Comparing Sets of Clusters

There is existing work used to compare one clustering (a set of clusters generated by a particular clustering algorithm) to another clustering based upon data commonality and/or cluster structure commonality [44, 46]. There are two flavors of clustering comparisons algorithms. The first flavor compares clusterings generated from the same set of data points with the same attributes (homogenous data from the same window) and the second compares clusterings generated from different sets of data points that have the same attributes (homogenous data from different windows). The first flavor of comparing clusterings is used to evaluate the quality of one clustering algorithm over another and can be used in clustering algorithms that attempt to converge on a best clustering solution. Examples include the RAND index, the Jaccard coefficient, and other mentioned by in Meila et al. [46]. The second flavor is useful in distributed data mining when trying to group together similar clusterings and when trying to determine pattern wide changes over time. Examples include the FOCUS and PANDA frameworks [44]. Clustering comparison techniques can be used to measure similarity and differences between clusterings. Cluster set comparison techniques based upon data commonality or cluster structure commonality are useful when trying to compare sets of clusters or clusterings.

3.4.11 Mining Heterogeneous Data

There are algorithms in recent literature that perform data mining tasks on heterogeneous data. Two such algorithms are one-sided convolutional nonnegative matrix factorization (OSC-NMF) presented in Wang et al. and contingency table based clustering in Hossian et al. [47, 48]. The OSC-NMF algorithm performs temporal pattern mining on medical records for individual patients, groups of patients, and multiple groups of patients. The patient data consists of a wide variety of time stamped heterogeneous data types including vital signs, immunizations, laboratory data, radiology reports, etc. The OSC-NMF algorithm can generate temporal sequence patterns for individual patients using only an individual patient's history, group temporal sequence patterns using a group of similar patient histories, and cross group temporal sequence patterns using several groups of patient histories. OSC-NMF generates temporal sequence patterns by viewing patient histories as an event matrix where rows are heterogeneous data types and columns are time windows. OSC-NMF assumes the patient matrices can be expressed as a series of reoccurring concatenated event sequences. OSC-NMF determines the reoccurring sequences using matrix

decomposition. The result is a set of temporal event sequences. The OSC-NMF algorithm is limited to mining sequential patterns. Contingency table based clustering presented in Hossian et al. clusters two sets of data over different data types, heterogeneous data, using relationships between the two sets of data and various other attributes. The clusterings are generated by sorting points into clusters to optimize, maximize or minimize, connection strengths between heterogeneous data. When maximizing connection strengths the goal is to have points in a cluster that only have relationships to data points in a single heterogeneous data cluster, which they call dependent clustering. When minimizing connection strengths the goal is to have points in a cluster that have relationships to data points in all heterogeneous data clusters, which they call disparate clustering. The algorithm performs the dependent or disparate clustering by first creating a prototype grouping for each data type using k-means. A contingency table showing the relationships between groups is generated where groups from one data type are represented as rows,V, and groups for the other data type are represented in the columns,W. The relational strength between two groups V and W is in the contingency table entry, VW. Once the contingency table is created an optimization cycle is started to re-arrange group membership and improve the dependent or disparate strengths between groups in V and W. The algorithm also includes several restarts to help to avoid local minima and maxima in the optimization problem. An advantage of dependent or disparate clustering is that it does not have to be limited to heterogeneous data. Homogeneous data can be considered a degenerate case of heterogeneous data as creating relational links between homogeneous data shouldn't be any harder than creating relational links between heterogeneous data. A disadvantage of dependent or disparate clustering is that it is limited to creating clusters for pairs of heterogeneous or homogeneous data. It becomes problematic to utilize dependent or disparate clustering when there are more than two windows. Performing dependent or disparate clustering on three data sets requires the re-clustering of two data sets if relating all three data sets to each other. Dependent or disparate clustering is best used when relating only a pair of data sets. In [49] the MVSim architecture is described. MVSim utilizes a co-similarity algorithm on multiple similarity matrices between different data types to generate relations between the different data objects. The data object relational strength output of the MVSim architecture allows for the application of a generic clustering algorithm to generate clusters between different data types. The MVSim architecture gives a method that enables generic clustering algorithms to cluster heterogeneous data based upon relations between the

heterogeneous data. The MVSim architecture's generation of similarity matrices between data sets allows it to handle more than a pair of data sets, unlike dependent or disparate clustering presented in Hossain et al. The MVSim architecture has an n squared runtime complexity with a coefficient equal to the number of data sets to be related. However, it is also built with parallelization in mind and can be configured to allow each data set comparison to be run on a separate processor or core. Thus the coefficient applied to the n squared runtime is removed. The heterogeneous data mining methods presented in Wang et al., Bisson et al., and Hossian et al. are interesting algorithms that are useful when trying to mine temporal event sequences and creating dependent or disparate clusterings. Ideas used in Wang et al. and Bisson et al will be useful when constructing relations between clusters and clusterings. Adding a preprocessing step would allow MVSim to work with groups of data points instead of single data points and develop relational strengths between data groups or sets of data groups.

3.5 Improvements

Current phishing detection and phish grouping process cannot be easily adapted to incorporate different types of phishing related data. Developing a general method to handle phishing related data can be accomplished using machine learning algorithms. Each machine learning algorithm presented has its own particular advantages and disadvantages. Existing phishing phishing detection and phish grouping can produce high quality results but sacrifice runtime.

There are ideas in various algorithms that can be combined to improve runtime and reduce maximum memory consumption, while maintaining the quality of the results produced. The various families of clustering algorithms contain useful ideas when trying to meet the goals of this dissertation.

Scalability issues mainly concerned with runtime can be overcome by creating small sub problems that can be solved in parallel by different hardware as seen in MVSim [49]. Hossian et al presents a method to cluster homogenous and heterogeneous data using dependent clustering, which can be used to compare data points from different sub problems generated from parallel processing [48].

The ideas found in hierarchical clustering and the idea of data bubbles in Nassar et al provides a conceptual method that is adaptable and can be used to unite clusters from different data sets

to create a global clustering of many smaller data sets [39]. Combining ideas and strategies in the various data mining algorithms leads to the creation of a new tool called the Simple Set Comparison tool that improves runtime and reduces maximum memory consumption, while producing high quality results.

CHAPTER 4

Algorithm

The SSC tool is a method that can be applied to a clustering algorithm to reduce the algorithm's runtime and maximum memory consumption. The SSC tool uses a multi-step approach incorporating ideas from several unsupervised machine learning algorithms. Its multi-step approach is similar to hierarchical clustering. When applied to the SLINK style clustering algorithm used for phish grouping by Malcovery security it reduces the runtime and maximum memory consumption, while producing comparable results.

The SSC tool consists of four broadly defined steps. The four broadly defined steps are creating windows, clustering windows, comparing windows, and merging windows. In the first step, a user specifies chronological division points in the phishing data set. These chronological divisions are used to create single windows and cross windows. A single window consists of all data points within a single chronological division and all relationships between data points within the single chronological division. A single window is created for each chronological division.

A cross time window consists of all data points in two different single time windows and only the relationships between data points from the two different single time windows. A cross window is created for every combination of user specified single time windows.

In the second step, the single and cross windows are clustered independently of one another. All single and cross window clustering processes can be run in parallel. The second step results in a clustering for each window.

The third step compares clusters from single windows to clusters in cross windows that utilize data points from the single window. The cluster comparison is based upon shared cluster members. The cluster comparisons between overlapping windows can be run in parallel. The third step results in a cluster similarity graph, where clusters from the windows are nodes and edges are based upon the number of shared members between single window clusters and cross window clusters.

The fourth step runs a clustering algorithm over the cluster similarity graph to merge similar clusters. The result is a clustering of the entire data set.

The SSC tool takes advantage of parallel processing in the second and third steps. To be able to process a large data set in parallel the data set must first be subdivided or partitioned. The phish and kit data is tagged with a received time which allows partitioning on a chronological basis. The parallel processing in steps two and three are the key to reducing the runtime and reducing the maximum memory consumption for the clustering algorithm. Next each of the four steps are covered in more detail.

4.1 Creating Windows

The data set is subdivided into windows. As well as single windows, windows incorporating two single windows are created, which are called cross windows. Cross windows are created for every combination of single windows. For example a single data set could be divided into four windows, A, B, C, and D. The four single windows would be A, B, C, and D. There would be six cross windows A:B, A:C, A:D, B:C, B:D, and C:D.

4.2 Clustering Windows

The phish and kit single windows and cross windows are clustered by comparing phishing websites using the Deep MD5 method as a similarity score and a SLINK clustering algorithm to sort the phish into groups based upon their similarity scores [50, 51]. Deep MD5 generates a score based upon file set similarity using the MD5 message digest algorithm to create a hash for each file in the file set [52]. Deep MD5 generates a score using the count of candidate one's files (count1), the count of candidate two's files (count2), and the number of matching MD5 values between candidate one and candidate two (overlap).

$$Kulczynski2Coefficient = 0.5(\frac{overlap}{count1}) + 0.5(\frac{overlap}{count2}) \tag{4.1}$$

A Kulczynski 2 coefficient, 4.1, is then applied to count1, count2, and overlap to generate the Deep MD5 score with a value between 0.0 and 1.0. For example two websites, website X and website Y, could be compared using Deep MD5. If website X's html code makes references to

files a,b,c,d,e and website Y's html code makes references to files a,b,f,g then the overlap count between the two websites' file sets is two (overlap). Website X's file count is five (count1) and website Y's file count is four (count2). Then the Deep MD5 score is 0.5(2/5) + 0.5(2/4) or 0.45.

The cross windows that incorporate phish and kit data are compared using a a Simpson coefficient instead of a Kulczynski 2 coefficient. The Simpson coefficient is used to connect phish to kits because when fetching the phishing website only the web facing content can be collected. Downloaded phishing kits contain not only the web facing content, but also all of the server side content. A phishing website created from a particular kit will have at most the same number of files as the kit, but often it will have fewer. Introducing the kit files would introduce server side files that cannot be contained in the phish and dilute the similarity score. The similarity score should not incorporate the total number of kit files. The similarity score of a phish to a kit should only be based upon how many of the files found in the phish are contained in the kit. The Simpson coefficient uses the smaller of the two file set counts as its denominator instead of both the phish and kit file set counts like the Kulczynski 2 coefficient. The smaller file set count should be the file set count for the phish.

$$SimpsonCoefficient = \frac{overlap}{smallestset} \tag{4.2}$$

The Deep MD5 score is modified by changing out the Kulczynski 2 coefficient for a Simpson coefficient in equation 4.2. The number of matching files between the two sets is still used as the numerator. The difference between the Kulczynski 2 and the Simpson coefficients is the denominator. The Simpson coefficient uses the smaller of the two set sizes (smallest set) as its denominator. After the similarity scores are generated the results are feed to a SLINK clustering algorithm. The SLINK clustering algorithm is a graph theoretic clustering algorithm. The graph has vertices of phishing websites and for each pair of vertices there exists a deep MD5 similarity score. Edges where the similarity score meets or exceeds a threshold are kept and edges not meeting the minimum threshold are discarded. An analysis of Deep MD5 scores between phish showed good matching results between the threshold values 0.5 and 0.75 with very little change [51]. A 0.6 value is chosen as a middle ground between the high and low end threshold values. After all edges not meeting the minimum threshold have been pruned, the SLINK clustering algorithm turns connected components into clusters.

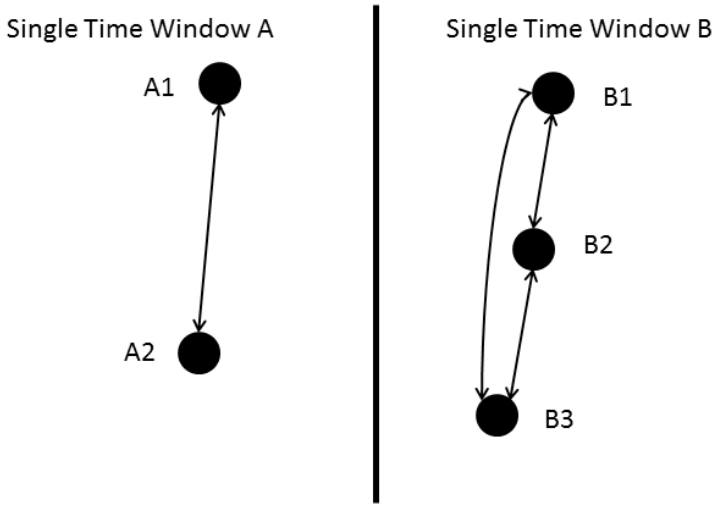

FIGURE 4.1: Similarity Score Generation for Single Time Windows A and B

Each single window is clustered by generating similarity scores for all phish from a single window and then applying a SLINK clustering algorithm. Figure 4.1 shows similarity scores being generated for two single windows before the clustering algorithm is applied.

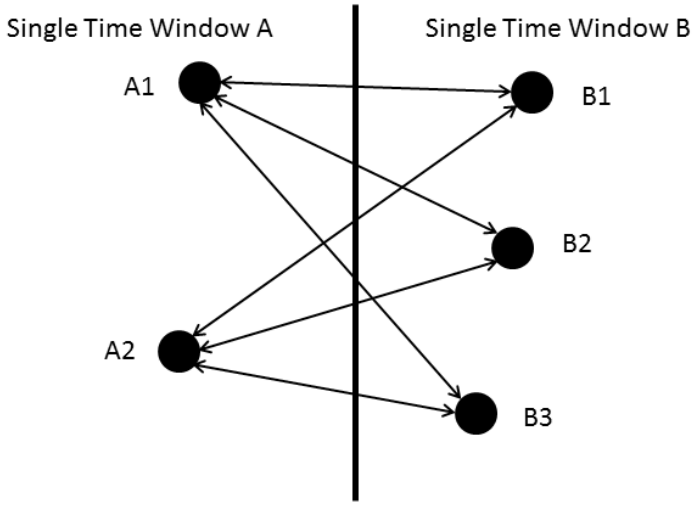

FIGURE 4.2: Similarity Score Generation for Cross Time Windows A and B

The six cross windows are clustered by generating a similarity score for all phish from one window compared to another window and applying a SLINK clustering algorithm. Phish from the same window are not compared, only phish from different windows are compared. Figure 4.2 shows similarity scores being generated for a single cross window before the clustering algorithm is applied. Clustering the four single time windows and six cross windows can be performed

independently of one another, allowing all clustering processes to be run in parallel instead of in sequence.

4.3 Comparing Windows

The cross window clusters are used to merge clusters from single windows together. Clusters from single windows are compared to clusters from overlapping cross windows. The clusters are compared by counting the number of members shared between the two clusters (overlap) divided by the total number of members in the single window cluster (count1) resulting in a score between 0.0 and 1.0.

$$ClusterMembershipSimilarity = \frac{overlap}{count1} \tag{4.3}$$

The cross window's size is not included as it will dilute the similarity score. Because the cross window clusters can incorporate members from two windows they are generally much larger. Each single window to cross window comparison can be run independently of one another. Comparing windows can be performed in parallel. Comparing windows based upon shared cluster members results in a cluster similarity graph for all clusters from all windows.

4.4 Merging Clusters

A clustering algorithm is then run over the cluster similarity graph to merge similar clusters. The same SLINK clustering algorithm used in step two is used here in step four. The clustering algorithm used in this step of the SSC tool is interchangeable. The only requirement is the clustering algorithm takes an edge based representation of a graph and produce non-overlapping clusters.

4.5 Computational Complexity

The SSC tool leverages parallelization in steps two and three to reduce runtime and maximum memory consumption, but step four is not run in parallel. Step two creates additional copies of the clusters in the single time windows via cross time windows clusters. Since step four is not parallelized and the number of clusters that it needs to cluster has been increased the

computational complexity may increase because of the use of the SSC tool. The fear is this may lead to a squaring of the computational complexity. Other factors need to be incorporated to limit the computational complexity growth. The computational complexity of the clustering algorithm without the SSC tool is used as a baseline and compared to the computational complexity of the SSC tool applied to the same clustering algorithm. The clustering algorithm's runtime and the clustering algorithm runtime using the SSC tool are both assumed to be dependent upon the number of data points to be clustered. To determine the number of data points it is assumed that all windows will have approximately the same number of data points. In any given window, W, it is assumed the number of data points, k, will be approximately the same. The number of data points in a data set is kW. The computational complexity for the clustering algorithm without using the SSC tool is the computational complexity to generate a clustering for the entire data set. C(x) is defined to be the computational complexity of the clustering algorithm, on x data points. The number of data points, x, will then be kW. However, as k is a constant it can be removed. Therefore the computational complexity for the clustering algorithm is:

$$C(W) \tag{4.4}$$

The computational complexity for the clustering algorithm using the SSC tool will be the computational complexity of the highest growth step out of the four steps. The first step, creating windows, is assumed to be determined before the SSC tool is run and would not be accounted for in the computational complexity. The second step is clustering windows. Each of the windows is clustered independently, so each window can be run in parallel. Since the windows are clustered in parallel, the computational complexity will be the window that has the largest number of data points. Single windows consist of only a single window's data points whereas cross windows consist of the data points for two single windows. Therefore, the largest number of data points occurs in cross time windows. A cross time window is assumed to have the longest runtime. The number of data points to cluster is 2k. Since 2k is a constant the second step, clustering windows, is constant and will not grow with an increase in the number of windows.

The third step is comparing windows. Each of the cross windows is compared to the two single windows that it overlaps. Each window comparison is performed independently and assumed to be performed in parallel. Since the window comparisons are performed in parallel

the window comparison with the most points will have the highest computational time. Each single window has k data points and each cross window will have 2k points. Since k and 2k are constants the computational complexity for the third step, comparing windows, is constant as long as there are enough processes available to scale with the increasing number of windows.

The fourth step is merging windows. The similarity graph generated from comparing windows is used to cluster the clusters generated in step two, clustering windows. There are no independent pieces when merging windows. Merging windows cannot be performed in parallel. To compute the input size the number of data points in the windows to be merged needs to be determined and the number of windows to be merged. Each window has k data points and each cross window has 2k data points. Since merging windows involves merging clusters from different windows and not data points a constant reduction percentage, r, is used to represent the reduction in the number of data points as they have been merged into clusters in the clustering windows step. Therefore each single window has rk data points and each cross window has 2rk data points. Next the number of single windows and cross windows needs to be determined. There are W single windows and W(W-1)/2 cross windows. The number of cross windows, once constants are removed, is W^2. The number of data points in single windows is Wrk and the number of cross window data points is W^2rk. Therefore the number of clusters is Wrk + W^2rk. Since rk and 2rk are constants they can be removed. Also, W^2 is the dominate growth rate compared to W. So the linear growth rate of W can be removed. The result is the input growth being governed by W^2, quadratic growth rate.

To remedy the quadratic growth rate the number of cross windows needs to be limited. Limiting the number of cross windows involves limiting the number of single windows to which any given single window can be compared. Limiting each single window to being compared to a constant number of other single windows, b, will solve the issue as there will only be bW cross windows instead of W^2. As b is now a constant it can be removed, resulting in W. Assuming the computational complexity of merging windows is C(x) where x is the number of data points the computational complexity is:

$$C(W) \tag{4.5}$$

The first step, creating windows, is assumed to not affect the computational complexity as it is performed before the process is run. The second step, clustering windows, has a constant

computational complexity. The third step, comparing windows, also has a constant computational complexity. The fourth step, merging windows, has a linear computational complexity once a limit, b, is used as an upper bound to the number of single windows to which any given single window can be compared. The largest computational complexity growth rate out of the four steps is the linear growth rate of the fourth step, merging windows. The computational complexity of the SSC tool is linear, which is the same as the traditional clustering algorithm's growth rate. However, limiting the number of cross windows limits the number of single windows to which any other single window can be compared. Limiting the single window comparisons to b will only limit the results when the number of windows is larger than b. Therefore when the number of windows is less than b every single window will be compared to every other single window. However, when the number of windows is greater than b every single window will not be compared to every other window and may result in clusters from different windows that should be merged failing to merge. We have not investigated and determined an ideal value for b.

CHAPTER 5

Experiments

Four experiments with real world phishing data are carried out to compare how a SLINK style clustering algorithm using SSC tool compares to the SLINK style clustering algorithm without the SSC tool.

The four experiments will evaluate the SLINK style algorithm using the SSC tool compared to the SLINK style clustering algorithm without the SSC tool. The evaluation is based upon the ability to process all of the data without exceeding maximum available memory, the runtime it takes to produce a clustering, and the quality of the clustering produced.

The first and second experiments use a memory hungry implementation of the SLINK style algorithm that loads all data points and all edges between data points into main memory. The third and fourth experiments use a memory lean implementation of the SLINK style algorithm called Union-Find that loads all of the data points and iterates through all edges only loading a single edge into main memory at any given time. The experiments involve real world data collected and labeled by a security company, Malcovery Security.

5.1 Data Set Overview

The data sets for the four experiments have been gathered by a security company, Malcovery Security and provided to the author. The data set for the first experiment consists of phish gathered over a one month period. The data set for the second experiment consists of phish and phish kits gathered over a six month period. The data set for the third experiment consists of phish data gathered over a one month period. The data set for the fourth experiment consists of phish data gathered over a six month period.

5.1.1 Phish Data

The Malcovery Security phish data mine gathers phishing URLs from a large spam-based URL provider, a large anti-phishing company, and a number of other feeds including private companies,

security companies, and financial institutions. The URLs are either URLs contained in spam advertising phish or URLs reported by the public to fraud alert email addresses. The data set favors financial institutions and under represents gaming and social media phish when compared to other phishing collections. A number of methods are used in the industry to count phish. Some methods count distinct URLs. If there is any randomization in the host name, directory path, or arguments it leads to 'over-counting'. Cases where this occurs include wild-card DNS entries, per user customized URLs, or virtual hosts allowing the same directory path for multiple domains to resolve to a single IP address. A conservative counting approach that attempts to de-duplicate URLs leading to the same phishing content is used by Malcovery Security.

The phishing data consists of all files referenced in the phishing website. The website files were fetched using an automated web crawler that makes use of a firefox mechanization tool [53]. After the files were downloaded, a hash value is generated for each file using the MD5 hashing algorithm. Screenshots and the domain information were manually reviewed to determine whether the potential phish was a phish. The phish data used only consists of confirmed phish.

5.1.2 Kit Data

The Malcovery Security kit data mine collects suspected kits by searching confirmed phishing URLs for zip files. Each confirmed phishing URL is scanned starting from the domain level up to the last directory in the URL. At each level the most common kit names and most common kit names for the confirmed phish brand are appended onto the current directory level and attempted to be fetched. If the fetching attempt is successful the zip file is downloaded and stored locally. Once the suspected kit zip file is stored locally an attempt is made to unzip the file. If successfully unzipped all unzipped files are hashed using the MD5 hashing algorithm. The unzipped files from the suspected kits are manually reviewed to determine if the suspected kit is a kit. The presence of an html page imitating a known brand and/or script pages used to create emails with phished credentials are indicators that a suspected kit is a kit. The kit data used only consists of confirmed kits.

5.2 First Experiment

The first experiment compares a memory hungry implementation of the SLINK algorithm using the SSC tool to a traditional clustering algorithm, SLINK.

The first experiment consists of phish data gathered over the course of a one month period. The first experiment also compares several different merging thresholds for the SSC tool's step four, merging windows, based on the quality of the results produced. By comparing the quality of the results produced the best merging threshold can be determined.

5.2.1 Data Set

Brand	Count
Tech Company 1	3815
Telecom Company 1	1720
Tech Company 2	1484
Financial Institution 2	1435
Financial Institution 3	829
Tech Company 3	786
Tech Company 4	709
Financial Institution 4	657
Tech Company 5	589
Financial Institution 5	529

TABLE 5.1: Ten Most Phished Brands

The one month data set consists of 19,825 confirmed phishing sites collected between September 1st 2014 and September 30th 2014 and does not include phishing kits. There are a total of 245 different brands attacked by phishing websites in the one month data set. The phishing websites are represented by MD5 hash values for the main html phishing website and all files that are referenced by the main html page. The referenced files include image files, script files, style sheets, etc on the same domain as the phishing website and on other domains. Table 5.1 shows an anonymized by sector listing of the 10 most phished brands in the data set.

5.2.2 Results

Before the SLINK algorithm using the SSC tool can be compared to the SLINK algorithm without, a threshold needs to be established for step four, merging clusters, in the SSC tool. The SSC tool merging thresholds are compared using three evaluation metrics: homogeneity, completeness, and V-measure.

After a merging threshold has been selected, the SLINK algorithm using the SSC tool is compared to SLINK algorithm without using the SSC tool on both quality measurements and on

runtime. The same Simple Set Comparison tool merging threshold found to be most effective in the one month test is used in the second, third, and fourth experiments.

5.2.2.1 Merging Thresholds

Before a complete clustering can be produced using the SSC tool a merging threshold used in step four, merging clusters, must be established. Various SSC tool merging thresholds are compared and evaluated on clustering quality metrics as seen in table 5.2. The clustering quality is measured using three different entropy based metrics; homogeneity, completeness, and V-measure [54]. All three measures are based upon evaluating the clustering results compared to a ground truth label assigned to all data points. The ground truth label assigned to the data points represents a perfect clustering of the data set. The three different measures evaluate how close to a perfect clustering is created. The ground truth label used is the phish brand. Homogeneity evaluates how well the clustering is at placing members that should be in the same cluster in the same cluster. A perfect homogeneity score is achieved when all clusters only contain members with the same label. Completeness evaluates how close to perfect the clustering came to determining the correct number of clusters. A perfect completeness score is achieved when there is only one cluster for each label. V-measure is the harmonic mean of the homogeneity and completeness scores, a blend of homogeneity and completeness scores. An exact definition for all three entropy metrics is found in the appendix section. A more detailed explanation for all three metrics can be found in Rosenberg et al [54]. The number of clusters created is also included in table 5.2.

Threshold	Number Clusters	Homogeneity	Completeness	V-Measure
0.001	1,248	0.9871	0.6778	0.8037
0.1	1,281	0.9872	0.6761	0.8025
0.2	1,321	0.9872	0.6739	0.801
0.3	1,324	0.9872	0.6729	0.8003
0.4	1,332	0.9872	0.6716	0.7994
0.5	1,332	0.9872	0.6716	0.7994
0.6	1,335	0.9873	0.6702	0.7984
0.7	1,368	0.9866	0.6683	0.7969
0.8	1,377	0.9851	0.6641	0.7934
0.9	1,386	0.9845	0.6606	0.7907
1	1,416	0.985	0.6352	0.7723

TABLE 5.2: Simple Set Comparison Tool Clustering Quality Measures

The Simple Set Comparison tool is evaluated over eleven different thresholds ranging from 0.001 to 1.0. These ranges are chosen as the smallest and largest thresholds to evaluate because the smallest cluster similarity score found above 0.0 is approximately 0.0094 and the largest cluster similarity score is 1.0. The other thresholds range from 0.1 to 0.9 with 0.1 increments to get the best coverage without an exhaustive search of all threshold values.

The cluster quality measures stay very consistent across the thresholds. As the threshold increases the homogeneity scores only changes in the third and fourth decimal places. Oddly though the homogeneity scores rise slightly until the 0.6 threshold and then fall slightly until the 1.0 threshold. This may be due to an unusual breakdown of good similarity clusters at very high thresholds. The completeness score decreases slightly but consistently from the 0.001 threshold to the 1.0 threshold. The V-measure score that measures the tradeoff between homogeneity and completeness slightly decreases from the 0.001 to the 1.0 threshold. The degradation of completeness without a corresponding improvement in homogeneity begins at 0.3. The relative degradation of the clustering is also reflected in the declining V-measure score from the 0.3 to 0.4 thresholds. The clustering result produced by the 0.3 merging threshold is chosen as a best representative to compare to the traditional clustering algorithm on clustering quality as it is the point where a degrading completeness score does not result in an improvement in homogeneity.

5.2.2.2 Quality Comparison

Now that a particular merging threshold has been set for the SSC tool it can be used in combination with the SLINK algorithm to produce a clustering of the one month phish data set. The clustering quality is measured using the same three entropy based metrics as used previously; homogeneity, completeness, and V-measure [54]. The number of clusters created is also included.

The 0.6 threshold value used in SLINK algorithm is the same value used in step one of the SLINK algorithm using the SSC tool. The 0.6 threshold should serve as a good benchmark. As is noted in a Deep MD5 evaluation paper [51] there is little difference between the 0.5 and 0.75 DeepMD5 threshold values.

Figure 5.1 compares the quality of the clusterings produced. Comparing the clustering quality measures shows almost no differences as the homogeneity, completeness, and V-measure scores

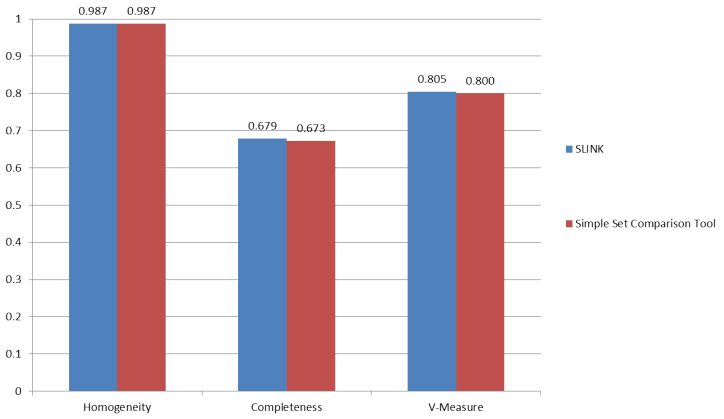

FIGURE 5.1: Comparing quality measures for SLINK with and without using the SSC tool.

are all relatively similar. SLINK without the SSC tool produces a slightly better completeness score. There are negligible differences between the quality of clusterings produced.

5.2.2.3 Anecdotal Comparison

An anecdotal comparison of the ten largest clusters generated by SLINK with and without the SSC tool is presented in table 5.3.

Cluster Brand	**SLINK Cluster Size**	**Simple Set Comparison Tool Cluster Size**
Telecom Company 1	1291	1291
Tech Company 1	915	915
Financial Institution 3	794	794
Financial Institution 2	770	770
Tech Company 3	637	637
Tech Company 2	567	567
Tech Company 1	365	365
Financial Institution 6	303	303
Telecom Company 1	303	303
Financial Institution 4	302	302

TABLE 5.3: Ten Largest Clusters Produced By SLINK With and Without the SSC Tool

SLINK with and without the SSC tool produces the same ten largest clusters. Meaning, both sets of ten clusters are perfectly homogeneous, have the exact same cluster sizes, and have the same brand label. The individual phish that make up both of the sets of ten were not compared to determine if they have exactly the same phish contained in each corresponding cluster.

5.2.2.4 Runtime

The total runtime for SLINK using the SSC tool is computed by adding the runtime for each of the four steps together. There is no runtime spent for the first step as the chronological dividing points for each time window are chosen before the tool is run. The second step is run in parallel, ideally each clustering process is run on a separate machine; the runtime for step two will be the longest runtime out of the group. The third step's runtime is the longest comparison runtime out of all single to cross time window comparisons. Since the third step is run in parallel, ideally each comparison process is run on a separate machine; the runtime for step three will be the longest runtime out of the group. The fourth step is not parallelized. The fourth step's runtime will be the runtime it takes to assemble a global clustering out of the cluster similarity graph generated in step three. The parallel clustering processes run in step two have the largest runtimes out of all of the steps and their runtimes are presented in figure 5.2.

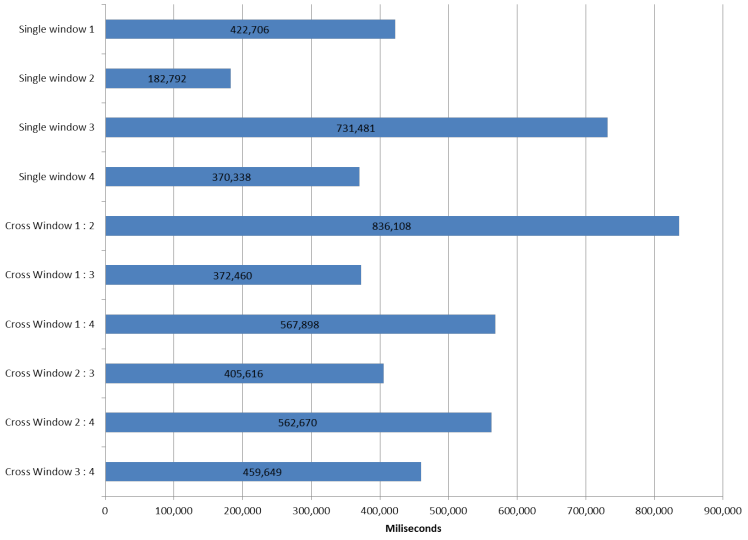

FIGURE 5.2: Clustering Runtimes for Single and Cross Time Windows

Clustering the single time windows takes between three minutes for the fastest and twelve minutes for the slowest. Clustering the cross time windows takes between almost seven minutes and almost fourteen minutes. The longest runtime out of the group is cross window 1:2 at almost 14 minutes, 836,108 milliseconds. Step two, comparing single to cross time window clusters, took very little time. All twelve of the comparisons took only 640 milliseconds combined. The longest comparison took 93 milliseconds and the shortest took 31 milliseconds. Step four, merging time windows, is not parallelized and has a single runtime of 1,324 milliseconds. Adding

the longest runtime for step two (836,108 milliseconds), the longest runtime for step three (93 milliseconds), and step four's runtime (1,324 milliseconds) results in a total runtime of almost fourteen minutes (837,525 milliseconds). The biggest contributor of total runtime comes from step two, clustering windows. The traditional clustering algorithm took over eight and a half hours, 31,044,322 milliseconds, to complete. The SSC tool's runtime is more than 37 times faster than the traditional clustering algorithm's runtime. There is almost no difference between the quality of clusterings produced by the SSC tool and traditional clustering. The runtime is the biggest difference between the two. The SSC tool is more than 37 times faster at producing results for the monthly dataset.

5.3 Second Experiment

After a merging threshold has been established for the SSC tool in the one month data set and the results compared, an experiment on six months of data was performed. The six month data set used in this experiment has been provided by Malcovery Security, the same security company that provided the one month data set. Unlike the one month experiment, the six month experiment includes two different data sources, phish data and phishing kit data. Other differences between the two experiments are described next followed by the results.

There are several differences between the one month experiment and the six month experiment. Some of the differences are based upon lessons learned from the one month experiment. Other differences are necessitated because of the increased volume of data used in the six month experiment.

5.3.0.1 SLINK Implementation

The SLINK implementation used in the first experiment could not be used without the SSC tool. Several attempts were made to use the SLINK implementation. However each attempt failed with a java error message of "java.lang.OutOfMemoryError". Increasing the amount of memory available to java via command line arguments up to 8 gigabytes of memory did not solve the problem. Instead a new implementation of the SLINK algorithm was created using plpgsql to create a postgres database function that implements the SLINK clustering algorithm. Implementing the SLINK algorithm inside of the database as a database function

was chosen because it allows for the creation of an algorithm that limits the amount of required memory and shifts the costs to runtime. However, this concession had to be made as the memory requirements of the original java implementation when using six months of data exceeded the hardware limitations. The postgres function implementation of the SLINK algorithm is only used for the SLINK traditional clustering benchmark on the six month data set. The java SLINK implementation is still used for the SLINK with the SSC tool.

5.3.0.2 Comparing Windows

The experiment on one month of phishing data showed some steps in the SSC tool benefited more than others when run in parallel. The SSC tool clustered the monthly data set in 837,525 milliseconds. Most of the runtime comes from the second step, clustering windows, which took 836,108 milliseconds. The third step, comparing windows, only added 93 milliseconds to the runtime. If the third step was not run in parallel, but run in sequence it would have added only 640 milliseconds to the runtime or less than 1% to the overall runtime. When comparing clusters there is some manual overhead with setting up and tracking each of the parallel processes. Since comparing clusters in parallel versus in sequence did not significantly add to the runtime in the first experiment in the second experiment comparing clusters is performed in sequence.

5.3.1 Data Set

Unlike the first one month experiment the six month experiment consists of both phish and kits. The phish and kit data for the six month experiment are also provided by the security company Malcovery. The following is a description of both the phish and phish kit or kit data.

5.3.1.1 Phish Data

The phish data was gathered from July 1st 2014 to January 11th 2015 and consists of 130,690 phishing websites covering 427 different brands. The phishing websites are represented by MD5 hash values for the main html phishing website and all files that are referenced by the main html page. The referenced files include image files, script files, style sheets, etc on the same domain as the phishing website and on other domains.

Brand	Count
PayPal	43,632
eBay	8,543
Bank of America	7,281
Yahoo	6,864
SFR.com	4,639
Alipay	3,527
Generic Email	3,476
Bradesco Internet Banking	2,988
Taobao	2,835
Wells Fargo	2,679

TABLE 5.4: Ten Most Phished Brands

The data set has been provided and manually tagged by Malcovery Security. The ten most phished brands are listed in table 5.4. The label "Generic Email" is used to describe phish targeting many different email providers using a single phish.

5.3.1.2 Kit Data

Brand	Count
PayPal	274
Generic Email	146
Wells Fargo	59
Bank of America	39
Yahoo	37
Sparkasse	29
SFR.com	18
Apple	17
Chase Bank	17
AOL	15

TABLE 5.5: Ten Brands with the Most Phish Kits

The kit data set was gathered from July 1st 2014 to January 11th 2015 and consists of 906 kits covering 90 different brands. The kit data consists of MD5 values for all files contained in the kit zip file. The kit data was gathered by Malcovery Security using the practices described earlier for gathering kit data. The ten brands with the most phish kits are listed in table 5.5.

5.3.2 Results

Now that the SSC tool merging threshold has been established in the one month experiment the same merging threshold is used for the six month data set. Just like the one month experiment the

six month experiment compares a SLINK algorithm using the SSC tool to a SLINK algorithm not using the SSC tool. Unlike the one month experiment the six month experiment involves phish and kit data. For the SLINK algorithm not using the SSC tool additional runtimes need to be incorporated. The additional runtimes incorporated for the SLINK algorithm include the runtime to compute comparisons between phish like the one month experiment but also the runtime to compute the comparisons between phish and kits, and the runtime to compute comparisons between kits and other kits. The SLINK algorithm with and without the SSC tool is evaluated on clustering quality and on runtime.

5.3.2.1 Cluster Quality

The clustering quality is measured using three different entropy based metrics; homogeneity, completeness, and V-measure [54]. These are the same three entropy based metrics used in the first experiment. An exact definition for all three metrics can be found in the appendix section and a more detailed explanation can be found in Rosenberg et al [54]. The same ground truth brand label is also used.

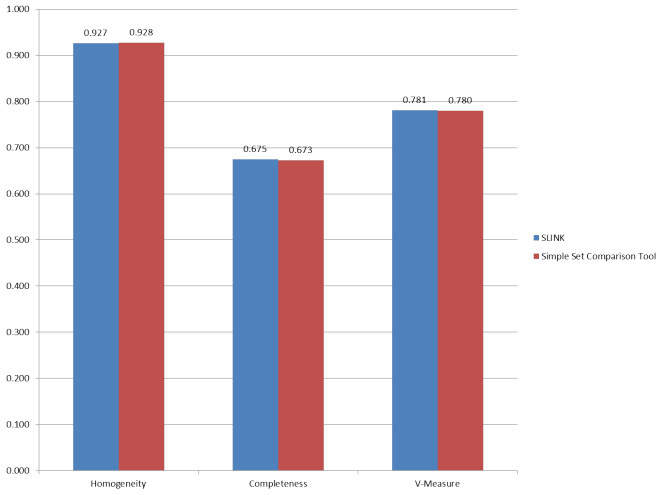

FIGURE 5.3: Traditional Clustering and Simple Set Comparison Quality Measures

Figure 5.3 compares the clusterings produced using the three quality metrics. There is very little difference in the quality metrics between the clusterings produced by SLINK with and without the SSC tool.

5.3.2.2 Anecdotal Comparison

The ten largest clusters created by SLINK with and without the SSC tool are compared for anecdotal evidence to support the claim that the clusterings produced by each are almost exactly the same. Clusters 1, 2, and 4 all have multiple brands, while clusters 3, 5, 6, 7, 8, 9, and 10 only have one brand. The ten largest clusters created by SLINK with and without the SSC tool are listed with their size and brand label in table 5.6. If there are multiple brands for a cluster the label Multi Brand is used.

Cluster Identifier	**Brand**	**Simple Set Comparison Tool Cluster Size**	**SLINK Cluster Size**	**Size Difference**
1	Multi Brand	36,690	36,787	97
2	Multi Brand	12,426	12,705	279
3	eBay	3,502	3,514	12
4	Multi Brand	2,491	2,491	0
5	SFR.com	2,420	2,697	277
6	Alipay	2,231	2,231	0
7	AXA Banque	1,907	1,907	0
8	SFR.com	1,713	1,713	0
9	Bancolumbia	1,592	1,592	0
10	PayPal	1,570	1,573	3

TABLE 5.6: Ten largest clusters produced by SLINK and the Simple Set Comparison Tool

The comparison shows clusters 1, 2, 3, 5, and 10 are not the same size. But the differences are not large for the most part with the largest difference being two hundred seventy nine between the second clusters. Clusters 6, 7, 8, and 9 are exactly the same size and have the same brand labels. Each of the SLINK with SSC tool clusters are either smaller or the same size as the equivalent SLINK cluster. Also, all mono brand clusters have the same brand label. The top ten mono brand clusters are very similar. The similarity between the ten largest clusters produced by the SLINK algorithm and the Simple Set Comparison tool anecdotally supports the claim that the clusterings are very similar. More detailed comparisons of clusters 1 and 2 are examined as these clusters are multi brand clusters and differ in size. Even though the counts are very similar for the multi brand clusters they could be significantly different with respect to their members' brand make up especially because the counts are different.

Figure 5.4 compares the largest cluster produced by SLINK with and without SSC tool. Cluster members are aggregated by brand and the brand counts are compared. Both clusters consist of members with the same eight brands in almost the exact same numbers. There are

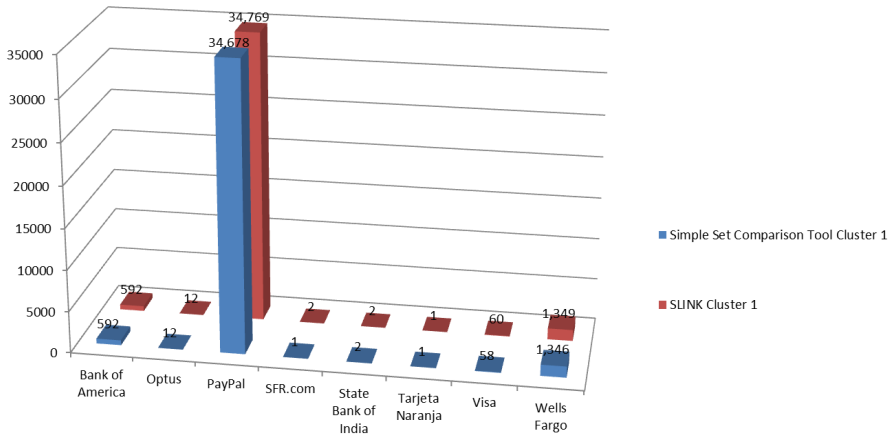

FIGURE 5.4: Comparing Simple Set Comparison Tool's Cluster 1 to SLINK's Cluster 1

only three brands with different counts; PayPal differs in count by one, SFR.com differs in count by one, Visa differs in count by two, and Wells Fargo differs in count by three. Examining the differences shows they consist of members with the same brands and have very similar brand counts. The close similarity between the largest cluster produced by SLINK with and without the SSC tool gives anecdotal support to the claim that clusterings produced by each are almost the same.

Figure 5.5 compares the second largest cluster produced by SLINK with and without the SSC tool. Cluster members are aggregated by brand and the brand counts are compared.

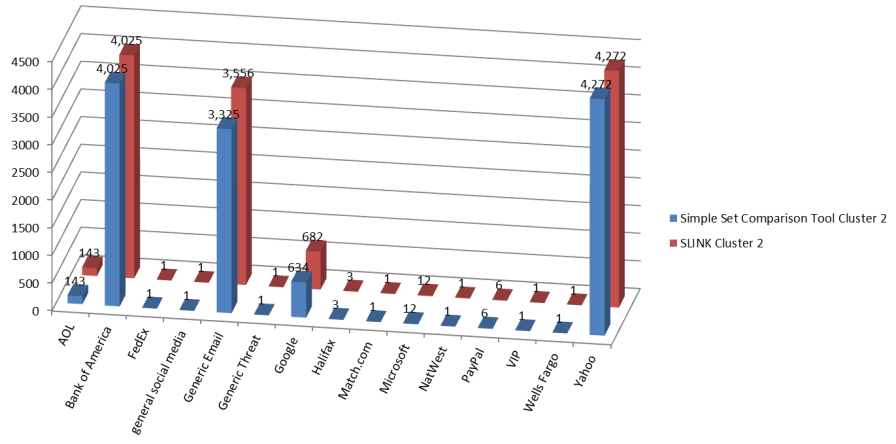

FIGURE 5.5: Comparing Simple Set Comparison Tool's Cluster 2 to SLINK's Cluster 2

Both clusters consist of members with the same fifteen brands with almost the exact same member counts for each brand. Thirteen of the fifteen brands have the same member count, while two brands have a different member count. The two brands with different member counts are

Generic Email with a difference of two hundred thirty one and Google with a difference of forty eight. The difference in the number of members with the Generic Email brand is the largest out of the two different member counts but still only makes a 6.7% difference between the two. This gives anecdotal support to the claim that the clusterings produced by SLINK with and without the SSC tool are almost the same. Comparing the ten largest clusters shows anecdotal evidence to support SLINK produces almost identical clusterings with and without the SSC tool.

5.3.2.3 Runtime

Outside of the quality of the clusterings produced the runtime also needs to be evaluated. The SSC tool allows a clustering algorithm to be partially run in parallel. So, there should be a large runtime gain when SLINK uses the SSCT compared to SLINK without using the SSC tool. SLINK's runtime is the time it takes to create the comparison scores between all data points and then to use the comparison scores to generate a clustering. The runtime is broken down into three categories of comparison score generation and the runtime to generate the clustering, presented in table 5.7.

Traditional Clustering Runtime in Milliseconds	
Compare Phish	111,449,382
Compare Kits	5,013
Compare Phish to Kits	543,437
Clustering	556,552,493
Total Runtime	**668,550,325**

TABLE 5.7: Breakdown of Traditional Clustering Runtime in Milliseconds

The total runtime is approximately 186 hours or almost eight days. Out of the three comparison runtimes the compare phish runtime is the largest. Comparing all phish to one another is going to involve many more comparisons than comparing all kits and comparing all phish to all kits. There are approximately 130,000 phish and 906 kits. Comparing all phish to one another involves approximately 130,0002 or 1.69 x 1010 comparisons. Comparing all kits to one another involves 9062 or approximately 8.2 X 105 comparisons, while comparing all phish to all kits involves 906 X 130,000 or approximately 1.1 x 108 comparisons. It is appropriate for the compare phish runtime to be significantly larger than the other comparison runtimes.

The clustering runtime is the single largest contributing factor to the total runtime. To generate a clustering the slower but less memory intensive implementation of the SLINK algorithm is

having to process a graph with 130,906 points and 661,636,369 edges connecting those points. It makes sense that the clustering runtime would be so large.

SLINK runtime using the SSC tool consists of the runtime for steps two through four. The first step, creating windows, is assumed to be decided before the algorithm is run and does not add any runtime. The second step, clustering windows, is parallelized and is the largest runtime out of all of clustering runtimes. The third step, comparing windows, can be parallelized, but was run in sequence as it did not impact the runtime as much as the second step. The fourth step, merging windows, is a non-parallelized step and is the runtime needed to merge all windows. The runtime broken down by step is presented in table 5.8.

The Simple Set Comparison Tool Runtime in Milliseconds	
Clustering Windows	1,587,458
Comparing Windows	11,986
Merging Windows	55,094
Total Runtime	**1,654,538**

TABLE 5.8: The Simple Set Comparison Tool's Runtime in Milliseconds

The Simple Set Comparison tool's total runtime of 1,654,538 milliseconds is just under a half an hour. The majority of the runtime comes from clustering the windows just as seen in the first experiment. In comparison SLINK without the SSC tool has a runtime of 668,550,325 milliseconds or just under eight days. The java SLINK implementation using the SSC tool is approximately 404 times faster than the plpgsql SLINK implementation without the SSC tool.

5.4 Third Experiment

The third experiment uses a memory lean SLINK implementation in java called Union-Find, UF. UF only loads a list of data points or nodes and a single link or edge at a time. UF produces the same results as the non-memory lean implementation, but uses much less memory. UF with and without the SSC tool is used to cluster the one month data set. UF with the SSC tool and UF without the SSC tool are compared on the quality of the clustering produced, and the runtime to produce a clustering.

5.4.1 Data Set

The third experiment uses the same one month data set used in the first experiment. The one month data set consists of 19,825 confirmed phishing sites collected and tagged by Malcovery Security between September 1st 2014 and September 30th 2014. There are a total of 245 different brands attacked by phishing websites in the one month data set. The phishing websites are represented by MD5 hash values for the main html phishing website and all files that are referenced by the main html page. The referenced files include image files, script files, style sheets, etc on the same domain as the phishing website and on other domains. The data set does not include phishing kits.

5.4.2 Results

As in the prior experiments the results are evaluated based upon the quality of the clusterings produced and runtime taken. UF with the SSC tool and UF without the SSC tool were able to complete a clustering of the one month data set without exceeding the maximum available memory.

5.4.2.1 Quality

The clusterings created by UF with and without the SSC tool are evaluated on their quality using the same three entropy based metrics used in the first, second, and third experiments. The number of clusters in each clustering is also noted. UF with the SSC tool produced 1,209 clusters and UF without the SSCT tool produced 1,193 clusters. Using the SSC tool resulted in only 16 more clusters being produced.

Each clusterings quality metrics are seen in figure 5.6. The homogeneity scores between the two clusterings are almost identical. The homogeneity score for UF with the SSC tool is only 0.0002 lower than the homogeneity score for UF without the SSC tool. The completeness scores are also very similar. The completeness score for UF with the SSC tool is only 0.0018 lower than the completeness score for UF without the SSC tool. As a result of the homogeneity and completeness scores being similar the derived V-measure scores are also very similar. UF with the SSC tool has a V-measure score 0.0012 lower than the V-measure score for UF without the SSC tool. All three entropy based metrics show the two clusterings are very similar.

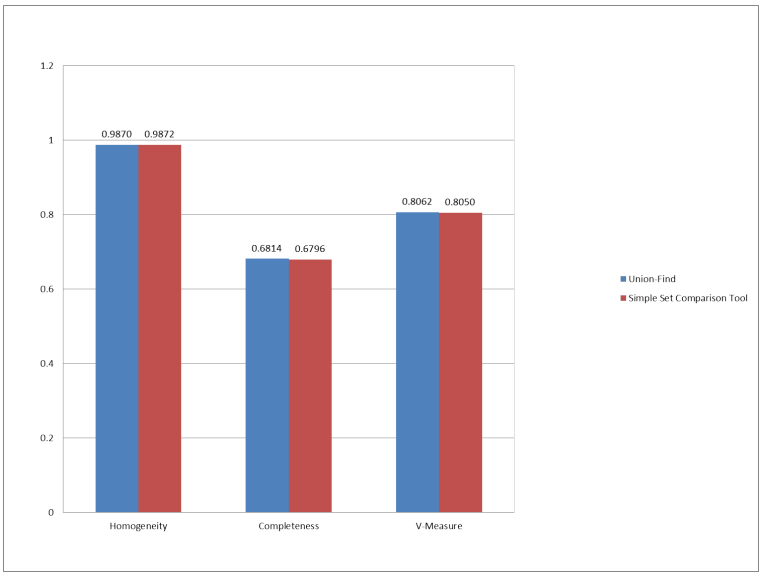

FIGURE 5.6: Comparing Quality Metrics Between UnionFind and UnionFind using SSCT

5.4.2.2 Runtime

The runtimes to produce a clustering with and without the SSC tool are compared. The UF runtime is made up of the time it takes to compare phish using DeepMD5 and the time it takes to cluster the resulting phish similarity graph.

Compare Phish	4,717,239
Cluster	5,054
Total Runtime	4,722,293

1

TABLE 5.9: Union-Find Runtime

UF without the SSC tool has a runtime of 4,722,293 milliseconds or approximately 78 minutes. The majority of the runtime is taken comparing phish with the UF clustering process taking only 5,054 milliseconds or approximately 5 seconds.

To compute the runtime of UF with the SSC tool the runtime of each step is added. Step one is assumed to not contribute to the runtime as it is user defined. The runtime for step two is the longest runtime to generate and cluster a single or cross time window as this step is run in parallel. The runtime for step three is the time it takes to compare clusters from different windows. The runtime for the fourth step is the time it takes to generate a clustering using the cluster comparisons from step three. Table 5.10 gives a detailed listing of each step's runtime with step two being split out between the runtime to compare phish in a window and the runtime to cluster a window.

Compare Phish	784,839
Cluster Windows	1,295
Compare Windows	390
Merge Windows	78
Total Runtime	786,602

TABLE 5.10: Union-Find Runtime using the SSC tool

UF with the SSC tool has a runtime of 786,602 milliseconds or approximately 13 minutes. The majority of the runtime is taken comparing phish in a window with the same window taking only 1,295 milliseconds to cluster. UF with the SSC tool has a runtime that is six times less than UF without the SSC tool.

5.5 Fourth Experiment

The fourth experiment compares UF to UF using the SSCT tool when used to cluster six months of phishing data. A subset of the data used in the second experiment is used in the fourth experiment. Only the phish data from the second experiment is used. The kit data is not used. Both UF with and without the SSC tool were able to produce a clustering for the six months of phishing data without exceeding the maximum amount of memory available. UF using the SSC tool is compared to UF without using the SSC tool. It is evaluated based upon the ability to produce a clustering, the runtime to produce a clustering, and the quality of the clustering produced.

5.5.1 Data Set

The fourth experiment uses the same phish data set of 130,690 phish used in the second experiment, but does not include the 908 kits that were included in the second experiment. The data set was collected from July 1st 2014 to January 11th 2015.

The kit data was excluded as the kit data only adds 908 data points, but requires the same amount of work to include as the much larger phish data set. Excluding the set will also not significantly decrease the memory costs as the number of data points is small and the number of links are relatively small when compared to the number of links created by the phish data alone.

5.5.2 Results

As in the prior three experiments the fourth experiment compares the clusterings produced by UF using the SSC tool and UF without the SSC tool. The runtime taken to produce a clustering is also examined. UF with and without the SSC tool were able to successfully produce a clustering without exceeding the maximum memory available.

5.5.2.1 Quality Comparison

Clusterings produced by UF without the SSC tool and UF with the SSC tool are compared based upon the number of clusters created and the three entropy quality metrics used in the prior three experiments.

UF produced 3,935 clusters, while UF using the SSC tool produced 4,150 clusters. Using the SSC tool increased the number of clusters produced by 215.

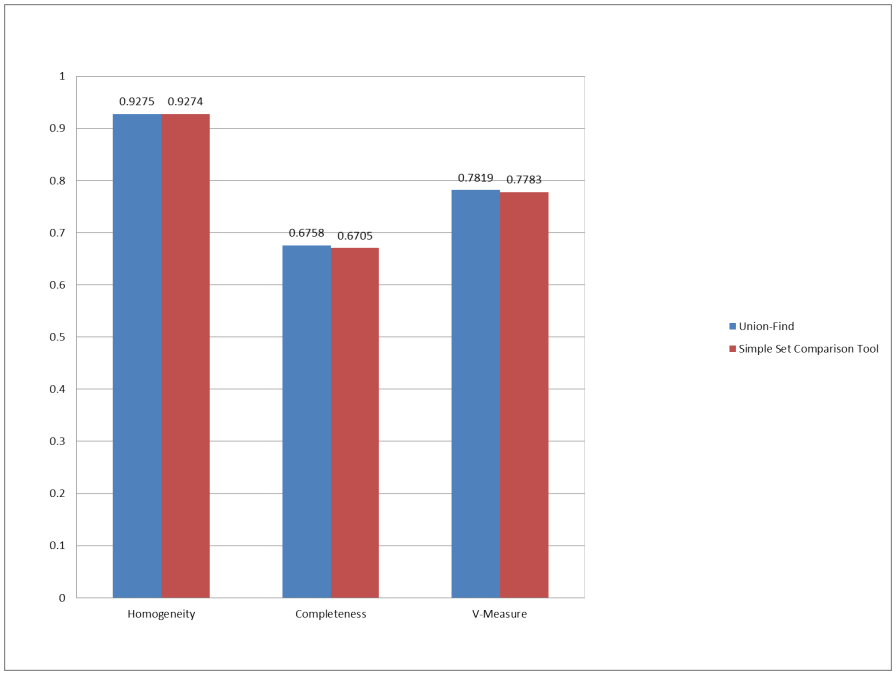

FIGURE 5.7: Comparing Quality Metrics Between UF and UF using the SSC tool

The homogeneity, completeness, and V-measure scores are very similar. For each score UF without the SSC tool has a slightly better score than UF with the SSC tool. UF without the SSC tool has a homogeneity score that is 0.0001 greater, a completeness score that is 0.0053 greater, and a V-measure score that is 0.0036 greater than UF with the SSC tool. All three quality measures and the number of clusters created are very similar.

5.5.2.2 Runtime

The runtimes to create a clustering using UF with the SSC tool and without the SSC tool are compared. The runtime for UF consists of the runtime to compare phish and the runtime to cluster the comparison results.

Compare Phish	28,771,848
Cluster	394,713
Total Runtime	29,166,561

TABLE 5.11: Union-Find Runtime without the SSC tool

The UF process took a total of 29,166,561 milliseconds or 486 minutes to complete a clustering. The majority of the runtime comes from comparing phish.

Compare Phish	2,530,337
Cluster Windows	5,085
Compare Windows	148,770
Merging Windows	3,092
Total Runtime	2,687,284

TABLE 5.12: Runtimes for Union-Find with SSCT

UF using the SSC tool took a total of 2,687,284 milliseconds or approximately 45 minutes to complete. UF with the SSC tool has a runtime that is 10.8 times faster than UF without the SSC tool. The SSC tool significantly improves the runtime to produce a clustering.

CHAPTER 6

Discussion

The discussion section takes a more detailed look at the quality and runtime results from the four experiments. The question of exceeding the maximum amount of memory available is also examined. The discussion section also covers the interchangeability of the comparison metric and clustering algorithm.

6.1 Comparable Quality

In all four experiments the clustering algorithm using the SSC tool produces comparable results to the clustering algorithm that does not use the SSC tool. The quality of clusterings produced differs almost negligibly in the homogeneity, completeness, and V-measure metrics.

The SSC tool is slightly worse when evaluated on quality measures in the six month experiments, but only slightly more than the one month experiments. The quality differences are negligible in both the one and six month experiments.

6.2 Improved Runtime

The computational complexity analysis shows using the SSC tool does not increase the growth rate as the number of data points increase as long as a maximum number of windows is set. The SSC tool is easily run in parallel without requiring specialty infrastructure or a reformulation of a clustering algorithm to run in parallel.

All four experiments show the SLINK style clustering algorithm using the SSC tool has a dramatically better runtime compared to the SLINK style clustering algorithm not using the SSC tool. In the first experiment, the clustering algorithm using the SSC tool has a runtime 37 times faster than the clustering algorithm that does not use the SSC tool. In the second experiment, the clustering algorithm using the SSC tool has a runtime 404 times faster than the clustering algorithm that does not use the SSC tool. In the third experiment, the clustering algorithm using the SSC tool has a runtime 6 times faster than the clustering algorithm that does not use the SSC

tool. In the fourth experiment, the clustering algorithm using the SSC tool has a runtime 10.8 times faster than the clustering algorithm that does not use the SSC tool.

The larger improvement in runtime in the second experiment is due to the implementation changes required to process the six month data set when not using the SSC tool. The SSC tool still used the original SLINK algorithm implementation. While this is not comparing apples to apples it shows the SSC tool's ability to reduce maximum memory cost and how that can affect runtime. All four experiments show using the SSC tool improves the SLINK algorithm's runtime.

6.3 Exceeding Maximum Memory

In the second experiment the memory hungry SLINK implementation that did not use the SSC tool was not able to produce a clustering as it exceeded the eight gigabytes of memory available. The memory hungry SLINK implementation loads all nodes and edges into memory. The data set in the second experiment is represented by a graph with 130,906 nodes and 661,636,369 edges. All of the nodes and edges can not be loaded into main memory as seen by the out of memory error that occurs in the second experiment. The SLINK algorithm only requires the nodes to be loaded and a single edge. However, other clustering algorithms such as DBSCAN require an undeterminable number of accesses to the same data. The result is having to load all nodes and edges into main memory or load them onto disk resulting in a large runtime increase. The SSC tool reduces the maximum amount of memory required by splitting the data into windows. The window with the largest number of edges had 14,398 nodes and 14,321,012 edges. This is only a fraction of the size of the entire data set. The SSC tool can allow a clustering algorithm to process a larger data set than could normally fit in main memory as seen in the second experiment.

6.4 Interchangeable Components

The SSC tool has three interchangeable components. The distance metrics used to compare data in the first step, the clustering algorithm used to cluster phish in the first step, and the clustering algorithm used when merging similar clusters in the third step. The SSC tool can make use of a variety of similarity metrics. The tool only requires the similarity metric be numeric and have an upper and lower bound. The SSC tool can make use of a variety of clustering algorithms. The tool requires a clustering algorithm to take an edge representation of a graph as input and produce

non-overlapping clusters. These loose requirements allow the SSC tool to work with a variety of similarity metrics and clustering algorithms.

6.4.1 Deep MD5

The Deep MD5 metric is being used as a similarity measure in the first step as it has been shown to be useful for clustering phishing websites [51, 55, 56]. The Deep MD5 metric is not fool proof as it relies on file reuse by phish. Small changes to a file will change the MD5 value for that file. If a phishing author was so inclined all content files referenced by a phish could be slightly changed each time a particular phish was created. The result would be a Deep MD5 score of 0.0 between two phish created by the same author that targeted the same brand with the same functionality and appearance. However, this has not been noticed to be prevalent in the wild at this time. If this does occur at some future date, the similarity metric used by the SSC tool is interchangeable and another more sufficient phish similarity metric can be used in place of Deep MD5. The only requirement the SSC tool has for a comparison metric is that the metric produces a single numerical value within a defined upper and lower bound. The Deep MD5 similarity metric is being used as an example similarity metric that is currently effective in this particular use case.

6.4.2 SLINK Algorithm

The SLINK clustering algorithm is used in step one for clustering windows and in step three when merging clusters. The same clustering algorithm does not have to be used in both step one and step three. Indeed there may be circumstances where using a different clustering algorithm in step one and step three may produce better results. However, in this particular case using a SLINK style clustering algorithm to cluster the time windows in step one and merge clusters in step three is effective as it produces a clustering of similar high quality to traditional clustering. The SLINK clustering algorithm is not the best or newest clustering algorithm. It is a simple clustering algorithm and has been shown to produce good results when applied to clustering phish [51, 55, 57, 58]. Like the similarity metric, the clustering algorithm used by the SSC tool is interchangeable. The SSC tool only requires a clustering algorithm take an edge based representation of a graph as input and produce non-overlapping clusters within a single data set.

CHAPTER 7

Conclusion

Phishing is a problem today and continues to grow in size. Phishing is diverse in its use of social engineering tactics to enable a successful attack. Countermeasures exist to combat phishing and each countermeasure has pros and cons. To select the most effective countermeasures and evaluate the effectiveness of countermeasures algorithms to perform phish grouping are needed.

A phish grouping process tries to achieve three different goals. It tries to produce dependable groups, quickly produce groups to allow phishing attacks to be evaluated in as close to real time as possible, and have the ability to analyze large volumes of phish.

A variety of phish grouping processes currently exist. One such process is a SLINK style algorithm run over DeepMD5 comparisons of phishing websites. Malcovery uses this process to automate phishing detection and brand identification.

The SSC tool has been developed to improve runtime and the volume of data that can be processed by a phish grouping process. The four experiments run use large phishing data sets collected over one month and six months. The data sets were clustered using an implementation of a SLINK style algorithm with and without the SSC tool. In each experiment, the SLINK style algorithm using the SSC tool had significantly improved runtime and produced clusterings with equivalent quality.

In the second experiment the SSC tool allowed the SLINK style clustering algorithm to produce a clustering when the same implementation failed to produce a clustering because it exceeded the eight gigabytes of memory available.

The SSC tool successfully improved the runtime of the SLINK style algorithm and increased the volume of data the algorithm can process without exceeding available memory. The SSC tool improves a currently used phish grouping algorithm.

CHAPTER 8

Future Directions

In the four experiments conducted DeepMD5 was used to measure the similarity of phishing websites. As noted though the DeepMD5 similarity metric can be rendered ineffective if phishing website authors make minor changes to files to alter the file's MD5 hash value. The SSC tool only requires a weight value be developed for a link or edge between two nodes. Another similarity metric can be used to replace the DeepMD5 score used in the four experiments.

In each of the four experiments conducted the SSC tool was used with a SLINK style clustering algorithm. The memory hungry implementation of the SLINK style algorithm provides in memory access to all nodes and edges. The in-memory access to all nodes and edges would allow other clustering algorithms to effectively run such as DBSCAN, OPTICS, Chameleon, and others. It may be possible to use the SSC tool with other clustering algorithms used in applications other than phish grouping.

LIST OF REFERENCES

[1] APWG. About the apwg, 2014. https://apwg.org/about-APWG/.

[2] PhishTank. Phishtank faq, 2014. http://www.phishtank.com/faq.php#whatisphishtank.

[3] Kaspersky Lab. Kaspersky lab, 2014. http://www.kaspersky.com/about/news/virus/2013/Malware_spam_and_phishing_the_threats_most_commonly_encountered_by_companies.

[4] G. Aaron and R. Manning. Phishing activity trends report 2nd quarter 2014, 2014. http://docs.apwg.org/reports/apwg_trends_report_q1_2014.pdf

[5] Gary Warner. Cybercrime & doing time, 2015. http://garwarner.blogspot.com/2014/11/university-accept-your-new-raisephish.html.

[6] Brian Krebs. Krebsonsecurity, 2015. http://krebsonsecurity.com/2015/02/phishers-pounce-on-anthem-breach/.

[7] Gary Warner. Cybercrime & doing time, 2015. http://garwarner.blogspot.com/2015/06/49-corporate-email-phishers-arrestedin.html.

[8] S. Sheng, B. Magnien, P. Kumaraguruand A. Acquisti, L. Faith Cranor, J. Hong, and E. Nunge. Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish, 2007. http://garwarner.blogspot.com/2015/06/49-corporate-email-phishers-arrestedin.html.

[9] APWG. Welcome to apwg & cmu's phishing education landing page, 2015. http://phish-education.apwg.org/r/en/index.htm.

[10] APWG. Phishing - owasp, 2015. https://www.owasp.org/index.php/Phishing.

[11] Wombat Security Technologies. Interactive training modules, 2015. https://www.wombatsecurity.com/security-education/educate.

[12] McAfee. Classroom and virtual instruction, 2015. http://www.mcafee.com/us/services/strategic-security-education/classroomtraining/index.aspx.

[13] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. In *Proceedings of Sixth Conference on Email and Anti-Spam (CEAS)*, 2009.

[14] Tyler Moore and Richard Clayton. An empirical analysis of the current state of phishing attack and defence. In *WEIS*, 2007.

[15] Cyveillance. Anti-phishing solutions to prevent and mitigate attacks, 2015. https://www.cyveillance.com/home/cyveillance-services/antiphishing/.

[16] Rhiannon Weaver and M Patrick Collins. Fishing for phishes: Applying capture-recapture methods to estimate phishing populations. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 14–25. ACM, 2007.

[17] Huajun Huang, Junshan Tan, and Lingxi Liu. Countermeasure techniques for deceptive phishing attack. In *New Trends in Information and Service Science, 2009. NISS'09. International Conference on*, pages 636–641. IEEE, 2009.

[18] Min Wu, Robert C Miller, and Simson L Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610. ACM, 2006.

[19] Philip J Nero, Brad Wardman, Heith Copes, and Gary Warner. Phishing: Crime that pays. In *eCrime Researchers Summit (eCrime), 2011*, pages 1–10. IEEE, 2011.

[20] Leo Lipis. Foreign banking organizations in the u.s., 2015. https://www.theclearinghouse.org/publications/2015/2015-q1-bankingperspective/real-time-payment-fraud.

[21] Philip K Chan, Wei Fan, Andreas L Prodromidis, and Salvatore J Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and Their Applications*, 14 (6):67–74, 1999.

[22] Yufeng Kou, Chang-Tien Lu, Sirirat Sirwongwattana, and Yo-Ping Huang. Survey of fraud detection techniques. In *Networking, sensing and control, 2004 IEEE international conference on*, volume 2, pages 749–754. IEEE, 2004.

[23] Binod Gyawali, Thamar Solorio, Manuel Montes-y Gómez, Bradley Wardman, and Gary Warner. Evaluating a semisupervised approach to phishing url identification in a realistic

scenario. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, pages 176–183. ACM, 2011.

[24] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254. ACM, 2009.

[25] Brad Wardman, Gaurang Shukla, and Gary Warner. Identifying vulnerable websites by analysis of common strings in phishing urls. In *2009 eCrime Researchers Summit*, pages 1–13. IEEE, 2009.

[26] Matthew Dunlop, Stephen Groat, and David Shelly. Goldphish: Using images for content-based phishing analysis. In *Internet Monitoring and Protection (ICIMP), 2010 Fifth International Conference on*, pages 123–128. IEEE, 2010.

[27] Ram Basnet, Srinivas Mukkamala, and Andrew H Sung. Detection of phishing attacks: A machine learning approach. In *Soft Computing Applications in Industry*, pages 373–383. Springer, 2008.

[28] R Suriya, K Saravanan, and Arunkumar Thangavelu. An integrated approach to detect phishing mail attacks: a case study. In *Proceedings of the 2nd International Conference on Security of Information and Networks*, pages 193–199. ACM, 2009.

[29] Jason Britt, Brad Wardman, Alan Sprague, and Gary Warner. Clustering potential phishing websites using deepmd5. In *Presented as part of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2012.

[30] Brad Wardman, Gary Warner, Heather McCalley, Sarah Turner, and Anthony Skjellum. Reeling in big phish with a deep md5 net. *The Journal of Digital Forensics, Security and Law: JDFSL*, 5(3):33, 2010.

[31] Shams Zawoad, Amit Kumar Dutta, Alan Sprague, Ragib Hasan, Jason Britt, and Gary Warner. Phish-net: Investigating phish clusters using drop email addresses. In *eCrime Researchers Summit (eCRS), 2013*, pages 1–13. IEEE, 2013.

[32] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[33] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

[34] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.

[35] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[36] Benjamin Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 870–878. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.

[37] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

[38] Joao C Xavier Junior, Anne Canuto, Alex A Freitas, Luis Marcos G Goncalves, and Carlos N Silla Jr. A hierarchical approach to represent relational data applied to clustering tasks. In *Proceedings of the 2011 International Joint Conference on Neural Networks*, pages 182–196. IEEE Press, 2011.

[39] Samer Nassar, Jörg Sander, and Corrine Cheng. Incremental and effective data summarization for dynamic hierarchical clustering. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 467–478. ACM, 2004.

[40] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):24–45, 2004.

[41] Aris Anagnostopoulos, Anirban Dasgupta, and Ravi Kumar. Approximation algorithms for co-clustering. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 201–210. ACM, 2008.

[42] Claudia Plant. Sonar: Signal de-mixing for robust correlation clustering. In *SDM*, pages 319–330. SIAM, 2011.

[43] Stephan Günnemann, Ines Färber, and Thomas Seidl. Multi-view clustering using mixture models in subspace projections. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 132–140. ACM, 2012.

[44] Irene Ntoutsi, Nikos Pelekis, and Yannis Theodoridis. Pattern comparison in data mining: a survey. *Research and Trends in Data Mining Technologies and Applications: Advances in Data Warehousing and Mining, Idea Group*, 2008.

[45] Charu C Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 575–586. ACM, 2003.

[46] Marina Meilă. Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd international conference on Machine learning*, pages 577–584. ACM, 2005.

[47] Fei Wang, Noah Lee, Jianying Hu, Jimeng Sun, and Shahram Ebadollahi. Towards heterogeneous temporal clinical event pattern discovery: a convolutional approach. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 453–461. ACM, 2012.

[48] M Shahriar Hossain, Satish Tadepalli, Layne T Watson, Ian Davidson, Richard F Helm, and Naren Ramakrishnan. Unifying dependent clustering and disparate clustering for non-homogeneous data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 593–602. ACM, 2010.

[49] Gilles Bisson and Clement Grimal. Co-clustering of multi-view datasets: a parallelizable approach. In *2012 IEEE 12th International Conference on Data Mining*, pages 828–833. IEEE, 2012.

[50] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

[51] Brad Wardman, Tommy Stallings, Gary Warner, and Anthony Skjellum. High-performance content-based phishing attack detection. In *eCrime Researchers Summit (eCrime), 2011*, pages 1–9. IEEE, 2011.

[52] Ronald Rivest. The md5 message-digest algorithm. 1992.

[53] M. Maischein. Www::mechanize::firefox, 2013. http://search.cpan.org/dist/WWW-Mechanize-FireFox/.

[54] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

[55] Brad Wardman, Gary Warner, Heather McCalley, Sarah Turner, and Anthony Skjellum. Reeling in big phish with a deep md5 net. *The Journal of Digital Forensics, Security and Law: JDFSL*, 5(3):33, 2010.

[56] Jason Britt, Brad Wardman, Alan Sprague, and Gary Warner. Clustering potential phishing websites using deepmd5. In *Presented as part of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2012.

[57] Shams Zawoad, Amit Kumar Dutta, Alan Sprague, Ragib Hasan, Jason Britt, and Gary Warner. Phish-net: Investigating phish clusters using drop email addresses. In *eCrime Researchers Summit (eCRS), 2013*, pages 1–13. IEEE, 2013.

[58] Brad Wardman, Jason Britt, and Gary Warner. New tackle to catch a phisher. *International Journal of Electronic Security and Digital Forensics*, 6(1):62–80, 2014.

APPENDIX A

The appendix section provides an exact definition for the three entropy based comparison metrics used to evaluate clusterings. The three entropy based metrics are homogeneity, completeness, and v-measure. A more detailed explanation of the three entropy metrics is provided in Rosenberg et al [54]. For the following definitions a few variables are needed to exactly define the homogeneity, completeness, and v-measure entropy based metrics. Assume the data set is comprised of N data points and two different partitions of N. The first partition is the set of classes or ground truth labels, C, where the set of classes in C is 1...n. The second partition is the set of clusters, K, where the set of clusters in K is 1...m. A is the contingency table produced by a clustering algorithm representing the clustering output. A has a set of members aij that are the members of class ci and elements of cluster kj. Homogeneity evaluates how well the clustering is at placing members that should be in the same cluster in the same cluster. A perfect homogeneity score is achieved when all clusters only contain members with the same label. Two preliminary calculations are needed before the homogeneity can be determined. The first is H(C/K) and the second is H(C).

$$H\left(C/K\right) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\frac{a_{ck}}{N}\log\frac{a_{ck}}{\sum_{c=1}^{|C|}a_{ck}} \tag{8.1}$$

$$H(C) = \sum_{c=1}^{|C|}\frac{\sum_{k=1}^{|K|}a_{ck}}{n}\log\frac{\sum_{k=1}^{|K|}a_{ck}}{n} \tag{8.2}$$

$$Homogeneity = 1 - \frac{H(C/K)}{H(C)} \tag{8.3}$$

Completeness evaluates how well the clustering came to determining the correct number of clusters. A perfect completeness score is achieved when there is only one cluster for each label. Two preliminary calculations are needed to calculate completeness. The first is H(K/C) and the second is H(C).

$$H\left(K/C\right) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{a_{ck}}{N}\log\frac{a_{ck}}{\sum_{k=1}^{|K|}a_{ck}} \tag{8.4}$$

$$H(K) = \sum_{k=1}^{|K|}\frac{\sum_{c=1}^{|C|}a_{ck}}{n}\log\frac{\sum_{c=1}^{|C|}a_{ck}}{n} \tag{8.5}$$

$$Completeness = 1 - \frac{H(K/C)}{H(K)} \tag{8.6}$$

V-measure is the harmonic mean of the homogeneity and completeness scores, a blend of homogeneity and completeness scores. A variable, beta, determines the weighting between the homogeneity and completeness scores. A beta value less than 1 weights homogeneity more than completeness. A beta value greater than 1 weights completeness more than homogeneity. A beta value of 1 equally weights homogeneity and completeness. The beta value used in this research is 1 to equally weight the importance of homogeneity and completeness.