CASTLE: A Social Framework for Collaborative Anti-Phishing Databases

Arash Nourian

School of Computer Science
McGill University
Montreal, Canada

October 2009

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Science.

Dedication

To My love Maryam for her devotional support
My Mom and Dad

Acknowledgements

I would like to thank my supervisor, Professor Muthucumaru Maheswaran, for his dedication and guidance which encouraged me to think thoroughly and work hard. I had a chance to receive advice from him for my thesis along with his profound knowledge which was shared delicately with me.

I would also like to thank my colleagues at the Advanced Networking Laboratory (ANRL) who made my time with them enjoyable and rewarding. In particular, I would like to thank Sameer Ishtiaq for several insightful discussions, suggestions, and for helping out with the simulations.

I would also like to thank Scott Assen and Martin Ashton for proofreading the thesis and translating the abstract into French. Many thanks to Diti Anastasopoulos for her support and making the procedure easier. Finally I'd like to mention my appreciation for the research facilities provided by the *Advanced Networking Laboratory*, *School of Computer Science* and *McGill University*.

Abstract

A Phishing attack is a type of identity theft attempting to steal confidential and personal data like Credit Card or banking account information. Different approaches have been proposed to defeat phishing attacks. Most of the approaches rely on a database lookup approach. In this thesis, we present a framework called CASTLE that allows a collaborative approach to build and maintain the databases containing information needed for anti-phishing services. We provide the full design and discuss how phishing sites can be captured using CASTLE. A prototype of this social framework for collaborative anti-phishing databases is partially implemented to evaluate the performance and effectiveness of the framework against phishing attacks.

Résumé

L'hameçonnage est un type de vol d'identité qui tente de voler des donnés confidentielles et personnelles comme l'information de cartes de crédit ou de comptes bancaires. Plusieurs stratégies ont été proposées pour vaincre l'hameçonnage ; la plupart d'entre elles dépendent d'une base de données. Dans cette thèse, nous présentons le cadre CASTLE, qui incite la collaboration pour construire et entretenir des bases de donnes contenant l'information nécessaire pour contrer l'hameçonnage. Nous fournissons la conception et discutons la manière avec laquelle les sites de hameonnage peuvent être capturés a l'aide de CASTLE. Un prototype de ce cadre est partiellement mis en œuvre pour évaluer la performance et l'efficacit du cadre contre les attaques de hameçonnage.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Websites are the most common tools for e-commerce and they are the core of internet business. However, despite vast usage of websites by users and industries and daily development based on those user's needs, security flaws of the websites are still bringing various dangers and financial burdens to their companies. Thus, web security as one of the fundamental issues of the today's internet is the one that is highly important and should be addressed accordingly.

In recent years, the number of phishing attacks has dramatically increased. The purpose of these attacks are to exploit the user's sensitive information such as credit card numbers or bank account information of the users to steal money from them. Lack of robust solutions against phishing is evident. Thus, researchers are exploring solutions that can address phishing efficiently, considering the fact that providing effective solutions against phishing websites is important nowadays.

1.2 Thesis Contribution

The key contribution of this thesis is to design and develop a new distributed framework for battle against phishing in the Internet. One of the key differences of our work from previous proposals [1, 2, 3] is providing decentralized solutions rather than the centralized common approaches which are based on the centralized repository

that are not effective for the zero day attacks [4], and, if the centralized repository is compromised, all of the requests for checking whether a particular URL is phishing or not would be useless.

We proposed the CASTLE as a social collaborative framework that is solely administered by people and their organizations (some of them could be community groups). Our social collaborative framework is built on online social networks. Using our framework a user can check the validity of a website if there is a responsible node for the queried website inside the social network by connecting to one of the nodes and forwarding the request to it.

CASTLE is structured as a P2P socially administrated network of phishing verification servers (PVC's) which are responsible for telling the user whether the given URL is phishing or not. Each PVC contains routing information for its trusted neighbors. They can search and find the PVC which is responsible for a given URL injected by the user who has been connected to one of the PVC's.

The other element of the CASTLE is its content analysis module which is responsible for finding the target of the phishing website if the injected URL by the user is not hosted in any of the PVCs in the CASTLE network.

Phishing attacks are among the most important problems which target millions of users every year. A successful phishing prevention solution should have the following characteristics [5]:

- Adaptability with the current internet infrastructure without requirements of fundamental changes in the internet infrastructure.

- Ability to support the users of all websites regardless of the technology that they use in their websites.

- User transparency.
- Ability to prevent different types of phishing attacks.
- No dependency to central service for prevention which may be target of the attackers too.

1.3 Thesis Organization

Considering the above goals, this thesis has been organized as follows:

1. Researching and evaluating the process of phishing attacks.
2. Researching and evaluating different types of phishing attacks.
3. Study of the proposed phishing prevention techniques and find their weakness and capabilities.
4. Proposing a solution for prevention of the phishing attacks.
5. Evaluating the proposed solution and its effectiveness in battle against phishing.

Chapter 2

Background Information

In this chapter, the term *Social Networks* and *Phishing* are explained, and we will show some common *Internet attacks* as well as *Phishing attacks* and their characteristics. Then we will propose some protection techniques for battling against phishing.

2.1 Social Network

2.1.1 Definition

The term *Social Network* [6] was first introduced in 1954, by Barnes [7]. According to his explanation, a social network is a social structure composed of people or organizations that are tied by specific types of relationship, such as values, ideas, friendship or interests. In [8], the social network is expressed as "patterns or regularities in relationships among interacting actors". Actors and their actions are viewed as interdependent rather than independent, autonomous units. Relational ties between actors are channels for transfer or "flow" of resources.

Recently, social networks have been mentioned in the context of the Internet. Since the emergence of Web 2.0, some websites have integrated social network aspects in order to provide online social networking service to end-user, which are often referred to as "Online Social Network".

2.1.1.1 Online Social Network

Online Social Network [9] is defined as web-based services that allow individuals to:

1. Construct a public or semi-public profile within a bounded system.
2. Articulate a list of other users with whom they share a connection
3. View and traverse their list of connections and those made by others within the system [10].

Nowadays, online social networking has become a fundamental part of the global Internet experience. The benefits of online social networking have convinced researchers to develop solutions that leverage the collaboration of users.

Phishing can also be prevented through the social networking of all people who are the targets of such attacks in order to build a robust network of trusted sites. Thus, the information that can be obtained from such a social network can be used as a trusted source of information for those who want to know whether a website is phishing by asking the corresponding node inside the social network.

2.2 Internet Attacks

Internet attacks can be categorized based on different parameters. One categorization of the attacks is based on the flaws that are present in the current websites. Accordingly, different types of attacks can be categorized into six group types [11]:

1. **Authentication Attacks**
2. **Authorization Attacks**
3. **Client-Side Attacks**

4. **Command Execution Attacks**

5. **Information Disclosure**

6. **Logical Attacks**

Authentication attacks and authorization attacks are those attacks which try to bypass the authentication and authorization such as brute force attacks and session prediction attacks. Command execution attacks focus on executing remote malicious commands on the websites such as LDAP injection and SQL injection attack. Information disclosure covers attacks which tries to acquire system specific information about a website such as directory indexing and path traversal attacks. Logical attacks cover the abuse or exploitation of a web application such as insufficient process validation and denial of service attack.

The Client-Side Attacks section focuses on the abuse or exploitation of a web site's users. When a user visits a web site, trust will be established between the two parties both technologically and psychologically. A user expects the delivery of the valid content by the visited web site. A user also expects the web site not to attack them during the browsing of the website. By leveraging these trust relationship expectations, an attacker may employ several techniques to exploit the user.*Content Spoofing, Cross Site Scripting* ,and *Phishing* are some examples of the client-side attacks.

2.3 Phishing

2.3.1 Introduction

The meaning of the phishing in webopedia [12] is "The act of sending an e-mail to a user falsely claiming to be an established legitimate enterprize in an attempt to scam the user into surrendering private information that will be used for identity theft." Phishing is a way of stealing users' information with the usage of tricks and techniques of social engineering. The stolen information includes credit cards, usernames, and passwords that could be used in order to access the user's bank account to steal money.

The first step to prevent the occurrence of phishing is to know the phishing type and its steps [13]. In this chapter, we will investigate different types of phishing along with their characteristics. First we define the phishing in detail and the importance of the prevention techniques according to the statistics that shows the increase of the phishing attacks [14] in recent years. Then, we will show the phishing steps along with the phishing emails and websites which will be the last covered topics in this chapter.

The major goal of phishing attacks is to steal valuable personal or identity related information from users. Once the attackers gain personal data such as passwords, date of birth, and bank account numbers, they use the information to their benefit by creating bogus identities or taking control of online accounts.

The phishing attacks are launched in many different ways [15]. The most common way is to send an email to users and persuade users to click on a forged link inside the email. When the user clicks on the forged link, the user will be sent to a site

controlled by the attacker which looks similar to a trusted site (e.g., the user's online bank). Since the site is under the attacker's control, all the information revealed by the user to the website such as usernames and passwords are obtained by the attacker. A vigilant user may detect the difference between the bogus site created by the attacker and the legitimate site. However, a sizeable population of the users do fall prey to these attacks. For example, legitimate websites (depending on the sophistication of the institution) can have welcome messages that include the user's name from previous successful logins. Such login information is not available for the bogus site and consequently it cannot customize the welcome message.

2.3.2 Phishing for stealing user identity

As we discussed in section 2.3.1, phishing is a method of stealing critical user information. Having such information, an attacker can represent himself as a trusted user with the stolen identity. Some types of phishing attacks incorporate social engineering to steal user's information [16]. In these attacks, attackers use bogus emails to deceive the users and redirect them to the un-trusted website which is controlled by the attackers. Attackers try to represent themselves as banks or credit card companies to users in order to persuade them to insert their usernames, passwords, or credit card numbers in the fake website.

Another type of phishing attack is to use some sort of deceiving techniques to steal the user's information [17]. In such an attack, attackers try to install crimewares on the user's machine to steal the information. Key Loggers [18] are another

type of crimewares which are generally used to steal the usernames and passwords of the users. DNS poisoning [19] is also used to change the real IP address of a trusted website. Thus, users will not see any difference in the website address while they are redirected to the bogus website by the DNS poisoning attack and their information would be hijacked.

Those phishing attacks that use social engineering methods to steal the user's information are generally initiated with an email. The bogus email tries to convince the user that it has been sent from an accredited company that the user knows and persuades the user to click on the provided web links inside the email. These links are generally malicious and redirect the users to an un-trusted website which looks exactly like the valid one that has been visited by the users previously. The bogus website asks the users to disclose the personal information such as credit card numbers, usernames, and passwords. This information will be sent to the attackers who steal the user's online identity and who then impersonate him/her on the web since they now have all of the user's critical information such as first name, family name, username, password, credit card number, and social insurance number.

2.3.3 The prevalence of the phishing attacks

The number of phishing attacks is increasing rapidly [14]. Because of their value, financial institutions are the favorite targets of the phishing attackers. Recently, this form of attack has diversified and started targeting social networking sites as well [14]. Further, the technologies adopted by the attackers are getting more sophisticated every day. Figure 2.1 shows the highlights of the statistics obtained by a recent study

on phishing attacks. According to recent studies, the average uptime for a phishing website is 4.5 days [20].

Statistical Highlights for Q2 2008	April	May	Jun
Number of unique phishing email reports received by APWG from consumers	24,924	23,762	28,151
Number of unique phishing websites detected	20,410	20,317	18,509
Number of brands hijacked by phishing websites	276	294	227
Country hosting the most phishing websites	China	Turkey	US
Certain some form of target name in URL	28.3%	23.2%	26.1%
No hostname; just IP address	5.5%	13.2%	4%
Percentage of sites not using port 80	.81%	.45%	.49%
Longest time online for website	30 days	31 days	30 days

Fig. 2.1 Highlights of a report from the Anti Phishing Working Group [20].

Figures 2.2, 2.3, 2.4 show the statistics related to the phishing attacks that have been collected the by the Anti Phishing Working Group (AWPG) [20].

The financial damage of phishing attacks is severe and shocking. Based on the statistics of Gartner Institute [21] more than 56 million people have received a phishing email in 2008 and 1.85 million people have disclosed their personal and financial information to phishing websites. Banks and credit card companies have lost more than 1.3 billion dollars due to phishing attacks [21]. Also 5% of internet users are the victims of phishing attacks in US every year [21].

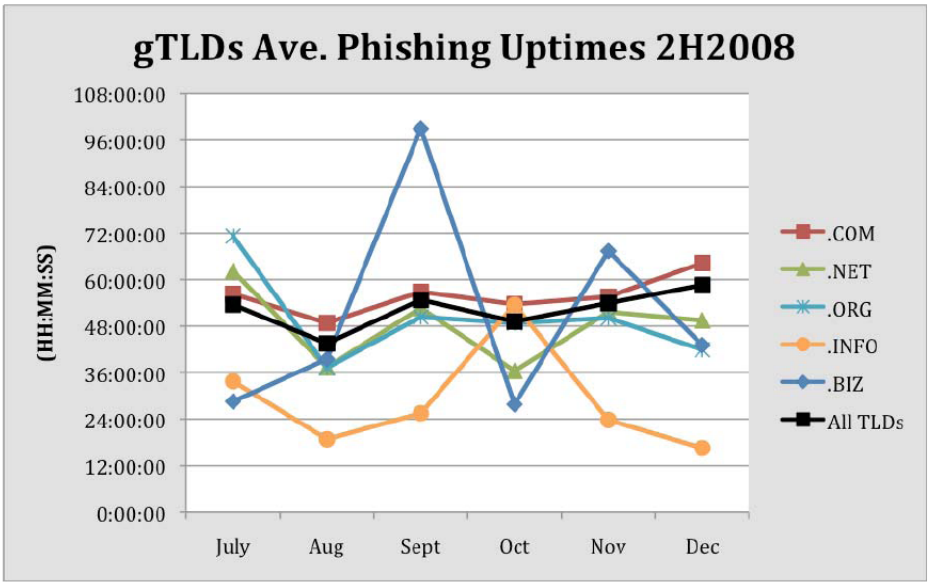

Fig. 2.2 Average of the phishing uptimes for the second half of 2008 [20].

Number of Unique Phishing Web Sites Detected	20,410	20,317	18,509
Unique Domains	6,176	5,849	5,633
Unique Brand-Domain Pairs	7,656	7,267	6,768
Unique Brands	276	294	227
URLs Per Brand	74	69	82

Fig. 2.3 Highlights of a report from the Anti Phishing Working Group [20].

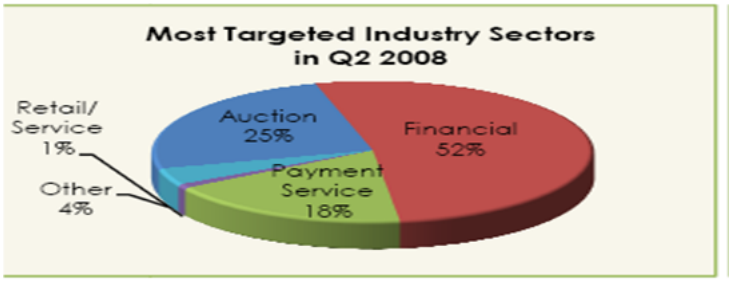

Fig. 2.4 Most targeted industries by the phishing attacks [20].

Due to the severity of phishing attacks, we need to have very careful knowledge of these kinds of attacks.

2.3.4 Phishing attacks steps

We need to precisely investigate the processes and steps of phishing attacks in order to find a comprehensive solution for them. In general, Phishing attacks consist of 4 steps [22]. Attackers will follow the following steps after designing a bogus website very similar to the users' trusted websites:

Distributing malicious links In this step, attackers try to send the user a malicious links through email or instant messaging. The designation of the users as phishing targets can be done intentionally if the attackers know the victims or it can be done randomly.

Visiting the phishing sites In this step, the victim will click on a malicious link that has been received through some sort of communication channel. Consequently, the victim will be redirected to the phishing website with a similar appearance to the real website that the user knows.

Revealing the sensitive information In this step, due to the comprehensive similarity between the bogus website and the real website, the user will be persuaded to disclose their personal and financial information as he/she does to the real website. **Transfer of the disclosed information to the attacker** After disclosing the information to the bogus website by the user, the information will be sent to the attackers and the user's identity will be hijacked by the attackers.

For a successful phishing attack all of the above 4 steps must be fulfilled. Preven-

tion techniques for a single step will lead to the failure of the whole phishing attack. For this reason, several prevention methods have been proposed for each of the above steps and some of them will be addressed later on.

2.4 Anatomy of Phishing URLs

The anatomy of the phishing URLs has been the subject of anti-phishing research to protect users against new types of phishing attacks and develop new tools to combat them. Phishing URLs can be classified in the following categories [23, 24].

- *Explicit Representation:* For instance, actual URL is `https://onlinebanking.bankofoklahama.com` but the phishing URL shown in the browser bar is `http://www.zglobia.com/OKLAHOMA/index.php`.
- *Similar Representation:* For instance, legitimate URL is `www.us-bank.com` whereas the fraudulent URL is `www.usbank.com`.
- *Spoofed Representation:* URL presented in address bar of the user's browser is the same as legitimate one. For example, when user accesses to a phishing web site, JavaScript makes the visual URL the same as the legitimate URL even though the actual URL is different.

Given the above approaches, there are many different opportunities for creating phishing URLs. Following are some example techniques used for phishing on the Internet [24]:

- Adding a suffix to the domain name of a URL. E.g. change `www.citybank.com` to `www.citybank.us.com`

- Actual link which is defined in the source code of the page is different from the visible link. For example, the HTML line `<ahref=‘‘http://www.citibank.com.us.ebanking’’>www.citibank.com</a>` represents a visible link as `http://www.citibank.com` but the actual link is `http://www.citibank.com.us.ebanking`.

- Using a bug in a web application to redirect the user to the phishing web page. For example, a bug of eBay's website can be used like this: `http://cgi.ebay.com/ws/eBayISAPI.dll?MfcISAPICommand=RedirectToDomain\&DomainUrl=PHISHINGLINK` to redirect users to any website specified in the PHISHINGLINK parameter.

- Replacing similar characters in the actual link such as, `WWW.CITIBANK.COM` to `WWW.C1TlBANK.COM` that the number 1 and the lowercase of the letter L is used instead of two I letters.

- Encoding the link to disguise its true value using hexadecimal, dword, or octal encoding such as, `http://www.visa.com@%32%32%30%2E%36%38%2E%32%31%34%2E%32%31%33` which is translated into http://220.68.214.213 instead of `http://www.visa.com`.

- The URL is actually a part of an image, which uses map coordinates to define the click area. In this case an image inside a webpage has more than one link which is defined for clicking certain part of an image.

- Using a URL masking service such as `cjb.net` or `tinyurl.com`. For example `http://jne9rrfj4.CjB.neT/?uudzQYRgY1GNEn` can hide the actual URL and redirect the user to the phishing website.

- Using the "@" sign in the URL. Everything to the left of the "@" is ignored while everything to the right is considered as an address. For example, `http://www.usbank.com/update.pl@81.109.43.102/usb/upd.pl` redirects user to `81.109.43.102/usb/upd.pl` instead of `http://www.usbank.com/update.pl`.

- Using a credible looking text string within the URL. For example, `http://81.109.43.102/ebay/account_update/now.php` which uses the eBay keyword in the URL that convincing the user as it is a legitimate URL.

2.5 Phishing websites characteristics

A typical phishing website has the following characteristics [22]:

1. Usage of the famous company's logo and visual similarities of the company's website.

2. Usage of actual links to the company's real website in some part of the fake website.

3. Usage of JavaScript codes to hide the actual address or actual links of the bogus website.

4. Usage of the IP address in the URL.

5. Usage of a redirection URL service such as tiny URL.

6. Usage of a different port number rather than the default ones such as, 8080 instead of 80.

7. Usage of the SSL certificate such as, `https://www.ebay.com@212.45.20.12/ebay` to indicate security using HTTPS in order to grab the attention of the users.

8. Main functionality is to gather the user's information only without providing other services.

9. Requiring the installation of some softwares to view the website.

10. Bogus address bar which shows the real company's URL while you are on the phishing website.

11. Usage of a Pop-Up menu and immediate redirection to the company's real website after the completion of the information stealing. In such a case, the attacker informs the user that actions such as activation of a certain service will need more time to be in effect since the user may not immediately notice the changes on the real website.

12. Disabling the right click of the mouse or any other usual functionalities of the keyboard.

13. Real data processing such as credit card validation to be sure that the numbers or the information which was entered by the user is correct.

14. Usage of hidden text that can be found in the webpage's source code to avoid text-based filtering solutions.

The above characteristics are the most common ones while additional features are revealed by new phishing attacks [20].

2.6 Different types of phishing attacks

Phishing attacks can be categorized to the following types based on their structures [22]:

2.6.1 Deceptive Phishing

While the term "phishing" originated in AOL account theft using instant messaging, the most common type for deceptive phishing today is email [5]. Generally, an attacker sends bulk deceptive emails with a "call to action" on the subject that demands the recipient to click on a provided link immediately.

A typical patterns of a "call to action" contains [22]:

- A statement showing a problem with the recipient's account at a financial insinuation or other businesses. The email requests an immediate visit to correct the problem, using a deceptive link in the email.
- A statement that the recipient's account is at risk, and necessity of enrollment in an anti-fraud program provided by the institution is required.
- A counterfeit invoice for the unordered market goods, with a link to cancel the fake order.
- A deceptive notice of an undesirable change to the user's account, with a link to block it.
- A statement indicating that the free period of a new service is ended at the financial institution, and limited-time opportunity to get the service for free is

proposed to the user as the current member.

In any of the above scenarios, the directed web site collects the user's confidential information. If a user discloses confidential information to the fraudulent website, the attacker can ultimately impersonate the victim to transfer funds from the victim's account, purchase market goods, take out a second mortgage on the victim's home, file for unemployment benefits in the victim's name, or inflict other damages.
Deception-based phishing schemes include many variations [5]. With client email softwares with the support of HTML, it is easy to provide a copy of a login page directly in an email, eliminating the need to click on a link and activate the user's web browser. Sometimes, a numeric IP address is used instead of a domain name in a link to a phishing site. In such cases, it is possible to use Javascript to take over the address bar of a browser or otherwise deceive the user into believing he or she is communicating with a legitimate site. For example, using a cousin domain which is controlled by an attacker that is deceptively similar to the legitimate domain name, such as `www.mcgil1.ca` instead of `www.mcgill.ca`. Sometimes, an initial deception-based message leads to an installation of a malware during the visit of the malicious site by the users.

2.6.2 Malware-Based Phishing

Malware-based phishing [5] refers to executing malicious software on the user's machine during the phishing attack. Malware-based phishing can occur in different forms. In general, malware is disseminated either by social engineering or by exploiting a security vulnerability. A typical social engineering attack is to persuade a user to open an email attachment or to download a file which is related to pornography, salacious

celebrity photos or gossip. Some free softwares can also contain a malware. Another malware propagation method is through a worm or virus that takes advantage of a security vulnerability to install the malware, or by making the malware available on the web site that exploits a security vulnerability. It can also disseminate through spam messages from a malicious web site via social engineering, promising some interesting content at the site, or by injecting malicious content into a legitimate web site by exploiting a security flaws such as a cross-site scripting vulnerability on the site.

2.6.3 Keyloggers and Screenloggers

Keyloggers [18] install themselves either as a plug-in in web browsers or as a device driver, which keeps track of the data being input and sends it to a phishing server. Keyloggers may be implemented using different technologies such as [5]:

- A browser helper object that detects changes to the URL such as parameters and logs information when it is necessary.
- A device driver that monitors keyboard and mouse as well as the user's activities.
- A *screenlogger* that monitors both the user's inputs and the display to thwart alternate on-screen input security measures [20].

Keyloggers may collect credentials that can be triggered by the user's location and only transmit credentials for particular sites. Often, tens of such websites including financial institutions, e-commerce portals, and social networks are attacked. Various secondary losses can be caused by a keylogger. In one example, the inclusion of a credit

reporting agency in a keylogger spread via pornography spam led to the compromise of over 50 accounts with access to the agency, which in turn were ultimately used to compromise over 310,000 sets of personal information from the credit reporting agency's database [20].

2.6.4 Session Hijackers

Session hijacking [20] is an attack which monitors a user's activities with a compromised browser component. The malicious component "hijacks" the session to perform malicious actions once the user has legitimately established his or her credentials during a log-in process or upon launching a transaction. Session hijacking can be launched locally through malware or remotely as part of a man-in-the-middle attack, which will be addressed later. When performed locally by malware, it is similar to the legitimate user interaction with the targeted site.

2.6.5 Web Trojans

Web Trojans [20] are malicious softwares that appear in login screens to collect credentials. The user may think that they are disclosing information to a trusted website while the information is being captured locally, then transmitted to the attackers for unauthorized use.

2.6.6 Hosts File Poisoning

Hosts File Poisoning [5] is changing the user's DNS (Domain Name System) cache information to redirect the user to phishing websites. If a user types `www.mcgill.ca` into the URL bar, the process of DNS look up is necessary to translate that address

into a numeric address before visiting the site. Many operating systems such as Windows, have a "hosts" file for looking up host names before a DNS lookup is performed. If this file is compromised, then `www.mcgill.ca` can be changed to refer to a malicious address which shows a legitimate-looking site for the disclosure of confidential information while the user is on a phishing website.

2.6.7 System Reconfiguration Attacks

System reconfiguration attacks [5] change the configuration of the user's machine to redirect all their traffic to phishing websites. A typical example of such an attack is to modify a user's DNS servers to redirect the user to the compromised website or install a web proxy, through which the user's traffic will be passed.

2.6.8 Data Theft

Malicious codes can directly steal confidential information stored on the victim's computer. By automatically filtering the data and looking for the desired information such as passwords, account numbers, and any other data that matches a certain pattern, attackers can obtain sensitive information. Data theft is also widely used for phishing attacks aimed at corporate espionage which may result in information leakage [5], since personal computers often contain the same confidential information as trusted enterprize computers.

2.6.9 DNS-Based Phishing ("Pharming")

Pharming [25] generally indicates any form of phishing that interferes with the integrity of the lookup process for a domain name such as hosts file poisoning which

is not part of the DNS. Another form of Pharming is polluting the user's DNS cache with bogus information to redirect users to the incorrect location. This can be caused by either DNS misconfiguration or a system reconfiguration attack that changes the user's legitimate DNS server to a malicious server.

2.6.10 Content-Injection Phishing

Content-injection phishing [5] is pushing malicious content into a legitimate website. The malicious content can redirect the user to other websites such as phishing websites or install a malware on a user's computer.

There are three primary types of content-injection phishing [5]:

1. Attackers can compromise a website through a security flaw and merge the malicious content with the legitimate content.

2. Malicious content can be pushed through a cross-site scripting flaw [26] into a website. A cross-site scripting vulnerability is a programming flaw involving content coming from an external source, such as a blog, a user review of a product on an e-commerce site, an auction, a message in a discussion board, a search term or a web-based email. Those externally pushed contents can be a malicious script which is not properly validated by the server-side applications and executes on the visitor's browser.

3. Malicious actions can be performed on a site through SQL injection flaws [27] which execute database commands on a remote server. similar to the cross-site scripting attacks, SQL injection attacks are a result of improper input validation.

Cross-site scripting and SQL injection are propagated through two different ways [27, 26]:

1. **Stored attacks** in which malicious content is injected into the legitimate web server.

2. **Reflected attacks** in which malicious content is embedded into the provided URL by an attacker. Malicious contents generally appear in the argument section of the URL to a search function.

2.6.11 Man-in-the-Middle Phishing

A man-in-the-middle phishing attack [5] is a type of phishing attack in which the attacker places himself between the user and the legitimate website. The attacker stores the user's messages or submissions to the legitimate website before transmitting them to their actual destination which is the legitimate website. Then, the attacker forwards the response generated by the legitimate website to the user. Session hijacking can be performed using Man-in-the-middle attack's techniques without the necessity of storing compromised credentials. Figure 2.5 shows the structure of Man-in-the-Middle Phishing.

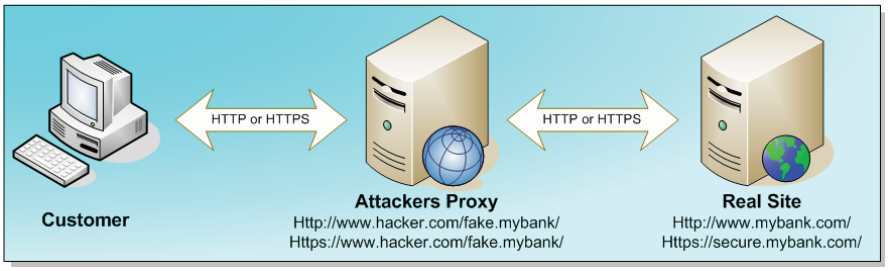

Fig. 2.5 Man-in-the-Middle Phishing structure [20].

Man-in-the-middle attacks are transparent to the users. Thus, they are difficult for a user to detect them, because the site will work properly without any misbehavior. Man-in-the-middle attacks may be used with many other types of phishing, including DNS-based phishing and deception-based phishing. This will augment the effectiveness of the phishing attack. Generally, a SSL protected website will not be vulnerable to a Man-in-the-middle attack. The handshake used by SSL ensures that the session is established with the website whose name is indicated in the server's certificate, and that an attacker cannot obtain the session key; SSL traffic is encrypted using the session key so it cannot be decoded by the Man-in-the-Middle. However, a malware-based attack can change a system configuration to install a new trusted certificate authority, in which such an attacker in a middle can create its own certificates for any SSL-protected site, decrypt the traffic and extract confidential information, and re-encrypt the traffic to communicate with the other side. Man-in-the-middle attacks can also compromise authentication credentials, such as one time or time sensitive passwords generated by hardware devices that can be used by the attacker for authentication as long as they remain valid.

2.6.12 Search Engine Phishing

Search Engine Phishing [5] refers to an attack in which attackers create web pages for invalid products that will be indexed by search engines. Then, they wait for users to visit the bogus website through searching the products in search engines and disclose their confidential information as part of an order. Such websites generally offer products at an enticing price. Fraudulent banking [5] is another type of Search Engine Phishing which refers to the indexed bogus websites that advertise an interest

rate slightly higher than any bank. Users find them through search engines and disclose their bank account information for balance transfer to the new account that they have created in the bogus website. Greed is a powerful tool in the hand of attackers that can cloud judgment.

2.7 General phishing attack scenario

The fundamental flow of information in a typical phishing attack based on both the described steps in section 2.3.4 and different types of phishing attacks described in section 2.6 can be presented as follows [22]:

1. The attacker prepares for the attack. For certain types of attacks, such as deceptive attacks using cousin domains [5], a domain must be registered by the attacker and the websites must be owned or must have been compromised by the attacker. The website is configured to receive information whether from a user or from malware.

2. A malicious payload reaches the user through some dissemination mechanisms. In a deception based phishing attack, the payload is typically a deceptive email. In a malware or system reconfiguration attack, the payload is malicious code in the form of an attachment to an email or an unintended software patch. Fake IP addresses are the payloads in DNS poisoning attacks. The payload is a search result referring to a fraudulent website in search engine phishing. In the case of a cross-site scripting attack, malicious code that is stored on a trusted website or embedded in a URL in an email is the payload depending on the attack details [5].

3. The user chooses an action that makes him or her susceptible to information disclosure. In a deception based phishing attack, the user clicks on a deceptive link. In a keylogger attack, the user visits a legitimate website. In DNS poisoning attacks, the user visits a legitimately-named website that is redirected to a bogus one.

4. The user is asked for sensitive confidential information, either by the visited website or locally by a malware.

5. The user disclosed sensitive information such as a passwords, either by providing it to a visited website, to malicious software executing on the local machine, or to a software which is used in the Man-in-the-Middle Phishing attack.

6. The confidential information is transmitted to the attacker. Depending on the type of the attack, this information can be sent by compromised websites, or in the case of locally executed malware such as a keylogger, The sensitive information can be sent by the user's computer.

7. The confidential stolen information is used to impersonate the user in the web.

8. The attacker obtains illicit money by using the stolen confidential information.

Each step in the phishing information flow is important to address and should be studied in order to find a solution which stops phishing in each steps.

Chapter 3

Related Work

3.1 Introduction

As we discussed earlier in Chapter 1 ,phishing is one of the cyber attacks that is widely used by attackers nowadays and has imposed a high financial loss to companies. Thus, the necessity of effective solutions for phishing prevention is an important issue. In this chapter we will examine current solutions along with their advantages and disadvantages in the battle against phishing.

3.2 Categorization of phishing prevention techniques

As phishing is an important issue that should be deeply studied, different methods have been proposed for phishing prevention. The main characteristics of these methods are their adaptation to the actual internet architecture without substantial need for change in it. In general, phishing prevention techniques can be categorized into 3 categories [22]:

- Restricting the reception of phishing scams by service providers.
- Restricting access to the phishing websites.
- Restricting interaction of users with phishing websites when they are visiting these websites.

In these techniques, proposed solutions are categorized into three categories [5]:

- Detection based solution.
- Prevention based solution.
- Correction based solution.

In this thesis, we propose a novel approach that integrates all of the required steps as described above.

3.3 Detection based solutions

Proposed solutions in this category are used after the phishing has occurred. In other words these solutions only run when they encounter some evidence of a phishing attack. Detection mechanisms are categorized into 5 main categories [22]:

1. Restricting the reception of spams and scams.
2. Detecting phishing websites similar to the most known websites.
3. Detecting malicious softwares.
4. Content filtering.
5. Monitoring account life cycle.

We will investigate important solutions in the above category in this section.

3.3.1 Restricting the reception of spams and scams

Spam is the abuse of electronic messaging systems (including most broadcast media, digital delivery systems) to send unsolicited bulk messages indiscriminately [28].

Scams [5] are the subset of spams that are not considered as important as spams. Although there are some solutions that detect spams, there are some issues which prevent the usage of spam filter solutions to detect scams effectively. Scams are sent to a small number of people in comparison to spams. This factor prevents the detection of scams automatically based on the email header. The attackers also send scams through different routes within a large period of time in order to avoid the detection of scams using spam filters.

3.3.2 Content filtering

In these solutions, the content of the visited websites and the information that the user tries to enter as inputs to these websites is analyzed. Based on the suspicious activities that may be encountered, the user will be informed or his/her access to these websites will be blocked. Content analysis can be done automatically in client side programs like SpoofGaurd [29] or can be done by other service providers that prepare some sort of black list repository which helps the users to know whether a website is phishing or not. Such solutions are primarily based on the users' report before doing any type of content analysis on a given website.

3.3.2.1 Client Side Methods

Client side methods [5] are the easiest way of detecting phishing. In these solutions additional information about the visited websites is shown to the user and the user decides by himself/herself ultimately if the visited website is phishing or not. To evaluate the client-side solutions, we introduce some of the selected ones as follows:

eBay toolbar [30] can identify if any particular URL is trying to phish `ebay.com`.

However, this toolbar does not work if any URL is trying to phish any other website since it is only dedicated to the eBay websites.

Microsoft and AOL use a *Black List Approach* [31, 32] by integrating black listing features into their browsers. It blocks users from entering any information while they are at a known phishing website. However it is ineffective since there is always a window of vulnerability during which the user is susceptible to attacks as described in Section 2.5.

PhishingGuard [5] makes the use of a white list approach. Since a website is identified by its IP address, a white list is generated by keeping the ip address of trusted URLs. Whenever any website is visited, the white list is checked in order to find corresponding information about it. If the URL is found in the white list, but the IP address of the founded URL is different from the ip address of the visited website, then this URL is classified as phishing. This tool is also capable of detecting pharming.

Bayesian Anti Phishing toolbar [33] is a Mozilla Firefox extension designed to help users identify phishing websites and protect their sensitive information. It also uses a white list based approach. For a given website, the Bayesian Anti Phishing toolbar sends the URL to the DOM analyzer [33]. The DOM analyzer then checks if the given URL is in the white list. If it is in the white list, it means it is a legitimate website. If it is not, then the DOM analyzer tokenizes the content of a given website and then forwards the token to the scoring module. This scoring module consults the database of tokens and their values, and then computes the final score. If the score exceeds a pre-set threshold, then the URL is classified as phishing.

SpoofGuard [29] is another example of a client-side toolbar designed and de-

veloped at Stanford University to prevent phishing attacks. *SpoofGuard* is a browser plug-in that uses domain names, URLs, and image checks to determine if a given page is a part of a phishing attack. It applies the following three tests to a given page and then combines the result using a scoring mechanism [34]:

1. Stateless methods are used to determine whether a downloaded page is suspicious.

2. Stateful methods are used to evaluate a downloaded page in light of the history and cache of the user's browser.

3. Stateful methods are used to evaluate outgoing HTML post data.

The final score determines whether SpoofGuard should alert users and also determines the severity of the alert. The false positive rate relies on how frequently the user creates new accounts or clears the browser history cache. If the user opens a new account, and uses the same password for another account, this will produce a false positive alarm.

Some of the tests compare passwords that are used at a particular site to passwords that were used at previous sites. An attacker could fool these tests by breaking the password field (on the spoofed page) into two adjacent fields, that would look contiguous, but would cause the test to fail. Some of the tests compare images on the phishing websites to the images that appear on legitimate pages. An attacker could circumvent these tests by slicing an image into adjacent vertical slides and presenting these slices next to each other.

CANTINA [1] uses a content-based approach to detect phishing websites. It combines a term frequency-inverse document frequency (TF-IDF) algorithm with other heuristics to determine whether a given website is phishing. It uses five words with the highest TF-IDF weight on a given website as a signature and then submits those five words to the Google search engine. If the URL of the site is found within the top results, then that URL is classified as a legitimate URL, otherwise it is classified as phishing. An attacker could circumvent CANTINA using several techniques. One technique would be to use an image instead of words in a given page. Another technique would be to add invisible text, text that is tiny or that matches background color of the page. Yet another technique would be to change a lot of words in order to confuse TF-IDF.

Spoof Stick [35] is a toolbar which is installed as a plug-in in both Internet Explorer and Mozilla Firefox. It examines the basic information of the visited domain and the visited website. For example, if the user is on eBay's website, this tool bar shows the "You are on eBay.com" as a message to the user. But if the user goes to the website whose address is based on an IP address, it shows "You are on 192.197.121.19" as a message to the user. However, the user must decide by himself whether the visiting website is phishing or not since the Spoof Stick only provides more information about the visiting website, it does not block the access.

We can see an example of Spoof Stick when it is launched against a real eBay website:

Trustbar [36] is another toolbar solution based on digital signatures. Trustbar uses SSL to verify whether the digital certificate of a given website is valid or not. If the digital certificate of the visiting website is valid, trustbar shows the logo of the

Fig. 3.1 Spoof Stick Usage.

visiting website along with the logo of the certificate issuer in its toolbar to the user. Since most phishing websites are not using a valid SSL certificate, trustbar shows nothing when the user visits those websites and this indicates phishing.

Figure 3.2 shows an example of Trustbar when it is used in the paypal website:

Fig. 3.2 SpoofGuard Usage.

3.3.3 Browsers's cache and history based solutions

Browsers cache and history based solutions [5] try to encrypt all the sensitive information that is entered as inputs to a website by users. These solutions capture the domain name and the IP address of the sites to which the sensitive information has been sent. Whenever a user visits a website and tries to enter the same information that has been previously sent to another domain, these solutions inform the user that the entered information has been used on another domain before; so that the user

can evaluate by himself/herself to see whether the distinguished website is phishing or not. Web wallet [37] and AntiPhish [38] are examples of such a solutions.

AntiPhish [38] is a Mozilla based plug-in that aims to protect inexperienced users against phishing attacks. AntiPhish keeps track of the sensitive information of a user and generates a warning whenever the user discloses sensitive information into a form on an untrusted website . Trusted websites must have been defined by the user to AntiPhish prior to its usage. Most of the browsers such as Mozilla Firefox have the functionality that allows form content to be stored and automatically inserted if the user desires. AntiPhish takes this common functionality one step further and also tracks where this information is sent. It also stores the mapping of where this information belongs to. However effectiveness of this approach is dependent on the user. Web wallet uses the same approach. Figure 3.3, 3.4 and 3.5 are examples of usage of AntiPhish and Web wallet.

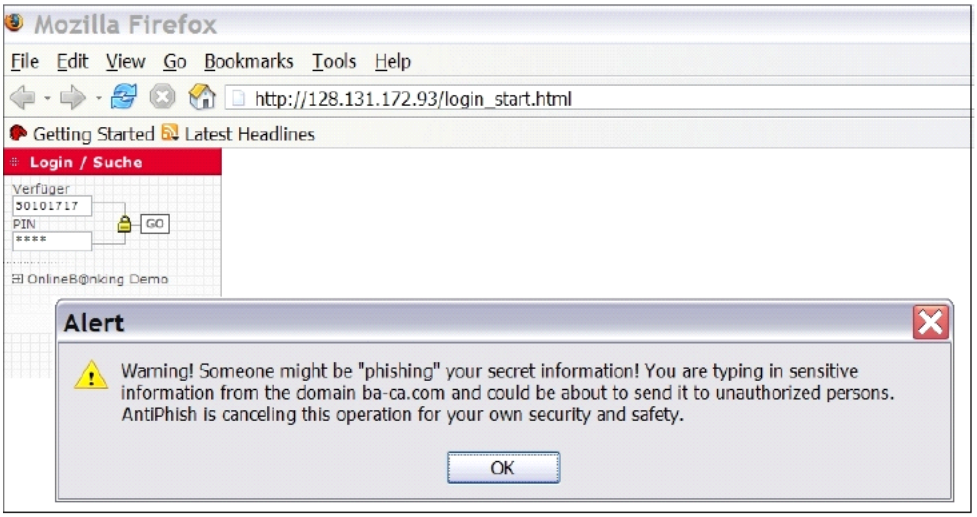

Fig. 3.3 Antiphish Usage.

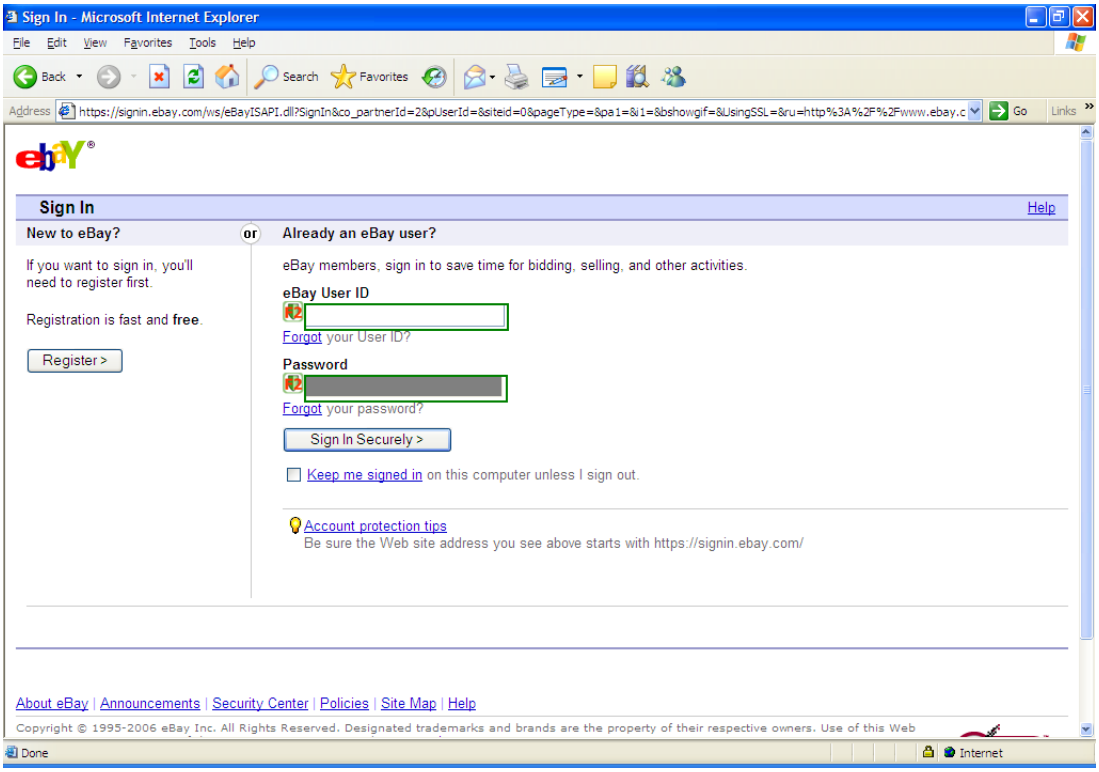

Fig. 3.4 Before Web wallet Usage.

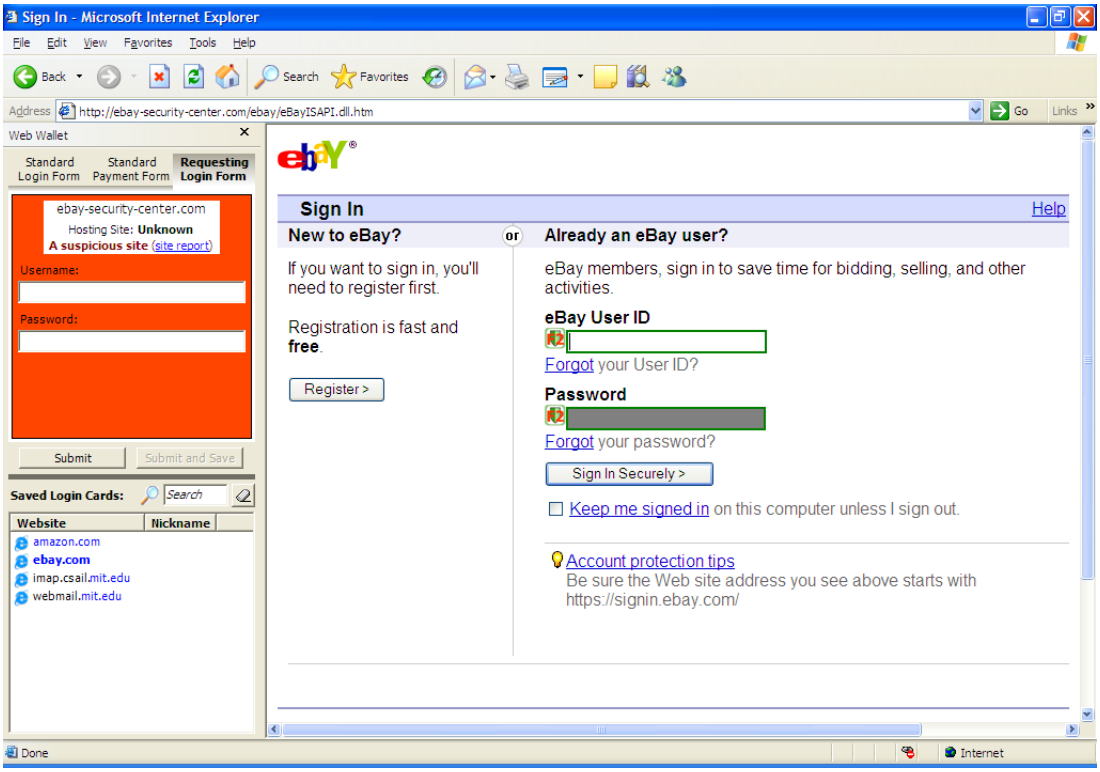

Fig. 3.5 After Web wallet Usage.

3.3.4 Central Black listed solutions

Different mechanisms are used in order to build a black list repository of phishing websites. Reports by users are the main source of constructing such a list. Giant service providers such as Google [39] or Cloudmark Security Network [40] provide ways that users can report a phishing or a malicious websites whenever they encounter one. Another solution for building up a black list is to combine the user's opinion about a given website and checking that website automatically to find phishing symptoms such as different port numbers, usage of IP addresses, URL in unicode or hexadecimal format. Also, the small lifetime of phishing websites [20] are another important characteristic that can be considered to build a black list.

After detecting phishing websites, they will be black listed by the black list providers and users can access their repository to know whether a website is phishing or not. Also, black list providers sometimes provide a client side application that is installed as a browser's plug-in which checks any queried URL against the black list repository to identify whether it is phishing or not. The access of users to these websites is granted if these websites are not published in the repository. Otherwise browsers prevent the access to the identified phishing website and users receive a warning about the website that they are going to visit.

We will explore some of these black list solutions that are offered by their providers as a service to their users.

Account guard It is a tool bar provided by the eBay [30]. It shows the black listed websites with the usage of traffic light signals. It has three states:

- Green is used if the user is visiting a website which is related to eBay or PayPal and is trusted by them.

- Gray is used If the user is visiting a website which is not trusted by eBay or PayPal and has not been identified as a phishing website in the black list.
- Red is used if the visiting website is in the black list which means that it is a phishing website.

Account Guard also provides some reporting mechanisms for users to explore the phishing black list about particular list of URLs.

Sanitizing Proxy System The main idea of the Sanitizing Proxy System (SPS) [41] for phishing detection is to omit the user's access to certain sections of the page which needs user's information as inputs. If a phishing website does not have a form object for entering information, then users can not enter any sensitive information which may led to stealing of their personal information. SPS can be used either in a proxy server, a firewall, or as a plug-in for the user's browser.

SPS uses a black list solution for its functionality. If a visiting webpage is in the black list, then SPS disables any input forms and any suspicious malicious JavaScript codes of the webpage.

Figure 3.6 shows a phishing webpage before the activation of the SPS:

Fig. 3.6 A Phishing Website Form.

However, figure 3.7 shows the same page after the activation of the SPS

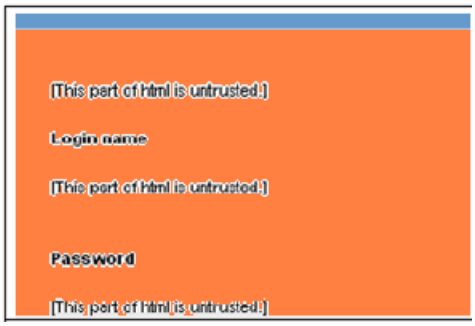

Fig. 3.7 SPS Usage on the Phishing website.

3.3.4.1 Google Safe Browsing

Google toolbar provides a service called Google Safe Browsing [39] that can be used only as a plug-in for the Mozilla Firefox browser to detect phishing websites based on the central black list repository provided by Google. Google uses some known phishing symptoms as well as different black list repositories in order to update its own black list repository. Also it considers users' reports about phishing websites to make its own black list up-to-date against zero-day attacks that may not have been detected by other blacklist repositories.

Figure 3.8 shows Google Safe Browsing service against a phishing website.

3.3.5 Detecting phishing websites similar to the most known websites

These solutions try to identify if the content of phishing websites is similar to the content hosted on the famous websites.

Site Watcher [42] is an example of the solutions that detects content similarities to the ones at famous websites and informs the users when they want to visit those websites. In these solutions, users should introduce the websites which are required to be pro-

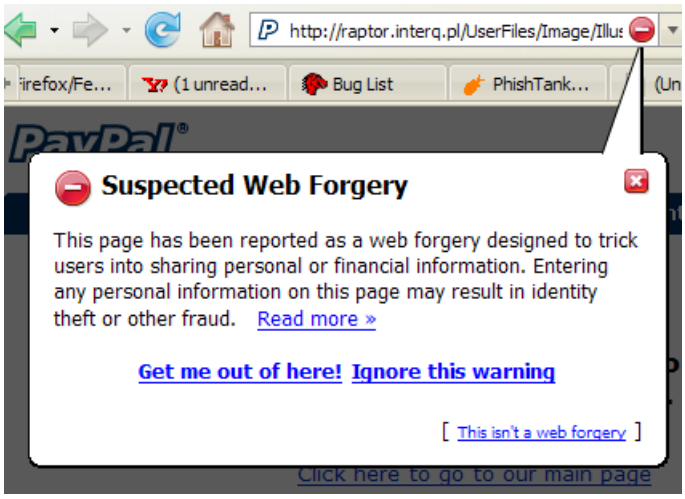

Fig. 3.8 Google Safe Browsing Usage .

tected against phishing to the software. Then, all the characteristics of the designated websites based on their content such as texts, pictures, page styles, and templates used, are fed into the system automatically by the software. The system then navigates through all the user's email and investigates all the pages that the URLs inside the emails are referring to. If the system finds any similarities to the famous websites that have been introduced to the system for protection, then it considers it as a phishing website and eliminates the corresponding email from the user's inbox and informs the user.

The textual content similarities that these solution uses is based on keyword similarities between the two webpages.

3.4 Prevention based solutions

These are solutions that prevent phishing. In contrast to the solutions in the detection category that are used after the phishing takes place, these solutions help users in

battling against phishing attacks.

The best known solutions in this category are [22]:

1. Authentication
2. Email Authentication
3. Patch Management
4. Web Application Security

3.4.1 Authentication

One of the fundamental flaws that helps phishing attacks, is not being able to identify correct users. Authentication is a one-way solution where only users can give their identities to the web providers and web providers cannot identify users in advance. Websites generally use digital certificates to prove their identity to users. Users often are not aware of the digital certificates and do not know the SSL properties either. Thus, attackers can easily deceive them by providing a visually similar website.

Using *visual customization* [43] is one way to protect users against phishing. In these solutions the Log-in page of each user is changed based on the layout that the user chooses for himself/herself in a certain time period. For example, the user can put his/her own favorite image as a background of log-in page. This makes it difficult for the attackers to build a replica of the website for such a user. If the selected template does not appear in the log-in page, the user will be suspicious and will not disclose any information to the suspected website.

However, this solution does not work against phishing attacks based on the DNS poisoning man-in-middle attack, since those templates can be detected and replaced by

the attackers.

Dynamic Security Scheme [43] is a solution that has incorporated the Secure Remote Protocol (SRP) to address the above issue. In this solution a toolbar is installed on the user's browser. Service providers produce images based on the user's password and then send it back to the user as a background of their current website or the background of the user's log-in page. Then the user generates an image based on its own password using the corresponding toolbar. Finally, the user compares those two generated images to see whether they are the same or not. Similarity between those two images informs the user that the service provider knows its password and can be considered as trusted. However, it may not protect the users against man-in-middle attacks completely.

The following figures show an example of the generated image by both the toolbar and the service provider.

Fig. 3.9 Picture generated based on the user's password which appears as the background image of other subsequent web forms .

There are other solutions that have used the same idea but in different ways to help users against phishing attacks.

Some toolbars use the idea of Passpet [44] that asks the user to assign a label to any trusted websites that he/she is willing to visit. Whenever a user visits a website that has been already assigned a label by the toolbar, the toolbar shows the label; so that

Fig. 3.10 The image generated by the service provider.

the user can identify the phishing websites, since phishing websites cannot generate the same label. Other solutions use hashes [45] to generate a new password based on the user's password and the address of the visiting website. Since the phishing websites have different addresses from the trusted websites, the generated password would be useless and the attackers only receive a password which is not the user's password. However, none of those solutions can tolerate the DNS poisoning Man-in-Middle attacks which can switch the address of a phishing website with the address of a real website.

Two Factor Authentication [46] is another example of such solutions to protect users against phishing. In this solution two different factors are used to designate a user's identity. Users need their passwords and their mobile devices in order to access the websites. In these solutions such as [46], service providers send a one-time password to the users' mobile devices through SMS. Service providers have already been informed by the users about their mobile phone numbers. Since attackers do not have access to the mobile network and users' mobile phones, they cannot access the generated password.

Also [46] has introduced another approach that uses a hardware token which generates a different six digit number every 60 seconds. Generated numbers are different for each user and can be used as a password which expires after 60 seconds and is useless if the attackers use them after they expire. This solution also cannot tolerate Man-in-Middle attack which allows attackers to use the password within the 60 seconds of their effective period.
There are some other solutions that are not susceptible to the man-in-middle attacks such as [47] which requires users to install hardware on their computers, which is not cost effective.

3.4.2 Patch Management and Web Application Security

Patch Management and Web Application Security [5] are the most important solutions in the battle against phishing. The attackers use the benefits of flaws in web applications in order to deceive people. However, these two solutions help users to identify any bad behavior of a website. They are also based on some sort of black list repository that helps their providers to know the flaws; so that providers can prepare a patch or an update to prevent malicious activities and remove the flaws.

3.5 Correction based solutions

These are solutions that can be used after the occurrence of a phishing attack in order to decrease the damage of such an attack [5]. There are two types of solutions in this category:

- Phishing Site Takedown [48].
- Forensics Investigation [20].

3.5.0.1 Phishing Site Takedown

To protect other users from phishing attacks, certain groups and organizations have been founded like APWG that disables the phishing websites after their detection and removes such websites from the internet. However, it is not easy to take down the phishing websites since a large number of them are hosted in different countries. These anti-phishing organizations do not have capability to disable such phishing websites because it requires international arrangements and requires arduous effort.

3.5.0.2 Forensics Investigation

After a phishing attack takes place, researchers try to analyze and understand them by investigating the techniques that have been used in them. This research can be used in the new solutions against phishing. It also helps investigators to know the possible threats of those attacks in order to measure the extent of the damage caused by them and to efficiently inform the users about it.

Chapter 4

CASTLE: A Social Framework for Collaborative Anti-Phishing Databases

4.1 Design Objectives

CASTLE is a decentralized framework for building and maintaining databases useful for anti-phishing services. It is a collaborative approach using social factors, content, and location for the evaluation. For instance, browser-based toolbars lookup centrally maintained databases to determine whether a given location is safe or not. If a site is previously black-listed as phishing, the browser-based toolbar will prevent the user from visiting it. Although the centralized databases are already in place for anti-phishing, their scalability is a cause for concern. In particular, the existing approaches do not handle the following issues in a scalable manner.

- *Enrollment process:* The name space that can be phished is vast. Few repositories cannot handle complaints for the whole name space. Because phishing is usually a impersonation attack against well known Internet brands, there is a concern of brand tarnishing. Therefore, valuable brands would want to take ownership of their namespace.

- *Decision process:* With few central repositories, administrators of those repositories should use automated tools and manual examinations to decide on phishing reports. Because a repository is likely to receive many reports, some of which being completely new, the decision process is an arduous one. If the namespace is partitioned based on URL prefixes, the decisions can be made by repositories that are familiar with URLs. In CASTLE, the social factor is used to reduce namespace grabbing attacks.

- *Lookup process:* One of the advantages of central repositories is the straightforward lookup process for determining whether a URL is safe or not. This advantage will be lost if multiple repositories need to be checked. A distributed network of repositories need to have efficient lookup process.

All proposed solutions that we discussed in Section 3 use either URL analysis or content analysis. We believe that these two factors should be used together in order to build a robust anti-phishing solution. While URL analysis is useful in detecting whether a URL name (i.e, domain) is banned due to phishing incidents, It is not sufficient to detect all phishing sites. This is because some phishing sites can use plain IP addresses or unknown URL names. Consequently banning all IP addresses or unfamiliar URLs is not a feasible solution. To address this problem, the URL analysis can be augmented with a content-based analysis. Thus, our proposed solution, CASTLE, is based on both URL and content analysis.

The CASTLE framework provides a novel decentralized peer-to-peer approach for maintaining anti-phishing databases for the Internet. It uses the social factor to limit the membership to the peer-to-peer network. This is necessary to prevent the attackers from locating their verification servers in the network of servers maintained

by CASTLE. Further, CASTLE admits location-based (URL-based) and content-based access to the repositories.

The key feature of CASTLE is that it uses multiple phishing verification servers (PVCs) to determine if the queried URL is phishing. The social network of PVCs would be protected using SybilGuard protocol in order to limit the corrupted influences of a Sybil attack where an edge between two PVCs indicate a human-established trust relationship [49].

In case of a lack of information about a queried URL at a certain PVC, the content analysis module will be employed to help finding the PVC responsible for the queried URL.

4.2 Overall Design

Figure 4.1 shows the overall system architecture of CASTLE. A host on the Internet contacts one of the *PVCs* and launches its search to the connected *PVC* in order to find whether the queried URL is phishing or not. Connected PVCs will run the routing process to find out the responsible PVC for the queried URL. After finding the responsible PVC, a decision will be returned by the PVC back to the user. If there is no information about the particular URL in the responsible PVC, the queried URL will be returned as a suspected URL and one or more URLs inside the white list of the responsible PVC that closely match the queried URL will be recommended to the user as a trusted substitute for the queried URL.

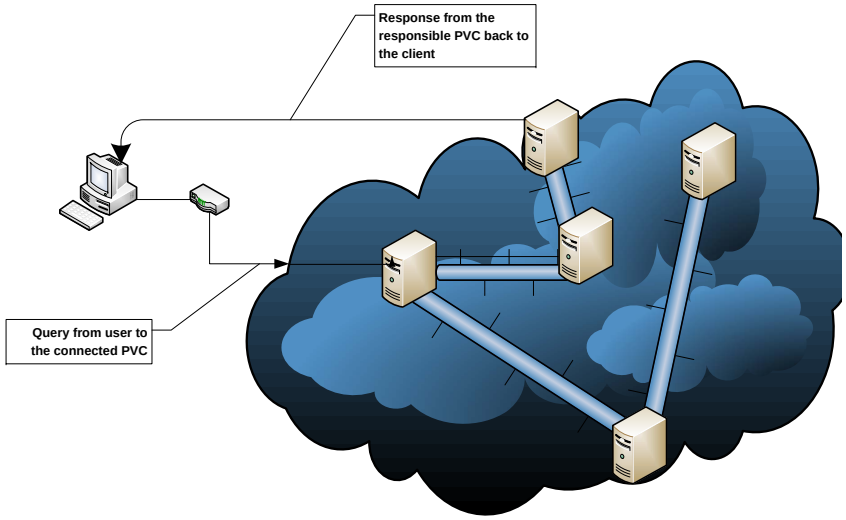

Fig. 4.1 The overall architecture of the CASTLE.

4.2.1 Architecture of PVC

A PVC is responsible for a certain domain or domain prefixes; however each PVC can handle either one or more domain prefixes. For example a PVC could be responsible for `cn*.com` instead of `cnn.com` which indicates that it will handle all URLs with the prefix *cn* such as `cnn.com` and `cnet.com`. In case of a single domain, the PVC is only responsible for that domain.

The information regarding other PVCs or the status of a URL is kept on certain lists. Each of the PVCs have two main tables:

- Decision table
- Routing table

PVCs will be queried based on the domain name they are responsible for inside the network of PVCs. A decision table is used to identify the status of a URL in a

particular PVC. Essentially, it shows whether a related URL to the PVC is phishing or not. Construction of the decision table is based on the content analysis module and administrators of the PVCs. The content analysis module identifies whether the queried URL has content similar to the one hosted at the trusted URLs of the PVC or not. If the URL returned by the content analysis module is similar to the URL handled by a certain PVC inside the network, then the content analysis module will put that URL inside the gray list of that PVC which contains the list of possible phishing URLs that should be investigated by the administrators of PVCs. Migration from gray list to other lists such as black list or white list will be done by the administrators of a PVC. Administrators of a PVC will investigate the gray list entries very carefully and change their locations to the appropriate list based on their relation to the PVC. This process can be done semi-automatically or manually.

The decision table has the following data structure which is based on the status of a URL in the PVC. It contains 5 lists as follows:

- Black list which contains the list of phishing URLs.
- Archived Black list which contains the list of phishing URLs which are more than 5 days old.
- White list which contains the list of all trusted URLs by the PVC.
- Gray list which contains the list of all possible phishing URLs identified by the content analysis module.
- Unknown list which contains the list of all URLs whose contents are similar to the content of the domain handled by one of the PVCs and corresponding domain name handled by that PVC.

4.2 Overall Design

All links at a PVC are searched to find the appropriate decision about a particular URL. As mentioned in Section 2.3.3, the average up time for phishing websites is around 4.5 days [20]. So, to prevent the blacklists getting larger with expired phishing URLs, we have time to live (TTL) values (counter TTL) for each entry in the blacklist. The value of TTL will increase by one every day. Each time, a blacklisted URL or blacklisted IP address is visited, the TTL for this URL or IP address would be reset to zero. If the value of TTL reaches 5, that entry is moved to the archived blacklist. This will help us to provide a better response time since the system will revise itself based on the ongoing phishing attacks and give higher priority to the recent phishing attacks in comparison to the old ones. After a search is performed on the black list appropriate decision is given. If we could not find any information about the queried URL, we would inform the user that the URL is not certified which means that it is neither good nor bad. However, we would continue searching on the archived black list in the background. If the queried URL is found in the archived black list, we move the queried URL to the black list. Due to the small size of the blacklist in contrast to the archived black list, response time would be lower if the URL is present in the blacklist.

Each routing table contains pointers to other PVCs by keeping a list of IP addresses of other PVCs. Each PVC would have the privilege of adding or removing entries to its own routing table. It would also be able to update its own routing table by contacting the other PVCs with whom it is socially connected.

4.2.2 URL Routing Mechanism

The routing table is to push the queries through the P2P network of PVCs. We use only the domain name and TLD parts of a URL. Therefore, the structure of the routing table is based on two arrays that help to determine the domain name and the TLD of the current PVC. These arrays are:

1. A one-dimensional array of size 270 called TLDArray which stores the list of all possible top level domains (TLDs) known by the PVC [50].

2. A two-dimensional array called DNArray which stores allowed characters of the domain name of the URL as well as null character.

All domain names are stored with a null character at the end. For example *cnet* would be stored as c, n, e, t, and a null character at the end. The structure of the routing table is shown in Figure 4.2.
The first row in this routing table represents the TLDArray which points to alternate PVCs. For example the PVC which is responsible for `mcgill.ca` contains a pointer to all other PVCs which handle domain names with different TLDs than the TLDs of domains handled by the current PVC.

The rest of the rows represent DNArray. The first row of the DNArray represents the first letter of the domain name handled by the PVC. It also contains pointers to other PVCs whose first letter of their domain names is not the same as the first letter of the domain name handled by the current PVC. The second row of the DNArray represents the second letter of the domain name handled by the current PVC along with pointers to other PVCs whose first letter of their domain names is the same as

the first letter of the domain name handled by the current PVC and their second letter of their domain names is not the same as the second letter of the domain name handled by the current PVC and so on. The DNArray will contain 64 rows which limits the size of the domain name to 63 character, the maximum of the domain name length [51]. If the URL contains IP address, the IP address would be stored in hex format in order to minimize the space usage. Null character in a row finalize the domain name of the PVC.

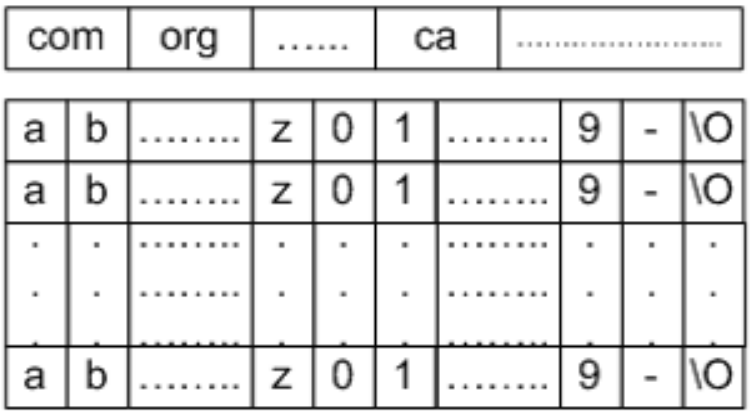

Fig. 4.2 The structure of the routing table

Figure 4.3 shows an example of the routing table of a PVC handling `cnet.com`. The first row represents the TLDArray containing the list of all possible TLDs, while the rest of the rows represent the DNArray containing all possible characters of a domain name. The entries in the first row (in bold) represent the TLD, which is handled by the PVC whereas entries in other rows (in bold) represent the character of domain or domain prefixes handled by this PVC. All the entries can point to other PVCs. Figure 4.3 shows the list of entries to which the PVC handling `cnet.com` points to.

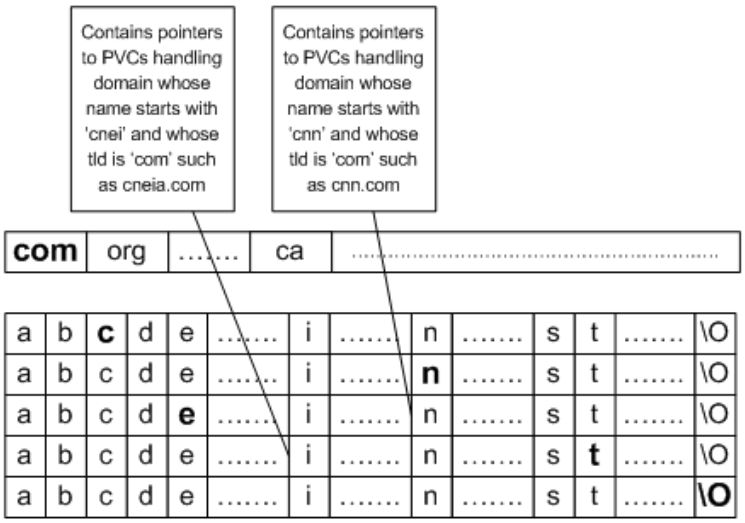

Fig. 4.3 Routing entries for PVC handling cnet.com.

The mechanism for inserting pointers to other PVCs in the routing table is defined as follows:

Entries containing the IP addresses of PVCs handling domains with different TLDs than the TLD of the domain name handled by the current PVC will go to the first row of the routing table in the appropriate position. Entries containing the IP addresses of PVCs that handle the domains with the same TLD as the current PVC and share any prefix with the domain name handled by current PVC will go to the appropriate rows of the routing table. Figure 4.6 shows an example of what happens when we add the new entry `www.cneia.com` to the PVC which handles `www.cnet.com`.

Pseudocode for the algorithm for inserting entries into the routing table is as follows:

Insertion(URL, IP address of PVC)
host $\leftarrow$ *domain name of URL*
domain_tld $\leftarrow$ *tld of domain handled by this PVC*

length ← *length of host*
tld ← *tld of URL*
domain ← *domain of URL handled by this PVC*
determine i such that $TLDArray[i] = tld$
if $domain\_tld \neq tld$ AND $url \neq ip\ address\ format$ **then**
  $ip\_add\_list \leftarrow TLDArray[i]$
else
  for $i = 1$ to length **do**
    find j where DNArray[i][j] = host[i]
    if $host[i] \neq domain[i]$ **then**
      $ip\_add\_list \leftarrow DNArray[i][j]$
    end if
  end for
end if

The Figure 4.4 shows how we route to the PVC handling `cneia.com` from the PVC handling `visa.org` assuming `visa.org` routing entries are as shown in Figure 4.5.

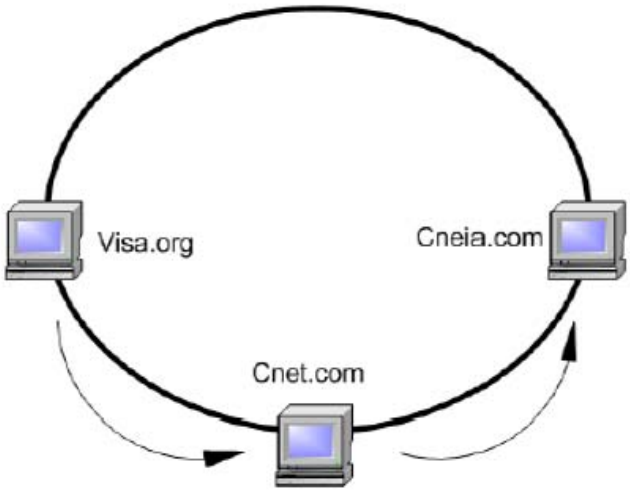

Fig. 4.4 A hypothetical example showing how routing takes place from the PVC handling visa.org to the PVC handling cneia.com. In this case, we assume the search starts from visa.org.

This URL routing mechanism would help us determine whether the queried URL is phishing. If none of the PVCs handle the queried URL, then this mechanism would route to the PVC which handles domains whose larger prefix matches with the queried URL. Afterwards, we will check if the queried URL is present in one of the lists in the target PVC as described in Section 4.2.1. If it is present in the black list, white list or gray list, the user would be informed about the decision based on the lists in which it is present. If it is present in the unknown list, the corresponding URL would be extracted and we would then route to the PVC handling that corresponding URL in order to find out about the appropriate decision. If it is not present in unknown list, content analysis would be performed, which would return a URL called *CA URL* whose content is similar to the content of the queried URL.

If the *CA URL* is similar to the URL handled by the current PVC, then that URL would be put into the gray list of the current PVC. Then, the owner of the PVC handling the URL with similar content as the content of the queried URL, would have an option of putting that URL in the appropriate list. If the *CA URL* is different from the domain handled by the current PVC, then the queried URL would be put into the unknown list of the current PVC which indicates content similarity to the current PVC rather than URL similarity and we would route to the PVC handling that *CA URL*. After reaching the PVC that handles the *CA URL*, the queried URL would be put into the gray list of that PVC.

In this situation, there would be two PVCs in the system handling such a URL:

- A PVC for content similarity

- A PVC for URL similarity

This will help us to notify both the PVC whose address is similar to the phishing URL and the PVC whose content is similar to the phishing website but the URL is not. So, both PVCs will investigate the suspected URL and determine an appropriate status for the phishing URL in their decision list.

The search process is one of the fundamental components of the CASTLE that helps PVCs to find the responsible PVC for the given URL in the network. Pseudocode for the search algorithm is as follows:

{Whenever search is initiated, the function search(0, URL, IP address of current PVC) is called. It returns 1 if the URL is phishing, 0 if the URL is suspected phishing and -1 if it is not phishing}
search(index, CA_URL, URL, ip address of current PVC)
if $CA\_URL = NULL$ **then**
  $CA\_URL \leftarrow URL$
end if
$host \leftarrow domain\ name\ of\ URL$
$domain\_tld \leftarrow tld\ of\ domain\ handled\ by\ this\ PVC$
$tld\_domain \leftarrow tld\ of\ domain\ handled\ by\ this\ PVC$
$host \leftarrow domain\ part\ of\ url$
$domain \leftarrow domain\ part\ of\ url\ handled\ by\ this\ PVC$
$tld \leftarrow tld\ part\ of\ url$
determine i such that $TLDArray[i] = tld$
if $tld\_domain\ handled\ by\ this\ PVC \neq tld$ AND $url \neq ip\ address\ format$ **then**
  $ip\ address \leftarrow randomly\ extracted\ ip\ address\ from\ TLDArray[i]$
  search(i, NULL, URL, IP address)
else
  for $i = index$ to length **do**
    find j where DNArray[i][j] = host[i]
    if $host[i] \neq domain[i]$ **then**
      $ip\ address \leftarrow randomly\ extract\ IP\ address\ from\ DNArray[i][j]$
      if $ip\ address = null$ **then**

```
        if CA_URL is present in white list then
          return 1
        else if CA_URL is present in gray list then
          return 0
        else if CA_URL is present in black list then
          return -1
        else if URL is present in unknown list then
          CA_URL ← extract corresponding URL
           search(0, CA_URL, URL, IP address of current PVC)
        else
          perform content analysis model and get CA_URL
          search(0, CA_URL, URL, IP address of the current PVC)
        end if
      else
        search(i, NULL, URL, ip address)
      end if
    end if
  end for
end if
```

Suppose we are looking for the URL `cneibbc.com` whose content is similar to the content in `rbc.ca` and currently no PVC has an entry for `cneibbc.com`. Assume `cneia.com` routing table entries are as shown in Figure 4.6. Figure 4.7 shows how routing would take place through PVC handling `cneia.com` in such a case.

Notice that an entry will be inserted to `cneia.com` which shares the longest prefix with the URL `cneibbc.com` in its unknown list and due to the content similarity to `rbc.ca` another entry regarding the URL `cneibbc.com` is inserted into the PVC which is responsible for `rbc.ca`.

In order to describe the routing mechanism in more detail, we will provide more examples in the following section.

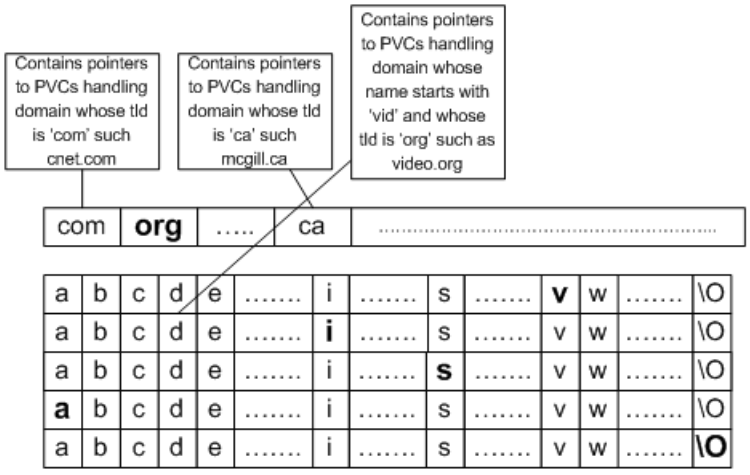

Fig. 4.5 Routing entries for PVC handling visa.org

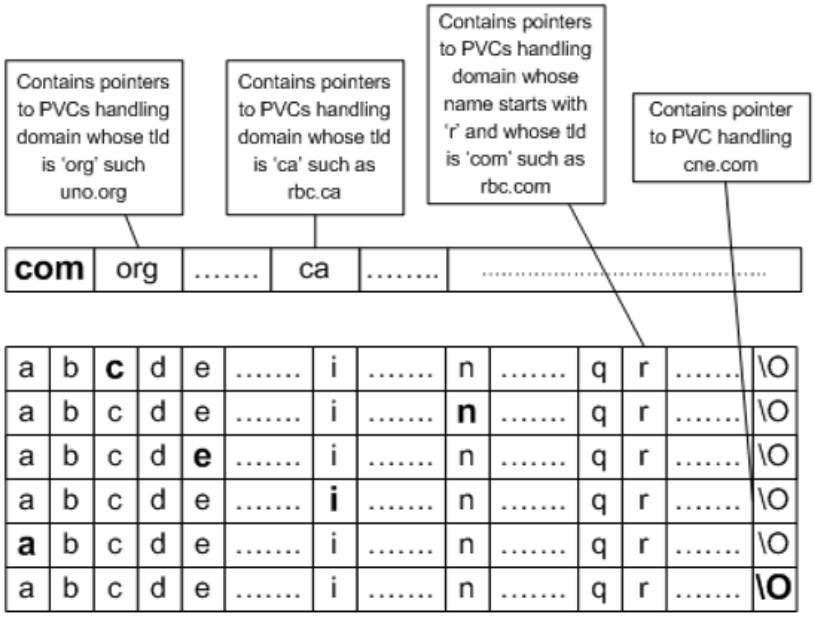

Fig. 4.6 Routing entries for PVC handling cneia.org

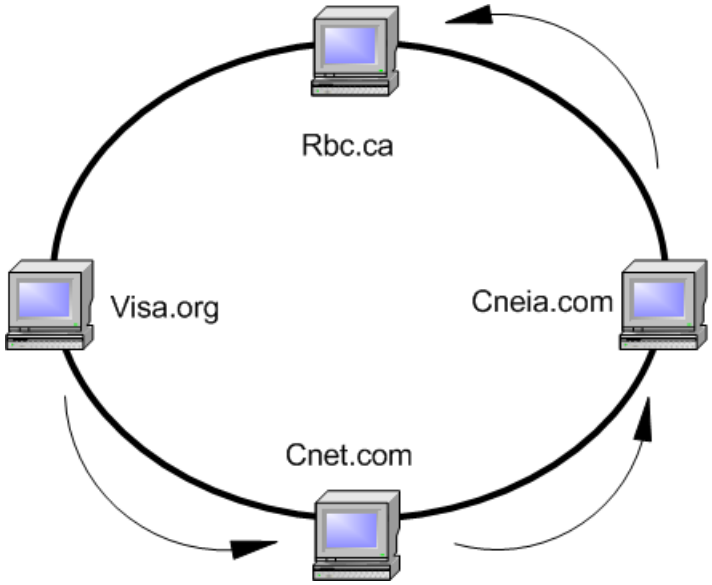

Fig. 4.7 Hypothetical example showing how routing would take place if we want to determine whether cneia.com is phishing. In this case we assume content in cneia.com is similar to the content in rbc.ca

4.2.3 Example of Routings

Example 1:

Suppose we would like to search for the domain name `cneia.com`. There is a PVC ‘‘`a`’’ which handles `cneia.com` as Figure 4.4 and we start search from PVC ‘‘`b`’’ which handles `visa.org`. In this scenario, we will first check if the TLD of the queried URL is the same as the TLD of the domain handled by the PVC `b` . Since it is not the same, the IP address of the domain with the TLD `com` would be extracted from the TLDArray. It could be possible to extract the IP address of the PVC handling `cnet.com`. So in this case, we would route to the PVC handling `cnet.com`. Then we determine the longest prefix shared by `cnet.com` and `cneia.com`. In this case it is `cne`. Since the fourth character is not the same, we extract one of the IP addresses stored in the fourth row of the DNArray. The extracted IP address would be the IP

address of PVC, which handles the domains whose prefixes are `cnei`. In this case we route to PVC handling domain `cneia.com` which is the only PVC handling domain with prefix `cnei`.

Example 2:

Suppose we would like to search for `cneiaf.com`, and assume there is no PVC which handles `cneiaf.com`, but there is a PVC called PVC `b` handling domain name `cneia.com` as in Figure 4.4, we start the search from PVC `a` which handles `visa.org`. Considering the routing procedure described in *Example 1*, we end up in PVC `b` handling domain name `cneia.com`, since there is no routing entry for PVC handling `cneiaf.com`. Consequently, we check its own decision tables to make an appropriate decision about `cneiaf.com`. If we are not able to reach any decision, we invoke the content analysis module.

4.2.4 Content Analysis

Content analysis is another component of CASTLE. It helps to find out which URL is pretending to look like one of the trusted URLs in the network. A URL is assumed to be trusted if there is a PVC which handles this URL. This can help us determine the target of the phishing URL and then we would inform the PVC handling that target URL about the possible phishing attack. The PVC, which has been informed about the target URL, should decide whether the URL is phishing or not and accordingly migrate the inserted URL by the content analysis module in its gray list to the appropriate list.

The content analysis module(CA) takes both text and image snapshot of the given URL by rendering them through a browser using one of the webkit based tools called

CutyCapt [52]. Then inside the CA we have two sub CA modules called TCA (textual CA) and VCA (visual CA). Figure 4.8 shows the flowchart for the content analysis module. The TCA uses text snapshot of the URL and extracts five random sentences from the text corpus. We form a textual signature of the URL using the five sentences. Then we push these sentences one by one into a search engine(e.g., Google) and find the top five URLs (returned by the search engine) for each sentence. A table is used to keep track of these 5 URLs. We refer to the top five results returned by the search engine for each sentence as the *search set*. Based on the result returned by the search engine, a score is computed using a scoring mechanism mentioned below. However before computing the textual signature, we determine if the given URL contains a copyright sentence. The copyright sentence is common in the most phishing websites. It is used to deceive the users and some textual based spam filters. The copyright sentence must contain one of the following sets of keywords:

- "CopyRight" and "All Rights Reserved" in the sentence.
- "©" and "All Rights Reserved" in the sentence.
- "©" and "CopyRight" in the sentence.
- "©" or "CopyRight" with any numbers in the sentence.
- "Trademark" and "Registered", or "©" and "Trademark" in the sentence.

If the given URL has more than one sentence with the above characteristics, we will return that URL as a suspected one and we will put it into the gray list. If it has only one sentence with the above characteristics, we extract this sentence and remove the date inside the sentence and feed that sentence into a search engine. We refer to

the first URL returned by the search engine as the *copyright URL*. The reason that we remove the number from the copyright sentences is that some phishing websites change the date of the copyright sentence in order to be indexed differently from the legitimate websites in search engines. Then we compute the textual signature. We feed each of the five randomly selected sentence separately into the search engine. A score of one would be assigned to the URL returned by the search engine for each sentence if it is present in top five results. The score would be increased if the search engine returns the same URL for the other sentences. Then we will check the presence of *copyright URL* in every search set. If the *copyright URL* is not present, we will repeat the above procedure of computing the textual signature up to five times. If the *copyright URL* is still not present in the search sets, we return the queried URL as a suspected URL and we would put it inside the gray list of the appropriate PVC. If the *copyright URL* is present, the *copyright URL* would be assumed as the target URL.

If the queried URL does not contain any copyright sentence, then the URL with the highest score would be returned as the target URL. After determining the target URL, we will route to the PVC, which is responsible for the target URL and we will put the queried URL into its gray list.

4.3 Complexity Analysis

For simplicity purposes, we will call domain or domain prefixes handled by a PVC h-domain (hosted domain) or h-domains respectively. We know in each routing step, either a PVC forwards the search request to other PVCs whose h-domain share a prefix

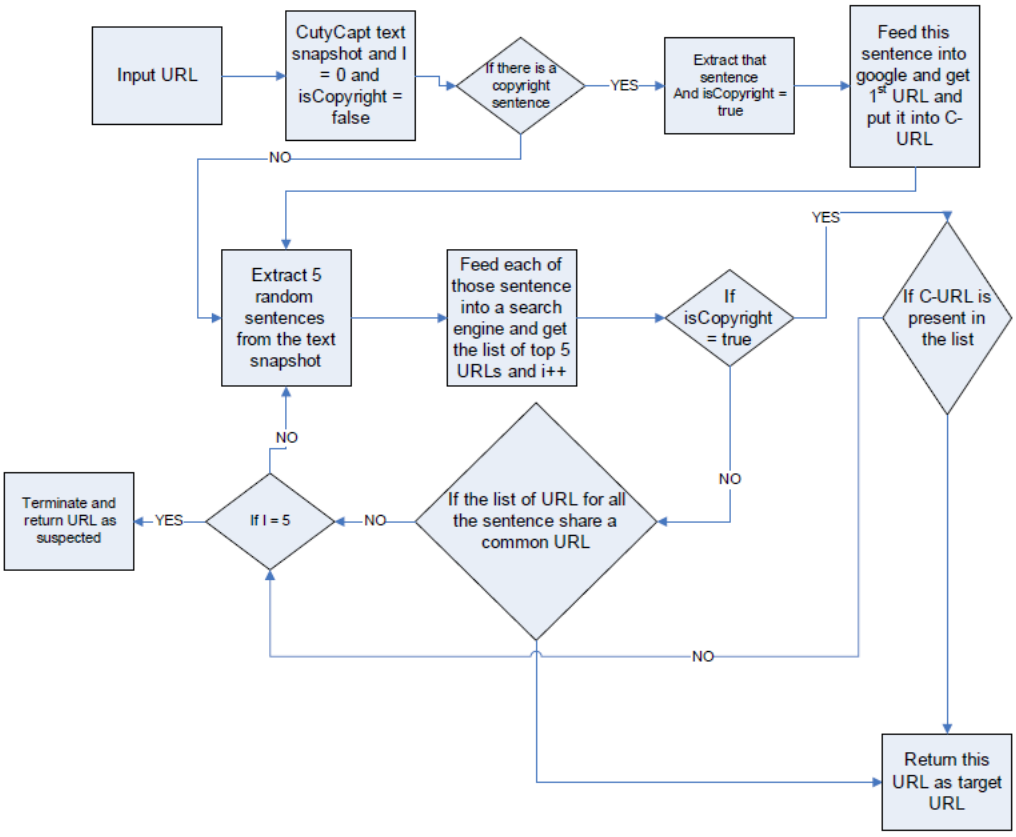

Fig. 4.8 Flowchart for content module.

with the domain name of the queried URL or a PVC forwards the search request to a PVC handling a different TLD.

In order to go to a PVC handling a different TLD, it takes only 1 hop since we pick up the IP address of the corresponding TLD from the TLDArray of the current PVC and route to that one. In order to go to a PVC handling a domain name whose first letter is the same as the first letter of the current PVC, it takes at most 2 hops since we extract the IP address of the domains who share the first letter with the current PVC from the first row of the DNArray of the current PVC, then we route to it. In order to go to a PVC handling a domain name whose first two letters are the same as the first two letters of the queried domain, the extracted IP address from the second row will be used so it takes at most 3 hops. So in order to go to a PVC handling a domain whose first n letters are same as the first n letters of the queried domain, it takes at most $n + 1$ hops. As mentioned in Section 4.2.2, the NULL character is also consider part of the domain name, so we will make use of the NULL character in order to restrict the number of routing hops to the domain length. For example if domain name is *cnet*, in our design it would be represented as *cnet* plus the NULL character. In this case it would take at most 6 hops assuming cnet.com is handled by one of the PVCs since there are 5 letters including the NULL character in the domain name. If the queried URL is stored in any of the PVC's decision list and assuming the URL length is n,then in this case it can take up to $n + 2$ hops to reach that PVC.

As mentioned in Section 4.2.2, it is also possible that the URL is not handled by any PVC and is not present in white list, gray list or black list of all the PVC. In this case, we would route to the PVC whose h-domain is similar to the queried domain. After routing to this PVC, we would use the unknown list or the content

analysis module to find the domain whose content is the most similar to the content of the queried domain, and then we would route to the PVC handling domain which is stored in the unknown list or the URL returned by content analysis module. For this scenario, assuming m is the length of the domain and this domain is handled by one of the PVCs whose content is the most similar to the content of queried domain, it would take up to $m + n + 4$ hops.

4.4 Security Analysis

Security threats to CASTLE can be categorized into two parts:

1. Threat to content analysis module (CAM).

2. Threat to the social network of PVCs.

4.4.1 Attacks to the CAM

Attackers can put some invisible text in order to circumvent the content analysis module of the CASTLE because sentences which are part of the invisible text, could be used by CAM to analyze the content. As a result, CAM can return URL whose content is not similar to the content of queried URL. In order to prevent such an attack, the process of selecting 4 random sentences from the content of the queried URL and feeding them into the search engine will be repeated twice. If the outcome of these two iterations are not similar, then the queried URL would be returned as suspected.

Also since we are dealing with the exact domain name of the visiting URL, all the misleading information in the domain name that the attacker tries to put in order to

mislead the users would be useless. For example if the URL is *www.ebay.online.com*, the user may think that this URL belongs to *eBay*, but in our design, it would be treated as *www.online.com* not as a subdomain of *eBay*.

In addition, all the changes in the user's browser such as hiding the address bar, faking the address bar, or displaying the URL in unicode format would be useless because we are dealing with the actual domain who hosts the websites and the content of the phishing website in order to find the website that the attacker is trying to phish.

4.4.2 Attacks to the social network of PVCs

Another attack category would be against the P2P network of PVCs which handle the information regarding the decision that should be rendered for a given URL. In this case we assume that all the PVCs are connected through the social network and propagation of the routing information will be taking place over the social network, which is guarded using SybilGuard that prevents a user to create many trust relationships; so the only attack against this would be a intrusion into the social network. An attacker can compromise a node in the social network even without the awareness of the authorities of that PVC and may try to give false information to other PVCs who may ask the compromised PVC. But in this case an attacker can only say something regarding the domain that it is responsible or propagate wrong routing information. We call that node a "bad node" inside the social network. In general bad nodes can do the following actions inside the social network:

- Ignore queries
- Misroute queries

- Propagating false routing updates
- Refuse to store or return items
- Return incorrect items

To address these issues, we assume that at least one node in the social network is alive and not malicious. Then we try to prove that a node is bad with some mechanism that will result in the eviction of such a node from the social network.
We introduce an online certificate authority(OCA), which will issue a certificate to each node upon their joining to the network. Also each node must inform the OCA about the address of two possible backup of itself in case of failure or attack. Each node must know the private key if it wants to propagate routing information to other nodes. So if a node is hacked, then the attacker can not connect to other nodes to get the routing information or update the routing information of that node if it is granted the required permission by the contacted node for the update process. Also each node must renew its own key after its expiration of the key. So if a node proved itself to be malicious, the OCA would refuse to issue a certificate to it. Thus it would be evicted from the social network. So any attempt from anyone inside the network to say something wrong will result as misbehavior and exile from the social network.

Misbehavior can be proven through some mechanisms. Any node will periodically check its information with its connected nodes. Connected nodes are nodes inside the routing table which have been connected through pointers to the current node. So each node will ask connected nodes about the information that it has in its black or whitelist and monitor the responses of the connected nodes. Connected nodes can be

responsive or not responsive. If they are not responsive, then there should be some problem with them or they might have been hijacked by attackers who do not want to respond and just want to give out false information about the current PVC's domain. In this case, the node who has checked the irresponsiveness will inform the *OCA* about the connected nodes which are not responsive. We call this a "complaint to the *OCA*". If the number of complaints for a certain nodes exceeds a threshold, then that node would be tagged as a suspected node and the *OCA* will perform its own procedure to validate the complaint. The *OCA* will select a random number of PVCs in the network excluding the ones which have complained to ask the suspected node about its information. The reason that we choose random nodes inside the network is because we do not want to show the suspected node that it is on probe. If the queries come from different sources, this can be considered as a normal query process inside the network and cannot be detected as a probe process. After the random nodes checked the suspected node about its information to see whether the information that it supplies is correct or not, they will inform the *OCA* about the correct or incorrect answers returned by the suspected node. If the number of incorrect answers exceeds a threshold, then the *OCA* will call that node malicious and will inform its immediate back up nodes, so that those backup nodes can revoke their links with that malicious node and take the responsibility of that node inside the network. Also the *OCA* will not issue to that node a certificate anymore; so no more queries will be forwarded to that node anymore and it will be evicted from the network at the time of renewal of the certificate for good. The reason that we ask the *OCA* to evaluate the complains by itself is because of avoiding the case where some nodes join together in order to evict one of their competitors from the network by providing false information the

OCA.

To protect the *OCA*, we can also consider a total distributed-limited state shared replicas for it. Also only nodes that are currently part of the network need to inform or ask the *OCA* so we can filter all other traffic to it.

Chapter 5

Prototype Design and Experimental Result

5.1 Prototype Design

We have developed a prototype of the CASTLE using the Java programming language, which can be deployed on the real Internet. In this prototype, the textual content analysis module (CAM) and the routing mechanism are also designed and implemented with all of their heuristics as we described them in Section 4.1.

5.2 Implementation

As we mentioned in Section 5.1, we implemented CASTLE in JAVA. CASTLE comprises of 2000+ lines of code as well as some modules such as CutyCapt, Google AJAX Search API, JAVA RMI, and JSON libraries. We used CutyCapt libraries to get the text snapshot of the content of a web page. Google AJAX Search API was used to access the Google database to search for a particular sentence and to retrieve the corresponding URLs. JAVA RMI library was used to build the network of PVCS, which connects trusted PVCs to each other. The content analysis module has a command line interface with some options, which returns a URL which has the similar content to the queried URL and puts it inside the gray list of the appropriate PVC. Each PVC also has a command line interface where the administrators of a PVC can modify the

routing tables or the decision tables.

5.3 Experimental results

We tested the content analysis module (CAM) and the routing algorithm which are the major components of CASTLE to evaluate our design and their performance in the real simulation on the Internet. We used PlanetLab to test our solution with different scenarios.

5.3.1 Routing algorithm

We tested the routing algorithms on PlanetLab with 100 machines (PVCs) in different locations. Each PVC handled one URL as a trusted URL. Also we add other URLs in each PVC to make it responsible for multiple URLs. the distribution of URLs is shown in Figure 5.1. Black lists, white lists and gray lists for each PVC were populated with 7 URLs. In total, each PVCs was populated with 21 URLs. Since we encountered scalability issues on PlanetLab, we could not test on more than 100 machines.

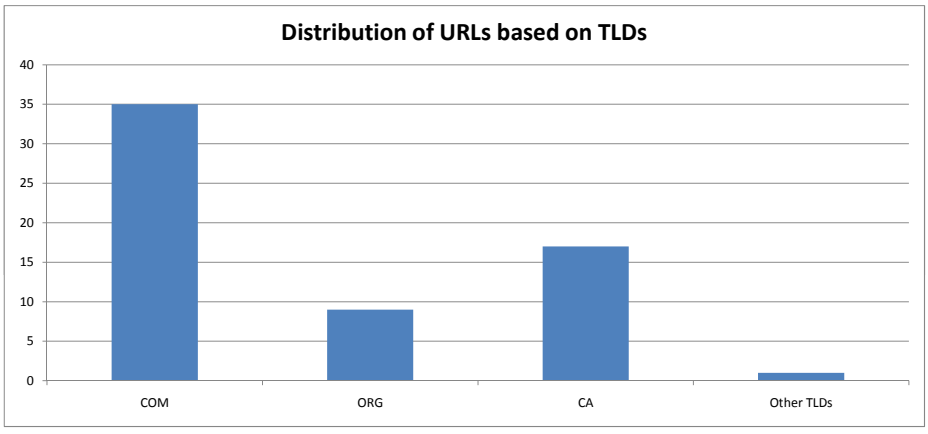

Fig. 5.1 Distribution of the trusted URLs on PlanetLab machines based on their TLDs

5.3 Experimental results

In the first part of this experiment, we carried out the search from four different PVCs handling different TLDs for all 100 URLs which were assigned to each of the 100 machines in PlanetLab. We repeated this experiment 15 times for each of the four different PVCs. The response times are shown in Figure 5.2, 5.3, 5.4 and 5.5 for various TLDs respectively. As shown in these figures, most of the time, the time for search is from 2-4 seconds. However in few cases, the time is 18 seconds which is due to the abnormal behavior of some of the nodes inside the PlanetLab such as network congestion or a node failure.

In the second part of this experiment, we carried out a search of 2200 URLs stored in decision lists of PVCs and 100 URLs assigned to each PVC from four different PVCs handling domains with different TLDs. We repeated this experiment 7 times. The response time are shown in Figures 5.6, 5.7, 5.8 and 5.9 for various TLDs respectively. As shown in these figures, most of the time, the search time is less than 4 seconds. However in a few cases, the search time is between 4 and 18 seconds. In Figure 5.8, the maximum response time that we found in this experiment is 35 seconds which is due to the abnormal behavior of some of the nodes inside the PlanetLab. As we notice in the figure 5.10, the response time for the 90% of searches is at most half a second. However, as can be seen on the figure 5.10, the response time for 2% of the searches is more than 2 seconds because of abnormal network conditions inside the PlanetLab.

In the third part of the experiment, in order to evaluate the response time of our system for the URLs which were not hosted by any of the PVCs inside the network, we collected about 350 URLs which were not handled by any of the PVCs and were not present in the decision list of any PVC. Then we carried out a search of these 350 URLs from four different PVCs, which handled domains with different TLDs. We repeated this experiment 5 times. The response times are shown in Figures 5.11, 5.12, 5.13 and 5.14 for various TLDs respectively. As shown in these figure, most of the time, the times for searching is below 2 seconds. However in a few cases, the search time is more than 4 seconds which is due to the abnormal behavior of some of the nodes inside the PlanetLab.

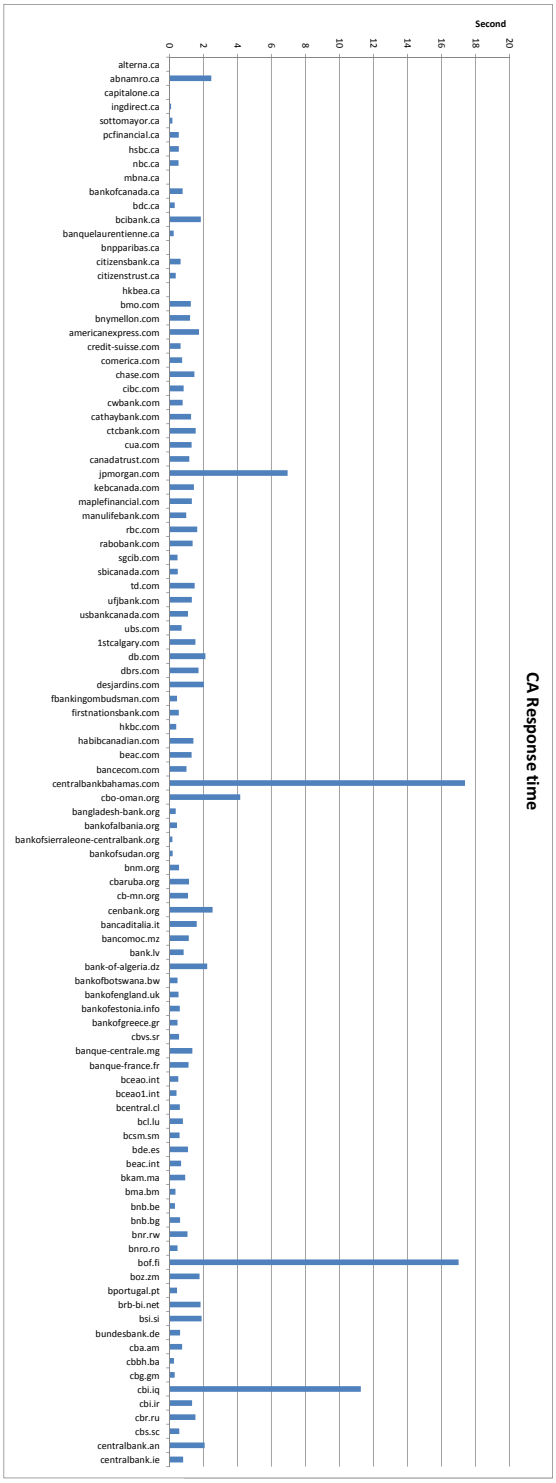

Fig. 5.2 The response time for search carried out from PVC handling domain with CA TLD

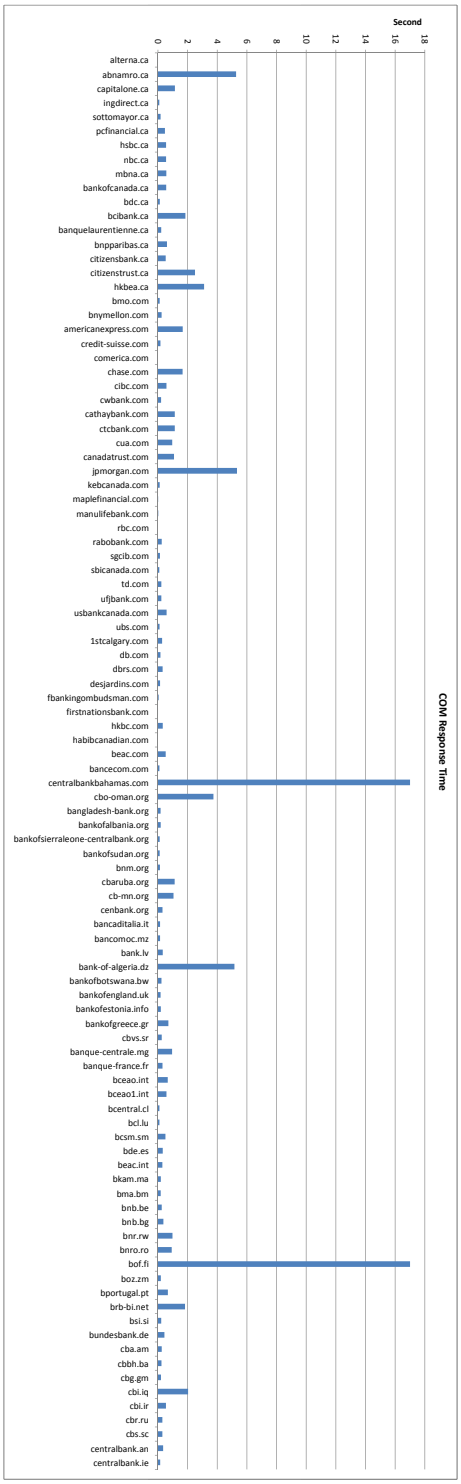

Fig. 5.3 The response time for search carried out from PVC handling domain with com TLD

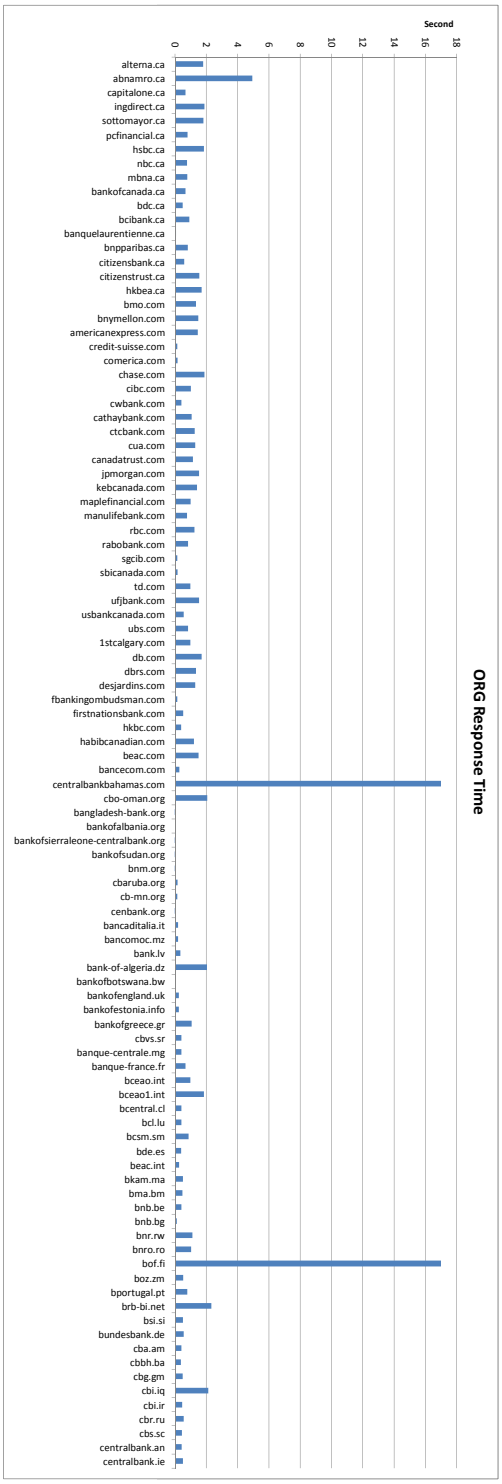

Fig. 5.4 The response time for search carried out from PVC handling domain with ORG TLD

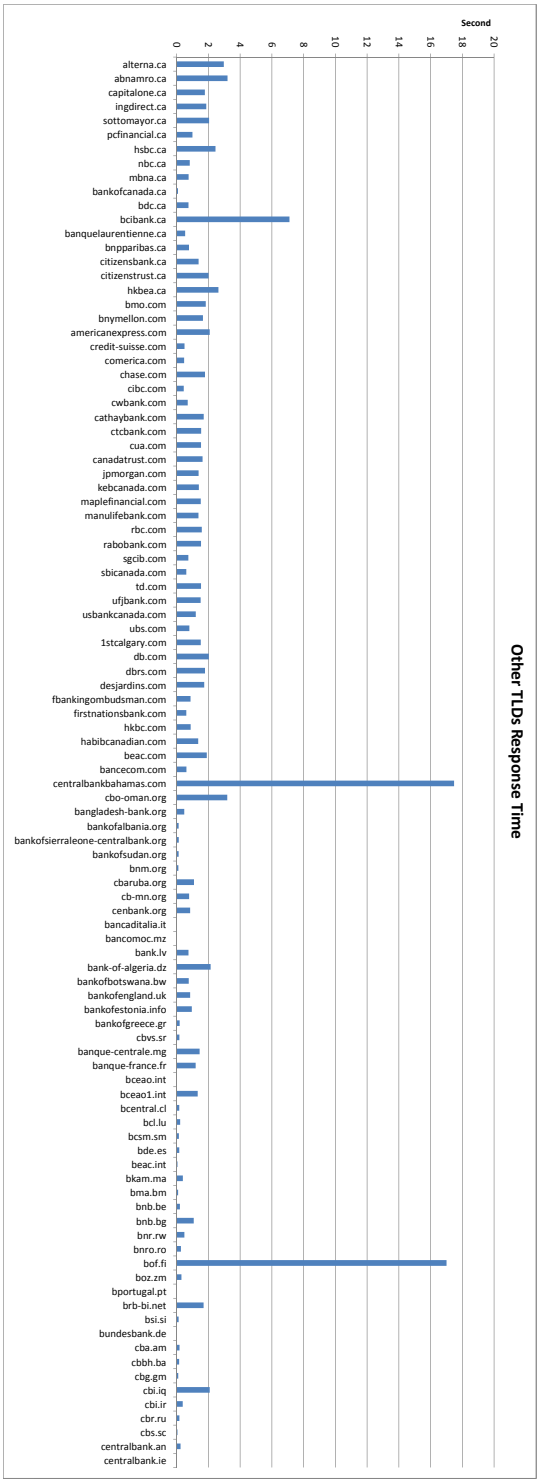

Fig. 5.5 The response time for search carried out from PVC handling domain with other TLD

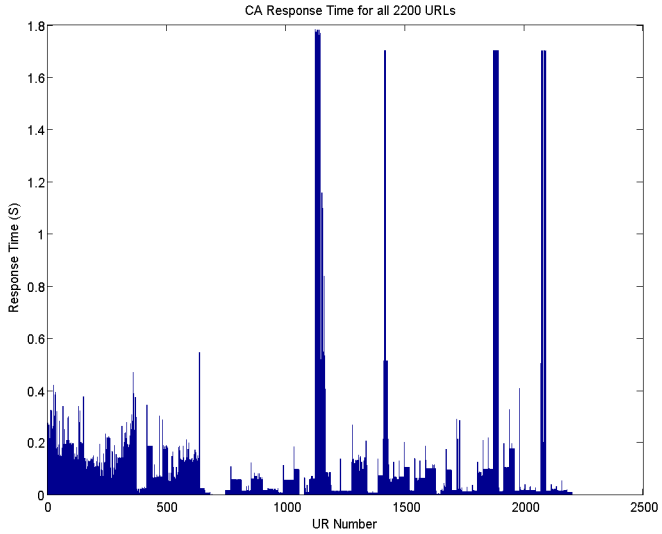

Fig. 5.6 The response time for search carried out for the 2200 URLs from PVC handling domain with CA TLD

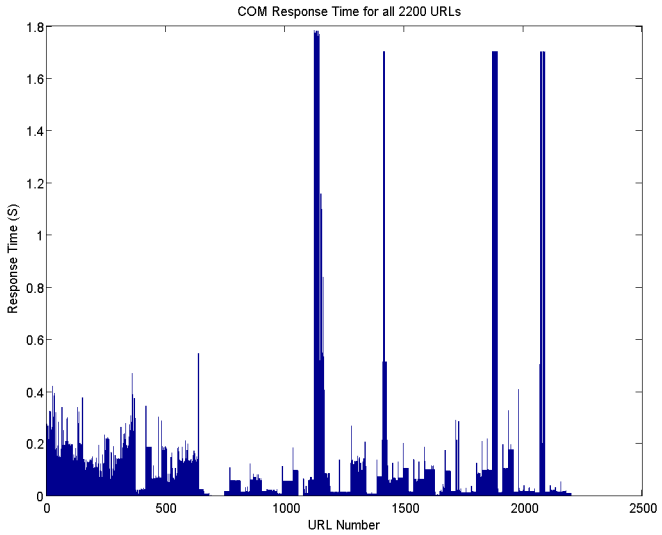

Fig. 5.7 The response time for search carried out for the 2200 URLs from PVC handling domain with COM TLD

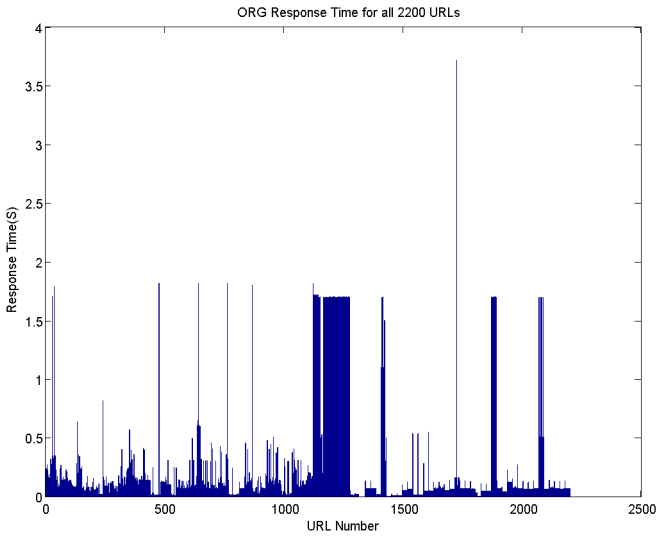

Fig. 5.8 The response time for search carried out from for the 2200 URLs PVC handling domain with ORG TLD

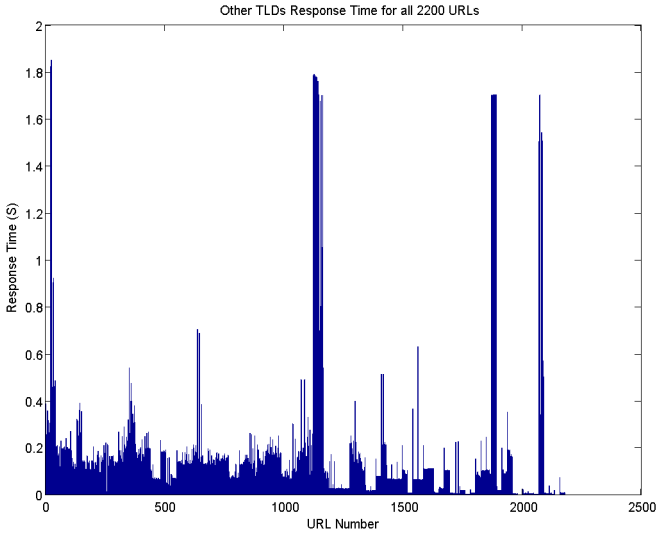

Fig. 5.9 The response time for search carried out for the 2200 URLs from PVC handling domain with other TLDs

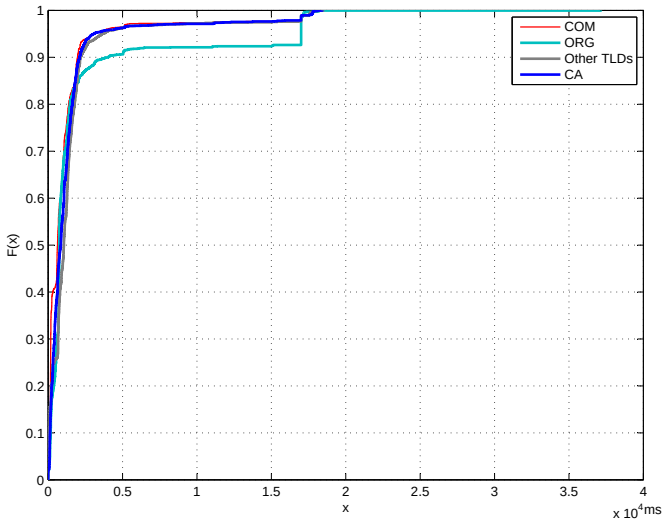

Fig. 5.10 The distribution of response time for search of 2200 URLs carried out from four different PVCs handling different TLDs

5.3.2 Content Analysis Module

We conducted an experiment which consists of 150 phishing URLs along with 50 legitimate URLs in order to measure the effectiveness of our content analysis approach which is described in section 4.2.4. From April 5 to April 22 2009, we collected most of the phishing URLs from `phishtank.com`. These phishing URLs from `phishtank.com` were selected within one hour of being reported. Some of the phishing URLs were also gathered from users' spam email accounts. We also manually chose 50 legitimate URLs in different categories such as banks, popular e-commerce websites, and top 20 popular sites from the Alexa Web search such as `www.rbc.com`, `www.citibank.com`, `www.ebay.com`, and `www.cnn.com`. In our test scenario, we fed all 200 URLs into our content analysis module. In the case of a phishing URL, which is fed into our module, we compared the URL returned by our content analysis module and the actual URL that is being phished by the phishing URL. In a case of a legitimate URL, which is

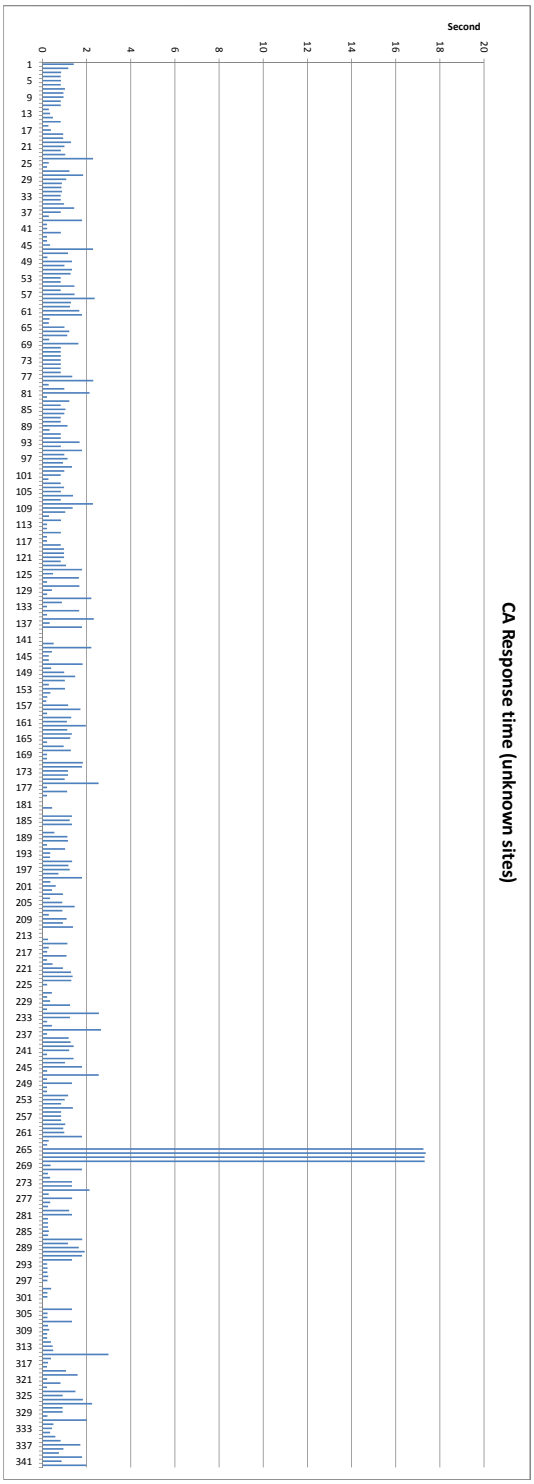

Fig. 5.11 The response time for search of 350 unknown URLs carried out from PVC handling domain with CA TLD

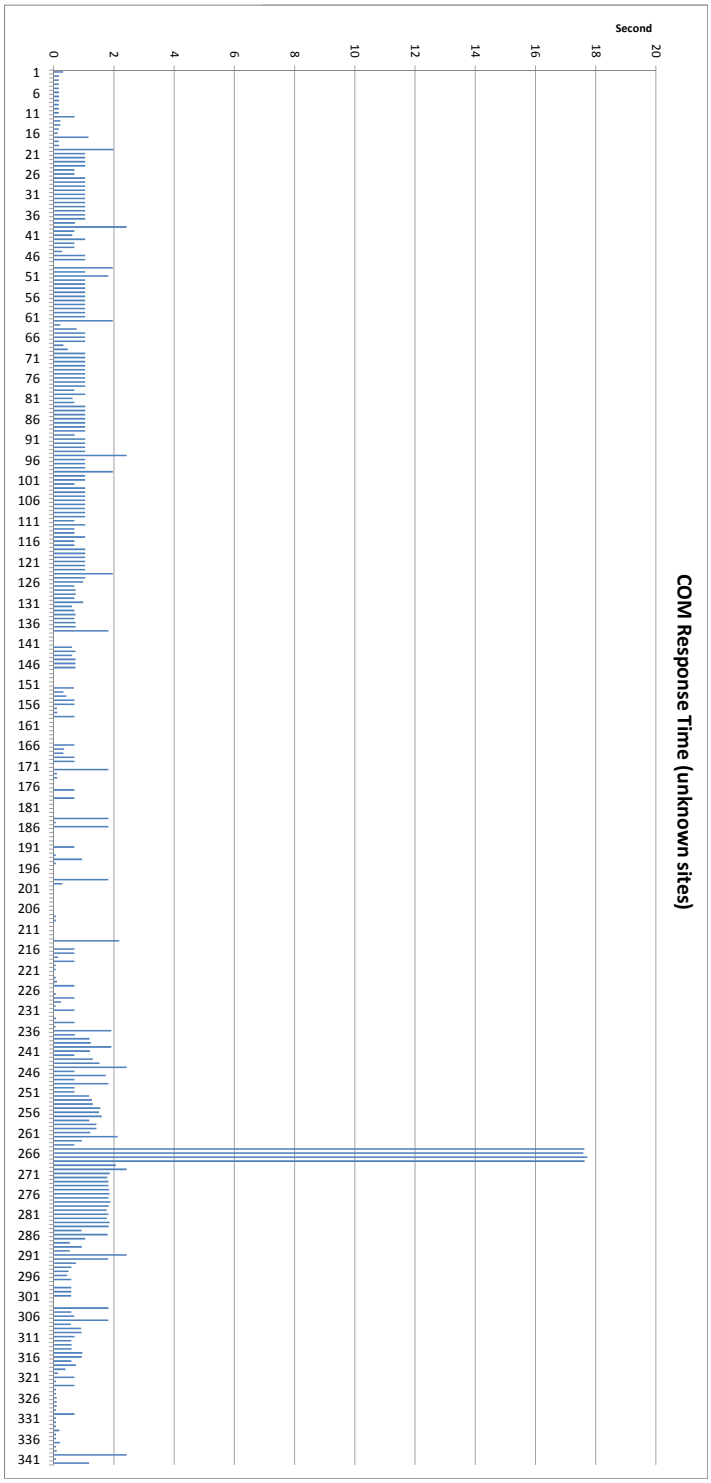

Fig. 5.12 The response time for search of 350 unknown URLs carried out from PVC handling domain with COM TLD

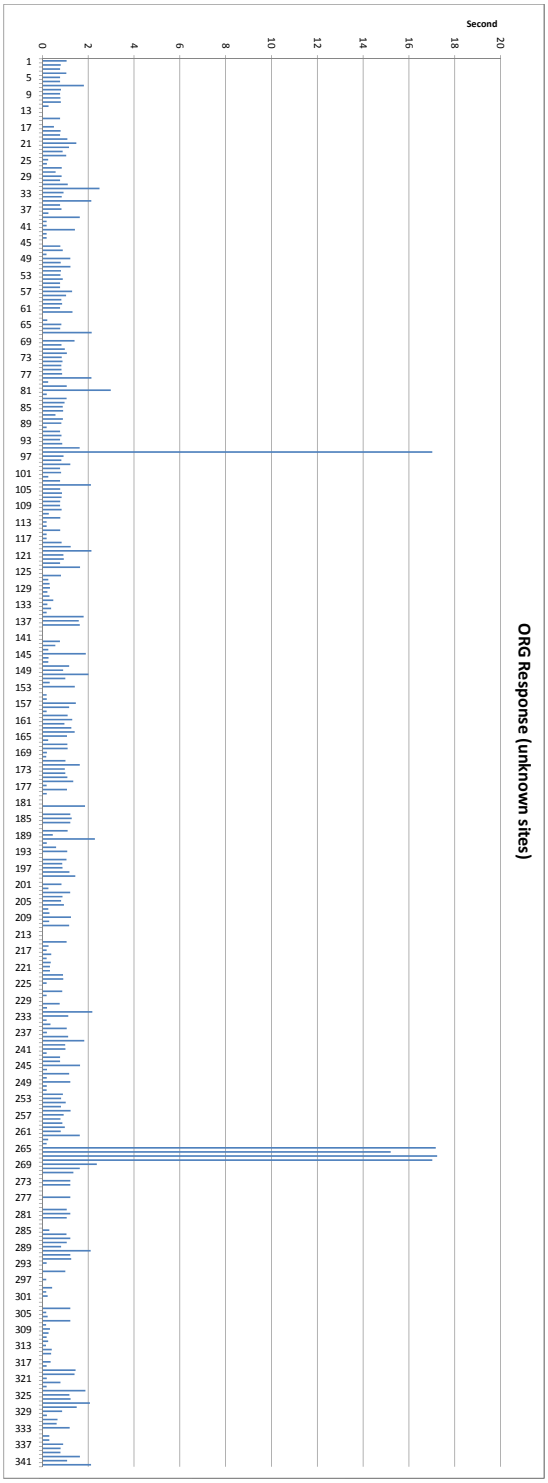

Fig. 5.13 The response time for search of 350 unknown URLs carried out from PVC handling domain with ORG TLD

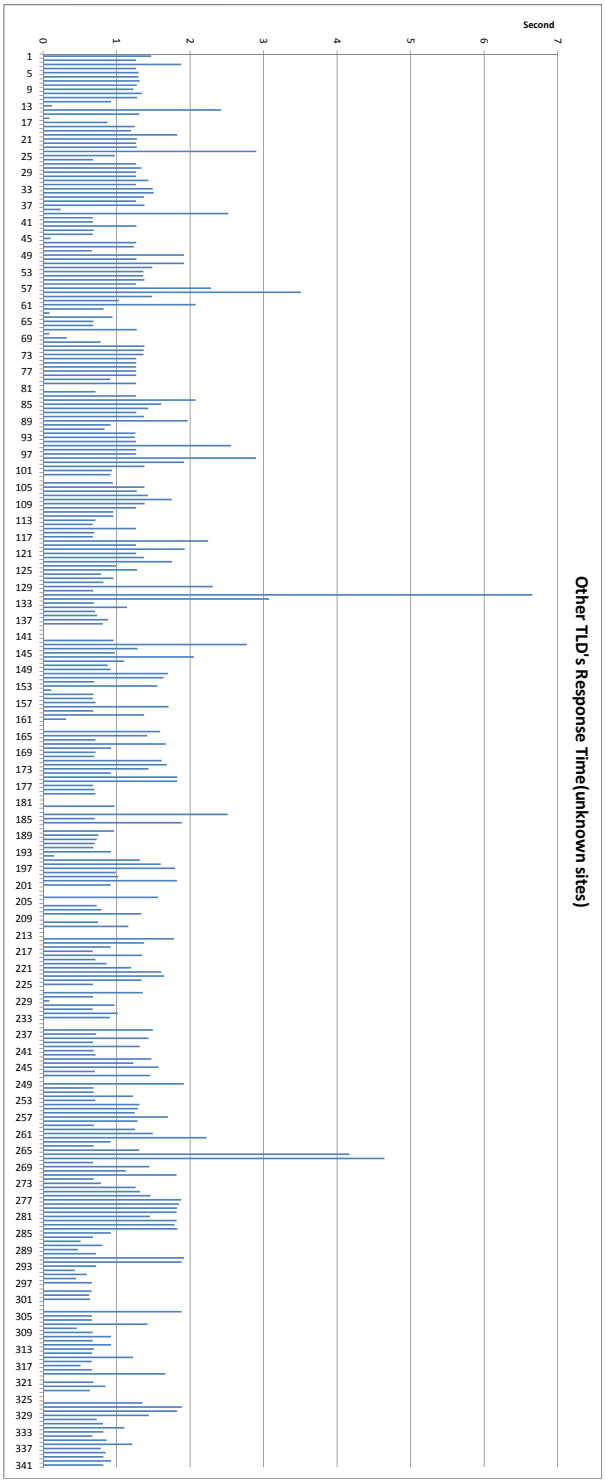

Fig. 5.14 The response time for search of 350 unknown URLs carried out from PVC handling domain with other TLDs

fed into our module, we compared the legitimate URL itself and the URL returned by our content analysis module. In this case, both URLs should be the same. Figure 5.16 shows our results. CASTLE content analysis module can detect the target websites of the phishing websites very well and had a fairly low false positive rate 2.5%. To decrease the false positive rate further, we applied a set of heuristics as described in Section 4.2.4 under the copyright property characteristic and ran another experiment with the same structure. After the new experiment was run, the false positive rate fell down to 1.5%. Figure 5.16 shows the results after applying these heuristics. The average response time for the content analysis module was also computed for the 200 URLs as described above which was 4.62 seconds.

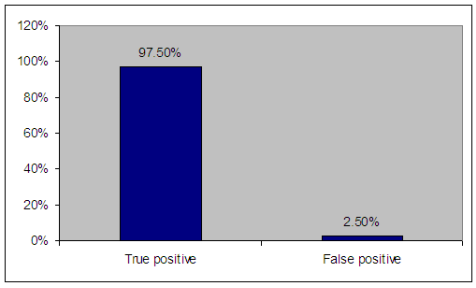

Fig. 5.15 Result before applying heuristics (described in section under copyright property characteristics).

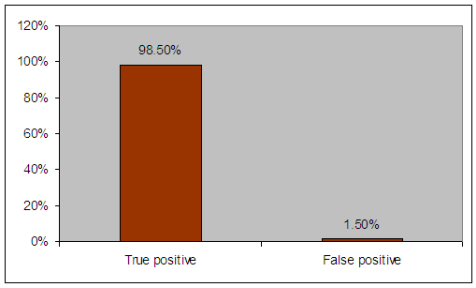

Fig. 5.16 Result after applying heuristics (described in section under copyright property characteristics).

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis describes a new framework called CASTLE that provides a collaborative approach for building anti-phishing databases. Such anti-phishing databases are at the core of browser-based toolbars, plugins, and extensions that prevent users from falling prey to phishing attacks. CASTLE embodies both detection and correction based strategies for anti-phishing based attack categorization that we presented in Section 3.2.

The advantage of the CASTLE approach is that it can provide a substrate for integrating many different algorithm for implementing detection and correction. For example, the PVCs that are part of the P2P network can employ diverse techniques for making the local decisions. Some PVCs can use a manual examination based approach while others could be using a combination of automated analysis and classification and manual examination. The use of diverse techniques increases the scalability of the anti-phishing effort.

Further, to the best of our knowledge this is the first anti-phishing framework that combines location-based and content-based techniques in a single system.

6.2 Future Work

Our framework is lightweight and can be deployed on the Internet without requiring major changes to the underlying infrastructure. Another benefit of our framework is that legacy applications such as emails or social networks can readily benefit from it without undergoing any changes.

We have implemented a prototype of CASTLE that can be deployed on the Internet and the initial performance results are reported here. In the future, the system design will be analyzed and updated in terms of the attacks and security issues. For example, PVCs can be compromised through malicious PVCs or malewares. There is a possibility that a PVC inside the network would delegate its role to an attacker. There are other security issues such as secure routing updates, secure message exchange among PVCs, and secure communication between PVCs and the OCA.

Also, our content analysis module now only performs textual analysis but not analysis on images. Some phishing websites may include a number of images or only an image to circumvent our textual analysis. So, the content analysis module should be expanded in order to capture those attacks.

References

[1] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*, (New York, NY, USA), pp. 639–648, ACM, 2007.

[2] A. Y. Fu, "Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd)," *IEEE Trans. Dependable Secur. Comput.*, vol. 3, no. 4, pp. 301–311, 2006.

[3] E. Kirda and C. Kruegel, "Protecting users against phishing attacks with antiphish," *Computer Software and Applications Conference, Annual International*, vol. 1, pp. 517–524, 2005.

[4] M. Wu, R. C. Miller, and S. L. Garfinkel, "Do security toolbars actually prevent phishing attacks?," in *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, (New York, NY, USA), pp. 601–610, ACM, 2006.

[5] M. Jakobsson and S. Myers, *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley-Interscience, 2006.

[6] "Social network wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Social_network.

[7] J.A.Barnes, "Class and committees in a norwegian island parish," *Human Relations*, vol. 7, pp. 39–58, 1954.

[8] S. Wasserman and K. Faust, *Social Network Analysis : Methods and Applications*. Cambridge University Press, 1994.

[9] *Online Communities and Social Computing: Second International Conference, OCSC 2007,*. Springer-Verlag Gmbh, 1., ed. ed., September 2007.

[10] D. M. Boyd and N. B. Ellison, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, 2007.

[11] "Threat classification." http://www.webappsec.org/projects/threat, 2004.

[12] "What is phishing." http://www.webopedia.com/TERM/P/phishing.html.

[13] S. A. Robila, "Don't be a phish: steps in user education," in *ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology*, pp. 237–241, ACM Press, 2006.

[14] "Phishing attack trends report - second half 2008." http://www.apwg.org/reports/apwg_report_H2_2008.pdf, Mar. 2009.

[15] M. Jakobsson, "Modeling and preventing phishing attacks," in *In Financial Cryptography*, Springer Verlag, 2005.

[16] C.-F. Yeh, C.-H. Mao, H.-M. Lee, and T. Chen, "Adaptive e-mail intention finding mechanism based on e-mail words social networks," in *LSAD '07: Proceedings of the 2007 workshop on Large scale attack defense*, (New York, NY, USA), pp. 113–120, ACM, 2007.

[17] "ecrime '07: Proceedings of the anti-phishing working groups 2nd annual ecrime researchers summit," 2007.

[18] O. Zaytsev, *Rootkits, Spyware/Adware, Keyloggers and Backdoors: Detection and Neutralization*. A-List Publishing, 2006.

[19] S. Abu-Nimeh and S. Nair, "Bypassing security toolbars and phishing filters via dns poisoning.," in *GLOBECOM*, pp. 2001–2006, IEEE, 2008.

[20] "APWG." http://www.antiphishing.org, Anti Phishing Working Group.

[21] "Gartner institute." http://www.gartner.com.

[22] M. Wu, *Fighting phishing at the user interface*. PhD thesis, Cambridge, MA, USA, 2006. Adviser-Miller, Robert C.

[23] D. K. McGrath and M. Gupta, "Behind phishing: an examination of phisher modi operandi," in *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, (Berkeley, CA, USA), pp. 1–8, USENIX Association, 2008.

References

[24] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 1245–1254, ACM, 2009.

[25] C. Karlof, U. Shankar, J. D. Tygar, and D. Wagner, "Dynamic pharming attacks and locked same-origin policies for web browsers," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 58–71, ACM, 2007.

[26] G. Wassermann and Z. Su, "Static detection of cross-site scripting vulnerabilities," in *ICSE '08: Proceedings of the 30th international conference on Software engineering*, (New York, NY, USA), pp. 171–180, ACM, 2008.

[27] M. Nystrom, *Sql injection defenses*. O'Reilly, 2007.

[28] "Spam wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Spam_(electronic).

[29] "Spoofguard." http://crypto.stanford.edu/SpoofGuard/.

[30] "Services - buyer tools - ebay toolbar." http://pages.ebay.ca/ebay_toolbar.

[31] "Netscape." http://www.netscape.com.

[32] "Microsoft." http://www.microsoft.com.

[33] D. H. T. H. J. Likarish, P. Eunjin Jung Dunbar, "B-apt: Bayesian anti-phishing toolbar," in *Communications, 2008. ICC '08. IEEE International Conference on*, (Washington, DC, USA), pp. 1745–1749, IEEE Computer Society, 2008.

[34] J. Kang and D. Lee, "Advanced white list approach for preventing access to phishing sites," in *ICCIT '07: Proceedings of the 2007 International Conference on Convergence Information Technology*, (Washington, DC, USA), pp. 491–496, IEEE Computer Society, 2007.

[35] "Spoof stick." http://www.spoofstick.com/.

[36] A. Herzberg and A. Jbara, "Security and identification indicators for browsers against spoofing and phishing attacks," *ACM Trans. Internet Technol.*, vol. 8, no. 4, pp. 1–36, 2008.

[37] M. Wu, R. C. Miller, and G. Little, "Web wallet: preventing phishing attacks by revealing user intentions," in *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, (New York, NY, USA), pp. 102–113, ACM, 2006.

[38] E. Kirda and C. Kruegel, "Protecting users against phishing attacks with antiphish," in *COMPSAC '05: Proceedings of the 29th Annual International Computer Software and Applications Conference*, (Washington, DC, USA), pp. 517–524, IEEE Computer Society, 2005.

[39] "Google safe browsing for firefox." www.google.com/tools/firefox/safebrowsing/.

[40] "Cloudmark security network." http://www.cloudmark.com/.

[41] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, "Sps: A simple filtering algorithm to thwart phishing attacks," in *AINTEC*, pp. 195–209, 2005.

[42] "Sitewatcher - anti-phishing service." http://www.sitewatcher.com/.

[43] R. Dhamija and J. D. Tygar, "The battle against phishing: Dynamic security skins," in *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*, (New York, NY, USA), pp. 77–88, ACM, 2005.

[44] K.-P. Yee and K. Sitaker, "Passpet: convenient password management and phishing protection," in *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, (New York, NY, USA), pp. 32–43, ACM, 2006.

[45] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger password authentication using browser extensions," in *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*, (Berkeley, CA, USA), pp. 2–2, USENIX Association, 2005.

[46] G. Yang, D. S. Wong, H. Wang, and X. Deng, "Two-factor mutual authentication based on smart cards and passwords," *J. Comput. Syst. Sci.*, vol. 74, no. 7, pp. 1160–1172, 2008.

[47] "RSA. Security-Technology." www.rsa.com/rsalabs/technotes/OneTimePWWP.pdf.

[48] T. Moore and R. Clayton, "Examining the impact of website take-down on phishing," in *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, (New York, NY, USA), pp. 1–13, ACM, 2007.

[49] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 267–278, 2006.

[50] "Top-level domain names by host count." http://ftp.isc.org/www/survey/reports/current/report.bynu
2008.

References

[51] "Domain name system." http://en.wikipedia.org/wiki/Domain_Name_System.

[52] "Cutycapt." http://cutycapt.sourceforge.net/.