NAIST-IS-DT0561032

Doctoral Dissertation

A Machine Learning Approach for Detecting Fraudulent Websites

Daisuke Miyamoto

Department of Information System
Graduate School of Information Science
Nara Institute of Science and Technology
February 18, 2009

Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Daisuke Miyamoto

Thesis Committee:

Professor Suguru Yamaguchi	(Supervisor)
Professor Hideki Sunahara	(Co-supervisor)
Dr. Hiromitsu Takagi	(Co-supervisor, National Institute of Advanced Industrial Science and Technology)
Associate Professor Youki Kadobayashi	(Co-supervisor)

Abstract

This dissertation presents machine learning based detection methods against phishing. Phishing is a fraudulent activity defined as the acquisition of personal information by tricking an individual into believing the attacker is a trustworthy entity. Phishing attackers lure people by using "phishing email", as if it were sent by a legitimate corporation. The attackers also attract the email recipients into a "phishing site", which is the replica of an existing web page, to fool a user into submitting personal, financial, and/or password data. Strategies against phishing tackle to protect users from fraud.

My research motivation is developing a method for detection of phishing sites to prevent users from browsing phishing sites. Currently, existing detection methods are far from suitable. URL filtering-based detection methods could not deal with new types of phishing attacks, i.e, spear phishing. Conversely, heuristics-based detection methods have a possibility to identify these sites. When users browse a site, the methods calculate the likelihood of being a phishing site for the site. The methods also classify the site as phishing if the likelihood is greater than the discrimination threshold. The problem in heuristics-based detection methods is that the detection accuracy is not high. Accordingly, users would become distrusting the system and would ignore the notification from detection systems.

In this dissertation, I employ machine learning algorithms to improve the detection accuracy, machine learning can facilitate the development of algorithms or techniques by enabling computer systems to learn. As my preliminary experiment, I investigate whether a machine learning algorithm is available or not. I construct a training dataset by analyzing 50 phishing sites reported on Phishtank.com and the same number of legitimate sites with 8 heuristics, namely, Age of Domain, Known Images, Suspicious URL, Suspicious Links, IP Address, Dots in URL, Forms, and TF-IDF-Final heuristics. I then let AdaBoost, one of the typical machine learning algorithms, study from the

training dataset in a supervised-learning manner. I also construct a testing dataset composed of 50 phishing sites and the same number of legitimate sites, and classify them based on the model derived from the training dataset. By comparing with the existing method, AdaBoost can provide higher detection accuracy in almost of all cases. In some cases, overfitting problems are observed. To avoid the overfitting problems, I attempt to increase the number of URLs in dataset by both implementing all heuristics and monitoring Phishtank.com periodically.

In my performance evaluation, I employ 9 machine learning techniques including AdaBoost, Bagging, Support Vector Machines, Classification and Regression Trees, Logistic Regression, Random Forests, Neural Networks, Naive Bayes, and Bayesian Additive Regression Trees. I let these machine learning techniques combine heuristics, and also let machine learning-based detection method(MLBDM)s distinguish phishing sites from others. I analyze our dataset, which is composed of 1,500 phishing sites and the same number of legitimate sites. These 1,500 URLs of phishing sites are reported Phishtank.com during November, 2007 – February, 2008, and are verified as phishing sites by registered users of Phishtank.com. I then classify them using the machine learning-based detection methods, and measure the performance. In my performance evaluation, I decide f_1 measure, error rate, and Area Under the ROC Curve (AUC) as performance metrics along with my requirements for detection methods. The highest f_1 measure is 0.8771, the lowest error rate is 11.96%, and the highest AUC is 0.9543, all of which are observed in the case of AdaBoost.

Next, I check whether or not MLBDMs are available even if the dataset or the set of heuristics are different. I test another dataset which contains phishing sites reported in different time period. I also use another dataset which contains 1,277 URLs of phishing sites, 223 URLs which are not phishing sites but treated as phishing, and 1,500 URLs of legitimate sites. I also change a set of heuristics and observe the performance. All results show that almost of all MLBDMs outperform the traditional detection method.

I then discuss utilization methods for MLBDMs. First, I explore a way for deciding the discrimination threshold for each user. Within my preliminary algorithm, I confirm that changing threshold can customize the detection strategy. Next, I argue the another approach which aims to cover the weak points of users by existing heuristics with machine learning techniques. The key idea of this approach, named "HumanBoost", is employing users' past trust decision as a new heuristic. As my pilot study, I conduct subject within test by calling 10 subjects. Subjects browse 14 emulated phishing sites

and 6 legitimate sites, and check if the site seems to be a phishing site or not. By using such types of subjects' judgments as a new heuristic, I let AdaBoost to incorporate the heuristic into existing 8 heuristics. The results shows that the average error rate in the case of HumanBoost was 9.5%, whereas the average error rate of subjects was 19.0% and that in the case of AdaBoost was 20.0%.

I also propose HTTP Response Sanitizing (HRS) which is a countermeasure against phishing. When a phishing prevention system focuses on reducing false negative errors, false positive errors would increase even if the system employs MLBDMs. My proposed HRS is designed to reduce the loss of convenience arisen from false positive errors. The key idea of HRS is removing all input forms from the sites. While users can browse the rest of content, the loss of convenience would be lower than the existing method which filters whole suspected web pages. I implement HRS-capable proxy servers and verify the function of removing by browsing 100 actual phishing sites. The performance overhead is 3.59 millisecond given content size is 10Kbytes and the content involves 13 HTML tags to be removed. I also compare HRS among existing countermeasures.

Finally, I discuss the development of MLBDM-capable phishing prevention systems. In order to clarify the discussion, I show the stakeholders of the system and the data flow in the system. I then introduce 3 types of implementation forms, named a security service provider (SSP)-side model, a client-side model, and a collaborative model. I also assume that forms of implementation should be selected after due consideration of both the system resources and the set of the heuristics. The SSP-side model would be suitable for thin clients such as handheld devices, and the client-side model would facilitate to develop HumanBoost. Although there are pros and cons between these 2 models, the collaborative model is designed to cover their disadvantages.

This dissertation demonstrates that machine learning algorithms are available for detecting phishing sites. Since MLBDMs contribute to improve the detection accuracy, users will believe that the notification from detection methods. Accordingly, users can easily avoid phishing sites.

Keywords:

Phishing, Web Spoofing, Machine Learning, Trust Decision, Sanitizing

Acknowledgments

My deepest thanks to my adviser and Committee Chair, Professor Suguru Yamaguchi for their invaluable scholarly advice, inspirations, and guidance that helped me through my Ph.D. dissertation works. I will always be indebted to him for all that he has done for me during. Thank you very much for being such a good supervisor.

I wish to thank Professor Hideki Sunahara for his helpful support on my committee. His useful comments helped me to finish my work. I would like to acknowledge the advice and support from my external examiner, Dr. Hiromitsu Takagi of National Institute of Advanced Industrial Science and Technology. His kind support encouraged me to make my research a fruitful one. My special thank to Associate Professor Youki Kadobayashi for his very long sustained support of my work.

I would thank Assistant Professor Hiroaki Hazeyama. Without the discussion about the countermeasures against phishing attacks, I could not write this dissertation. I also thank Assistant Professor Takehshi Okuda, Assistant Professor Shigeru Kashihara. Their continuous support pushed me to my goals. I would thank Assistant Professor Teruaki Yokoyama of Cyber University. He provides me many points of view which I could not have.

I also thank Professor Masayuki Jimichi of Kwansei Gakuin University. He is my first teacher on both Statistics and Internet when I was in Kwansei Gakuin University as a Bachelor's Degree. He also gave me a time for discussing machine learning algorithms.

During my school days, I had a lot of fan and was encouraged by Takuji Iimura. Through our discussion, I could contribute to make useful soft wares such as Packter, Nu-Firefox and so on. I could also have a lot of happy memories with Mio Suzuki and his enjoyable words, Kenji Masui and his fruitful words and Masayoshi Shimamura and his intelligible words.

I gratefully acknowledge members of Web Application Security Forum. The experi-

ence on the analysis for Web Application Security gave me skills and let me realize the importance of my research area. I also thank to the member of Accelia Corporation. I could have much experience for web server management.

I am grateful to IPLAB members and my friends. Talking with them was refreshing me and helped to push me toward my goal. I really appreciate their long-lasting friendships. I also thank to StarBED members to assist our team to experiment.

I also thank to the members of Fuzzcat, Event, and Magprin in Fenrir Server at Final Fantasy XI. Especially, my special thanks go to Diea as my avatar, Kanbo and Warumon for inviting me to FFXI world, Ultima for inviting me first wonderful party, Haruzou for inviting me to his great guild, Magni for teaching me how to battle against heavy monsters, Cepter and Crawd for helping me many times, Aizen, Kamikaze, Vesta, and Selfiria for having hardy night at the beginning in the Chain of Promathia, Acacia and Yukimimi for leveling much times, and Duvan for encouraging my graduation.

Finally, I with to thank my family who are always there to cheer me up every time. Their sincere encouragement as well as their love and support have allowed me to gain my degree.

Dedication

To my father, Minoru Miyamoto, lives in Heaven since August 2008, who instills the importance of education above other things in this world.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There is no human who never make mistakes. In the cyberspace, many security issues have been arisen from human errors. For example, buffer overflow vulnerabilities are the result of a programming error. Due to the weakness of human, "phishing", which is a form of identity theft whose targets are computer users, is one of the serious threats in the cyberspace. Phishers, phishing attackers, attract victims to a spoofed web site, a so-called "phishing site", and attempt to persuade them to send their personal information such as user identification, password, debit card number, credit card number, and so much on.

The damage suffered from phishing is growing. In 2005, the Gartner Survey reported 1.2 million consumers lost 929 million dollars through phishing attacks [1]. In the modern survey conducted in 2007, they also reported 3.6 million consumers lost 3 billion dollars [2]. According to the survey, of those consumers who lost money to phishing attacks, 47 percent said a debit or check card had been the payment method used when they lost money or had unauthorized charges made on their accounts. This was followed by 32 percent of respondents who listed a credit card as the payment method, and 24 percent who listed a bank account as the method.

The number of phishing sites is also increasing. According to trend reports published by the Anti-Phishing Working Group [3], the number of the reported phishing sites was 25,630 in March 2008, far surpassing the 14,315 in July 2005 [4].

Phishing attacks can be separated into 5 distinct phases, and key phases are the attraction phase and the acquisition phase. In the attraction phase, a phisher sends victims an email as if it were sent from a legitimate corporation. I explain 5 phases in

```
The security questions and answers of PayPal account were changed
on 21 Nov. 2008.

If you did not authorize this change, please contact us
immediately using the phone number found on the following page:

https://www.paypal.com/uk/login/

Thank you for using PayPal!
The PayPal Team.

Please do not reply to this email. This mailbox is not monitored
any you will not receive a response. For assistance, lo in to your
PayPal account and click the Help link located in the top right
corner of any PayPal page.

--------------------------------------------------
Copyright 1999-2008 PayPal. All rights reserved.
PayPal EmailID PP232
```

Figure 1.1: PayPal Phishing Scams reported on millersmiles.co.uk

Section 3.1.

As an instance of a phishing attack, I present an actual phishing email, which is reported on millersmiles.co.uk [5] and mimics PayPal, as shown in Figure 1.1. The email sender seems to be a legitimate PayPal, and the subject shows "This security questions and answers of PayPal account were changed" to the recipient. The email content recommends the recipients to click the link in the email. Because this email is HTML format, this link in the email points a phishing site even if it seems to be a PayPal site, `https://www.paypal.com/uk/login/`.

The acquisition phase starts when a phisher could attract victims to browse phishing sites. The phishing site is well-designed to be a look-alike of the actual PayPal site, therefore, some victims cannot aware of being deceived. Hence, they enter their

personal information into a phishing site.

1.1 Issues on phishing

The fundamental problem in phishing is that people are deceived. In psychology, many research formulated "Why people are deceived" [6]. Unfortunately, I could not found the consensus for protecting users to fraudulent claims. In cyber security, many researchers tackle the phishing by supporting people to recognize both phishing and legitimate sites. In other words, they attempt to support end users to make trust decision.

Essentially, an SSL should be used for people to make trust decision. When users visit a site which employs a valid SSL certification, the modern browsers appear a padlock icon in the browser window. The padlock icon indicates that the trustworthy third parties had verified the server belongs to the legitimate enterprises. The padlock icon also indicates that users' inputted data are encrypted to prevent anyone from stealing while the data are transferred from a client PC to a server host. Ideally, users should not enter personal information into any sites which do not employ a valid SSL certification. However, some users browse Internet without knowledge of SSL. These users are likely to make trust decision by checking web content of the site.

Unfortunately, a phishing site is well-designed to be a look-alike of the targeted legitimate site except from the padlock icon and the address bar; attracted users easily believe that a site is legitimate, and the users are likely to disclose their personal information to phishers. Moreover, some site requires inputting such information without using a valid SSL certification, even if the site is maintained by legitimate enterprises. In the view of this, it is naturally to assume that users have a habit to enter the password into non-SSL site. Users often assess a site's credibility without the rigorous criteria [7].

So, how can we, security researchers, support users to make trust decision ? Some researchers would answer "By educating web users.", to the question. They have educated many end users and developed various educational materials. Ideally, to avoid the phishing, every user should distinguish between phishing sites and legitimate sites by themselves, and should pay attention to phishing attacks while browsing web sites. Their education would facilitate to do so. Other researchers would answer "By developing new interfaces". They have created new both human and user interface

which people can easily identify legitimate sites. Since they did not adhere the legacy user interface, many interfaces have been developed. These novel interfaces would instill a trusted path for browsing web sites. Other researchers would answer "By notifying that they are just visiting phishing site". If some browser or browser extension could success to inform users that the site is not legitimate, users would not input their personal information. Someone answers "By adding insurance against phishing", that is a way for blocking or limiting the abuse of the stolen password. Some other one answer "By arresting a phisher'. They propose a scheme for tracing phishers' cyber crime [8].

1.2 Issues on detecting phishing sites

My first motivation against phishing is for supporting end users by informing that they are just visiting phishing sites. For doing so, such system that can identify phishing sites and can notice to end users is necessary.

There are two distinct approaches for distinguishing phishing sites. One is URL filtering. It detects phishing sites by comparing the URL of a site where a user visits with a URL blacklist, which is composed of the URLs of phishing sites. If the URL is listed on the blacklist, the site is detected as a phishing site. However, the effectiveness of URL filtering is limited. The rapid increasing of phishing sites hinders URL filtering to work sufficiently due to the difficulty of building a perfect blacklist. Moreover, in the case of distributed phishing attacks [9], which I will explain in Section 2.1, a phisher prepares distinct URLs of phishing sites per each target. It is not difficult for the phisher because there are many bot-installed PCs in the Internet [10] and several tools such as Rock Phish Kit, facilitate to generate phishing sites on these PCs. Registering these types of phishing sites into a blacklist is tedious work.

The other approach is a heuristic-based solution. A heuristic is an algorithm to distinguish phishing sites from others based on users' experience, and a heuristic checks if a site seems to be a phishing site. Based on the detection result from each heuristic, the heuristic-based solution calculates the likelihood of a site being a phishing site and compares the likelihood with the defined discrimination threshold. Different from URL filtering, a heuristic-based solution has a possibility to identify unreported phishing sites. Notice that heuristics only give a hint to detect phishing sites, but do not provide the accurately information. Thus, Heuristics-based solutions usually employ 2

or more heuristics to improve the detection accuracy.

Unfortunately, the detection accuracy of existing heuristic-based solutions is far from suitable for practical use, even if various studies [11–14] discovered heuristics. Zhang et al. [15] mentioned that SpoofGuard [16], which is one of the heuristics-based solutions, identified more than 90% of phishing sites correctly, but incorrectly identified 42% of legitimate sites as phishing. If a heuristic-based solution often makes mistakes and identifies legitimate sites as phishing sites, end users may become distrusting the system. In this case, users may not believe the notification from the heuristics-based solution even if it could correctly identify a phishing site.

1.3 Machine learning-based approach

In this dissertation, I focus on employing machine learning [17] algorithms to calculate the likelihood of being a phishing site. Machine learning is the study of algorithms that enable computers to improve their performance and increase their knowledge base. Research in machine learning has taken place since the beginning of artificial intelligence in the mid-1950s. The first notable success was Arthur Samuel's program that learned to play checkers well enough to beat skilled humans. If it is available for detection of phishing sites, I expect such system which can correctly identify whether a site is phishing or not, rather than identifying a site by humans.

As my preliminary experiment, I test one machine learning algorithm for detecting phishing sites and observe the weak point of machine learning. Based on the result, I evaluate the performance of 9 machine learning algorithms including AdaBoost, Bagging, Support Vector Machines(SVM), Classification and Regression Trees(CART), Logistic Regression(LR), Random Forests(RF), Neural Networks(NN), Naive Bayes(NB), and Bayesian Additive Regression Trees(BART). I also compare the performance of machine learning-based detection methods with that of the traditional method.

1.3.1 Preliminary evaluation

As my preliminary experiments, I employ AdaBoost, the typical one of the machine learning algorithm, for detection of phishing sites. The key feature of AdaBoost is to build a strong classifier by combining several weak classifiers. In the context of phishing sites detection, a heuristics is corresponding to a classifier. Each weak classifier is only

Table 1.1: Detection result by heuristics

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	Type
Site #001	1	0	1	0	1	1	1	1	phishing
Site #002	1	1	1	1	0	0	0	1	phishing
Site #003	1	1	0	0	1	1	1	0	phishing
⋮									⋮
Site #199	1	0	0	1	1	0	0	0	legitimate
Site #200	0	0	0	0	1	0	0	0	legitimate

required to make the correct detections slightly over half the time. I expect AdaBoost and its feature to create strong heuristics-based solution, even if the detection accuracy of each heuristic is not high.

I then prepare my training dataset, which is composed of 50 URLs of phishing sites reported on Phishtank [18] and the same number of legitimate sites. I also construct my testing dataset which is composed of 50 URLs of phishing sites and the same number of legitimate sites. The URLs of the test dataset are different from those of the training dataset.

Next, I employ 8 heuristics, presented by Zhang et al. [19], to check each site. Each heuristic return binomial variable; If a site is deemed as legitimate, heuristics return 0. If not, they return 1. By converting the detection result from each heuristic, I make binary vectors shown in Table 1.1, where $H_1, H_2, \cdots H_8$ denote the 8 heuristics, and Site #001 $\cdots$ Sites #200 denote the identification number of the URLs in my dataset, 1 or 0 denotes the detection result by each heuristic, Type denote the actual condition indicating phishing or legitimate.

AdaBoost is designed for supervised learning. In the supervised learning, the training data consist of pairs of explanation variables such as $H_1, H_2, \cdots H_8$, and desired output such as Type field in Table 1.1. AdaBoost can solve two different types of problem, that is, regression and classification. The output of the function can be a continuous value on the regression problem or can predict a class label of the input object on the classification problem. Of course, my purpose is classifying phishing sites or not.

After the training, I test AdaBoost with my testing dataset and measure the accu-

racy. The false positive rate is 0.0%, the false negative rate is 6.0%. Through my paper, the false positive denotes labeling a legitimate site as phishing, and the false negative denotes labeling a phishing site as legitimate. I also check the detection accuracy in the case of existing combination method for heuristics. The false rate is 4.0%, the false positive rate is 8.0%. By comparing the AdaBoost-based combination method with the existing method, AdaBoost can provide higher detection accuracy.

However, I observe that AdaBoost sometime occurs overfitting [20]. When machine learning algorithms find the very best fit to the training data, there is a risk that they will fit the noise in the data by memorizing various peculiarities of training data rather than finding a general predictive rule. This phenomenon is usually called "overfitting". In my experiment, I find one phishing site was labeled as legitimate by many heuristics which performed better to label other sites. To identify the site, AdaBoost assigns high weights on other heuristics which could label correctly, even if these heuristics could not provide higher accuracy to detect other sites.

To avoid the overfitting, several regularization techniques have been proposed. Regularization often penalizes parameters and/or error terms to thwart the effectiveness of overfitting. These types of penalization is called shrinkage in statistics, is also called weight decay in neural networks. Support Vector Machine [21] uses soft margin to admit classification error in training, and some extension of AdaBoost such as MadaBoost [22], are often used to avoid overfitting. Removing confusing samples [23] from the dataset also helps thwarting the overfitting.

In this dissertation, I employ the most straightforward way to avoid overfitting, that is, increasing the number of samples in training dataset. Accordingly, I attempt to collect much number of phishing sites and legitimate sites in long time period.

1.3.2 Performance evaluation

At first, I implement an automated system to increase the number of URLs in my dataset. The system accesses to Phishtank.com periodically, obtains newly reported phishing sites, checks sites with heuristics, and stores the results from heuristics for later analysis. In my preliminary experiments, I implemented several heuristics, however, some heuristics were not implemented due to the difficulty of implementation. I explain how to implement these heuristics in Section 7.1.

I built my dataset, which is composed of 1,500 phishing sites reported during

November, 2007 – February, 2008, and the same number of legitimate sites. In addition, these 1,500 phishing sites are verified as phishing sites by registered users of Phishtank.com.

In my comparative study, it is important to define reasonable metrics for evaluating detection methods. I decide my metrics along with my 2 requirements for the detection method. One is accuracy. User safety would obviously be compromised if detection methods labeled phishing sites as legitimate. Users would also complain if the method labeled legitimate sites as phishing sites because of the interruption in browsing caused by false alarms. For this requirement, I employ both the f_1 measure(higher is better) and error rate(lower is better). The other is adjusting capability. Basically, there are trade-off between reducing false negative errors and reducing false positive errors. If a user is a novice, who is easily taken in by phishing attacks, detection methods should decrease false negative errors instead of increasing false positive errors. I require detection methods to adjust the discrimination threshold for reducing false negative errors for novices. Conversely, if a user is a security expert, the methods focus on decreasing false positive errors. I also require detection methods to adjust the threshold for reducing false positive errors for experts. For these requirements, I perform ROC analysis. With various discrimination thresholds for a binary classifier system, I plot the true positive rate vs. false positive rate into the graph space. I then estimate the Area Under the ROC Curve(AUC) value and use AUC(higher is better) as my performance metrics of adjusting capability.

Then I employ 9 machine learning algorithms, including AdaBoost, Bagging, SVM, CART, LR, RF, NN, NB, and BART. Based on these algorithms, I test 9 machine learning-based detection method(MLBDM)s and evaluate their performance. I also select the optimal parameters for MLBDMs. My criterion for choosing parameters is to minimize the error rate in training. For decision tree-based machine learning techniques such as RF, I test them using different numbers of trees. I also select the iteration time for AdaBoost, kernel function for SVM, the number of units in hidden layer for NN. To average out the result, I perform 4-fold cross validation 10 times in order to perform evaluation.

The result shows that the highest f_1 is 0.8777 in AdaBoost, followed by SVM(0.8770), BART(0.8765), CART(0.8755), Bagging(0.8751), NN(0.8751), RF(0.8749), NB(0.8735), and finally LR(0.8609). The lowest false positive rate is 06.73% in LR, and the highest is 14.37% in CART. The lowest error rate is 11.96% in AdaBoost, followed by

SVM(12.03%), BART(12.19%), NN(12.21%), RF(12.34%), NB(12.58%), Bagging(12.60%), CART(12.69%), and finally LR(13.08%). The highest AUC is 0.9543 in AdaBoost, followed by BART(0.9540), RF(0.9539), LR(0.9523), NN(0.9518), Bagging(0.9502), NB(0.9486), CART(0.9449), and finally SVM(0.9180). I also compare the results with the existing detection method, and find 8 out of 9 MLBDMs outperform the traditional detection method.

To assess the availability of machine learning, I then modify test conditions by changing the dataset or the set of heuristics. At first, I construct the dataset whose phishing sites are collected in different time period. I used 1,500 URLs of modern phishing sites reported on Phishtank.com during August, 2008 – November, 2008. As a result, the highest f_1 measure is 0.8721, the lowest error rate is 12.49%, the highest AUC is 0.9454, all of which are observed in the case of AdaBoost. By comparing with the traditional detection method, AdaBoost, Bagging, LR, RF, NN, NB, and BART-based detection methods outperform the traditional method. I also let MLBDMs train from the dataset contains phishing sites reported during November, 2007 – February, 2008, and test with that contains newly created phishing sites reported during January, 2009 – February, 2009. In this case, MLBDMs also outperforms the traditional detection method. Second, I test the dataset which contains 1,500 URLs of phishing sites but these are not verified. Actually, this dataset contains 1,277 phishing sites, 223 unknown sites, and 1,500 legitimate sites. In this test, these 223 unknown sites are not verified as phishing sites, but are treat as phishing sites. The result can be summarized that the highest f_1 measure is 0.8581, the lowest error rate is 14.15%, the highest AUC is 0.9342, all of which are observed in the case of AdaBoost. These 7 out 9 MLBDMs also perform better than the traditional detection method. Finally, I change serious of heuristics by adding new heuristics. In another aspect, I check if MLBDMs can incorporate newly developed heuristics. Otherwise my 2 heuristics are not so accurately, MLBDMs can incorporate these heuristics into the existing 8 heuristics. I observe that the highest f_1 and the lowest error rate are 0.8984 and 09.91% in the case of AdaBoost, respectively. I also observe that the highest AUC is 0.9607 in the case of RF. By comparing the traditional detection method, 7 out 9 MLBDMs outperform. I also simulate the effectiveness of newly created heuristic by disabling the TF-IDF-Final heuristic. I compare the performance with and without using TF-IDF-Final heuristics, and the result shows that MLBDMs can incorporate the TF-IDF-Final heuristic. Accordingly, I assume that machine learning is available

for detection of phishing sites.

1.4 Utilization methods for MLBDMs

I also discuss the customizing method of MLBDMs for each user. Aside from phishing, Denial-of-Service(DoS) attacks aim both network routers and hosts. There are several defense systems against DoS attacks, and it is different between the system for network routers and that for hosts. In the view of this, per-user customized system against phishing might be necessary as long as phishers aim various people. I find if novices can accept 25.49% of false positive errors, 95.00% of phishing sites would be blocked. Similarly, if security experts could accept 20.49% of false negative errors, 95.00% sites of legitimate sites would be browsed normally. I also make my preliminary algorithm for deciding threshold, and confirm that changing threshold can customize various strategies against phishing.

Next, I explain HumanBoost, which aims to cover the weak point of human-being by using AdaBoost. The key idea of HumanBoost is employing users' past trust decision as a new heuristic. As my pilot study, I conduct subject within test by calling 10 subjects. Subjects browse 14 emulated phishing sites and 6 legitimate sites, and check if the site seems to be a phishing site or not. By using such types of subjects' judgments as a new heuristic, I let AdaBoost to incorporate the heuristic into existing 8 heuristics. The results show that the average error rate in the case of HumanBoost was 9.5%, whereas that of subjects was 19.0% and that in the case of AdaBoost was 20.0%.

1.5 HTTP Response Sanitizing

Based on the detection results from MLBDMs, phishing prevention system can inform end users that they are just visiting phishing sites. Some systems indicate in some portion of web browser and/or show alerting window by interrupting users' browsing. However, end users can ignore such types of warning. Unfortunately, reducing both false negative error and false positive error is difficult even if MLBDMs perform better, so there is still possibility that end users may distrust the system due to the false alarms. Apart from warning, proposed countermeasures against phishing attacks are categorized as compulsory blocking. Compulsory blocking can filter whole suspected

web pages regardless of users' intention or wrong behavior. However, a user may complain when a legitimate site is blocked.

I find that such system is needed that can also reduce users' inconvenience arising from false positive errors. I propose a method for countermeasure against phishing, which I named HTTP Response Sanitizing (HRS). Instead of blocking the entire web content, HRS removes the HTML tags of web content which may generate input forms, and pads with a warning message as a substitute for the removed input forms. In this paper, I refer to this action of removing and padding as "sanitizing".

After I decided the target of sanitizing described in Section 9.1, I implement several HRS-capable web proxy servers. By using these implementations, I browsed 100 phishing sites to check if all input forms can be removed. The result shows that all input forms are removed; end users cannot input any information into the site whereas they can browse the rest of content. Next, I measure the processing overhead by browsing a web site which is 10K bytes HTML content and contains 13 tags to be sanitized. In addition, this site was mimicked the login page of Morgan Stanley collected in July, 2005. The processing overhead for the phishing site was 3.59 millisecond. When I change the file size from 10Kbytes to 100Kbytes, the processing overhead increase to 43.37 millisecond. However, I confirm that the performance of users' browsing would penalize when users visit phishing sites. Finally, I compare HRS among warning and blocking. When browsing phishing sites, HRS can provide higher safety than warning, and the provided safety of HRS is as same as that of blocking. When browsing legitimate sites, HRS penalize users' convenience, however, the loss of convenience would be lower than compulsory blocking. Thus, I assume that HRS is useful countermeasure against phishing attacks.

1.5.1 Development of MLBDM-capable systems

I explore a suitable way for development of MLBDM-capable phishing prevention systems. The features of the system are that the system requires a dataset, learns from the dataset, detects with the detection algorithms constructed by machine learning techniques, and takes countermeasure.

At first, I explain the stakeholders for detecting phishing sites by showing the users' browsing. I assume that the stakeholders are web client developers such as browser vendors, and trustworthy third parties placed in the Internet such as security service

providers. Next, I also show the data flow in the system by drawing a flow diagram. As a result, I find that there are pros and cons between a web client and an SSP when learning and detection are performed. Finally, I introduce 3 forms of implementation, named a security service provider (SSP)-side model, a client model, and a collaborative model which learns in the SSP and detects in the web client.

I also find that forms of implementation should be selected after due consideration of both the system resources and the set of the heuristics. The SSP-side model is designed to reduce the client load by performing learning and detection in the SSP. The client-model is also designed to facilitate the development of the per-user customized systems such as HumanBoost by performing learning and detection in the web client. Although there are pros and cons between these 2 models, the collaborative model is designed to cover their disadvantages; in the collaborative model, the SSP performs learning for reducing the client load, and the client performs detection for reducing the number of queries sent toward the SSP.

1.6 Contributions

My ultimate goal is that all web users can avoid phishing attacks. Toward the goal, there are various ways, but I approached by improving the accuracy for detection of phishing sites.

This dissertation's contribution can be summarized as follows:

- Confirmation of the availability of machine learning algorithms for detecting phishing sites
  I confirm that machine learning algorithms are available for detection of phishing sites. I tested several machine learning algorithms, and checked if the performance would be worse by modifying the test conditions such as organize of dataset and that of heuristics. In almost all cases, MLBDMs perform better than the existing detection methods.

- Improvement of the detection accuracy in heuristics-based solutions
  There are 2 distinct methods, URL filtering and heuristics-based solution for detecting phishing sites. However, URL filtering cannot deal with newly created phishing sites, and heuristics-based solution often mistakes to identify the site, so end users may distrust the detection results from heuristics-based solutions.

If MLBDM are widely used, the detection accuracy would increase. Accordingly, users can believe the detection results.

- Discussion of the availability of the MLBDMs in future
  I need to continue this work, but I show that MLBDMs can outperform even if I build another dataset which contains URLs of phishing sites reported in different time period. It means that the effectiveness of employing machine learning algorithms would not be temporal.

- Effectiveness of cleansing dataset
  I test both (i) the dataset whose phishing sites are verified as phishing by registered users of Phishtank.com and (ii) the dataset whose phishing sites are not verified. In the case of dataset (ii), I found 223 URLs sites, which are not verified, in 1,500 URLs which are reported during November, 2007 – February, 2008. Be comparing the performance evaluation of (i) with that of (ii), I confirm that cleansing dataset would improve the performance.

Toward the ultimate goal, I find open issues as follows:

- Revealing all phishing sites
  In this dissertation, I employed limited numbers of URLs in web. To remove the bias in my dataset, obtaining two or more anti-phishing databases are necessary. Aside from using existing resources, discovering newly created phishing sites with an MLBDMs-capable web-crawler would give a hint to researchers. To reveal the spear phishing sites, such systems that can pretend novice users to receive phishing emails. For doing so, I would attempt to forge fictional persona, and to expose fake personal information along with the persona.

- Collaboration with other research fields

  I would incorporate other countermeasures against phishing into MLBDMs. For example, if an email is suspected a phishing email, the detection methods for phishing sites should perform strictly. Constructing an interconnect architecture with other countermeasures is my open issue.

In research fields in cyber security, there are many contributions to characterize botnet [10]. Detecting botnet is useful to detect phishing sites, as long as phishing sites are often hosted in bot-installed PCs [24]. Bots have also function of organizing DoS attacks, virus propagation, sending mails such as SPAM, virus attached-emails, and phishing emails. In the view of this, I have already undertaken to detect virus emails by using AdaBoost [25]. I would collaborate with researchers of cyber security.

- Research of human factor
  In the dissertation, I discuss the optimal threshold for each user, and explain my preliminary algorithms. I confirm that changing threshold can provide the different strategies for each user, but not confirm the reasonability of proposed algorithm. I need to perform subject within test for test the effectiveness. I also continue to investigate if users' past trust decision is used as a new heuristic.

  Per user customization against phishing sites is my open issue. Imagine if a person has never use PayPal, he/she would not disclose secret to phishing sites which mimicked PayPal. I assumed that the user would not complain when the sites which seem to be PayPal are labeled as phishing. Such kind of information gives a hint to detect phishing sites for protecting particular people.

- Research of phishing prevention systems
  I develop 2 heuristics but the effectiveness of these heuristics are marginal. Even so, I confirm that MLBDMs can incorporate new heuristics into existing 8 heuristics. I assume that adding novel heuristics would straightly improve the detection accuracy in the case of MLBDMs.

  My proposed HRS must sacrifice users' convenience, but the loss of convenience would be lower than compulsory blocking. I need to determine an index for measuring the convenience, and conducted a test. I also need to develop a notification method for HumanBoost, since it is difficult to reverse the users' trust decision.

1.7 Organization

The rest of dissertation is organized as follows: In Chapter 2, I present my related work and analyze the problem in Chapter 3. In Chapter 4, I introduce the machine learning techniques, and explain heuristics that I use in my evaluation in Chapter 5. I describe my preliminary evaluation in Chapter 6, and evaluate the performance by using 9 machine learning algorithms in Chapter 7. I introduce several utilization methods MLBDMs in Chapter 8, propose a countermeasure against phishing, named HTTP Response Sanitizing in Chapter 9, and discuss the development of MLBDM-capable phishing prevention systems in Chapter 10. Finally I summarize my contributions in Chapter 11.

Chapter 2

Related Work

In this chapter, I briefly explain the tricks of phishers at first, and then explain why the victim fall into such kind of tricks. Next, I show the countermeasure of phishing attacks, including education, user interface for end users, and detection methods against phishing attacks. Finally, I show the online anti-phishing databases and their trends of reported phishing sites.

2.1 Analysis of phishing

There are various research contributions to characterize phishing attacks. In this section, I introduce these contributions aspect from the tricks of phishing attackers.

2.1.1 Rewriting URLs

According to Felten et al. [26], one of the phishers' tricks is to rewrite all of the URLs on some web page so that they point to the phishing sites rather than to some legitimate websites. For example, `http://www.attacker.org/http://home.netscape.com` is a URLs that points `www.attackers.com`, but end users would think that they are just visiting `home.netscape.com`.

In another case, the phisher employ IP address instead of showing the domain name. For example, if phisher copied the content of `paypal.com`, the legitimate enterprise, into `angelfire.com`, the free web space, and also prepare the URLs like below.

`http://204.238.155.37/biz2/headlines/topfin.html`

IP address "202.238.155.37" points the `angelfire.com`, but end users would not be aware and misunderstand that this site as legitimate. Moreover, they show the another case which abuse URL scheme a follows:

`http://paypal.com:biz@3438189349/headlines/topfin.html`

In this case, "paypal.com:biz" is not treat as a hostname, but as a username and password, and "3438189349" is a hostname. This number is decimal number of "202.238.155.37", and so the URL points `http://204.238.155.37/biz2/headlines/topfin.html`. Accordingly, end users browses this site no awareness of being just visiting a phishing site.

Fette [27] also surveyed that some phishing emails have a link to pages by an IP-address. Some phishing sites are hosted in compromised PCs. These machines may not have DNS entries, and the simplest way to refer to them is by IP address although legitimate enterprises rarely link to pages by an IP-address.

2.1.2 Confusing URL

A spoofed website is typically made to look like a well known site with a slightly different or confusing URL. Phishers will register a similar or otherwise legitimate-sounding domain name such as paypal.com or paypal-update.com are increasingly common.

The one of the modern phishing techniques is IDN Spoofing [28, 29]. The phishing site `bq--abyoc6lq4bwa.com` is evidently different from `www.paypal.com`, however, it can be shown as a `www.páypàl.com` in users' web browsers. Anthony et al. also indicated [30] that the letter "a" in Cyrillic alphabet is quite similar to the letter "a" in the alphabet.

Type Jacking directs users to a phishing site by making a typo when typing in the URL for a domain. For example, the attacker will substitute the letter "L" in a URL with the letter "K" which resides next to the "L" on the keyboard. If the end-user were to mistakenly type the wrong letter in that exact location within the URL, they will be taken to a fraudulent website.

2.1.3 Mimicking legitimate sites

Phishers also have tricks for generating their phishing sites for convincing victims. According to [24], one of the most successful vectors for gaining control of customer information and resources is through man-in-the-middle attacks. In this case, a phisher

situates themselves between the customer and the real web-based application, and proxies all communications between the systems. From this vantage point, the attacker can observe and record all transactions. This form of attack is successful for both HTTP and HTTPS communications. The customer connects to the phisher's server as if it was the real site, while the phisher server makes a simultaneous connection to the real site. The phisher's server then proxies all communications between the customer and the real web-based application server, typically in real-time.

A popup-window can be used as a trick. This trick displays the real site in the browser but puts a border less window from the phishing site on top to request user's personal information. Even though the user is at the legitimate web site, this popup will claim to be a "Security Confirmation" box for the user to login to. Their credentials are then harvested by the popup code and sent to the phisher. In addition, if the victims use ancient web browser, a phisher can easier hide the address bar in the popup-window, whereas the modern web browsers have a function for showing the URL to users.

2.1.4 Targeting the particular person

Jakobsson et al. [14] explained the model of phishers' activities. Basically, when phisher aimed the particular person, phisher should know or guess the person's information such as banks. Then prepare the phishing sites to persuade victims to input their personal information. Jakobsson also explained the distributed phishing attacks [9], commonly known as spear phishing. The phisher works by a per-victim personalization of the location of sites collecting credentials and a covert transmission of credentials to a hidden coordination center run by the phisher. Jakobsson showed how distributed phishing attack can be simply and efficiently implemented and how it can increase the success rate of attacks while at the same time concealing the tracks of the phisher.

Kuo [31] alerted that phishers often aim children rather than parents. In background, children likely use SNS such as MySpace. MySpace is an SNS that is particularly popular with the under-21 age group. In the second half of 2006, there were two major phishing attacks against MySpace users. In addition to MySpace-targeted phishing scams, there has also been a dramatic increase in Trojans that target massive multiplayer online (MMO) games, specifically World of Warcraft (WoW), the most popular of the genre. To infect victims' PC by such Trojans, phisher can steal the

victims' accounts and can gain the game money used in WoW.

2.1.5 Poisoning

DNS cache poisoning [32] may be used to disrupt normal traffic routing by injecting false IP addresses for domain names. For example, the attacker poisons the DNS cache of a victim's DNS server, so that all traffic destined for the PayPal's IP address now resolves to the phisher's server. Pharming is well-known techniques of phishing. In pharming attack, phishers try to control end users' node to modify its hosts file [33]. If pharmers, pharming attackers, succeeded to control a hosts file, then registered a line like this: `204.238.155.37 paypal.com`. By adding this line into the host file, the PC would connect IP address 204.238.155.37 when end user browse paypal.com, rather than obtaining legitimate IP address of `paypal.com`.

Search engine poisoning also directs victims to a website by showing up in early in search engine results. This could be for commonly used terms, could be used in combination with typo's in search terms.

2.1.6 Social engineering

Social engineering is a significant problem involving technical and no technical ploys in order to acquire information from unsuspecting users. Social engineering may involve both psychological and technological ploys in order to leverage the trust of the target. Karakasiliotis et al. [34] described the phisher can exploit characteristics of human behavior in order to increase the chance of the user doing what is desired. Cialdini introduced 6 weapons of influence [6], which is 6 tendencies of human behavior that may influence compliance with a request, namely authority, scarcity, liking, reciprocation, commitment and social proof.

Mosley [35] analyzed some psychological factors of successful phishing. The legacy phishing emails said "Hey buddy - want to buy a Rolex ?". This emails intended to input victims' personal information for pandering victims' greed. People' innate resistance to being motivated by greed, such as, "Why is the cheap Rolex watch not actually ticking ?", prevented themselves for being deceived. However, recent phishing emails said "Verify your existing data or be terminated" or "Please help us update our record" or "Confirm your account credentials to protect from fraud". The resistance cannot come into play.

2.2 Analysis of victims

The targets of phishing attacks are end users, so there have various contributions to analyze end users and their activities. Fogg et al. [7] analyzed 2,684 people and found that when people assessed a real web site's credibility they did not use rigorous criteria. 46.1 percent checked by design look of the site, 26.5 percent site's design and/or structure. Ye et al. [36] also stated that end users would convince by the content of HTML and URL, regardless of checking SSL padlock icons.

According to Kumaraguru et al. [37], there are the difference in the model for making trust decision between novices and experts. In comparison to experts, novices tended to receive meaningless signals when they made trust decision. Novices also ignored some signals such as SSL, address bar, and so on, where experts received these signals.

Dhamija et al. [38] reported their subject within tests for identifying phishing sites. They found that phishing caused of lack of knowledge. For example, subjects thought www.ebay-members-security.com belongs to www.ebay.com due to the lack of system knowledge. Also, many subjects did not understand security indicators. They did not know that a closed padlock icon in the browser indicates that the page they are viewing was delivered securely by SSL. Even if they understand the meaning of that icon, users can be fooled by its placement within the body of a web page. They also found that the best phishing websites fooled 90% of participants. The URL of the site is "www.bankofthevvest.com", with two "v"s instead of a "w" in the domain name.

Wu et al. also measured the effectiveness of security toolbars, which informs end users that they are visiting phishing sites [39]. They tested 3 types of security toolbars, namely, Neutral-information toolbar such as NetCraft Toolbar [40], SSL-Verification toolbar such as TrustBar [41], System-Decision toolbar such as SpoofGuard [16]. Each of the 3 security toolbars was tested with 10 subjects, and they browsed both phishing sites and legitimate sites with one security toolbar, and they also classified the site was phishing or not. Wu et al. concluded that all toolbars failed to prevent users from being spoofed by high-quality phishing attacks. Users failed to continuously check the browser's security indicators, since maintaining security was not the user's primary goal. Although users sometimes noticed suspicious signs coming from the indicators, they either did not know how to interpret the signs or they explained them away.

Aside from these reports, Jakobsson et al. [42] mentioned their pivotal observations:

People look at URLs. In 2007, they conducted a test with 17 subjects and succeeded, and observed that subjects looked carefully at URLs of web pages, and on the URLs obtained by mouse-over in emails. Subjects were also good at detecting IP addresses as being illegitimate, but were not highly suspicious of URLs that were well-formed, such as www.chase-alerts.com. On the other hand, subjects were good at detecting syntactically peculiar addresses such as www-chase.com.

Possible reasons for the difference among earlier research included the demographics of the subjects and substantial media coverage about phishing and identify theft in the intervening 2 years. However, I found some biased in the subjects; all subjects were college students and university staff and faculty. Even if Jakobsson et al. excluded computer science students and staff/faculty with a computer science background, however, there were still bias.

2.3 Countermeasures against phishing

In this section, I show existing countermeasures against phishing. As I mentioned in Chapter 1, there have many research contribution against phishing attacks. Especially, I introduce education, human interface, and detection methods against phishing.

2.3.1 Education for end users

While the phishing problem is caused by person's knowledge, education is one of the straightforward ways to counter phishing. There were provided much number of educational materials . For example, Merve et al. [13] proposed educational material prevention techniques and a strategy on preparing to avoid phishing attacks. Their educations included that "Visit web sites by typing the URL into your address bar", "Use spyware detection tools to detect any intrusion by spyware" and so on. Web-based training materials, contextual training, and embedded training have all been shown to improve users' ability to avoid phishing attacks.

Despite claims by security and usability experts that user education about security does not work [43], there are some evidence that well designed user security education can be effective. Kumaraguru et al. proposed to employ a comic as an educational material [44]. They were tested the educational effectiveness of 30 subjects with 3 types of educational materials. There were 10 participants in each material; The group(i)

received typical security notices, the group(ii) studied with a text and graphics, and group(iii) studied the comics for identifying phishing sites. Their results suggested that the current practice of sending out security notices(group(i)), was ineffective. Their results also indicated that that their comic strip format(group(iii)) was the more effective than the text and graphics(group(ii)).

Sheng et al. [45] found that the game is a novel educational material. The main character of the game was Phil, a young fish living in the Interweb Bay. Phil wanted to eat worms so he can grow up to be a big fish, but has to be careful of phishers that try to trick him with fake worms (representing phishing attacks). Phil is rewarded with 100 points if he correctly accesses a legitimate site or correctly rejected a phishing site. He was slightly penalized for rejecting a legitimate site by losing 10 seconds off the clock for that round. He was severely penalized if he accessed a phishing site and was caught by phishers, losing one of his 3 lives. They developed this scoring scheme to match the real-world consequences of falling for phishing attacks, in that correctly identifying real and fake web sites was the best outcome, a false positive was the second best, and a false negative was the worst. They conducted the total correctness of subjects' classification before and after the education. By using this game, the correctness was increased from 69%, before the education, to 87%. In the case of using existing training materials, the correctness was increased from 66% to 74%.

Anandpara et al. [46] gave me another point of view: They mentioned that phishing IQ tests fail to measure susceptibility to phishing attacks. They conducted a study where 40 subjects were asked to answer a selection of questions from existing phishing IQ tests in which I varied the portion (from 25% to 100%) of the questions that corresponded to phishing emails. Their tests had 3 distinct stages. (i) They performed a first phishing IQ test, then performed (ii) phishing education for subjects, and finally they performed (iii) a second phishing IQ tests, and analysis. They did not found any correlation between the actual number of phishing emails and the number of emails that the subjects indicated was phishing. The number of times that subjects labeled a stimulus as phishing increased from the first to the second test for most subjects. Accordingly, they thought that the tests did not measure the ability of the subjects.

Ideally, to avoid the phishing, every user should distinguish between phishing sites and legitimate sites by themselves, and should pay attention to phishing attacks while browsing web sites. In the view of this, education is the remedy, however, the effectiveness of education was limited to the small number of users; there can be much number

of users who were not educated. Moreover, in real life, security is rarely user's primary goal [47]. The user is primarily concerned with other tasks, such as reading mail, buying a book, or editing a document. Avoiding disclosure of passwords or personal information may be important, but it isn't foremost in the user's mind.

2.3.2 Interfaces for end users

Another approach is to develop human and/or user interfaces for users to identify the site correctly. Due to the smallness of the padlock icon in the current web browser, users receive weak signal for SSL. There are some approaches to reinforce the signal. For example, Herzberg et al. developed TrustBar [41] for indicating the information of the site. If a user installs the TrustBar, user's web browser shows the information in the browser area. Aside from a small padlock icon, this area is highly visible. However, such toolbars often alerted when end users browsed the legitimate sites which were not used SSL, so subjects thought the notice of the toolbar can be ignored [39]. Extended Validation SSL(EV SSL) certificates also present information prominently to users. If a site used EV SSL certificates, the background of the address bar in users' browser turned from white to green; it can be useful to notice that users are visiting legitimate enterprises.

To protect personal information such as a password, Kinda et al. [48] proposed to new user interface which extends users' web browser. This extension can register the password of the site that users once inputted into the site. If the user inputted the same letters of the registered password into another site, then the extension informed users to that the site just users inputting the password was different from the site users once registered the password. Ross et al. [49] created a browser plug-in called PwdHash. This application hashes the password entered by a user with the domain name of the site to which the password is being transmitted. However, many web users have habit to share one password for roughly 6 different sites [50]. Even if users installed these extensions, they would have a habit to ignore the alert from these extensions. It cannot be useful to protect users who have habit of ignoring alert. In addition, PwdHash does not defend against pharming.

Some types of multi-factor authentication reinforced the existing authentication scheme. For example, America Online's Passcode program was proposed as a phishing defense [51]. This program distributes RSA SecureID devices to AOL members. The

device generates and displays a unique 6 digit numeric code every 60 seconds, which can be used as a secondary password during login to the AOL website. This scheme reduces the value of collecting passwords for attackers because the passwords can not be used for another transaction.

Probi et al. [52] proposed to employ PIN/TAN-approach with a paper-based challenge-response-technique. It differed from PIN/TAN in the way that the user gets a list with challenge-response-pairs instead of a list with TANs. Before completing a transaction a challenge was presented to the user who must enter the corresponding response. If the response was incorrect the transaction was not carried out.

In Passmark and Verified by Visa [53, 54], a user provides the server with a shared secret such as and image and/or pass phrase, in addition to his regular password. The server presents the user with this shared secret, and the user is asked to recognize it before providing the server with his password. In the Passmark scheme, the bank server places a secure cookie on user machine, which must be presented at login. This prevents a classic man-in-the middle (MITM) attack where an attacker interposes himself between the client and the bank.

Ye et al. proposed Synchronized Random Dynamic(SRD) boundaries to make trusted path for users' browser [55]. This scheme uses a random number generator to set a bit that determines whether the browser border is inset or outset. The browser border alternates between inset and outset at a certain frequency in concert with a reference window. Within this scheme, phishers who did not know the random number could not provide content that changed at the right time.

In YURL solution [56], user's browser maintains a mapping of public key hash to pet name. When the user visits a page identified by YURL, the browser displays the pet name the user previously associated with the public key hash contained in YURL. If no such association exists, the browser allows the user to create one. After a site transition, the user verifies that the expected pet name is displayed. The absence of a pet name indicates the absence of a trust relationship between the user and the site.

Dhamija et al. [57] verified the authentication and anti-phishing schemes their 5 requirements. The scheme should be easy for a particular class of computers to pass, and it also should be hard for other computers to pass. A scheme can meet if a specified server can reliably authenticate itself to the user without extraordinary resources and if it is difficult for illegitimate computers to masquerade as the legitimate server, even after observing a number of successful authentications. Moreover, the scheme should

Table 2.1: Analysis of authentication and anti-phishing schemes using HIP criteria

Scheme	Easy for specific computer ?	Hard for other computers?	Easy for humans to verify ?	Protocol available ?	Tools required
SSL	Yes	No	No	Yes	modified browser
Trustbar	Yes	No	No	Yes	modified browser
PGP	Yes	Yes	No	Yes	PGP client/plug-in
3rd Party Seals	Yes	No	No	No	No
AOL Passcode	Yes	No	No	No	SecureID device
SMS Passwords	Yes	No	No	No	cell phone SMS
Passmark	Yes	No	No	No	secure cookie
SRD	Yes	Yes	No	Yes	modified browser
YURL	Yes	No	No	Yes	modified browser
DSS	Yes	Yes	Yes	Yes	modified browser

produce results that are easy for a human to verify, it should employ a protocol that is publicly available, and it should not require the user to have specialized tools. The results can be summarized in Table 2.1. In short, they stated that no tools were suitable. Thus, they [58] proposed Dynamic Security Skin(DSS) to meet all requirements.

The DSS scheme allows the remote server to generate a unique abstract image for each user and each transaction. This image creates a "skin" that automatically customizes the browser window or the user interface elements in the content of a remote web page. Their extension also allowed the user's browser to independently compute the image that it expected to receive from the server. To authenticate content from the server, the user can visually verify that the images match.

Aside from the web server authentication, the client authentication also studied. Fu et al. [59] provided a description of the limitations, requirements, and security models specific to a web client authentication. They presented a set of hints on how to design a secure client authentication scheme, based on experience gained from our informal survey of commercial schemes.

Using client-side SSL is generally considered the most secure option for web authen-

tication. In the SSL specification, a server can also request client-side authentication, where the client also presents an X.509 certificate and proves knowledge of the corresponding private key. Using client-side SSL, servers can identify users with her SSL public key and authenticate them using the SSL protocol. However, several contributions have shown there were significant usability problems with client-side SSL [60–62]. To use client-side SSL, users must import a certificate and corresponding key pair into their browser. Legitimate sites may provide users these certificates or users may be required to obtain their own certificate signed by a certificate authority. Regardless, studies have shown that obtaining and installing a certificate and key pair is a cumbersome and confusing process for users, and when users make mistakes, they are hard to correct.

A cookie is also used to web authentication, but they are vulnerable to pharming. Pharmers create an environment where users' browser directed to the web server legitimately associated with a particular domain instead connects to a spoofed site. Pharmers can then harvest the cookies associated with the attacked domain. Cached cookies [63] and Locked cookies [64] were designed to prevent pharmers from harvesting cookies.

Oiwa et al. proposed [65] mutual HTTP authentication. They explained that both users and servers must be authenticated to detect phishing, so employed Password Authenticated Key Exchange for web authentication. This method also provides the trusted area to input a pairs of username and password, instead of using web input forms. If they could instill the method in all web users, the threat of phishing would be thwarted. For doing so, they implemented extension modules for Apache and Firefox to deploy this method in both web servers and web clients.

2.3.3 Detection of Phishing Attacks

There have been various anti-phishing extensions for web browsers. The CallingID Toolbar [66] performs 54 different verification tests in order to determine the legitimacy of a given site. The Cloudmark Anti-Fraud Toolbar [67] relies on user ratings. When visiting a site, users have the option of reporting the site as good or bad. The EarthLink Toolbar [68] are based on users' report for suspected phishing sites to EarthLink. The eBay Toolbar uses a combination of heuristics and blacklists. When the user visits a site known to be operated by eBay or PayPal, the eBay Toolbar indicates that the

sites are safety. GeoTrust's TrustWatch [69] works with several third parties' reputation services and certificate authorities to verify sites as trusted. The Netcraft Anti-Phishing Toolbar [40] uses legitimacy of a web site. The Netcraft web site explains that the toolbar "traps suspicious URLs containing characters which have no common purpose other than to deceive," "enforces display of browser navigation controls (tool & address bar) in all windows, to defend against popup windows which attempt to hide the navigational controls," and "clearly displays sites' hosting location, including country helping you to evaluate fraudulent URLs". The Netcraft toolbar also uses a blacklist, which consists of fraudulent sites identified by Netcraft as well as sites submitted by users and verified by the company. SpoofGuard [16] is different from other toolbars; it does not employ blacklist. The toolbar is a heuristics-based solution, that is, checks a site with a set of heuristics to identify phishing pages. To detect phishing attacks, SpoofGuard uses 3 groups of tests: stateless methods which evaluate a downloaded page, stateful methods that evaluate a page with respect to user activity, and post data from the page. Based off of how the page scores with these methods, the total spoof score is calculated as a standard aggregation function, summing products of pairs, triples and larger subsets as well as individual test results, because certain combinations of attributes make a page drastically more suspicious. SpoofGuard applies these tests to all downloaded pages and combines the results from heuristics using.

Aside from anti-phishing extensions, the modern web browser such as Microsoft Internet Explorer 7, Firefox 2.0, Netscape 8.1 or later have the function of distinguishing phishing sites from others. These browsers usually employ blacklist, also accepts users' report if the users deems the website as phishing. These web browser vendors verified the users' report and add them into their blacklist.

Zhang et al. [15] evaluated the detection accuracy of such kind of anti-phishing tools. The result can be summarized in Table 2.2, where NA denoted that they did not check the sites with the specified tool. The number denoted the successful rate of identifying phishing sites. When a phishing site was reported, and they checked the phishing site with these anti-phishing tools before 1 hour past. They tested 100 phishing URLs reported on Phishtank [18] in November 4-5, 2006 and same number of phishing URLs reported on APWG in November 21 and 27, 2006. The results indicated almost anti-phishing tools depend on blacklist is not successful to identify phishing sites. The successful rate of SpoofGuard was higher than other tools, however, Zhang et al. explained that SpoofGuard incorrectly classify legitimate sites as phishing.

Table 2.2: Number of phishing sites correctly identified by anti-phishing tools

	reported on Phishtank.com	reported on APWG
CallingID	NA	23%
Cloudmark	68%	22%
EarthLink	83%	54%
eBay	28%	52%
IE7	68%	75%
Firefox	NA	28%
Firefox/Google	70%	53%
Netcraft	77%	60%
Netscape	8%	31%
SpoofGuard	91%	96%
TrustWatch	49%	44%

They prepared a list of 516 legitimate URLs to test false positives, and they observed that 42% of legitimate sites was classified as phishing in the case of SpoofGuard.

To improve the detection accuracy in the case of heuristics-based solution, Zhang et al. contributed to propose a novel heuristic, named TF-IDF-Final heuristic. I explain these heuristics that they use in Chapter 5. They incorporated TF-IDF-Final heuristic to existing heuristics drawing primarily from SpoofGuard and PILFER [27], and implemented anti-phishing tools, named CANTINA [19]. In CANTINA, each heuristic returns 1 binomial variable. Based on the result of each heuristic, CANTINA calculates the likelihood of being a phishing site (L) by weighted majority as shown in Equation 2.1. They then classified the site by comparing L with the discrimination threshold.

$$L = \sum W_i * h_i \tag{2.1}$$

Zhang et al. mentioned that a heuristic should have high accuracy in detecting phishing sites while also having a low false positive rate. Accordingly, they assigned weight by calculating the true positive rate minus the false positive rate. Given the

effect e_i of each heuristic, they calculated each weight proportionally, that is:

$$W_i = \frac{e_i}{\sum e_i} \tag{2.2}$$

They let all heuristics to classify if the site deems a phishing or not by using training dataset, which composed of 100 phishing sites reported on Phishtank from November 15-16, 2006, and the same number of legitimate sites. They then assigned weights for each heuristics and tested another 100 phishing sites reported on Phishtank from November 17-18, 2006, and the same number of legitimate sites. The result showed that the true positive rate of CANTINA is 89% and the false positive rate is only 1%. CANTINA did not use blacklist or whitelist, however, it could achieve the higher accuracy in comparison to other anti-phishing tools.

To improve the detection accuracy, some researchers proposed to employ machine learning algorithms. Machine learning can facilitate the development of algorithms or techniques by enabling computer systems to learn. Aside from phishing sites, machine learning has already been used to filter phishing emails. PFILTER, which was proposed by Fette et al. [27], employed SVM to distinguish phishing emails from other emails. According to [70], Abu-Nimeh et al. compared the predictive accuracy of several machine learning methods including LR, CART, RF, NB, SVM, and BART. They analyzed 1,117 phishing emails and 1,718 legitimate emails with 43 features for distinguishing phishing emails. Their research showed that the lowest error rate was 7.72% in the case of Random Forests. Basnet et al. [71] performed an evaluation of 6 different machine learning-based detection methods. They analyzed 973 phishing emails and 3,027 legitimate emails with 12 features, and showed that the lowest error rate was 2.01%. Nevertheless the experimental conditions were different between [70] and [71], the machine learning provided high accuracy for the detection of phishing emails.

A machine learning method was also used to detect phishing sites. According to [72], Pan et al. presented an SVM-based page classifier for detection of phishing sites. They analyzed 279 phishing sites and 100 legitimate sites with 8 features, and the results showed that the average error rate was 16%.

2.4 Online anti-phishing databases

The biggest two online anti-phishing databases are Anti-Phishing Working Group (APWG) [73] and Phishtank [18]. APWG has more than 3000 members from more than 1700 companies and agencies worldwide. Member companies include leading security companies such as Symantec, McAfee and VeriSign. Financial Industry members include the ING Group, VISA, MasterCard and the American Bankers Association. APWG on offers the possibility to submit emails which are considered phishing emails.

Phishtank.com is operated by OpenDNS [74], who provides a DNS resolution service for consumers and businesses as an alternative to using their Internet service provider's DNS servers. Actually, Phishtank offers free submission of suspect URLs considered to be phishing sites. The submission is limited to registered users of Phishtank.com, but anyone can register on it. Due to the free submission to Phishtank.com, some sites are reported phishing sites but they are not. To improve the reliability of anti-phishing database, the reported URLs are validated by registered users of Phishtank.com. They discuss whether the reported sites are really phishing or not.

Phishtank's statistics about phishing activity can be shown in `http://www.phishtank.com/stats.php`, and Figure 2.1 summarizes the data during January, 2008 – December, 2008. It shows that the number of the reported phishing sites reported in 2008, where x axis denotes month and y axis denotes the number of the sites. The normal line denotes the number of reported phishing sites, the dotted line denotes the number of the site which were verified as phishing sites, and the bold line denotes the number of that were not verified as phishing sites. 73.81% of the reported sites were verified as phishing sites, and 4.65% of the sites were not. Other sites were unknown. I assumed the reason is that these sites were expired before registered users attempt to check. In addition, median time to verify the site is 11.0 hours. The reports also showed that 46.87% of verified 158,056 phishing sites were look-alike PayPal, followed by JPMorgan(16.15%), eBay(11.54%), Bank of America(4.59%), HSBC(1.54%), Internal Revenue Service(1.03%), and other sites. It was United States which hosts the most numbers of phishing sites (39.63%), followed by United Kingdom(7.38%), Germany(5.63%), Russia(5.38%), South Korea(3.75%), China(3.63%), and so on.

APWG also published the trend reports of phishing. The latest APWG report [3] presented that the number of the reported phishing sites was 24,908 in March, 2008. 99.48% of the reported sites used HTTP port 80, and 4% of the sites' URL used IP

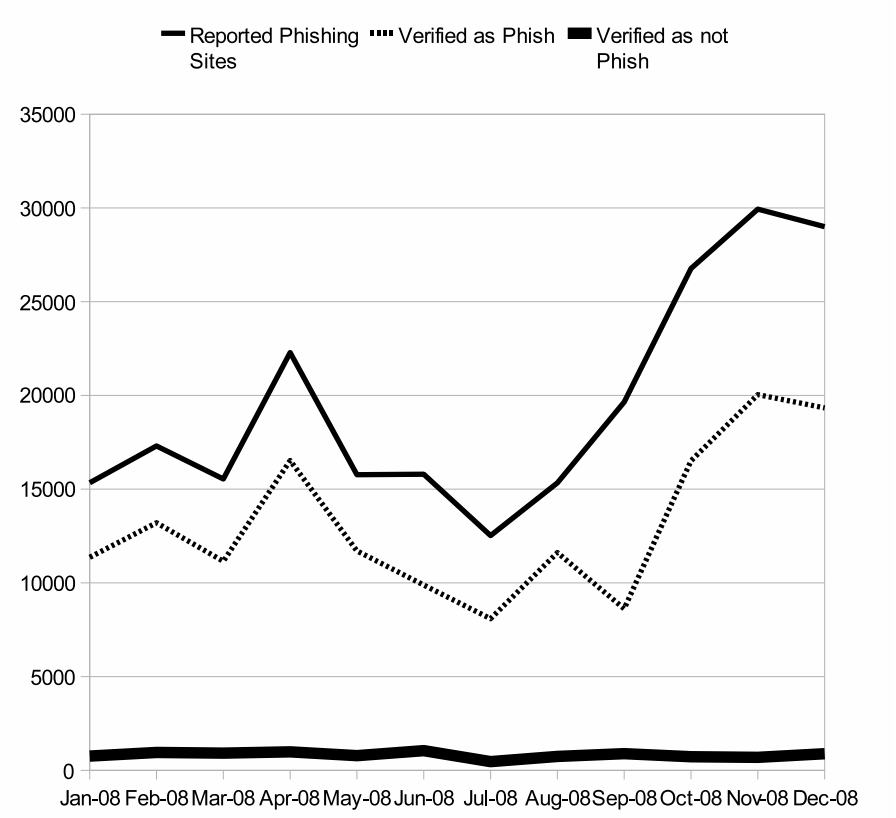

Figure 2.1: The number of the reported phishing sites, verified as phishing sites, and verified as not phishing sites

address instead of using FQDN. 92.9% phishing sites mimicked financial services.

Chapter 3

Problem Analysis

In this chapter, I illustrate phishing attacks aspect from the activities of the phishers. There are various stakeholders and their countermeasure. Based on my examination of these countermeasures, I present the issues on the current countermeasures against phishing attacks.

3.1 Approaches to counter phishing

Table 3.1 shows the activities of both phishers and stakeholders, who can thwart the criminal activities. First, phishers would attempt email harvesting [75], that is, the collection of valid e-mail addresses through automated web-crawlers, called spambots. Spambots scan web pages, mailing lists or chat rooms looking for the @ symbol. In order to avoid harvesting, address-munging (i.e. inserting random text such that spambots cannot recognize email address while humans can) or the use of "at" instead of @ is recommended. Andreolini et al. proposed HoneySpam [76], which can send endless web content against harvesting to slowdown of its process, and can return invalid email addresses for poisoning of phishers' email databases. Moreover, phishers also investigate the particular people's information whenever they try to perform spear phishing. Web users should not expose their e-mail addresses, main banks, credit card companies and so on.

Second, phishers setup their phishing sites. In some cases, phishers penetrate web servers and setup phishing sites in penetrated web server. Unfortunately, there are various bugs in web applications, web application frameworks, and/or programming

Table 3.1: Stakeholders on phishing

Activities of Phishers	Counter to Phishers	Stakeholder
Target Victims	Do not expose email addresses on web sites	Web users
	Set a trap against harvesting	Security service provider
Setup phishing sites	Prevent penetration	Web server administrator
	Fix vulnerability	Web sites developer
	Detect uploading phishing sites	Web hosting service provider
	Prohibit to register confusing domain	DNS registrar
	Hinder to setup phishing sites look alike official sites.	Legitimate enterprise
Send phishing emails (attraction)	Do not click links in emails	Email recipient
	Reject illegitimate sender	Email service provider
	Provide a way to validate emails	Legitimate enterprise
	Filter phishing emails	MUA developer
Let victims to browse phishing sites (acquisition)	Check if a site is a phishing site	Web users
	Provide a way to validate websites	Legitimate enterprise
	Use stronger authentication	Legitimate enterprise
	Detect phishing sites	Security service provider
	Detect phishing sites	Web client developer
Abuse stolen information	Expire stolen information	Legitimate enterprise
	Stop and/or Detect abusing	Legal Enforcement

languages for web applications. The web sites developers should fix such vulnerabilities and server administrator should apply bug fixes to thwart penetration. In other cases, phishers abuse the web hosting service providers. There are free web space which anyone can setup their own web sites; of course, phishers can setup their phishing sites. Web hosting service providers should verify if an uploaded web site is phishing sites or not. Phishers also can prepare phishing sites in boot PCs, and assign the confusing FQDN. According to actual criminal act, the URL "`www.paypaI.com`" is used to deceive users of Papal, `www.paypal.com`. DNS registrar should not accept such types of applications. Legitimate enterprises also should show the symbols which indicate that the sites are legitimate. As I mentioned in Section 2.3.2, these interfaces are designed for being aware of that they are visiting legitimate sites. While these symbols are uneasy to copy, phishing sites cannot have visual appearance as same as that of legitimate enterprises. For example, if all legitimate enterprises' web sites could install an EV-SSL certificate, users would identify the sites by checking whether the background of the address bar in users' browser is green or not.

Third, phishers sends phishing emails to attract targets. The recipients of phishing emails should check the email is trustworthy or not. If not, they should not click any links in the emails. If email server authenticates an email sender and rejects if

the sender is not authorized, then phishing emails would not deliberated to targets as long as phishers are authorized by email servers. For doing so, some methods such as Outbound Port 25 Blocking [77], Selective SMTP Rejection [78] are widely used in Japan. As I mentioned in Section 2.3.3, phishing email filtering schemes in MUA would also useful to eliminate phishing emails. Legitimate enterprises should provide a way for users to validate that the email is legitimate; the consumer should be able to identify that the email is from the institution, not a phisher. To do that, the sending institution must establish a policy for embedding authentication information into every email that it sends to consumers.

Fourth, phishers induce victims to click the link in phishing emails. Web users should check if the site is legitimate or not by checking information from their web browsers. Legitimate enterprises should construct their websites for users to validate that the website is legitimate. If the websites does not ask consumers for sensitive information when logging into the sites, it would be more difficult for phishers to obtain such information. Modern web browsers, web proxy servers and web application firewalls can check if the URL of the sites is registered in the blacklist or not. If these types of web proxy servers work as transparent network proxy servers, phishing sites are not able to browse as long as the URL of the sites is listed on the blacklist. Aside from URL filtering, heuristics-based solutions analyze the web content that are sent from web servers where host phishing sites.

Finally, phishers obtain users' secret such as passwords, credit card numbers, and so on. However, there are still chances for reducing the damage suffered from phishing. Florencio et al. [79] proposed to monitor that the stolen information was used from other PCs. Birk et al. presented a scheme for tracing phishers' cyber crime [8]. They contributed to model the phishers' activity such as money laundering, all of them are performed after they stolen the money by abusing the stolen password. Similar to the entrapment in police, these schemes can help arresting phishers. In order to prevent or report money laundering activities, the financial and legal industries published anti-money laundering softwares [80] to catch up the large cash transaction.

3.2 Issues on current approaches against phishing

My ultimate goal is that all web users can avoid phishing attacks. Due to the difficulties of phishing attacks, I divide phishing attacks into 5 distinct phases to clarify in

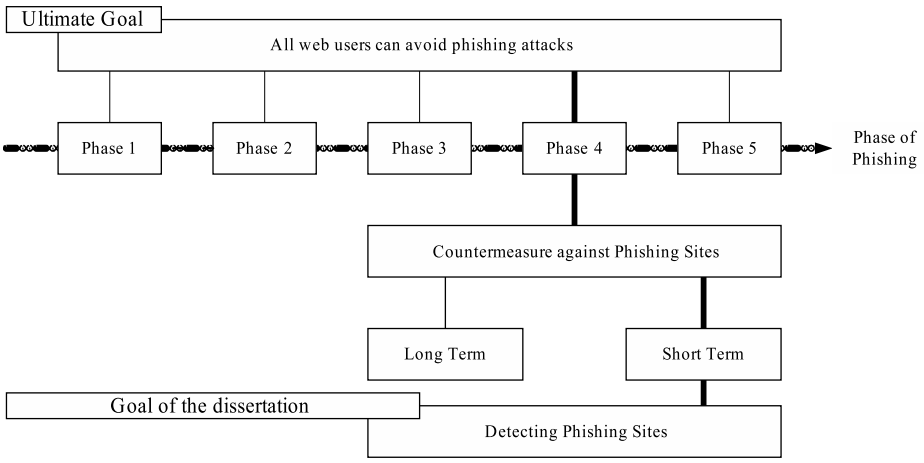

Figure 3.1: The ultimate goal and the goal of the dissertation

Section 3.1 and summarize in Figure 3.1.

Toward this goal, I present issues on each phase.

- Issues on the first phase
  Countermeasure against email harvesting would not help people whose email addresses have been already registered by phishers databases. Such people are required to change their email addresses. Announcing new email address is generally intractable problem.

- Issues on the second phase
  It is uneasy to prohibit phishers to setup phishing sites as long as there are vulnerable PCs or web applications in the cyber space. Detecting and isolating botnet and/or fixing security holes in vulnerable programs are necessary.

- Issues on the third phase
  Phishers can still create domains that appear to be real and register SPF records for those domains. Some validation methods for emails provides some protection, but only if the recipient checks the validity. Filtering phishing emails is useful, however, other sources such as Instant Messaging, facebook and social network services are used to attract to phishing sites instead of phishing emails [81, 82]. According to their scenario, a victim responds to an instant message from someone posing as a legitimate enterprise. The fraudsters persuade victims to visit

their sites.

- Issues on the five phase
  In the case of the countermeasures against abusing stolen information, there are some barriers such as privacy issues on tracing users' secrets and/or authority for tracing across the country. In another aspect, the cache flow caused by phishing are estimated much less than anti-money laundering softwares. Gartner's reported that 3.6 million consumers lost 3 billion dollars [2] in 2007; According to Asao [83], the target of money laundering in AML is large cash transaction, usually 10,000 dollars whereas each victim roughly lost 833 dollars.

In this dissertation, I attempt to solve issues in the fourth phase, that is, preventing victims for entering personal information into phishing sites. The typical approaches are education, developing authenticate and/or user interfaces, and detecting phishing sites.

The effectiveness of education is limited to the small number of users, whereas the number of Internet users is dramatically increasing. Ideally, to avoid browsing phishing sites, every user should distinguish between phishing sites and legitimate sites by themselves, and should pay attention to phishing attacks while browsing web sites. However, considering the growth rate of phishing attacks, many novice users are likely to disclose their personal information into phishing sites.

Developing authentication and/or user interfaces requires long-term effort. In the case of an EV-SSL certificate, there are few servers which have been employed an EV-SSL certificate; there can be numerous servers which were not installed. Developing authentication and/or user interfaces would also take a long time for web users to instill that the turning green in address bar indicates the site is legitimate. In other aspects, developing interfaces indiscriminately would result in users' confusion. Interface developers also have to standardize their interfaces even if it takes a long time.

I hereby focus on detecting phishing sites. Because phishing is a growing problem, I cannot adhere the long-term solution such as education and/or developing interfaces. Aside from these methods, detecting phishing sites has an immediate effect. If a detection method could distinguish phishing sites from other sites accurately, my ultimate goal would achieve immediately.

As I mentioned in Section 1.2, URL filtering methods have a problem that registering the URLs of the sites into URL databases is tedious work. URL filtering

methods also cannot deal with unreported phishing sites, e.g., spear phishing sites. Heuristics-based detection methods have a possibility to identify these sites, whereas the detection accuracy is far from sufficient.

Imagine if the heuristics-based solutions can achieve higher detection accuracy. Every phishing sites can be detected even if the sites were not reported. I will implement the function of heuristics-based solutions into web-crawler to discover unreported phishing sites. In addition, web hosting service providers can also detect that phishers construct phishing sites.

My goal of this dissertation is detecting phishing sites accurately, as shown in Figure 3.1. I find that there are two approaches. One is discovering innovative heuristics. There are various studies [11–14] which discovered heuristics. However, SpoofGuard [16], one of heuristics-based solutions incorrectly identified 42% of legitimate sites as phishing whereas it could identify more than 90% of phishing sites correctly [15]. The other is refining the calculation algorithm of the likelihood are important. I hypothesize that the inaccuracy is caused by heuristics-based solutions that can not use these heuristics appropriately.

My approach is employing machine learning techniques for detection of phishing sites. In order to calculate the likelihood of being a phishing site, I assume that machine learning facilitates the development of algorithms or techniques by enabling computer systems to learn. I find that there are two problems in earlier research for detection of phishing sites with machine learning. One is that the number of features for detecting phishing sites is lesser than that for detecting phishing emails. It indicates that the detection of phishing sites is much difficult than that of phishing emails. Studies on discovering innovative heuristics should be continued. The other is that no research contribution confirmed whether any kind of machine learning based detection methods were available to distinguish phishing sites from legitimate sites. To the best of my knowledge, earlier research tested only one machine learning technique. In this dissertation, I check if AdaBoost, the most typical one of the machine learning, can be applied to detect phishing sites in Chapter 6, and then evaluate 9 machine learning-based detection methods and show their performance in Chapter 7.

Chapter 4

Overview of Machine Learning Techniques

In this chapter, I briefly explain each machine learning technique which is used in my evaluation.

4.1 AdaBoost

Adaptive Boosting (AdaBoost) [84,85] algorithm learns a "strong" algorithm by combining a set of "weak" algorithms h_t and a set of weight α_t:

$$H_{Ada} = \sum \alpha_t * h_t \tag{4.1}$$

The weights are learned through supervised training off-line [86]. Formally, AdaBoost uses a set of input data $\{x_i, y_i : i = 1, \ldots, m\}$ where x_i is the input and y_i is the classification.

Each weak algorithm is only required to make the correct detections slightly over half the time. The AdaBoost algorithm iterates the calculation of a set of weight $D_t(i)$ on the samples. At $t = 1$, the samples are equally weighted so $D_1(i) = 1/m$.

The update rule consists of 3 stages. First, AdaBoost chooses the weight as shown in Equation 4.2.

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{4.2}$$

where $\epsilon_t = Pr_{i \sim D_t} [h_t(x_i) \neq y_i]$. Second, AdaBoost updates the weight by Equation 4.3.

$$D_{t+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if \quad h_t(x_i) = y_i \\ e^{\alpha_t} & if \quad h_t(x_i) \neq y_i \end{cases} \tag{4.3}$$

where Z_t is a normalization factor, $\sum_{i=i}^{m} D_{t+1}(i) = 1$.

4.2 Bagging

Bootstrap Aggregation (Bagging) [87] is a typical ensemble learning method, and its key feature is that dataset is perturbed by resampling with replacement. Given n samples in dataset, bagging selects m (m <n) samples for training and constructs a classifier h. Taking B iterations, it outputs the final classifier by majority vote of $h_1, h_2, \ldots h_B$ as shown in Equation 4.4.

$$f_{bagging} = \text{argmax}_y \sum_{i=1}^{B} (h_i = y) \tag{4.4}$$

4.3 Support Vector Machines

Support Vector Machines (SVM) [21] is also one of the typical machine learning methods for classification and regression. The key idea of SVM is to map data from the input space into a higher dimensional feature space, and to find the optimal separating hyperplane between 2 classes by maximizing the margin between the classes' closest points.

A discriminating hyperplane will satisfy Equation 4.5.

$$\begin{aligned} w'x_i + w_0 \geq 0 \qquad & if \quad t_i = +1 \\ w'x_i + w_0 < 0 \qquad & if \quad t_i = -1 \end{aligned} \tag{4.5}$$

where the distance of any point x to a hyperplane is $|w'x_i + w_0|/||w||$ and the distance to the origin is $|w_0|/||w||$.

4.4 Logistic Regression

Logistic Regression (LR) [88] is a model used for binary data prediction. LR is designed to deal with confounding variables, and its model typically uses the *logit* function as

shown in Equation 4.6.

$$log\frac{p(x)}{1-p(x)} = \beta_0 + \sum \beta_i x_i \tag{4.6}$$

where x is a vector of predictors and β is a vector of regression parameters.

4.5 Classification and Regression Trees

Classification and Regression Trees (CART) [89] is a typical decision tree algorithm. The modeling and prediction within the CART analysis is accomplished through a recursive binary partitioning of a training dataset. The term "binary partitioning" implies that the parent nodes are always split into two child nodes and "recursive" means that the process is repeated by treating each child node as a parent node. This process is repeated until further partitioning is impossible or is limited by some criterion set by the user. When a node data cannot be split into additional child nodes, it is called a terminal node. Once the first terminal node has been created, the algorithm repeats the procedure for each set of data until all data are categorized as terminal nodes.

CART requires a measure of node impurity and generally employs Gini Index as an impurity function. In a node t, the Gini Index criterion assigns a sample to a class c_i with the probability $p(c_i|t)$. The estimated probability of misclassification under this rule is as shown in Equation 4.7.

$$\text{Gini Index} = 1 - \sum (p(c_i|t))^2 \tag{4.7}$$

4.6 Random Forest

Random Forest (RF) [90] is a classifier that consists of many decision trees and outputs the class that is the mode of the classes output by individual trees. In building each decision tree model based on a different random subset of the training dataset, a random subset of the available variables is used to choose how best to partition the dataset at each node. Each decision tree is built to its maximum size, with no pruning performed.

The basic idea is similar to Bagging. The main difference between Bagging and RF is that RF uses a random subset of the available variables whereas Bagging uses

all available variables. So, RF is suitable for handling a very large number of input variables.

4.7 Neural Network

Neural Network (NN) is a non-linear and parallel computation model which is referred to a network of biological neurons. NN has overwhelming strengths in learning ability, auto-adapting ability, generalization performance and anti-noise ability. However, it may be easily influenced by model parameters and varieties of training data, so it has drawbacks of instability and low predicting precision.

The neurons are organized into 3 types of layers. The input layer presents the feature vector of input variables. The next layer is called a hidden layer; NN assumes that there may be several hidden layers. The final layer is the output layer, where there is one node for classification. Since interconnections do not loop back or skip other neurons, the network is called *feedforward*.

4.8 Naive Bayes classifier

The Naive Bayes (NB) classifier is a simple but effective classifier that has been used in numerous applications such as email filtering. Generally, NB's computational time is less than the non-naive Bayes approach because NB is based on Bayes' theorem with the independent feature model. The model of prediction was formulated in Equation 4.8.

$$f_{nb} = \text{argmax}_y P(y)\Pi(x_i|y) \tag{4.8}$$

4.9 Bayesian Additive Regression Trees

Bayesian Additive Regression Trees (BART) [91, 92] is designed for discovering unknown function f that predicts an output Y using a p dimensional vector of inputs x. The basic idea of BART is to model by a sum of regression trees,

$$f(x) = \sum g(x) \tag{4.9}$$

where function g denotes a binary regression tree. Replacing f in Equation 4.9 by modeling or approximating $f(x)$, BART obtains Equation 4.10 where ϵ is the random error.

$$Y = \sum g(x) + \epsilon, \epsilon \sim N(0, \sigma^2). \tag{4.10}$$

Conceptually, BART can be viewed as a Bayesian nonparametric approach which fits a parameter rich model using a strongly influential prior distribution. BART does not require variable selection, which is performed automatically as the trees are built. In addition, in order to fit the sum-of-trees model, BART uses a tailored version of Bayesian backfitting Markov Chain Monte Carlo simulation that iteratively constructs and fits successive residuals.

Chapter 5

Heuristics

In this chapter, I explain heuristics that I used for performance evaluation. These 8 heuristics were also employed in CANTINA [19].

5.1 Age of Domain

The Age of Domain heuristic checks if the domain was registered more than 12 months ago. If the site has been registered more than 12 months, the heuristic deems it a legitimate site, and otherwise it deems it a phishing site.

The weak point of this heuristic was that newly created legitimate sites are not registered in 1 year. In this case, the heuristic would fail. Another weak point is that there are many phishing sites registered on 1 year ago. Especially, the modern phishing sites are often discovered on the host which owned by legitimate enterprises. Some vulnerability in the host allowed phisher to penetrate into and to setup a phishing sites. In this case, the domain name was often registered in long time ago, thus, the heuristic fails to classify correctly.

5.2 Known Images

The Known Images heuristic checks if a page contains inconsistent well-known logos such as eBay, PayPal, and so on. For example, if a site contains the eBay logos but is not on an eBay domain, the heuristic deems this site a phishing site.

In this heuristic, some legitimate sites are labeled as phishing when the site put

the logo files of these 9 legitimate enterprises. And, some phishing sites are labeled as legitimate which mimics the site except from these 9 sites.

5.3 Suspicious URL

The Suspicious URL heuristic checks if a URL of the site contains an "at" symbol (@) or a "dash" (-) in the domain name. If so, the heuristics deems it a phishing site because phishing attackers are likely to use these symbols in the domain name of a phishing site. When "@" is used in a URL, all text before the "@" is ignored and the browser references only the information following the "@" as a hostname. Phishing attackers likely abuse this URL scheme: For example, if `http://paypal.com@phishing.com` is used, web browsers would be directed to the `phishing.com`. Even if it seemed like `paypal.com`, web browsers would ignore this.

The weak point is that some legitimate sites, e.g., `aist-nara.ac.jp`, employs "-" for their domain name. And, several phishing sites are not altogether containing "@" or "-" in the domain name.

5.4 Suspicious Links

Similar to the Suspicious URL heuristic, this one checks if a link on the page contains an "at" symbol or a dash. The weak points of the heuristics are as same as that of the Suspicious URL heuristic.

5.5 IP Address

The IP Address heuristic checks if the domain name of the site is an IP Address. Although legitimate sites rarely link to pages by an IP address, phishers often attract victims to phishing sites by IP address links. The heuristic fails if the URL of a phishing site is FQDN, or that of a legitimate site is IP Address.

5.6 Dots in URL

The Dots in URL heuristic checks if the URL of the site contains 5 or more dots. According to [27], dots can be abused for attackers to construct legitimate-looking URLs. One technique is to have a sub domain. Another is to use a redirection script, which to the user may appear like a site hosted at google.com, but in reality will redirect the browser to phishing.com. In both of these examples, either by the inclusion of a URL into an open redirect script or by the use of a number of sub domains, there are a large number of dots in the URL.

The heuristic fails if the number of dots in URL of phishing sites is less than 5. For example, a phishing site, which was reported in November 30th, 2008 and was placed in `http://kitevolution.com/os/chat6/plugins/safehtml/www.paypal.com/canada/cgi-bin/webscr.php?cmd=_login-run`, includes only 4 dots. Conversely, the URL of some legitimate sites has 5 or more dots.

5.7 Forms

Checking if the page contains any web input forms. In the case of CANTINA, it scans the HTML for `<input>` tags that accept text and are accompanied by labels such as "credit card" and "password." If so, the heuristic deems it a phishing site.

It is not strange for legitimate sites, thus the heuristic labels legitimate sites as phishing. Instead of using such words, employing a digital image which are painted the words are not seems to be a phishing site.

5.8 TF-IDF-Final

This heuristic checks if the site is phishing by employing TF-IDF-Final, which is an extension of the Robust Hyperlinks algorithm [93]. When the heuristic attempts to identify phishing sites, it feeds the mixture word lexical signatures and a domain name of the current web site into Google. If the domain name matches the domain name of the top 30 search results, the web site is labeled legitimate.

However, there is some possibility of manipulating Google's search result by abuse of search engine optimization. Even if the site is legitimate, some of them would not be appeared in higher rank of the search result.

Chapter 6

Preliminary evaluation

In this chapter, I check if the AdaBoost, the most typical one of the machine learning algorithms, is available to detect phishing sites. I also checked that the AdaBoost-based combination method can improve the detection accuracy.

6.1 Implementation of heuristics in preliminary evaluation

At first, I implemented heuristics described in Chapter 5, because CANTINA is not available to download yet. In this preliminary evaluation, I implemented 5 of 8 heuristics that are Suspicious URL, Suspicious Links, IP Address, Dots in URL, and Forms heuristic all of which only analyze the downloaded content or the URL of the site.

I also implemented Age of Domain and TF-IDF-Final heuristic, but some of their functions were not implemented. In the case of Age of Domain heuristic, the format of the WHOIS server differs in different countries, so my implementation derived the domain name from a URL and showed me the search result of WHOIS. In the case of TF-IDF-Final heuristic, I browsed the target URL with Firefox 2.0, inputted all text content to my implementation by copy-and-paste from the browser screen. In the case of Known Images heuristic, it is difficult to judge whether or not the well-known logo is used without browser's rendering support. Thus, I manually performed the checking of Known Images heuristic, and labeled as phishing if the site used well-known logos.

Table 6.1: Assigned Weight by CANTINA

CANTINA	True Positive Rate	False Positive Rate	Effect	Weight
Age of Domain	62.0%	8.0%	54	0.19
Known Images	88.0%	0.0%	88	0.31
Suspicious URL	8.0%	6.0%	2	0.01
Suspicious Link	6.0%	6.0%	0	0.00
IP Address	14.0%	0.0%	14	0.05
Dots in URL	10.0%	0.0%	10	0.03
Forms	88.0%	22.0%	66	0.23
TF-IDF-Final	98.0%	44.0%	54	0.19

6.2 Detection accuracy

My first purpose is to check if AdaBoost is available, and my second purpose is to check if AdaBoost can increase the detection accuracy. For measuring the detection accuracy, I used 3 metrics to evaluate; true positive rate, false positive rate, and total accuracy rate which is calculated by dividing the number of correctly identified sites by the number of all sites in the dataset. Next, I evaluate the validity of my two datasets, which contains 50% phishing sites and 50% legitimate sites.

First, I built a training dataset. I have chosen 50 phishing URLs from Phishtank.com in May, 2007 according to the following requirements: a phishing site which (i) can still be browsed (is not expired), (ii) looks like a well known legitimate site, and (iii) can be labeled as a phishing site by a URL of the site. Next, I have also selected 50 legitimate URLs, top 25 URLs of Alexa [94], 15 URLs listed as Good URLs in 3Sharp [95], 10 URLs chosen randomly from Yahoo! Random Link [96], and all 100 URLs (50 phishing and 50 legitimate) were English language sites. In addition, the criteria for choosing URLs was referred to [19]. Moreover, I have also built a test dataset which was composed of 50 phishing URLs from Phishtank.com and 50 legitimate URLs from 3Sharp. I note that the URLs of the test dataset were different from those of the training dataset.

Second, I applied each heuristic to the training dataset, and assigned the weight to each heuristic. The result of CANTINA's weight assignment is shown in Table 6.1.

$$W_i = \frac{\alpha_i}{\sum \alpha_i} \tag{6.1}$$

Table 6.2: Assigned Weight by AdaBoost

AdaBoost	Total Accuracy Rate	α	Weight
Age of Domain	54.0%	0.03	0.01
Known Images	88.0%	1.38	0.42
Suspicious URL	2.0%	0.00	0.00
Suspicious Link	0.0%	0.00	0.00
IP Address	14.0%	0.23	0.07
Dots in URL	10.0%	0.14	0.04
Forms	66.0%	0.56	0.17
TF-IDF-Final	54.0%	0.95	0.29

Table 6.3: Accuracy of normal CANTINA

	(i)	(ii)	(iii)	(iv)	(v)
True Positive Rate	92.0%	90.0%	92.5%	95.0%	80.0%
False Positive Rate	4.0%	4.0%	4.0%	4.0%	4.0%
Total Accuracy Rate	94.0%	93.3%	94.4%	95.7%	93.3%

Third, I also assigned the weight to each heuristic in the case of employing an AdaBoost algorithm, and the result of weight assignment is shown in Table 6.2. In order to facilitate comparing the weight assignment of AdaBoost with that of CANTINA, the weight α_i was normalized by Equation 6.1, which has same meaning as Equation 2.2. In preliminary evaluation, I used 1 time of iteration for AdaBoost. Notice that AdaBoost has a function to learn iteratively for reducing training errors, and I will explain the optimal number of iteration in Section 7.5. Moreover, in Boosting theorem, heuristics should be applied into a training dataset in order of higher accuracy, so I measured the total accuracy rate on each heuristic. When there are same total accuracy rate among several heuristics, the heuristic with lower false positive rate was applied. In the case of my training dataset, heuristics were applied in order of Known Images, Forms, Age of Domain, TF-IDF-Final, IP Address, Dots in URL, Suspicious URL and Suspicious Links.

By comparing the weight assignment of AdaBoost with that of CANTINA, I found that Known Images heuristic was assigned the highest weight, and Suspicious Links heuristic was assigned zero weight in both cases. However, the assigned weight of Age of Domain heuristic was lower than that of TF-IDF-Final heuristic, otherwise

Table 6.4: Accuracy of CANTINA with AdaBoost

	(i)	(ii)	(iii)	(iv)	(v)
True Positive Rate	94.0%	90.0%	92.5%	95.0%	90.0%
False Positive Rate	0.0%	0.0%	0.0%	0.0%	0.0%
Total Accuracy Rate	97.0%	95.6%	96.7%	98.6%	98.3%

Table 6.5: Accuracy of normal CANTINA and CANTINA with AdaBoost

Algorithm	True Positive Rate	False Positive Rate	Total Accuracy Rate
CANTINA	92.0%	4.0%	94.0 %
AdaBoost	94.0%	0.0%	97.0 %

normal CANTINA assigned higher weight to Age of Domain heuristic than TF-IDF-Final heuristic. This reversion was caused by that almost of all the sites which Age of Domain heuristic correctly labeled had been already identified correctly by Known Images heuristic. Conversely, TF-IDF-Final heuristic often labeled correctly where Known Images heuristic had mislabeled. Thus, AdaBoost assigned higher weight to TF-IDF-Final heuristic and lower weight to Age of Domain heuristic.

Finally, I applied both algorithms to my test dataset and the results(Table 6.5) showed that false positive rate deceased from 4.0% to 0.0%. This means that the AdaBoost-based combination method never labeled legitimate sites as phishing in this case.

True positive rate increased from 92.0% to 94.0%, and the total accuracy rate increased from 94.0% to 97.0%. According to this result, it can be assumed that AdaBoost is available for detection of phishing sites, and the AdaBoost-based weight assignment algorithm has more effective than CANTINA's algorithm to detect phishing sites.

6.3 Percentage of phishing sites in dataset

Essentially, the number of legitimate sites is much larger than that of phishing sites, whereas my dataset mentioned in Chapter 7 contained 50% phishing sites and 50% legitimate sites.

Here, I presented the verification result with 5 pairs of a training dataset and a test dataset. They were composed of (i) 50 phishing sites and 50 legitimate sites, (ii) 40

Table 6.6: Weight Assignment by CANTINA

	(i)	(ii)	(iii)	(iv)	(v)
Domain Age	0.19	0.17	0.19	0.17	0.18
Known Images	0.31	0.30	0.29	0.31	0.26
Suspicious URL	0.01	0.01	0.01	0.00	0.04
Suspicious Links	0.00	0.01	0.00	0.00	0.04
IP Address	0.05	0.05	0.05	0.05	0.06
Dots in URL	0.03	0.04	0.05	0.02	0.03
Forms	0.23	0.23	0.23	0.25	0.23
TF-IDF-Final	0.19	0.19	0.18	0.20	0.16

Table 6.7: Weight Assignment by AdaBoost

	(i)	(ii)	(iii)	(iv)	(v)
Domain Age	0.01	0.01	0.00	0.06	0.00
Known Images	0.42	0.41	0.41	0.51	0.34
Suspicious URL	0.00	0.00	0.03	0.00	0.10
Suspicious Links	0.00	0.00	0.00	0.00	0.00
IP Address	0.07	0.08	0.08	0.00	0.22
Dots in URL	0.04	0.04	0.18	0.00	0.00
Forms	0.17	0.17	0.14	0.16	0.25
TF-IDF-Final	0.29	0.28	0.25	0.28	0.10

phishing sites and 50 legitimate sites, (iii) 30 phishing sites and 50 legitimate sites, (iv) 20 phishing sites and 50 legitimate sites, and (v) 10 phishing sites and 50 legitimate sites. In the case of (i), both the training dataset and the test dataset are the same as those I used in Section 6.2. In the cases of (ii), (iii), (iv) and (v), I have chosen the phishing sites by random sampling manner.

The result of weight assignment is shown in Table 6.6 and that of AdaBoost is shown in Table 6.7. Based on assigned weight, I measured the accuracy of CANTINA as shown in Table 6.3 and that of CANTINA with AdaBoost in Table 6.4. I found the accuracy of CANTINA with AdaBoost is as well or better than that of CANTINA in every case.

I also found that the weight assigned by AdaBoost was concentrated on particular heuristics, unlike that of normal CANTINA. Figure 6.1 showed that the variance of

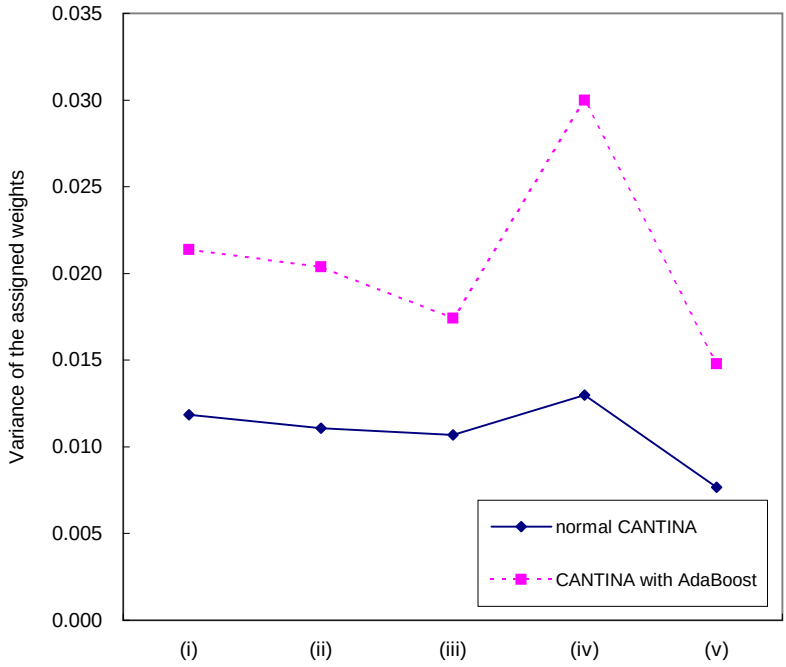

Figure 6.1: Variance of Assigned Weight

each heuristic in the case of AdaBoost is higher than the case of normal CANTINA.

Notably, Known Images heuristic of AdaBoost was always assigned higher weight than that of normal CANTINA. Within my dataset, I observed that Known Images heuristic showed the best accuracy among 8 heuristics; The total accuracy rate of Known Images is 88.0% in the case of (i), and 96.7% in the case of (v). Thus, I assumed that the concentration of the weight on Known Image heuristic has affected the accuracy of distinguishing between legitimate sites and phishing sites.

6.4 Effect of the Known Images heuristic

According to [19], the true positive rate of Known Images was 37.0%, while that of my experiments was 88.0%. I assumed the difference was caused by my manually checking whether or not the site contains well-known logos which were listed in Chapter 5. If I implement Known Image heuristic, it is necessary to have the function of pattern

Table 6.8: Accuracy of CANTINA's weight assignment

	(i)	(ii)	(iii)	(iv)	(v)
True Positive Rate	98.0%	95.0%	97.5%	95.0%	90.0%
False Positive Rate	28.0%	28.0%	28.0%	28.0%	20.0%
Total Accuracy Rate	85.0%	82.2%	83.3%	78.6%	81.7%

Table 6.9: Accuracy of the AdaBoost-based weight assignment

	(i)	(ii)	(iii)	(iv)	(v)
True Positive Rate	98.0%	95.0%	70.0%	65.0%	60.0%
False Positive Rate	20.0%	20.0%	10.0%	10.0%	10.0%
Total Accuracy Rate	89.0%	86.7%	81.1%	82.9%	85.3%

matching in a digitized image, although this might result in many misjudgments.

In my previous experiments, AdaBoost assigned the highest weight to Known Images heuristic, so it can be assumed that the accuracy of the AdaBoost-based combination method depends heavily on the accuracy of Known Images heuristic.

In order to verify whether or not AdaBoost can build a "strong" learning algorithm even if the accuracy of Known Images decreases, I tested the accuracy of detecting phishing sites by combining heuristics which were removing Known logo heuristic and using only the other 7 heuristics. I calculated the weight with the training dataset of (i), (ii), (iii), (iv) and (v), and measured the accuracy. The result is shown in Table 6.8 and 6.9.

In the case of (iii), I observed that the false positive rate of AdaBoost was rapidly decreased and the total accuracy rate of AdaBoost was lower than that of CANTINA. This was caused by overfitting, which I will discuss in Section 6.5. However, the rest of the result showed that false positive rate was decreased by applying AdaBoost, and I find that AdaBoost increased the total accuracy rate in almost of all cases.

6.5 Overfitting problem

In the training dataset of (iii) in Section 6.4, I found that one legitimate site was labeled as phishing by Forms, Age of Domain and TF-IDF-Final heuristic all of which were the top 3 heuristics in accuracy in this case. The AdaBoost-based combination method has checked this site by Forms heuristic, at first. Form heuristic showed the highest

accuracy among 7 heuristics, but could not identify this site correctly . In this case, an AdaBoost algorithm assigns higher weight to the heuristic which is able to label the site correctly, and assigns lower weight to heuristic which labeled incorrectly. Age of Domain and TF-IDF-Final heuristic, as well, could not identify the site correctly, so the weights of these two heuristics have been reduced.

I analyzed that it was an overfitting problem which caused this reduction. The overfitting problem is known as the one of the weak points of AdaBoost, it decreases the accuracy otherwise AdaBoost attempted to fit the weight for identifying the site. In the case of (i) and (ii), there are much phishing sites that both Age of Domain and TF-IDF heuristic labeled correctly. Therefore, these heuristics could be assigned high weight otherwise they lost weight by the issued site. However, in the case of (iii), (iv) and (v), there are fewer phishing sites, and these two heuristics could not be assigned high weight. Hence, the true positive rate of AdaBoost was falling. If the site was not contained in training dataset, the true positive rate of (iii) could increase to 97.5%.

Increasing samples in dataset is the remedy for reducing the effect of overfitting. Thus, I attempt to collect much number of phishing sites and legitimate sites by monitoring Phishtank in long time period.

Chapter 7

Performance evaluation

In this chapter, I evaluate 9 Machine Learning-based Detection Methods(MLBDMs) and show their performance. First, I explain how I implemented heuristics which I could not fully implement in Chapter 6. To reduce the effectiveness of overfitting problem, I need to employ much number s of URLs than that in my preliminary evaluation. I then build the dataset for performance evaluation. Next, I explain my metrics of the performance which along with my requirements for detection of phishing sites. Finally, I show the evaluation result.

7.1 Implementation of heuristics in performance evaluation

At first, my implementation might check if a site is written in English or not, because CANTINA has not been and is not working well if the sites were not written in English. I employed Perl and its module, `Lingua::LanguageGuesser` [97], which can identify the language by N-gram-based text categorization [98]. N-gram-based approach to text categorization that is tolerant of textual errors. Cavnar et al. mentioned that their prototype system is small, fast and robust. The system also worked very well for language classification, achieving in one test a 99.8 % correct classification rate on Usenet newsgroup articles written in different languages.

For the Age of Domain heuristic, I employed GNU Whois [99], which was a generic whois client and provides the exact whois response due to the connecting the right WHOIS server corresponding to requested domain name. I then attempted to extract

the domain creation date from the whois response. I checked each line if contained the beginning of date record such as `create`, `regist`, or`connect`, in the WHOIS response. As the year field in the date record, some server responded in YYYY format such as 2009, and another server responded in YY format such as 09. As the month field, some server responded in MM format such as 12 which denotes December, some server responded "December", and another server responses "Dec" or "DEC". Moreover, there are various format of presenting the date such as "2008/12/15", "Dec. 15 2008", "15/12/08", and so much on. To deal with the difference in the date record, I observed the site which could not identify the date and fix my implementation to obtain the date record. My implementation can identify roughly 100 patterns of the date record format, however, there have been some date format which my implementation could not extract. For example, some server confused me because it responded "15/11/07" as the domain creation record. It can be interpreted two or more ways such as "November 15th, 2007", "November 7th, 2015". In this case, my implementation checked the current date and assumed "November 15th, 2007" is the collect answer.

To implement the Known Images heuristics, I employed both `WWW::Mechanize` [100] and `Image::Compare` [101]. `WWW::Mechanize` was a Perl module and facilitated to analyze the source code of the target website. I obtained the list of URLs which pointed some images by using `WWW::Mechanize`. `Image::Compare` was also a Perl module and designed to compare the image with some other images. My implementation had some pixel images of legitimate enterprises as I mentioned in Chapter 5, and compared the images in the target websites with the prepared images of legitimate enterprises. When I checked the difference in two pixel images, I hired 25, the most popular threshold in `Image::Compare` module. Notice that differences are measured in a sum of squares fashion (vector distance), so the maximum difference is $255 \cdot \sqrt{3}$, or roughly 441.7.

In the TF-IDF-Final heuristics, I also used `WWW::Mechanize` to obtain plain text which was removed HTML tags from HTML source code of the target site. I then extracted words from HTML source by analyzing Document Object Model structure with `HTML::TreeBuilder` [102], and counted TF value for each word by using `Lingua::Fathom`. For obtaining IDF value for each word, I searched the word with Google and estimated `DF` by checking the number of records in search results. Next, I calculated the TF-IDF value by Equation 7.1. In the context of TF-IDF, N is the amount number of documents, so I hired 4,285,199,774 for N according to [103], at first. However, Google said that the number of records was 12 billion when I entered

the letter "a" into Google. So, I hired 12 billion for N. Moreover, my implementation selected 5 words of the top 5 rank in the TF-IDF value from the site, and created the lexical signature by concatenating these words with the domain name. My implementation also searched Google by inputting the lexical signature, and checked the search result. Finally, my implementation decided the site seems to be a phishing site when (i) the number of record in result page was 0 or (ii) the domain name of the site could not appeared in top 30 higher ranks in the Google's search result.

$$\text{TF-IDF} = TF * \log_{10} \frac{N}{DF} \tag{7.1}$$

7.2 Dataset

I built a dataset with the criteria for choosing URLs. Based on the criteria in the original CANTINA, I collected URLs with the same number of phishing sites and legitimate sites. All sites were English language sites. First, I chose 1,500 phishing sites that were reported on Phishtank.com [18] from November, 2007 to February, 2008. The reason that I used only phisihng sites which were reported in Phishtank.com is to facilitate to compare my detection methods with the existing method. In CANTINA [19], Zhang et al. used to obtain the data from Phishtank.com.

Notice that these 1,500 sites were verified as phishing sites by the registered users of Phishtank.com. Basically, Phishtank.com accepts report from registered users of Phishtank.com, however, everyone can register to Phishtank.com. It is naturally to assume that phishers attempt to register legitimate sites as phishing. To compete with the bogus reports, registered users discuss to remove these reports on Phishtank mailing list and vote whether the sites are really phishing or not. I confirmed these 1,500 phishing sites by checking the voting results in December, 2008. In addition, I call such verification as "cleansing" and discuss the effectiveness of cleansing dataset in Section 7.7.3.

I also selected 227 URLs from 3Sharp's [95] study of anti-phishing toolbars. There were listed 500 URLs of legitimate sites, however, I could not connect to many listed URLs. Third, I attempted to collect 500 URLs from Alexa Web Search [94] and observed 477 URLs. Finally, I gathered 796 URLs from Yahoo! Random Link [96].

Each site was checked with my implementation of heuristics, and was converted into a vector $\vec{x} = (x_1, x_2 \ldots x_p)$, where $x_1 \ldots x_p$ are the values corresponding to a specific

Table 7.1: Environment of Performance Evaluation

Machine Learning	Library	Version
AdaBoost	ada	2.0-1
Bagging	ipred	0.8-5
SVM	e1071	1.5-17
CART	tree	1.0-26
LR	glm	-
RF	randomForest	4.5-23
NN	nnet	7.2-39
NB	predbayescor	1.1-2
BART	BayesTree	0.2-0

feature. The dataset consisted of 8 binary explanatory variables and 1 binary response variable.

To perform my evaluation in a less biased way, I employed 4-fold cross validation. Cross validation is a method to estimate the error rate efficiently. The procedure works as follows: the dataset is divided into k sub-samples (in my experiment $k = 4$). A single sub-sample is randomly chosen as testing data, and the remaining k-1 sub-samples are used as training data. The procedure is repeated k times, in which each of the k sub-samples is used exactly once as the testing data. Furthermore, my cross validation was repeated 10 times in order to average the result, so I had 40 patterns of training and testing dataset.

7.3 Environment

For repeatability of my experiment, I showed my environment of performance evaluation. My analysis was performed in R 2.6.1 running on Mac Pro, which has 2 Intel Xeon Quad-Core CPUs and 8GB Memory. The library name and version of each machine learning algorithm are shown in Table 7.1.

Table 7.2: Test Result

	actual phishing sites	actual legitimate sites
predict phishing sites	tp	fp
predict legitimate sites	fn	tn

7.4 Evaluation metrics

I also defined metrics for evaluating performance along with requirements for detection methods. My requirements were as follows.

1. *Accuracy*
   An MLBDM must achieve high detection accuracy. User safety would obviously be compromised if phishing prevention systems labeled phishing sites as legitimate. Users would also complain if prevention systems labeled legitimate sites as phishing sites because of the interruption in browsing caused by prevention systems.

2. *Adjustment Capability*
   An MLBDM must adjust its strategy for detecting phishing sites for web users. If a user is a novice, who is easily taken in by phishing attacks, phishing prevention systems should decrease false negative errors instead of increasing false positive errors. Conversely, if a user is a security expert, the system should focus on decreasing false positive errors.

For Requirement 1, I used the f_1 measure (higher is better) and the error rate (lower is better) as metrics to evaluate the detection accuracy. Statistically, f_1 measure has been used as an index of a test's accuracy. This measure is the harmonic mean of precision and recall. Given the test result as shown in Table 7.2, precision p equals $tp/(tp + fp)$ and recall r equals $tp/(tp + fn)$. The f_1 measure can be calculated by $2 \cdot p \cdot r/(p + r)$. The average error rate has been also a reasonable metric to indicate the detection accuracy. It is calculated by dividing the number of incorrectly identified sites by the number of all sites in the dataset. So, the error rate equals $(fp + fn)/(tp + tn + fp + fn)$.

For Requirement 2, I performed Receiver Operating Characteristic (ROC) analysis. Generally, detection methods calculate the likelihood of being phishing sites L and compare the likelihood with the defined discrimination threshold θ. In my experiment, MLBDMs distinguish a phishing site by checking if L is less or equal than $\theta(= 0)$. Imagine that θ was higher than 0. In this case, MLBDMs would tend to label a site as phishing rather than as legitimate. Conversely, MLBDMs would tend to label a site as legitimate if θ was lower than 0. Accordingly, I assumed that adjusting θ provides different detection strategies. Based on this assumption, I employed ROC analysis because it has been widely used in data analysis to study the effect of varying the threshold on the numerical outcome of a diagnostic test. I also used the Area Under the ROC Curve (AUC; higher is better) as a metric to evaluate adjustment capability.

7.5 Experimental setup

Next, I adjusted the parameters for MLBDMs to minimize the error rate in training. Through my parameter selection, I let MLBDMs studied from 3,000 URLs and measured the classification error rates. In addition, finding the optimal parameter is important, however, the choice of the exact value of the optimal parameter is not often a critical issue since the increase in test error is relatively slow.

For decision tree-based machine learning techniques such as RF, I tested them using different numbers of trees, namely 100, 200, 300, 400, and 500 trees. The minimum error rate (11.23%) was observed when the number of trees was 100, followed by 200 and 500 (11.37%), and 300 and 400(11.77%). Thus, I set the number of trees to 100 for RF-based detection methods.

The iteration time was set to 300 in all of my experiments if the machine learning technique needed to analyze iteratively for reducing training errors. The minimum error rate (11.46%) was observed when the number of iterations was 300, followed by 200, 400 and 500 (11.50%), and 100 (11.63%).

For some types of machine learning techniques, I used threshold value to approximate the prediction output. For example, BART is designed for regression, not for classification. Therefore, BART gives quantitative value whereas I need an MLBDM to output binary value that indicates whether a site is a phishing site or not. In such cases, I employed threshold value and observed if the result of BART regression was greater than the threshold. I decided the threshold in the same fashion as the original

Table 7.3: Precision, Recall and f_1 measure, False Positive Rate(FPR), False Negative Rate(FNR), Error Rate(ER), and AUC

	Precision	*Recall*	f_1 *measure*	FPR	FNR	ER	AUC
AdaBoost	0.9016	0.8554	0.8777	09.45%	14.46%	11.96%	0.9543
Bagging	0.8739	0.8801	0.8751	13.10%	11.99%	12.60%	0.9502
SVM	0.9008	0.8548	0.8770	09.52%	14.52%	12.03%	0.9180
CART	0.8643	0.8908	0.8755	14.37%	10.92%	12.69%	0.9449
LR	0.9244	0.8068	0.8609	06.73%	19.32%	13.08%	0.9523
RF	0.8931	0.8593	0.8749	10.54%	14.07%	12.34%	0.9539
NN	0.8994	0.8529	0.8751	09.68%	14.71%	12.21%	0.9518
NB	0.8822	0.8654	0.8735	11.67%	13.46%	12.58%	0.9486
BART	0.8923	0.8622	0.8765	10.55%	13.78%	12.19%	0.9540
CANTINA	0.9878	0.7048	0.8226	00.87%	29.52%	15.26%	0.9367

CANTINA. In the case of CANTINA, the maximum likelihood of being a phishing site is -1 and that of being a legitimate site is 1; therefore, it employs the middle value 0 as the threshold value.

In SVM, I tested both linear and non-linear kernel functions. The training error by using *Radial Based Function* (RBF), one of the typical non-linear kernel functions, was 11.23%, less than 14.63% of linear kernel.

In NN, I selected the number of units in the hidden layer, namely 1, 2, 3, 4, and 5 units, for finding the minimum average error rate. The minimum error rate (11.40%) was observed when the number of units was 4, followed by 3 (11.43%), 2 (12.13%), 1 (12.47%), and 5 (16.33%).

7.6 Performance evaluation

In this section, I evaluate the performance of all MLBDMs by measuring f_1 measure, error rate and AUC, and studying them comparatively. I also compare MLBDMs with the original CANTINA.

I calculated precision, recall, and f_1 measure for each pattern of dataset respectively, and also calculated their average as shown in Table 7.3. For readability, I summarized

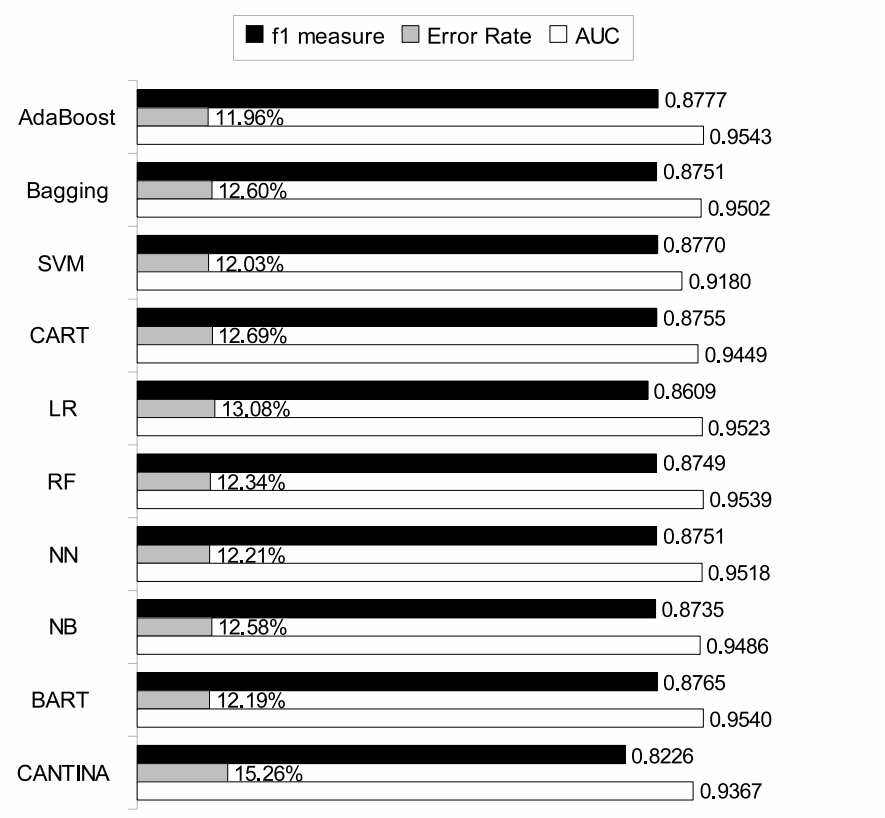

Figure 7.1: Test Result of f_1 measure, Error Rate, and AUC

the performance of MLBDMs in Figure 7.1 where black bars denoted f_1 measure. The highest f_1 was 0.8777 in AdaBoost, followed by SVM(0.8770), BART(0.8765), CART(0.8755), Bagging(0.8751), NN(0.8751), RF(0.8749), NB(0.8735), and finally LR(0.8609). The highest precision was 0.9244 in LR, and the lowest was 0.8643 in CART. The highest recall was 0.8908 in CART, and the lowest was 0.8068 in LR.

I then calculated the error rate in Figure 7.1, where gray bars denoted the error rate. The lowest error rate was 11.96% in AdaBoost, followed by SVM(12.03%), BART(12.19%), NN(12.21%), RF(12.34%), NB(12.58%), Bagging(12.60%), CART(12.69%), and finally LR(13.08%). The lowest false positive rate was 06.73% in LR, and the highest was 14.37% in CART. The lowest false negative rate was 10.92% in CART, and the highest was 19.32% in LR.

I also calculated AUC as shown in Figure 7.1, where white bars denoted AUC.

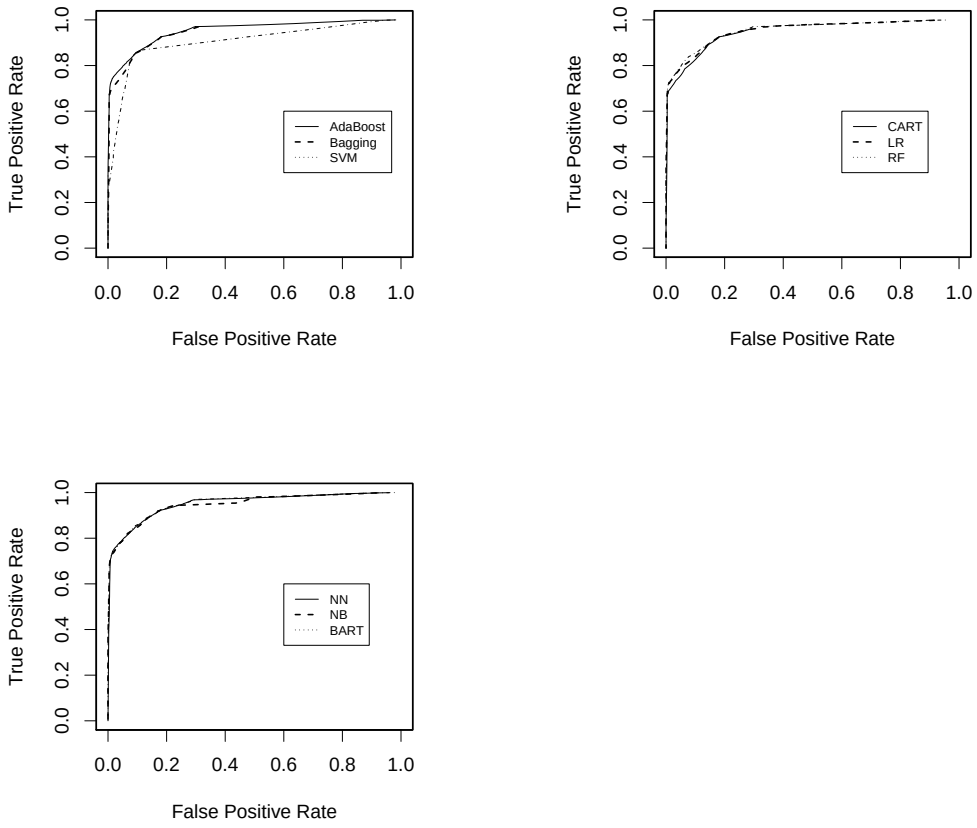

Figure 7.2: ROC curves of MLBDMs

The highest AUC was 0.9543 in AdaBoost, followed by BART(0.9540), RF(0.9539), LR(0.9523), NN(0.9518), Bagging(0.9502), NB(0.9486), CART(0.9449), and finally SVM(0.9180). Additionally, I plotted ROC curves of all MLBDMs as shown in Figure 7.2. For readability, each graph presented 3 ROC curves. I observed that all ROC curves passed through the upper left space in the graph. It indicated that all MLBDMs could achieve both high true positive rate and lower false positive rate because the best possible detection method would yield a point in the upper left corner (0,1) of the ROC space, representing that the true positive rate is 100% and the false positive rate is 0%.

Finally, I compared all MLBDMs with CANTINA's detection method. I evaluated

Table 7.4: Comparison of f_1 measure

	1	2	3	4	5	6	7	8	9	10
1. AdaBoost			n	n						
2. Bagging			n	n		n	n	n		
3. SVM	n	n		n		n	n		n	
4. CART	n	n	n			n	n	n	n	
5. LR										
6. RF		n	n	n			n	n	n	
7. NN		n	n	n		n				
8. NB		n		n		n				
9. BART	n		n	n		n				
10. CANTINA										

the performance of CANTINA in the same way as that described in Section 7.5, and observed f_1 measure was 0.8226, error rate was 15.26%, and AUC was 0.9367 as shown in Table 7.1. According to my comparison, 8 out of 9 MLBDMs, namely AdaBoost, Bagging, CART, LR, RF, NN, NB, and BART-based detection methods, outperformed CANTINA.

In addition, I performed analysis of variances by using 40 patterns of outputs. My Kolmogorov-Smirnov test showed that the output values of each detection method were fit to a normal distribution. Accordingly, I then performed paired t-test ($p < 0.05, \nu = 39$) for all detection methods. The results were shown in Table 7.4, 7.5, and 7.6, where "n" denoted that no statistical difference was observed, and empty denoted difference in f_1 measure was statistically significant. For example, I observed statistical differences in the average f_1 measure between AdaBoost and Bagging. In such cases, I did not mark anything. I also observed no statistical differences between Bagging and NB. In such cases, I marked "n" in the table. In short, my paired t-test showed that there were statistical difference between performance of MLBDMs and that of CANTINA.

Table 7.5: Comparison of Error Rates

	1	2	3	4	5	6	7	8	9	10
1. AdaBoost			n							
2. Bagging				n				n		
3. SVM	n						n		n	
4. CART		n						n		
5. LR										
6. RF							n	n		
7. NN			n			n			n	
8. NB		n		n		n				
9. BART			n				n			
10. CANTINA										

7.7 Discussion

In this section, I modified several test conditions to verify whether or not machine learning algorithms are available for detection of phishing sites. I changed the dataset which were built in different time period, and also changed the dataset which were not cleansed. Finally, I changed the set of heuristics by adding my heuristics.

7.7.1 Performance evaluation in different time period

Here, I discuss whether or not MLBDMs will continue to provide better performance. As I mentioned in Section 7.2, my collected phishing sites were reported on Phishtank.com from November, 2007 to February, 2008. Phishing attackers would attempt to build new phishing sites which are designed to evade detection, so I need to verify whether or not MLBDMs can keep higher performance in future.

As a first step, I built new dataset which contained modern phishing sites. I collected 1,500 URLs of phishing sites reported on Phishtank.com from August, 2008 to November, 2008. I also gathered 1,500 URLs of legitimate sites and checked that these 1,500 sites were verified as phishing sites in the same fashion as I described in Section 7.2. By using these 3,000 URLs, I measured their performance by doing 4-fold cross validation 10 times.

Table 7.6: Comparison of AUC

	1	2	3	4	5	6	7	8	9	10
1. AdaBoost						n			n	
2. Bagging										
3. SVM										
4. CART										
5. LR							n			
6. RF	n								n	
7. NN					n					
8. NB										
9. BART	n					n				
10. CANTINA										

The results were shown in Figure 7.3 where black bars, gray bars, and white bars denoted the average f_1 measure, error rate, and AUC, respectively. The highest f_1 was 0.8721 in AdaBoost, followed by SVM(0.8709), BART(0.8689), NN(0.8688), RF(0.8684), NB(0.8650), LR(0.8574), CART(0.8561), and finally Bagging(0.8555). The lowest error rate was 12.49% in AdaBoost, followed by SVM(12.55%), BART(12.69%), NN(12.73%), RF(12.79%), NB(13.21%), LR(13.34%), CART(13.52%), and finally Bagging(13.58%). The highest AUC was 0.9454 in AdaBoost, followed by BART(0.9452), RF(0.9451), LR(0.9429), NN(0.9419), Bagging(0.9343), NB(0.9321), CART(0.9212), and finally SVM(0.9038). Aside from MLBDMs, f_1 measure was 0.8494, error rate was 13.77%, and AUC was 0.9246 in the case of CANTINA. Similar to the comparison described in Section 7.6, 7 out of 9 MLBDMs, namely AdaBoost, Bagging, LR, RF, NN, NB, and BART-based detection methods, outperformed CANTINA. Accordingly, I predict that employing machine learning for detection of phishing sites has effectiveness in future.

In comparison between Figure 7.1 and 7.3, the performance of MLBDMs were tended to decrease in the case of using new dataset. In both cases, Kolmogorov-Smirnov test showed that the output values of each detection method were fit to a normal distribution. Thus, I performed two-sample Student t-test for homoscedastic data and two-sample Welch t-test for others. My two-sample t-test ($p < 0.05, \nu = 39$)

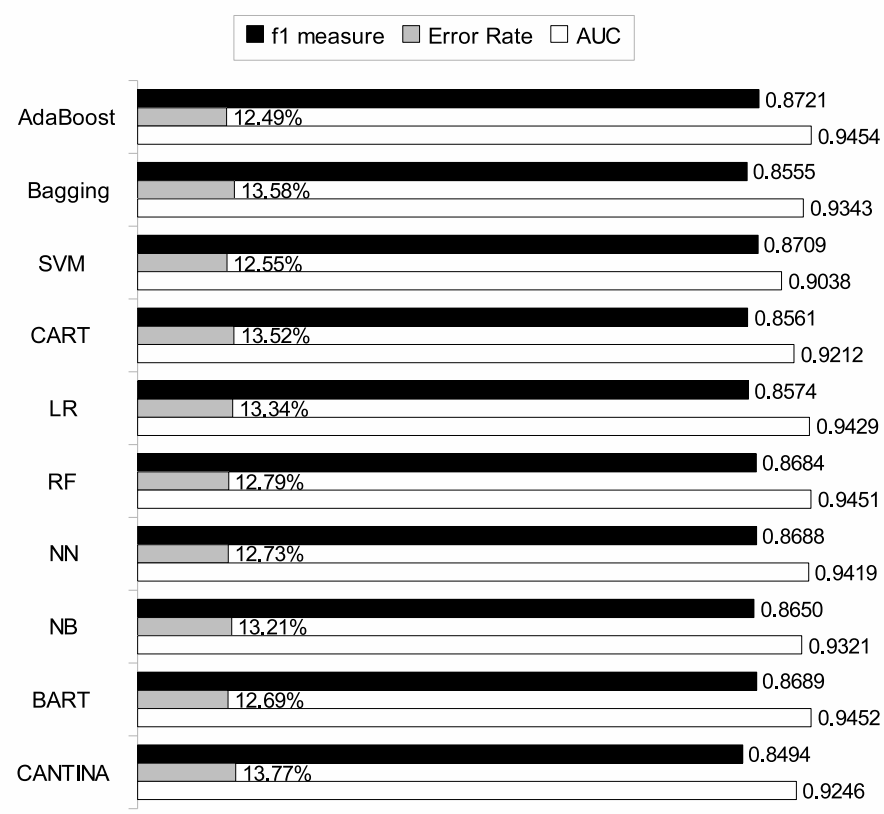

Figure 7.3: Test Result of f_1 measure, Error Rate, and AUC by using new dataset

showed that there were statistical difference in almost all of the cases; the error rates in the cases of LR, RF and NN were exceptions. I assumed that the reason of decreasing was caused by increasing the average error rate of heuristics. Figure 7.4 showed the error rate of each heuristic where black bars denoted error rates in the case of using old dataset, and gray bars denoted that of using new dataset. This indicated that I need to refine these heuristics and/or to develop new heuristics.

7.7.2 Detection of newly created phishing sites

In this section, I attempted to check if MLBDMs can detect newly created phishing sites. As I mentioned in Section 3.1, URL filtering methods cannot deal with newly created phishing sites. In spear phishing, a phisher can make new phishing sites for each victim and also can assign a different URL on each site. These sites are unreported,

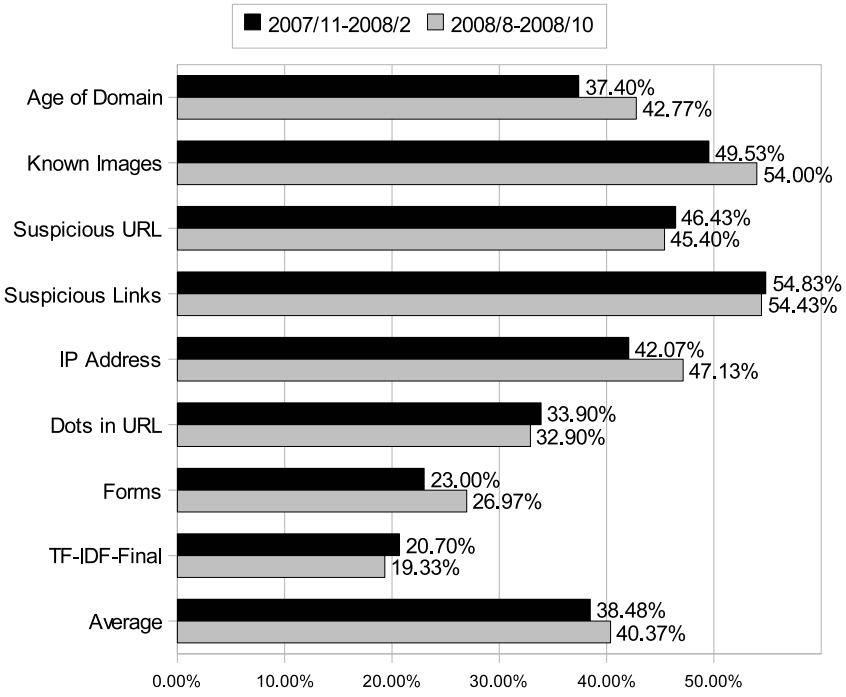

Figure 7.4: Error Rate of each heuristic in different time period

however, if heuristics-based solutions can detect such spear phishing sites correctly, the safety of URLs would be improved.

My evaluation performed in Section 7.6 could not show the sufficient evidence of this function. Because I performed 4-fold cross validation, the phishing sites in the testing dataset were reported in the same time period of that in the training dataset.

I explored the way for assessing that MLBDMs can deal with newly created phishing sites. I assumed that the best way is collecting enough numbers of spear phishing site, and analyze with these phishing sites. Unfortunately, only few people, who were targeted as phishers, could know the URLs of spear phishing sites. Collecting the spear phishing sites is my open issues.

Instead of using spear phishing sites, I selected 1,500 URLs of phishing sites reported in Phishtank.com during January, 2009 – February, 2009. I also collected 1,500 URLs of legitimate sites. For data independence between this dataset with other datasets, I employed 1,500 URLs from Yahoo Random Links, so did not use both 3Sharp's good URLs and Alexa's Top 500 URLs list.

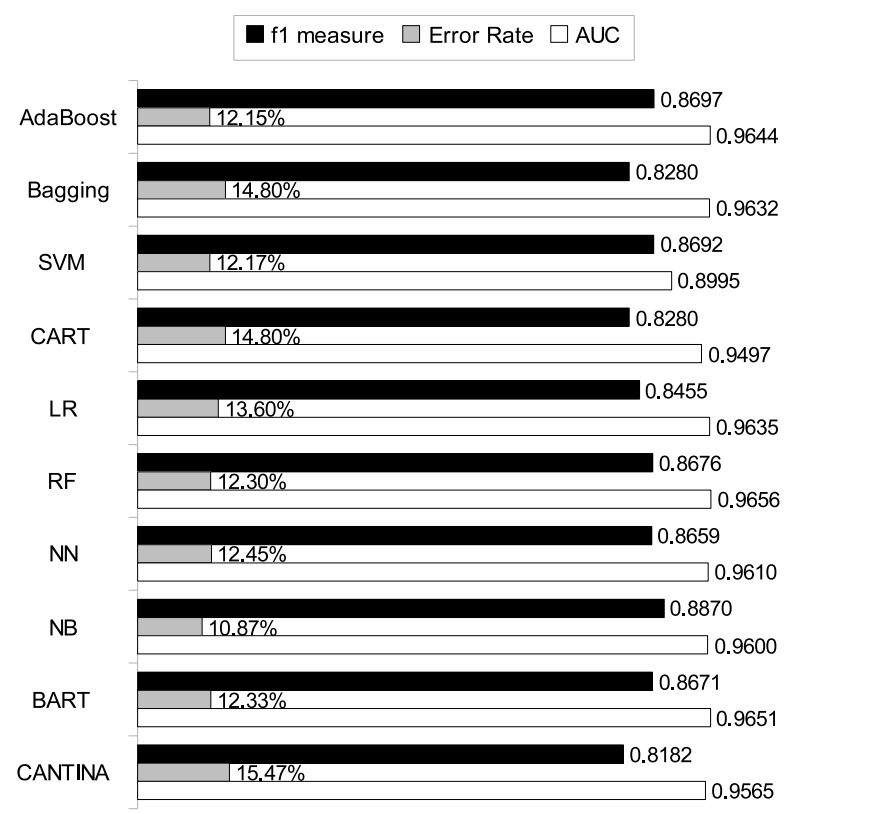

Figure 7.5: Test Result of f_1 measure, Error Rate, and AUC by training from 2007/11-2008/02, testing 2009/01 - 2009/02

For my performance evaluation, I let MLBDM to train from the dataset described in Section 7.2. Next, I tested with the newly created dataset. In short, MLBDMS were trained from phishing sites reported during November, 2007 – February, 2008, and they were tested with phishing sites reported during January, 2009 – February, 2009.

The results are show in Figure 7.5. The highest f_1 was 0.9027 in Bagging, followed by NN(0.9018), RF(0.9007), BART(0.9004), AdaBoost(0.8994), CART(0.8953), SVM(0.8941), NB(0.8889), LR(0.8811), and finally CANTINA(0.8447). The highest precision was 0.9781 in CANTINA, and the lowest was 0.8383 in CART. The highest recall was 0.9607 in CART, and the lowest was 0.7433 in CANTINA. The lowest error rate was 09.87% in Bagging, followed by NN(09.93%), RF(10.07%), BART(10.10%),

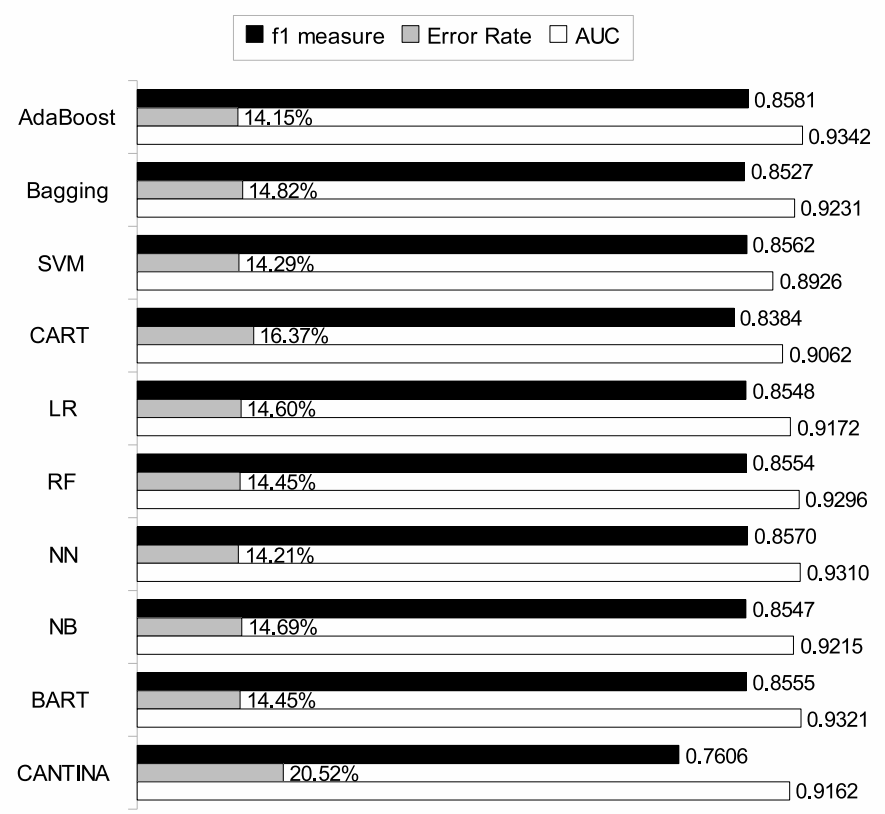

Figure 7.6: Test Result of f_1 measure, Error Rate, and AUC without cleansing

AdaBoost(10.23%), SVM(10.80%), CART(11.23%), LR(11.33%), NB(11.40%), and finally CANTINA(13.67%). The lowest false positive rate was 01.67% in CANTINA, and the highest was 18.53% in CART. The lowest false negative rate was 03.93% in CART, and the highest was 25.67% in CANTINA. The highest AUC was 0.9677 in BART, followed by RF(0.9675), LR(0.9666), NB(0.9663), Bagging(0.9656), AdaBoost(0.9647), CART(0.9615), NN(0.9610), CANTINA(0.9587), and finally SVM(0.9503). Similar to the comparison described in Section 7.6, 7 out of 9 MLBDMs, namely AdaBoost, Bagging, LR, RF, NN, NB, and BART-based detection methods, outperformed CANTINA. Thus, I believe that MLBDMs can deal with newly created phishing sites.

7.7.3 Effectiveness of cleansing

In my performance evaluation, I constructed my phishing dataset by both monitoring Phishtank.com and cleansing the dataset. However, due to the lifetime of phishing sites, my implementation checks the newly reported phishing sites as soon as possible regardless of cleansing. Hence, my dataset contains a data which is treated as a phishing site, however, explanation variables of the data would be similar to that of legitimate sites rather than that of phishing sites. After the dataset was generated, then I checked URLs by using Phishtank.com users' verification to build the cleansed dataset.

When I constructed my dataset described in Section 7.2, I obtained 1,500 URLs from Phishtank.com during November, 2007 – February, 2008. After my verification performed in December, 2008, I found that 223 sites were not verified as phishing sites. Thus, I removed these 223 sites from dataset and added other 223 phishing sites which were reported in February, 2008 and were verified as phishing sites. In addition, I obtained 19,443 sites during August, 2008 – November, 2008, and verified these sites December 2008. My verification showed that 14,115 sites were verified as phishing sites, whereas the rest of the sites were not verified.

The problem in cleansing dataset is that takes a long time for verifying. After phishing sites were reported to Phishtank.com, registered users carefully check if the site is really phishing or not. Some of the reported sites were difficult to judge. For example, `http://ebay.careerone.com.au/` was look-alike eBay, the domain name was also similar to eBay but different, so many users thought the site as phishing. The site was legitimate a career website (like careerbuilder.com) and it provided a co-branded career website to eBay Australia. Because of increasing such legitimate sites, these verification required users to have the knowledge of both phishing and legitimate sites.

I assumed that waiting verification would be unnecessary whenever MLBDMs could perform better even if the dataset were not cleansed. In the view of this, I attempted to measure the performance by using another dataset which were not cleansed. The dataset also contained 1,277 URLS of phishing sites, and 223 URLs of unknown sites which treat as phishing sites, and 1,500 URLs of legitimate sites. I also performed 4-fold cross validation 10 times.

The results were shown in Figure 7.6. The highest f_1 was 0.8581 in AdaBoost, fol-

lowed by NN(0.8570), SVM(0.8562), BART(0.8555), RF(0.8554), LR(0.8548), NB(0.8547), Bagging(0.8527), CART(0.8384), and finally CANTINA(0.7606). The lowest error rate was 14.15% in AdaBoost, followed by NN(14.21%), SVM(14.29%), BART(14.45%), RF(14.45%), LR(14.60%), NB(14.69%), Bagging(14.82%), CART(16.37%), and finally CANTINA(20.52%). The highest AUC was 0.9342 in AdaBoost, followed by BART(0.9321), NN(0.9310), RF(0.9296), Bagging(0.9231), NB(0.9215), LR(0.9172), CANTINA(0.9162), CART(0.9062), and finally SVM(0.8926). In short, 7 out of 9 MLBDMs, namely AdaBoost, Bagging, LR, RF, NN, NB, and BART-based detection methods, performed better than CANTINA. I also found that there was statistical difference between performance in MLBDMs and that in CANTINA. Accordingly, I assumed that MLBDMs performed better even if I changed the dataset.

By comparing Figure 7.1 with Figure 7.6, I observed that cleansing has effectiveness; In the case of AdaBoost, f_1 measure increases from 0.8581 to 0.8782, error rate decreased from 14.15% to 11.94%, and AUC increased from 0.9342 to 0.9544. My two-sample t-test ($p < 0.05, \nu = 39$) showed that there were statistical difference between with and without cleansing in all of the cases. Thus, I assumed that cleansing is useful. For expediting Phishtank.com users' verification, it is important to increase the number of registered users to gather many reputations of the issued sites.

7.7.4 Incorporating ability

Aside from the calculation methods for likelihood of being a phishing site, developing new heuristics are also contributed to thwart phishing attacks. In this section, I discuss whether or not MLBDMs can incorporate newly developed heuristics. First, I attempted to create new heuristics. Eventually, the detection accuracy of my heuristics was marginal, however, I observed if the MLBDMs can incorporate my heuristics. Instead of adding new heuristics, I next measure the performance of MLBDMs by disabling the TF-IDF-Final heuristics. Imagine if Zhang et al. have not proposed the TF-IDF-Final heuristic. Under the circumstances, MLBDMs could be constructed with other 7 heuristics. I assume that comparing the performance in the case of using 7 heuristics with that in described in Section 7.6 would be helpful to check if MLBDMs can incorporate new heuristics.

Effectiveness of adding new heuristics

Table 7.7: FPR, FNR, and ER of each heuristics

	FPR	FNR	ER
Age of Domain	41.33%	17.67%	29.50%
Known Images	00.00%	99.40%	49.70%
Suspicious URL	05.80%	92.67%	49.23%
Suspicious Links	20.20%	97.20%	58.70%
IP Address	00.00%	88.13%	44.07%
Dots in URL	01.07%	50.20%	25.63%
Forms	15.13%	22.40%	18.77%
TF-IDF-Final	24.73%	08.33%	16.53%
Old OS	03.80%	67.13%	35.47%
Country Mismatch	07.80%	62.20%	35.00%

Here, I introduce my heuristics.

- Old Operating System(OS)
  This heuristic checks if the OS of the web server is old. If so, the heuristic deems the site as phishing. Recently, phishing sites are often hosted in botnet. According to [104], bot installed PCs run Microsoft Windows are not properly patched and/or are not guarded by a firewall. In particular, old OSs such as Windows 95/98/Me/2000 are not equipped with firewall or their support is out of date. Hence, I assumed that a site is potentially a phishing site whenever the OS version of the site is old. In addition, I employed `p0f` [105], a passive OS fingerprint detection tool, to distinguish the old versions of OSs from others.

- Country Mismatch
  Checking if the Country Code Top Level Domain (ccTLD) of a site is equal to the geographical location of the site. For example, the ccTLD of `iplab.aist-nara.ac.jp` is "jp" and the geographical location of this server is also "jp". However, the ccTLD and the geographical locations of phishing sites are often different. Notice that this heuristic is not designed to catch a generic TLD, e.g., com, net, and org. If a site uses a generic TLD, this heuristic deems it is a legitimate, because it is not strange that a server placed in Japan uses such kinds of generic TLD

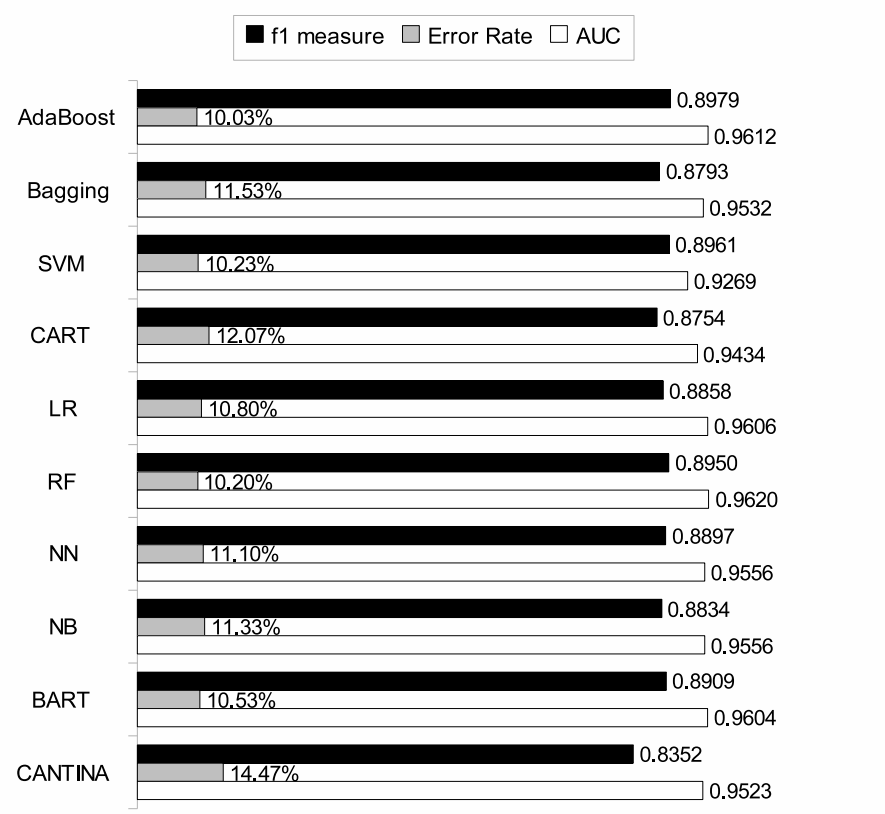

Figure 7.7: Test Result of f_1 measure, Error Rate, and AUC by adding new heuristics

in a domain name. In addition, I employed `GeoIP` [106] to distinguished the geographical location of the site.

I tested these 2 heuristics the reported phishing sites during November, 2007 – February, 2008, and found the characteristics of these heuristics. The results were shown in Table 7.7. The false positive rate of the Old OS heuristic was 03.80%, that of the Country Mismatch heuristic was 07.80%. However, these heuristics usually passed over many phishing sites, so the false negative rate of the Old OS heuristic was 67.13% and that of the Country Mismatch heuristic was 62.20%.

By using 2 of my heuristics with existing 8 heuristics, I test all MLBDMs by using the same dataset described in Section 7.2 and the same parameters described in Section 7.5. The results were shown in Figure 7.7 where black bars, gray bars, and white bars denotes the average f_1 measure, error rate, and AUC, respectively.

The highest f_1 was 0.8979 in AdaBoost, followed by SVM(0.8961), RF(0.8950), BART(0.8909), NN(0.8897), LR(0.8858), NB(0.8834), Bagging(0.8793), CART(0.8754), and finally CANTINA(0.8352). The lowest error rate was 10.03% in AdaBoost, followed by RF(10.20%), SVM(10.23%), BART(10.53%), LR(10.80%), NN(11.10%), NB(11.33%), Bagging(11.53%), CART(12.07%), and finally CANTINA(14.47%). The highest AUC was 0.9620 in RF, followed by AdaBoost(0.9612), LR(0.9606), BART(0.9604), NB(0.9556), NN(0.9556), Bagging(0.9532), CANTINA(0.9523), CART(0.9434), and finally SVM(0.9269). In short, 7 out of 9 MLBDMs outperforms than the existing detection method. Thus, I assumed that adding new heuristics would not decrease performance of MLBDMs.

I observed that the performance slightly increased. My two-sample t-test ($p < 0.05, \nu = 39$) showed that there were statistical difference in almost all of the cases; the f_1 in the cases of Bagging and CART, the error rate in the case of NN, and the AUC in the case of Bagging were exceptions. However, the contribution of my heuristics was not so obviously. I need to investigate if MLBDMs can treat new heuristics, and also to develop new heuristics.

Effectiveness of disabling the TF-IDF-Final heuristics

In the case of disabling TF-IDF-Final heuristics, I also measured the perform acne as shown in Figure 7.8 where black bars, gray bars, and white bars denoted the average f_1 measure, error rate, and AUC, respectively. The highest f_1 was 0.8540 in RF, followed by SVM(0.8540), NB(0.8531), AdaBoost(0.8530), NN(0.8524), LR(0.8470), BART(0.8466), Bagging(0.8377), CART(0.8376), and finally CANTINA(0.6098). The lowest error rate was 13.29% in SVM, followed by RF(13.30%), NB(13.37%), AdaBoost(13.40%), NN(13.44%), LR(13.87%), BART(13.92%), Bagging(14.54%), CART(14.54%), and finally CANTINA(28.28%). The highest AUC was 0.9214 in AdaBoost, followed by RF(0.9214), BART(0.9208), NN(0.9197), LR(0.9190), NB(0.9092), Bagging(0.9083), CART(0.8966), CANTINA(0.8880), and finally SVM(0.8788).

By comparing Figure 7.1 with Figure 7.8, I observed that performance in MLBDMs were improved by enabling the TF-IDF-Final heuristics. For example, performances (f_1, error rate, and AUC) of AdaBoost increased from (0.8530, 22.05%, 0.9214) to (0.8777, 11.96%, and 0.9543). In all detection methods, I observed that my two-sample t-test ($p < 0.05, \nu = 39$) showed that there were statistical difference between with and without using TF-IDF-Final heuristics. Thus, I assumed that MLBDMs can incorporate new heuristics.

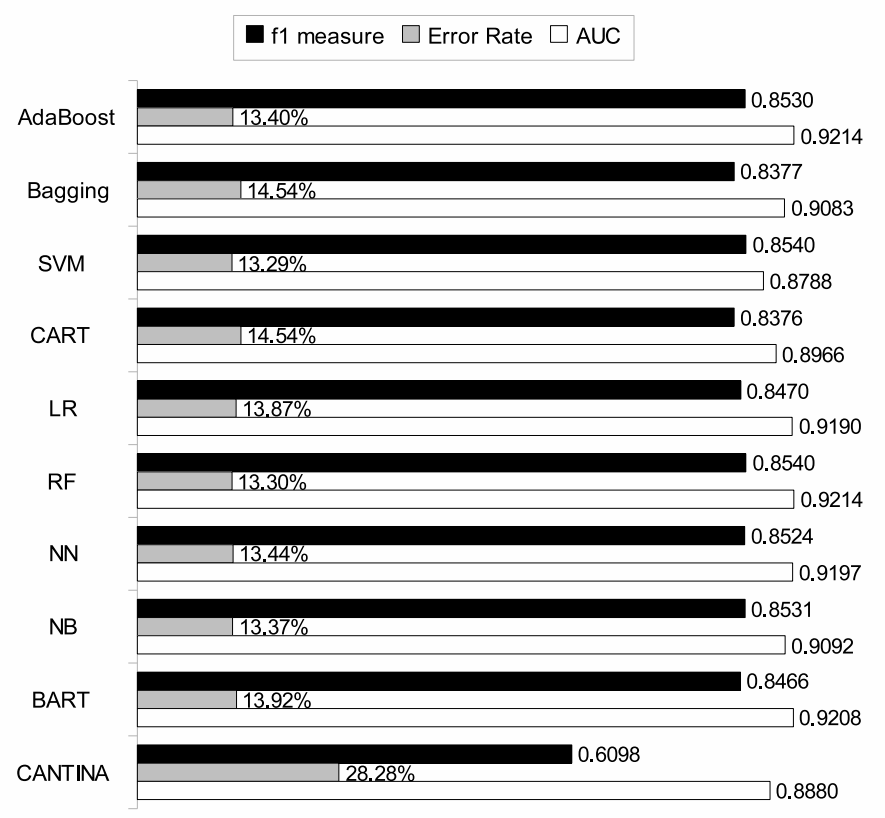

Figure 7.8: Test Result of f_1 measure, Error Rate, and AUC by disabling the TF-IDF-Final heuristic

In addition, performance of CANTINA increased from (0.6098, 28.28%, 0.8880) to (0.8226, 15.26%, 0.9367). The effectiveness of enabling the TF-IDF-Final heuristics in CANTINA would be more than that in MLBDMs. Paradoxically speaking, CANTINA might not perform better without the TF-IDF-Final heuristic.

Chapter 8

Utilization methods for MLBDMs

Based on MLBDMs, I argue about how to adjust the discrimination threshold for each user. Next, I introduce another approach which covers user's weak point, named HumanBoost.

8.1 Adjusting discrimination threshold

In my future work, I will implement MLBDMs-capable system, which is a phishing prevention system according to the detection result. Within such systems, I should adjust the discrimination threshold for each web user, as I mentioned in Section 7.4.

Aside from phishing, Denial-of-Service(DoS) attacks aim both network routers and hosts. There are several defense systems against DoS attacks, and it is different between the system for network routers and that for hosts. While their target of phishing is end users, so I assume that phishing prevention systems should provide different strategies for each users.

For a simple instance, I considered the cases of novices and security experts. If a user is a novice, who is easily taken in by phishing attacks, the system should decrease the false negative rate instead of increasing the false positive rate. Conversely, if a user is a security expert, the system should emphasize decreasing the false positive rate. Table 8.1 shows the false positive rate when the false negative rate was less than 5.00%, and the false negative rate when the false positive rate was less than 5.00%. The lowest false positive rate was 25.49% in the case of BART, and the lowest false negative rate was 20.49% in the case of RF. This indicated that if novices could accept

Table 8.1: FPR given FNR rate < 5.00%, and FNR given FPR rate < 5.00%

	FPR(FNR<5.00%)	FNR(FPR<5.00%)
AdaBoost	25.59%	20.55%
Bagging	25.97%	23.86%
SVM	71.19%	34.79%
CART	26.55%	24.51%
LR	25.83%	21.20%
RF	25.25%	20.49%
NN	25.75%	20.59%
NB	36.13%	21.05%
BART	25.49%	21.30%

Algorithm 1 Simple algorithm to decide threshold for end users

1: **Set** $\theta = 0.75$
2: **if** (The user input secret without checking the padlock icon) **then**
3: **Set** $\theta = \theta - 0.25$
4: **end if**
5: **if** (The user input secret without checking the address bar in Non-SSL sites) **then**
6: **Set** $\theta = \theta - 0.25$
7: **end if**

25.49% of false positive errors, 95.00% of phishing sites would be blocked as phishing sites. Similarly, if security experts could accept 20.49% of false negative errors, 95.00% sites of legitimate sites would be browsed normally.

Aside from the cases of novices or security experts, there have various users. The study for finding the optimal threshold for each user have not been conducted, however, the threshold should be decide by measuring each user's knowledge for the detection of phishing sites.

My prototype algorithm is show in Algorithm 1. The algorithm are derived from that I mentioned in Section 1.1; When inputting personal information, users should check the padlock icon, or check the URL appeared in browser's address bar in Non-SSL sites. Thus, I assume that it is naturally to use these 2 criteria to decide the

Table 8.2: FPR and FNR given θ was 0.75, 0,5 and 0.25

	$\theta = 0.75$		$\theta = 0.5$		$\theta = 0.25$	
	FPR	FNR	FPR	FNR	FPR	FNR
AdaBoost	01.51%	25.62%	09.45%	14.46%	26.64%	04.53%
Bagging	00.71%	31.90%	13.10%	11.99%	19.84%	06.99%
SVM	08.24%	16.44%	09.52%	14.52%	10.52%	13.71%
CART	00.87%	31.28%	14.37%	10.92%	24.98%	05.47%
LR	04.43%	21.85%	06.73%	19.32%	11.43%	14.45%
RF	00.51%	29.96%	10.54%	14.07%	22.82%	05.95%
NN	01.47%	26.10%	09.68%	14.71%	26.90%	04.35%
NB	03.90%	22.58%	11.67%	13.46%	17.50%	07.92%
BART	00.75%	29.96%	10.55%	13.78%	25.08%	05.15%

threshold for end users.

I set 0.75 to θ where MLBDMs would many sites as legitimate, rather than phishing. If an end user of the MLBDMs-capable system could not check the padlock icon by his or herself, θ decreased from 0.75 to 0.50. If the user cannot, then *theta* also decreases 0.25. The system would tent many sites as phishing.

Within the preliminary algorithm, I calculate the false positive rates and false negative rates given *theta* was 0.75, 0,5, and 0.25 as shown in Table 8.2.

In order to verify the reason-ability of this algorithm, I will perform a subject within study. At first, I check the ability of each subject and calculate threshold for each subject. I then let them to browse some phishing sites or legitimate sites by using the optimal threshold. I also let some of them to browse the sites by the normal threshold, 0,5, instead of using my algorithm. Next, I ask them to answer if they complain for the false positives and/or if they feel fear for the false negatives. Finally, I compare the results between 2 types of subject groups.

8.2 HumanBoost

Essentially, the boosting algorithms assign high weight to a classifier which correctly label a site where other classifiers labeled incorrectly, as I checked in Chapter 6. Imag-

Table 8.3: Conditions in each site

#	Web site	Real / Spoof	Lang	Description
1	Live.com	real	EN	URL (login.live.com)
2	Tokyo-Mitsubishi UFJ	spoof	JP	URL(www-bk-mufg.jp)
3	PayPal	spoof	EN	URL (www.paypal.com.%73%69 ... %6f%6d) (URL encoding abuse)
4	Goldman Sachs	real	EN	URL(webid2.gs.com), SSL
5	Natwest Bank	spoof	EN	URL(onlinesession-0815.natwest.com.esb6eyond.gz.cn) (Derived from PhishTank.com)
6	Bank of the West	spoof	EN	URL (www.bankofthevvest.com)
7	Nanto Bank	real	JP	3rd party URL (www2.answer.or.jp), SSL
8	Bank of America	spoof	EN	URL(bankofamerica.com@index.jsp-login-page.com) (URL scheme abuse)
9	PayPal	spoof	EN	URL (www.paypal.com) but first `a` letter is a Cyrillic small letter `a` (U+430) (IDN abuse)
10	Citibank	spoof	EN	URL(IP address)
11	Amazon	spoof	EN	URL (www.importen.se), contains `amazon` in its path (Derived from PhishTank.com)
12	Xanga	real	EN	URL (www.xanga.com)
13	Morgan Stanley	real	EN	URL (www.morganstanleyclientserv.com), SSL
14	Yahoo	spoof	EN	URL(IP address)
15	U.S.D of Treasury	spoof	EN	URL (www.tarekfayed.com) (Derived from PhishTank.com)
16	Sumitomo Mitsui Card	spoof	JP	URL (www.smcb-card.com)
17	eBay	spoof	EN	URL (secuirty.ebayonlineregist.com)
18	Citibank	spoof	EN	URL (シテイバンク.com) (is pronounced “Shi Tei Ban Ku”, look-alike “CitiBank” in Japanese Letter) (IDN abuse)
19	Apple	real	EN	URL (connect.apple.com), SSL, popup warning by accessing non-SSL content
20	PayPal	spoof	EN	URL (www.paypal.com@verisign-registered.com) (URL scheme abuse)

ine if user's trust decision can be treat as a classifier. AdaBoost would cover users' weak points by assigning high weights on heuristics that can correctly judge the site where a user is likely to misjudge. In other words, I use user's trust decision as new heuristic.

As my pilot study, in November 2007, I performed a subject-based test by showing legitimate enterprise web sites and emulated phishing sites. I called 10 subjects who belonged to Nara Institute of Science and Technology (NAIST), all of them were male, 3 of 10 had received an M.Eng degree in the last 5 years, and the rest of them were master's degree students.

Similar to the typical phishing IQ tests performed by Dhamija et al. [38], I constructed my dataset by preparing 14 emulated phishing sites and 6 legitimate ones, all

Table 8.4: The detection result by each subject

#	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	#Right	#Wrong
1							F	F			8	2
2	F	F									8	2
3		F									9	1
4			F			F			F		7	3
5							F	F			8	2
6		F	F		F	F					6	4
7	F	F		F		F					6	4
8											10	0
9											10	0
10				F							9	1
11							F				9	1
12							F		F		8	2
13		F	F								8	2
14								F			9	1
15				F							9	1
16		F		F				F			7	3
17		F		F				F			7	3
18											10	0
19	F			F			F	F	F		5	5
20			F								9	1

of which contained web input forms on which users could input their personal information such as a pair of user ID and password. Some phishing sites were derived from actual phishing sites according to a report from Phishtank.com. Other phishing sites were emulated phishing sites that were set up by phishing subterfuges [26, 27, 30] to induce subjects to input personal information.

In my test, I told them to browse the web sites freely. I did not prohibit subjects to access web sites that were not listed in Table 8.3. So, some subjects inputted several terms into search engines and compared the URL of the site with the URLs of those listed in Google's search result pages. As an environment for subjects to browse webs sites, I prepared Internet Explorer (IE) version 6.0 on Windows XP. I also configured IE to display IDN because some of the emulated phishing sites employed IDN spoofing techniques [29]. The detection results by each subject are shown in Table 8.4. In addition, "#" denotes the number of the web site in Table 8.3, S1 - S10 denote ten subjects, the letter "F" denotes that a subject failed to judge the web site, and empty denotes that a subject succeeded in judging correctly.

Next, I test AdaBoost-based detection methods. In the test, I used 8 heuristics

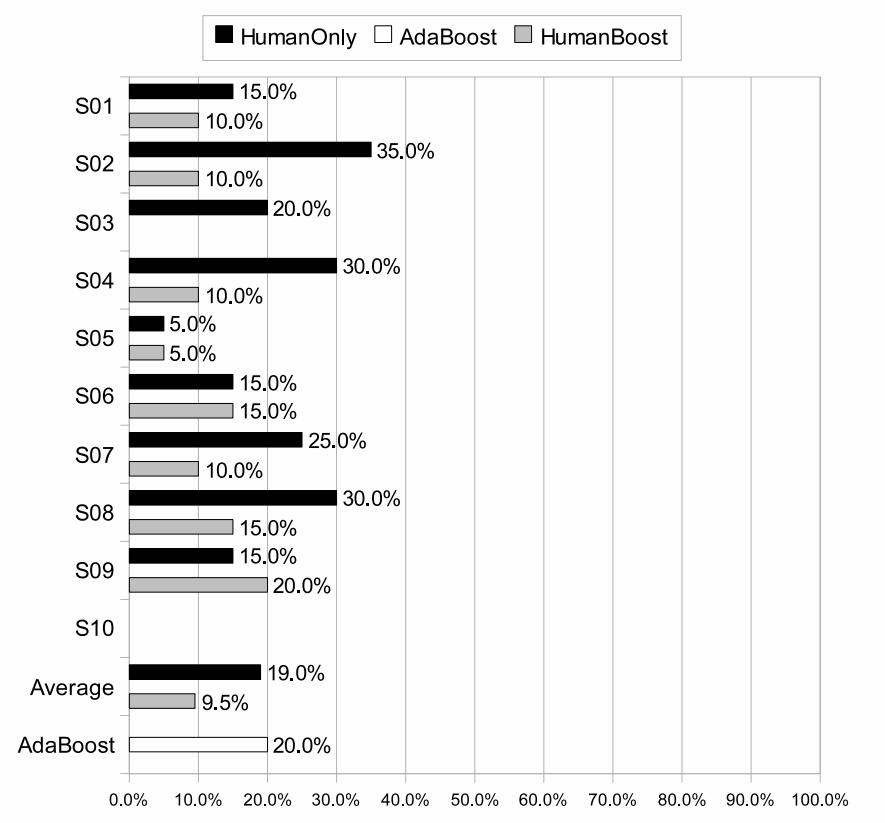

Figure 8.1: Error Rate in HumanOnly, AdaBoost and HumanBoost

described in Chapter 5, and define the number of iteration set to 1. I also employ 4-fold cross validation to average the result. The result showed the average training error rate in AdaBoost-based detection methods was 20.0%.

I then test HumanBoost. I let AdaBoost to test by using both user's trust decision and 8 heuristics, as shown in Figure 8.1. By comparing the case of HumanBoost with the case of HumanOnly, which denotes Human without AdaBoost, the error rate was lesser or equal in almost all cases. The average error rate in the case of HumanBoost was 9.5%, whereas the average error rate in the case of HumanOnly was 19.0% and that in the case of AdaBoost was 20.0%,

However, I found that a notification method to users is a significant problem on HumanBoost. In my pilot study, I showed 10 sites for a subject at first, then let AdaBoost study from the 10 sites in supervised-learning manner. When the subject

judge later 10 sites, then showed that the detection result using 9 heuristics, that is , his past trust decision and existing 8 heuristics. I observed that subjects did not change their trust decision in almost of all cases. After my experiment, I asked 4 subjects the reason that they ignored the detection messages. All of them answered that they trusted their own decision rather than system prediction. In view of this, detection message may be shown before a subject finished judging.

Chapter 9

HTTP Response Sanitizing

In this chapter, I discuss how to inform end users that they are just visiting phishing sites. Based on the detection result, phishing prevention systems can inform users to avoid browsing phishing sites. Some systems indicate in some portion of web browser and/or show alerting window by interrupting users' browsing. However, end users may distrust the system as long as the detection system sometimes makes mistakes and identifies legitimate sites as phishing sites. Accordingly, users would ignore the detection result.

Apart from indicators, proposed countermeasures against phishing attacks are categorized as compulsory blocking. Compulsory blocking is usually employed in application firewall [107, 108], and can filter whole suspected web pages regardless of users' intention or wrong behavior. The issue on compulsory blocking is that users have to sacrifice their convenience when a system identifies a legitimate site as phishing. A user may complain when a legitimate site is blocked by compulsory blocking due to the misjudgment by the detection algorithm.

I find that such system is needed that (i) can prohibit users to ignore the detection results and (ii) can also reduce users' inconvenience arising from false positive errors. For (i), application firewalls is available. For (ii), I propose a method for countermeasure against phishing, which I named HTTP Response Sanitizing (HRS). Instead of blocking the entire web content, HRS removes the HTML tags of web content which may generate input forms, and pads with a warning message as a substitute for the removed input forms. In this paper, I refer to this action of removing and padding as "sanitizing".

In the context of Cross Site Scripting(XSS), a sanitizing method is often employed. This method converts special characters to safe ones, since special characters may lead to unexpected actions. For example, sanitizing on XSS replaces "⟩" with ">". HRS sanitizes an HTTP response. Different from the sanitizing in XSS, HRS replaces dangerous parts of an HTTP response with warning messages.

A phishing site sanitized by HRS is always safe for browsing because it carries no input form for personal information. Even if false positive errors occur in detecting phishing sites, HRS removes only the input forms and users can browse the rest of the web content. Compared to compulsory blocking, the user's inconvenience is reduced.

9.1 Target of sanitizing

At first, I have to define what parts of an HTTP response are "dangerous". I defined as "dangerous" those data which generate an input form where a user can input personal information. According to this definition of "dangerous", I surveyed what kind of data in the web page generate an input form which should be sanitized.

As the result of the survey, the various types to be sanitized were seen to be:

- *FORM tags*
  Web input forms are typically composed of several HTML tags, as shown in Table 9.1. `<form>` and `</form>` tags state that there are web input forms. Other tags such as `<input>` are components of a web input form.

- *Active scripts*
  Active scripts, as shown in Table 9.2, are used to display dialog boxes where users can input information or are used to generate some kind of web content dynamically. Essentially, the `<script>` tag is used to describe active scripts. Without using a `<script>` tag, phishers can describe script in event handlers such as `onClick` and `onMouseOver`, for example.

  Regarding the latest version of Internet Explorer(IE), both `expression` and `url` functions in Cascading Style Sheet (CSS) enable almost all tags to take a function equivalent to active scripts. These functions are original extensions of IE; however, I assume that they could be the targets of sanitizing because IE is widely used.

Table 9.1: FORM Tags to be sanitized

<form>	<input>
<textarea>	<select>

Table 9.2: Tags, event handlers and CSS for Active Scripts to be sanitized

<script>	<style>	onclick
ondblclick	onkeydown	onselect
onkeypress	onkeup	onmousedown
onmouseup	onmouseover	onmouseout
onmousemove	onload	onunload
onfocus	onblur	onsubmit
onreset	onchange	onresize
onmove	ondragdrop	onabort
onerror	expression	url

HRS sanitizes all cases of active scripts by exploring tags, event handlers and CSS.

- *Active content*
  Phishing sites can be composed of active content such as Active X, Adobe Flash, Java Applet, and so on. Although HRS aims to remove only a bare minimum of web content, it is difficult to remove any input form involved in active content because active content is generally formatted in binary code, instead of being written in ASCII text. Hence, HRS must remove active content in order to avoid its abuse by phishers.

  HRS also sanitizes the tags shown in Table 9.3, which can be loaded into active content web pages. HRS can also detect that users are browsing active content by watching a Content-Type field in an HTTP response header.

- *HTTP headers*
  HRS sanitizes HTTP headers shown in Table 9.4, which can popup an input form to ask users to input ID and a password.

Table 9.3: Active Content Tags to be sanitized

<object>	<applet>	<embed>

Table 9.4: HTTP Headers to be sanitized

401 Unauthorized
402 Payment Required
407 Proxy Authentication Required
WWW-Authenticate

9.2 HRS Algorithm and Implementation

Here, I consider how to sanitize the "dangerous" data mentioned in Section 9.1 on an HTTP response. Algorithm 2 shows the pseudo code of my designed prototype HRS implementation. HRS is very simple, taking only 20 steps.

At first, HRS checks if the type of HTTP header matches one of the types referred to in Table 9.4. Next, HRS verifies whether the content is active content or not. In prototype implementation, I checked the content type according to MIME type [109–111]. After checking the HTML header and the content type, HRS tries to replace "dangerous" HTML tags or active scripts in the HTML text.

Figure 9.1 and 9.2 show an example of a web page sanitized by our HRS implementation. HRS alerts users to the "dangerous" parts of a sanitized web page with warning messages shown in Figure 9.2. The warning messages on sanitized pages are useful, when browsing, for identifying malicious intent.

9.3 Security Verification

The safety provided by HRS is in sanitizing to prevent leaks of personal information. To verify the sanitizing function of HRS, I used my prototype implementation of HRS and performed safety verification with 100 actual phishing sites.

I also implemented an HRS function as an ICAP [112] module incorporating to Squid [113]. Notice that the server could not identify a phishing or not; It only sanitized the content. Within my prototype implementation, I browsed 100 phishing sites from registered phishing sites reported on Phishtank [18] reported in April 2007, according to the following requirements; a phishing site which (i) can be still browsed (has not

Algorithm 2 HRS Algorithm

```
1: procedure HRS
2: SC points httpResponse.httpStatusCode
3: HC points httpResponse.httpContent
4: if SC includes MALICIOUS_HEADER then
5:     generate SANITIZED_HEADER
6:     replace SC to SANITIZED_HEADER
7: end if
8: if HC includes ACTIVE_CONTENT then
9:     remove ACTIVE_CONTENT from HC
10: end if
11: repeat {search for input-form in HC}
12:     for all place where input-form is found do
13:         generate SANITIZED_MESSAGE
14:         replace input-form to SANITIZED_MESSAGE
15:     end for
16: until all input-form are checked
17: return httpResponse
```

Figure 9.1: Before HRS

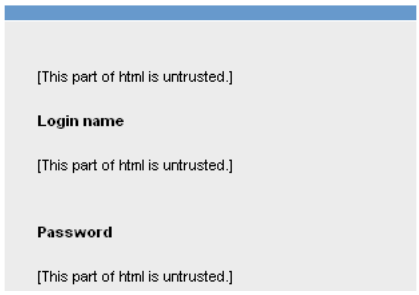

Figure 9.2: After HRS

expired), (ii) looks like a well known legitimate website, and (iii) is easily distinguished as a phishing site by its URL.

As a result of this verification, I confirmed that HRS sanitized all input forms on each phishing site and no personal information could be leaked. Hence, I concluded that the safety provided by HRS is at the same level as the safety provided by compulsory blocking.

9.4 Overhead of sanitizing

In this section, I evaluated the processing overhead of HRS. The processing overhead is incurred when HRS-capable proxy servers are sanitizing HTML content. My test-bed

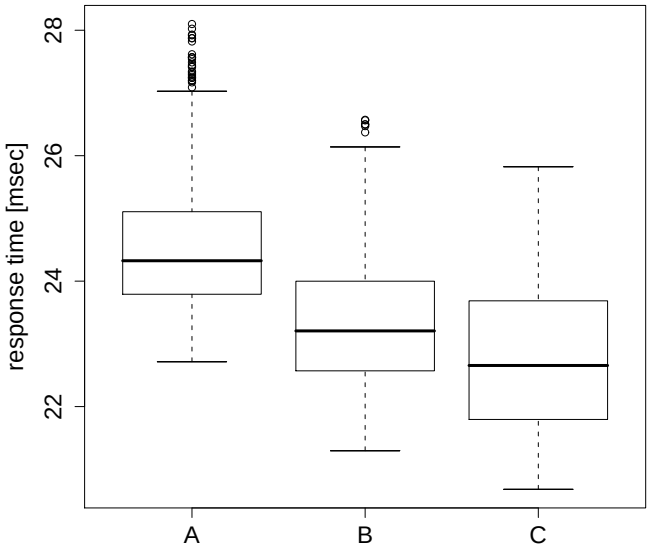

Figure 9.3: Overhead of sanitizing

environment was comprised of a client host, a web server host, and an HRS-capable host.

I also tried to implement HRS in Privoxy [114]. I implemented an HRS-capable proxy by adding only 60 lines to 37,449 lines of Privoxy. I prepared GNU Wget [115] HTTP client program in the client host, that ran Linux 2.4.22 with 1.7GHz Celeron processor and 256MB RAM. I also prepare Apache [116] HTTP server in the server host, that ran FreeBSD 4.11 with 800MHz Pentium III processor and 256MB RAM. HRS-capable server ran on a FreeBSD 4.11 host with 1.8GHz Pentium processor and 512MB RAM.

Figure 9.3 showed the processing overhead where y axis denotes response times. The target page was 10K bytes, and contained 13 tags to be sanitized. 1% of top and 1% of bottom data are omitted for accuracy. I assumed that pattern A showed the processing overhead for phishing site, so that when the client used HRS-capable proxy and sanitizing was needed. Also, I assumed that pattern B showed the processing overhead for the legitimate sites, so that the client used HRS-capable proxy but sanitizing is not needed. Pattern C showed that the client did not use HRS-capable proxy.

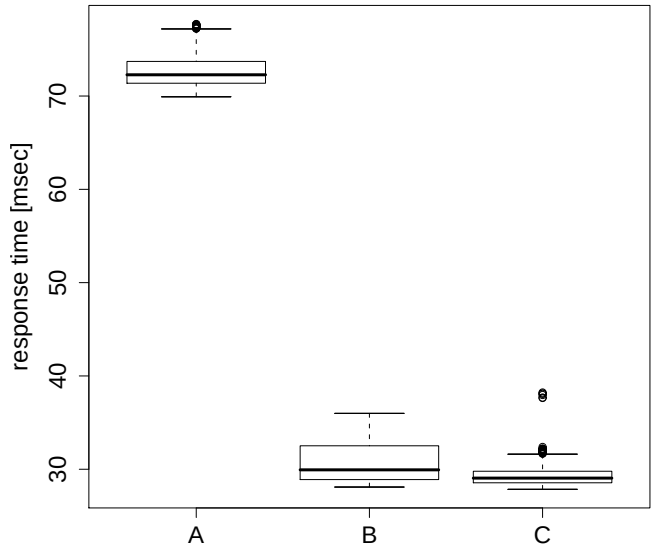

Figure 9.4: Overhead of sanitizing in increasing the content size

The processing overhead for the phishing site was 3.59 millisecond, which was obtained by subtracting from the average of pattern A to the average of pattern C. I considered the 3.59 millisecond of overhead not to be critical, because it was not sensible and there were too many factors of long delays on the Internet. On the other hand, the processing overhead for the legitimate site was 1.45 millisecond, which was obtained by subtracting from the average of pattern B to the average of pattern C. In order to evaluate the performance overhead of the increasing content size, I changed the content size from 10Kbytes to 100Kbytes. The result was described in Figure 9.4. The processing overhead for phishing was much higher Figure 9.3, and the average of overhead for phishing site was 43.37 millisecond, otherwise the average of overhead for the legitimate site was 1.38 millisecond. It assumed that the performance of users' browsing would penalize when users visit phishing sites.

Table 9.5: Comparison among countermeasures (to phishing sites)

	result of action	Admit users to ignore the action ?	Admit users to browse phishing site?	Robustness against phishing
Warning	popup warning message	yes	yes (original page)	weak
Compulsory Blocking	drop original page and report the result	no	no (report page)	strong
HRS	sanitize original page	no	yes (sanitized page)	strong

Table 9.6: Comparison among countermeasures (to legitimate sites)

	result of action	Admit users to ignore the action?	Admit users to browse legitimate site?	loss of the service
Warning	popup warning message	yes	yes (original page)	nothing
Compulsory Blocking	drop original page and report the result	no	no (report page)	everything
HRS	sanitize original page	no	yes (sanitized page)	input form active content proper page design

9.5 Comparison among countermeasures

Here, I compared HRS with other countermeasures against phishing attacks. Table 9.5 showed the results of applying each countermeasure to the phishing sites, and Table 9.6 presented the results of applying each countermeasure to the legitimate sites. According to Table 9.5, HRS was as safe as compulsory blocking. A warning was not as safe as other methods because it allows user to choose whether to continue to browse the page or not. When each countermeasure was applied to legitimate sites, though both HRS and compulsory blocking interfere with user's convenience, HRS provided more convenience than compulsory blocking, as shown in Table 9.6.

Chapter 10

Development of MLBDM-capable systems

In this chapter, I discuss the development of MLBDM-capable phishing prevention systems. In order to clarify the discussion, I show the stakeholders of the system and the data flow diagram in the system. I then introduce several forms of implementation for the system.

10.1 Stakeholders of the system

As I mentioned in Section 3.1, the stake holders of detecting phishing sites are web client developers and security service providers. To support the explanation, I show how victims disclose personal information as shown in Figure 10.1. Generally, victim's activities can be divided into several phases as follows:

1. A web user operates a web browser to show particular web content. The URL of the content has already given via phishing email, and so on.

2. A web browser sends an HTTP request to a web server, and a web server receives it.

3. A web servers process the request. Some phishing sites are static content such as HTML files, and some others are dynamic content such as CGI, PHP and so on.

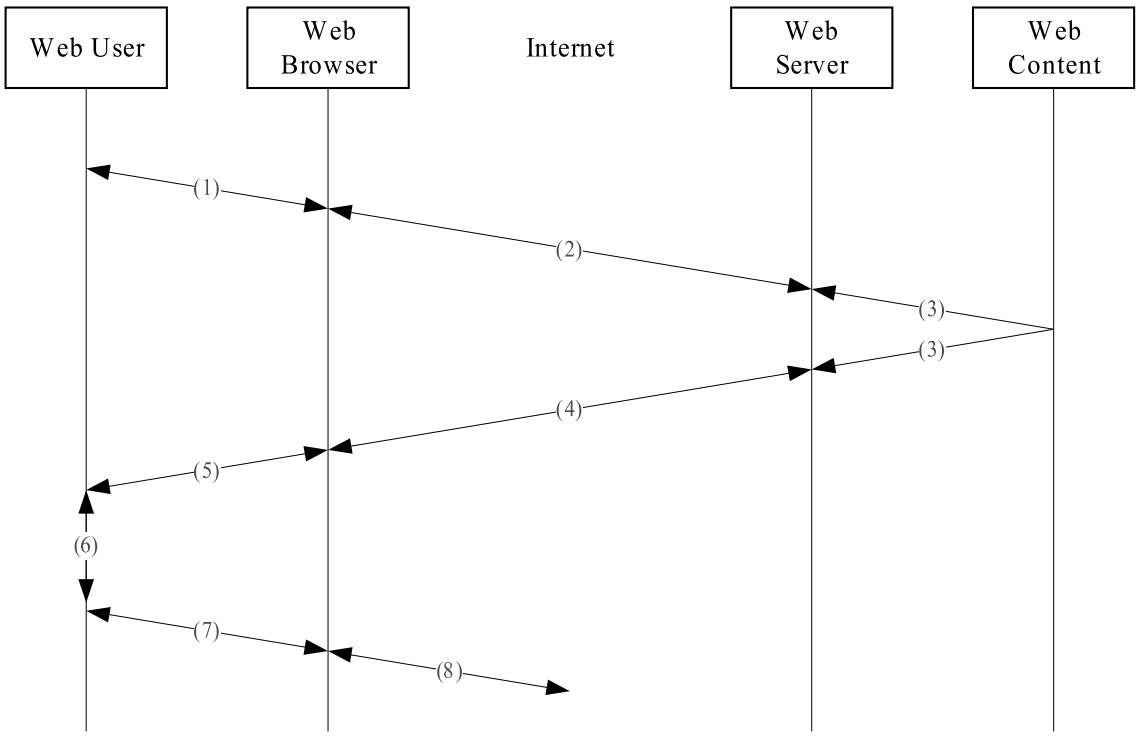

Figure 10.1: Phases of Users' browsing

4. A web server sends an HTTP response to a web browser, and a web browser receives it.

5. A web browser renders the response, and appears to the user.

6. A web user browses web content. In addition, phases 2, 3, 4 and 5 are repeated until the web browser downloads all elements in the content or the web browser decides timeout.

7. A web user enters their personal information to the displayed content.

8. A web browser HTTP requests involves the user's secret.

Notice that phishing sites are often hosted in botnet, therefore, web severs of the site cannot be trusted; these servers are under the phishers' control. In the view of this, the system should work in phase 2, 4, 5 and/or 7.

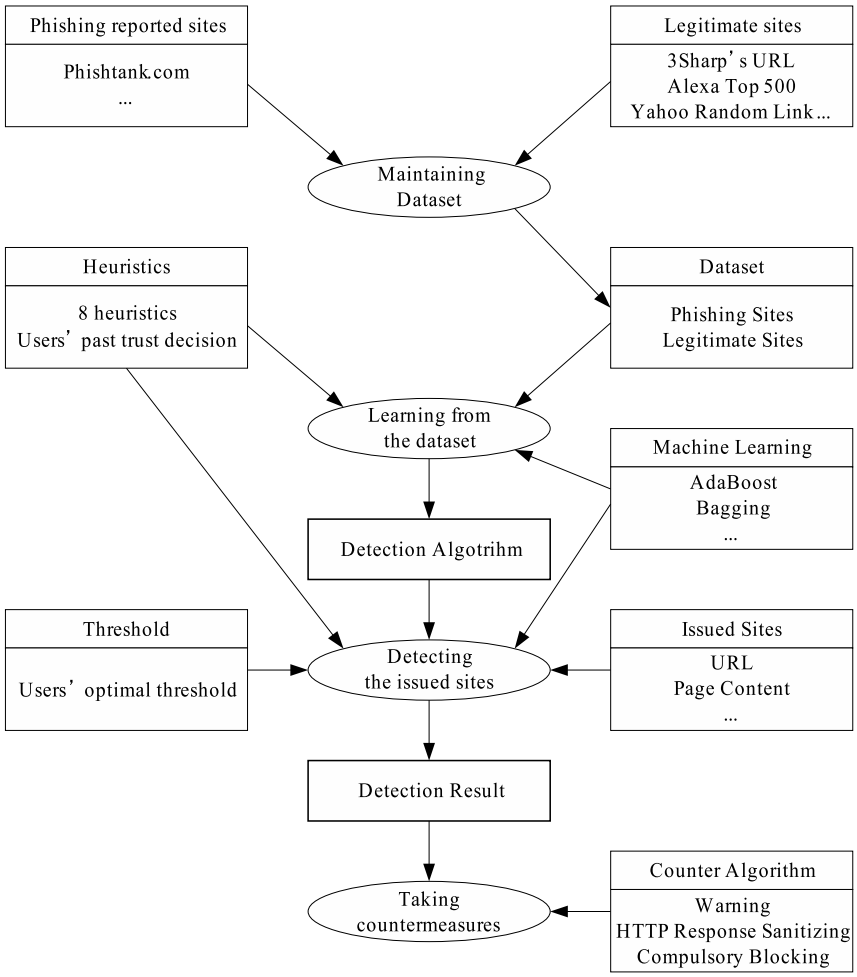

Figure 10.2: A data flow diagram for MLBDM-capable phishing prevention systems

Accordingly, the stakeholders of detecting phishing sites are web client developers such as browser vendors, and trustworthy third parties placed in the Internet such as security service providers. As a fact, existing anti-phishing solutions are usually integrated in web browsers, and communicate with the trustworthy third parties. For example, Internet Explorer 7.0 contacts a trust worthy third party (Microsoft) for each web page, asking whether the page's URL was or was not listed as a phishing site.

10.2 Data flow in the system

In this section, I explain the data flow for MLBDM-capable systems as shown in Figure 10.2, where an operation is surrounded by an eclipse, data required for the operation is surrounded by a rectangle.

- Maintaining a dataset
  An MLBDM-capable system needs a dataset contains both phishing sites and legitimate sites. It is also necessary to maintain the dataset by removing sites which are not phishing sites but registered as phishing, as I explained in Section 7.7.3.

- Learning from the dataset
  An MLBDM-capable system learns from the dataset with machine learning techniques to construct an algorithm for the detection of phishing sites. The system employs at least 8 heuristics described in Chapter 5. Additionally, the system requires the users' past trust decision if the system supports HumanBoost.

- Detecting the issued sites
  Based on the learned algorithm, an MLBDM-capable system checks if a site where a user is just visiting is phishing or not. The system analyzes the site with heuristics, calculates the likelihood of being a phishing site, and compares the likelihood with the discrimination threshold. Additionally, the systems should manage the optimal discrimination threshold for each web user, as I mentioned in Section 8.1.

- Taking countermeasures
  An MLBDM-capable system takes approximate countermeasure such as Warning, HTTP Response Sanitizing, and Compulsory Blocking according to the detection result.

I assumed that maintaining a dataset should be performed by a security service provider (SSP). If web clients attempt to maintain a dataset, numerous web clients crawl phishing reported sites; the load of such sites would increase seriously. Conversely, the load of the phishing reported sites would be marginal if a small number of the SSP crawl the Phishtank.com. In the view of this, this operation should be performed by the SSP.

There are pros and cons between a web client and an SSP when learning is performed. While the SSP maintains the dataset, a web client obtains the dataset from the SSP for learning. Empirically, machine learning algorithms require computational resources, whereas some web clients such as handheld devices or mobile phones have limited resources. However, the SSP should store the users' past trust decision to support HumanBoost.

I also found that there are also advantages and disadvantages between a web client and an SSP on detecting phishing sites. In order to check the issued site with heuristics, the system requires resources such as memory, processing speed, and bandwidth. Basically, the SSP cannot analyze the issued site when the issued site is encrypted by SSL. In this case, a web client provides decrypted content to the SSP for the detection.

When the system focuses on novice users, taking countermeasures should be performed by an SSP. A web client is usually controlled under a web user who is attracted by phishers; they can ignore the warning and/or stop the detection system. In order to enforce the detection results, the SSP should operate transparent HTTP proxy in the web user's ISP. Regardless of both the users' intention, the web traffic can be always forced to pass through a transparent HTTP proxy.

10.3 Forms of implementation

Based on the explanation in Section 10.1 and 10.2, I introduce 3 forms of implementation for the MLBDM-capable phishing prevention systems, named an SSP-side model, a client-side model, and a collaborative model.

Within the SSP-side model, both learning and detection are performed in the SSP. The advantage of this model is that the model doe not require delivering the dataset and/or the detection algorithm to a web client. This model also does not require computational resources to each web client. However, a web client must contact the SSP-side implementation of the system asking whether a site is phishing or not. If the number of users is numerous, the load of the system would seriously increase. To avoid the heavy load, it is useful to store the detection results of the system and the check results of each heuristic. Additionally, the SSP performs N times of learning where N is the number of users in the case of HumanBoost.

Conversely, a client-side implementation performs both learning and detection in the web client. Basically, user oriented information such as past trust decision and

optimal threshold should not be shared with third parties. In comparison to the SSP-side implementation, a web client can detect phishing sites regardless of using SSL and a web client does not contact to anyone while users' browsing. However, the system requires bandwidth and memory resources for obtaining the dataset, computational resources for learning from the dataset.

The feature of the collaborative model is that learning and detection are performed by different stakeholders, in respectively. For example, imagine if the learning is performed in an SSP and the detection is performed in a web client. In this model, a web client obtains the detection algorithm which is created by machine learning from the SSP. The system also calculates the likelihood of being a phishing site, and then compares the likelihood with the threshold in the client-side. Accordingly, the system does not require contacting to SSP while a web user is browsing. In addition, I assumed that the size of the algorithm is not so big, as long as an MLBDM-capable system employs 8 heuristics all of which output binomial variable; the combination of explanatory variables are only 256 ($= 2^8$) patterns.

Accordingly, I considered that forms of implementation should be selected after due consideration of both the system resources and the set of the heuristics. The SSP-side model would be suitable for thin clients such as handheld devices, and the client-side model would facilitate to develop HumanBoost. Although there are pros and cons between these 2 models, the collaborative model would be available for reducing both the client load and the number of queries sent toward the SSP.

Chapter 11

Conclusion

In this dissertation, I presented a machine learning approach for detecting phishing sites. This dissertation addressed the problems faced to phishing. The issues included how to support for end users to make trust decision, how to detect the phishing sites to inform the users, and how to improve the detection accuracy. Unfortunately, URL filtering methods could not detect unreported phishing sites. Conversely, heuristics-based solution could deal with such sites whereas the detection accuracy was not so high. For improving the accuracy, I proposed machine learning-based methods for detection of phishing sites. I tested machine learning algorithms, namely AdaBoost, Bagging, Support Vector Machines (SVM), Classification and Regression Trees (CART), Logistic Regression (LR), Random Forests (RF), Neural Networks (NN), Naive Bayes (NB) and Bayesian Additive Regression Trees (BART).

In my preliminary evaluation, I checked if machine learning techniques were available for detection of phishing sites. At first, I employed AdaBoost, one of the typical machine learning algorithms, to combine 8 heuristics presented by Zhang et al. [19], namely, Age of Domain, Known Images, Suspicious URL, Suspicious Links, IP Address, Dots in URL, Forms, and TF-IDF-Final heuristics. Next, I prepared 2 datasets; one is used for training, and the other is used for testing. Each dataset was composed of 50 URLs of phishing sites reported on Phishtank [18] and the same number of legitimate sites. In addition, the URLs of the testing dataset were different from those of the training dataset. I let AdaBoost to study from the training dataset in supervised-learning manner. Then I measured the detection accuracy by using the testing dataset. I also let an existing method to study from the training dataset and

measured the accuracy from the testing dataset. The result showed that true positive rate was 94.0%, false positive rate was 0.0%, and accuracy was 97.0% in the case of AdaBoost, all of them performed better than CANTINA where true positive rate was 92.0%, false positive rate was 4.0%, and accuracy was 94.0%. Moreover, I modified both heuristics and dataset, and then I found that the overfitting problem was the weak point of AdaBoost.

To thwart the overfitting problem, I attempted to increase the number of samples in dataset. I implemented 8 heuristics to construct automated systems for analyzing phishing sites. My automated system crawled Phishtank.com periodically and analyzed newly reported phishing sites. By using such system, my dataset composed of 1,500 phishing sites reported during November, 2007 – February, 2008, and the same number of legitimate sites. I also checked if the 1,500 phishing site was verified as phishing sites by registered users' of Phishtank.com. Without such kind of verification, 223 sites were not verified as phishing sites, but were regard as phishing sites in my dataset. I removed these 223 sites and add other phishing sites which were verified as phishing sites and were reported in February, 2008. All of the sites were checked by 8 heuristics.

By using the dataset, I employed 9 machine learning algorithm to combine heuristics. At first, I decided my metrics for my performance evaluation. Because I assumed that the detection method must be accurate and must have adjustment capability, I used f_1 measure, error rate and AUC as performance metrics. Next, I adjusted the parameters for MLBDMs to minimize the error rate in training. In addition, I performed 4-fold cross validation 10 times to average out the result.

The result showed that the highest f_1 was 0.8777 in AdaBoost, followed by SVM(0.8770), BART(0.8765), CART(0.8755), Bagging(0.8751), NN(0.8751), RF(0.8749), NB(0.8735), and finally LR(0.8609). The lowest error rate was 11.96% in AdaBoost, followed by SVM(12.03%), BART(12.19%), NN(12.21%), RF(12.34%), NB(12.58%), Bagging(12.60%), CART(12.69%), and finally LR(13.08%). The lowest false positive rate was 06.73% in LR, and the highest was 14.37% in CART. The highest AUC was 0.9543 in AdaBoost, followed by BART(0.9540), RF(0.9539), LR(0.9523), NN(0.9518), Bagging(0.9502), NB(0.9486), CART(0.9449), and finally SVM(0.9180). Additionally, I plotted the ROC curve and found that all MLBDMs could achieve both high true positive rates and low false positive rates. I also compared MLBDMs with CANTINA. The result showed that AdaBoost, Bagging, CART, LR, RF, NN, NB, and BART-based detection methods outperformed than the existing method.

Next, I measured the performance by using another dataset which contained modern phishing sites reported on Phishtank.com during August, 2008 – November, 2008. I also let MLBDMs to train from the dataset contains phishing sites reported during November, 2007 – February, 2008, and test with that contains newly created phishing sites reported during January, 2009 – February, 2009. By comparing with the traditional detection method, AdaBoost, Bagging, LR, RF, NN, NB, and BART-based detection methods outperformed than the existing method.

I also measured the effectiveness of cleansing dataset. I checked 1,500 URLs of phishing sites in my old dataset reported in November, 2007 – February, 2008, which has not been cleansed. This dataset contained 1,277 phishing sites, 223 unknown sites, and 1,500 legitimate sites. The result could be summarized that the highest f_1 measure was 0.8581, the lowest error rate was 14.15%, the highest AUC was 0.9342, all of which were observed in the case of AdaBoost. By comparing between before and after cleansing dataset, I found that performance can be improved after cleansing. Thus, I assumed that such verification is important.

Moreover, I checked if MLBDMs can incorporate newly developed heuristics. To deal with phishing attacks, both developing new heuristics and calculating the likelihood of being a phishing site are important. In the view of this, I also tried to make new heuristics and introduced the Old OS heuristics and the Country Mismatch heuristics. Otherwise my 2 heuristics were not so accurately, MLBDMs can incorporate these heuristics into the existing 8 heuristics. To simulate the effectiveness of newly created heuristic, I compared the performance with and without using TF-IDF-Final heuristics. By comparing between before and after adding new heuristics, I found that the performance increased in all cases by adding new heuristics. Thus, I predicted that MLBDMs can incorporate new heuristics.

Based on the results, I then discussed several utilization methods for MLBDMs. I found that if novices could accept 25.49% of false positive errors, 95.00% of phishing sites would be blocked as phishing sites. Similarly, if security experts could accept 20.49% of false negative errors, 95.00% sites of legitimate sites would be browsed normally. I also explained my preliminary algorithm for deciding the discrimination threshold of MLBDMs. In short, changing the threshold can provide different strategies against phishing.

Next, I showed the another approach, named HumanBoost, which aimed to cover the weak point of human-being by using machine learning techniques. I prepared 20

sites which were composed of 14 URLs of emulated phishing sites and 6 URLs of legitimate sites. I called 10 subjects and let them to classify if the site seems to be phishing or not. By using users' judgment as a new heuristic, I let AdaBoost to the heuristic to combine the 8 heuristics. Finally, I measured the detection accuracy in the case of AdaBoost-based detection method, the average error rate of subjects, and the average error rate in the case of HumanBoost. My pilot study showed that the average error rate in the case of HumanBoost was 9.5%, whereas the average error rate of subjects were 19.0% and that in the case of AdaBoost was 20.0%.

My proposed HTTP Response Sanitizing (HRS) was a new countermeasure against phishing. Instead of blocking whole web content, HRS only removed the dangerous part in HTML tags, so the loss of convenience in the case of HRS would be lower than that in compulsory blocking. The processing overhead for the phishing site was 3.59 millisecond. When I changed the file size from 10Kbytes to 100Kbytes, the processing overhead increase to 43.37 millisecond. However, I confirmed that the performance of users' browsing would penalize only when the users visited phishing sites.

Finally, I discussed the development of MLBDM-capable phishing prevention systems. In order to clarify the discussion, I showed the stakeholders of the system and the data flow in the system. I then introduced 3 types of implementation forms, named a security service provider (SSP)-side model, a client model, a collaborative model. I also assumed that forms of implementation should be selected after due consideration of both the system resources and the set of the heuristics. The SSP-side model was designed to reduce the client load by performing both learning and detection in the SSP. The client-model was also designed to facilitate the development of the per-user customized systems such as HumanBoost by performing learning and detection in the web client. Although there are pros and cons between these 2 models, the collaborative model was designed to cover their disadvantages; in the collaborative model, the SSP performs learning for reducing the client load, and the client performs detection for reducing the number of queries sent toward the SSP.

This dissertation has shown the countermeasures against phishing, strategies of supporting end users to make trust decision, problems on detecting phishing sites, proposed machine learning-based detection methods for detection of phishing sites, utilization methods to MLBDMs, and development of MLBDM-capable phishing prevention systems.

In summary, this dissertation has demonstrated that machine learning algorithms

were available for detection of phishing sites. This dissertation has focused on detecting of phishing sites with machine learning techniques. Finally, this dissertation has shown the evaluation results which indicated that MLBDMs were feasible solutions against phishing.

11.1 Recommendations for web sites aspect from detecting phishing sites

In this section, I describe my recommendations for both the owners and the developers of web sites. Because my work focuses on detecting phishing sites with machine learning-based techniques, my recommendations for web sites are also focusing on facilitating to distinguish between legitimate sites and phishing sites.

- Checking uploaded content in web space
  I observed some phishing sites were hosted in free web spaces. Even if a site has some advertisements injected by web hosting service provider, the site seems to work as a phishing site. In the view of this, free web spaces should check uploaded content. Otherwise, the number of phishing sites in such free web spaces would increase.

- Removing "word formation" from legitimate websites
  In linguistics, word formation is the creation of a new word. Word formation is sometimes contrasted with semantic change, which is a change in a single word's meaning. However, such words might prevent natural language analysis methods such as the TF-IDF-Final heuristics from distinguish phishing sites from legitimate sites. If the terms were in heavy usage in the web page, these words would be used for the lexical signature, because both TF values and IDF values would be high. If the Google could not catch up the words, the sites would not be appeared in higher ranks.

- Integrating domain names for legitimate enterprises
  Many companies have various domain names as their assets; when they create new products, they usually obtain the domain names oriented to the product. Phishers can abuse the habits of companies and assign such brand oriented domain names to phishing sites. It makes their sites look more convincing. In

addition, the Age of Domain heuristic would not work correctly such newly created brand.

- Avoiding active content for user authentication
  Active content such as Adobe Flash, are available for authenticating web users. It is meaningless for protecting web sites from phishing whenever phishers attempt the man-in-the-middle attack. In the view of detecting phishing sites, it only improves the barrier. Of course, using obfuscation techniques in HTML and/or Script are also doing so.

11.2 Open issues

Toward my ultimate the goal, I found several open issues as follows:

11.2.1 Revealing all phishing sites

Currently, there is no formulated way to counter spear phishing. US-CERT only reminds users that if you are not certain if an email request is legitimate, try to verify it by contacting the company directly [117].

To asses whether or not heuristics-based solutions can deal with spear phishing sites, I must catch up these sites. Otherwise it is difficult to reveal the spear phishing sites, several approaches are available.

One is discovering new phishing sites with a web-crawler which equips the function of MLBDMs. Aside from phishing, a crawler-based detection of spywares were studied. Moshchuk et al. [118] presented the top 10 spyware programs and sites with their crawler-based detection method. These studies are useful to detect phishing sites. If a site is a determinately phishing site and the site is unreported, I assume that the site is spear phishing site. Because the URLs of spear phishing sites are revealed to selected victims, this web-crawler should start from web mailers and/or web-based messaging services, all of which were used for attraction to spear phishing by phishers.

The other is luring phishers to sending spear phishing emails. If I could receive spear phishing emails, I can easily catch up the spear phishing sites. HoneySpam [76] would facilitate to do so, however, it is not enough because phishers want to know the victims' banks and so on. I would attempt to forge fictional persona, and to expose fake personal information along with the persona.

The distinct approach is developing MLBDMs-capable web browsers. The browsers will display spear phishing sites as long as numerous web users employ such web browsers. Even if MLBDMs cannot fully grasp that the site is spear phishing sites when users are just browsing, it would become clear in future. The MLBDMS-capable web browser should expedite the forensic of both web browser and web users' activity.

In addition, all of the sites were written in English because of facilitating to compare with the traditional detection methods. However, there are several phishing sites written in Japanese. I assume that several localization for heuristics must be studied.

11.2.2 Collaboration with other research fields

In Section 3.1, I illustrated 5 phases of phishing attacks and corresponding countermeasures. I would attempt to incorporate these countermeasures into MLBDMs. For example, if an email is suspected a phishing email, the detection methods for phishing sites should perform strictly. Constructing interconnect architecture with other countermeasures is my open issue.

I also assume that solving one security issue also contributes to solve other security issues. Because many phishing sites are hosted on bot-installed PCs [24], studies of detecting bots would help detecting phishing sites. Bots have also function of organizing DoS attacks, virus propagation, sending mails such as SPAM, virus attached-emails, and phishing emails. In the view of this, I have already undertaken to detect virus emails by using AdaBoost [25]. I would continue these research and incorporate them into detection algorithms against phishing.

11.2.3 Research of human factor

In this dissertation, I assumed that phishing prevention systems should provide different strategies for each user. Thus, I employed AUC as a performance metric. I also discussed the optimal threshold for each user and explained the prototype algorithm for deciding the threshold. The rest of problems were to assess the algorithms by performing subject within test and checking the users' safety and convenience.

I also continue to investigate if users' past trust decision is used as a new heuristic in a lesser biased way. If I could call many subjects in a filed test, the bias would be thwarted. I positioned my laboratory test as a first step, and would challenge to perform a field test in a large-scale manner. In order to make a field test be a

successful, my experiment should be designed to attract public attention and should prepare compensation for time cost while subjects judge whether a site is phishing or not.

Per user customization against phishing sites is my open issue. In my preliminary experiment for HumanBoost, I performed Phishing IQ test with subjects because many earlier research also had employed such fashion. However, I assumed that a person does not often enter his/her personal information to the site where the person has no experience to use the site. Imagine if a person has never use PayPal, he/she would not disclose secret to phishing sites which mimicked PayPal. I assumed that the user would not complain when the sites which seem to be PayPal are labeled as phishing. Such kind of information gives a hint to detect phishing sites for protecting particular people.

11.2.4 Research of phishing prevention systems

In this dissertation, I developed 2 heuristics but the effectiveness of these heuristics is marginal. I assumed that feature vector extraction for the automatic classification of phishing sites is available. I also assumed that detecting confusing samples in my dataset would be useful to establish new heuristics. Even if the effectiveness of new heuristics is marginal, I confirmed that MLBDMs can incorporate new heuristics into existing 8 heuristics.

My proposed HRS must sacrifice users' convenience, but the loss of convenience would be lower than compulsory blocking. However, I did not estimate the convenience which could be saved by HRS. Unfortunately, the modern web sites and/or web applications such as Ajax-based applications, often use active scripts to transfer some data to other sites. These sites would not be available whenever phishing prevention systems label such sites as phishing.

I also explore the suitable phishing prevention systems for HumanBoost. The system work after the user made trust decision, and monitor the trust decision by verifying whether or not the user trust the site. The system should cancel the users' data submission when HumanBoost identifies the site as a phishing site.

References

[1] Tom McCall and Radley Moss. Gartner Survey Shows Frequent Data Security Lapses and Increased Cyber Attacks Damage Consumer Trust in Online Commerce. Available at: `http://www.gartner.com/press_releases/asset_129754_11.html`, Jun. 2005.

[2] Tom McCall. Gartner Survey Shows Phishing Attacks Escalated in 2007; More than $3 Billion Lost to These Attacks. Available at: `http://www.gartner.com/it/page.jsp?id=565125`, Dec. 2007.

[3] Anti-Phishing Working Group. Phishing Activity Trends Report - Q1, 2008. Available at: `http://www.apwg.com/reports/apwg_report_Q1_2008.pdf`, Aug. 2008.

[4] Anti-Phishing Working Group. Phishing Activity Trends Report - June, 2005. Available at: `http://www.antiphishing.org/`, Jun. 2005.

[5] MillerSmiles.co.uk. PayPal Phishing Scams. Available at: `http://www.millersmiles.co.uk/report/8221`.

[6] Robert B. Cialdini. *Influence: Science and Practice, 2nd Edition.* Allyn & Bacon, 4th edition, 1981.

[7] Brian Jeffrey Fogg, Leslie Marable, Julianne Stanford, and Ellen R. Tauber. How Do People Evaluate a Web Site's Credibility? Results from a Large Study. Technical report, Stanford, Nov. 2002.

[8] Dominik Birk, Sebastian Gajek, Felix Grobert, and Ahmad-Reza Sadeghi. Phishing Phishers - Observing and Tracing Organized Cybercrime. In *Proceedings of*

The 2nd International Conference on Internet Monitoring and Protection, Sep. 2007.

[9] Markus Jakobsson and Adam Young. Distributed phishing attacks. In *Proceedings of Workshop on Resilient Financial Information Systems*, Mar. 2005.

[10] RSA Security, Inc. Phishing Special Report: What We Can Expect For 2007. Technical report, Nov. 2006.

[11] Abhishek Kumar. Phishing - A new age weapon. Technical report, Open Web Application Securitry Project, Jan. 2005.

[12] Gregg Tally, Rshan Thomas, and Tom Van Vleck. Anti-Phishing: Best Practices for Institutions and Consumers. Technical report, McAfee Research, Mar. 2004.

[13] Alta Van der Merwe, Marianne Loock, and Marek Dabrowski. Characteristics and Responsibilities Involved in a Phishing Attack. In *Proceedings of the 4th International Symposium on Information and Communication Technologies*, Jan. 2005.

[14] Markus Jakobsson. Modeling and Preventing Phishing Attacks. In *Proceedings of Financial Cryptography and Data Security, 9th International Conference*, Mar. 2005.

[15] Yue Zhang, Serge Egelman, Lorrie Cranor, and Jason Hong. Phinding Phish: Evaluating Anti-Phishing Tools. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium*, Feb. 2007.

[16] Neil Chou, Robert Ledesma, Yuka Teraguchi, Dan Boneh, and John C. Mitchell. Client-side defense against web-based identity theft. In *Proceedings of 11th Annual Network and Distributed System Security Symposium*, Feb. 2004.

[17] Arthur Lee Samuel. Machine Learning. *Technology Review*, 62:42–45, 1959.

[18] OpenDNS. PhishTank - Join the fight against phishing. Available at: `http://www.phishtank.com`.

[19] Yue Zhang, Jason Hong, and Lorrie Cranor. CANTINA: A Content-Based Approach to Detect Phishing Web Sites. In *Proceedings of the 16th World Wide Web Conference*, May 2007.

[20] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, 27(3):326–327, 1995.

[21] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[22] Carlos Domingo and Osamu Watanabe. MadaBoost: A Modification of AdaBoost. In *Proceedings of the 13th Annual Conference on Computational Learning Theory*, Jun. 2000.

[23] Alexander Vezhnevets and Olga Barinova. Avoiding Boosting Overfitting by Removing Confusing Samples. In *Proceedings of the 18th European Conference on Machine Learning*, pages 430–441, Sep. 2007.

[24] Gunter Ollmann. The Phishing Guide - Understanding & Preventing Phishing Attacks. Technical report, Next Generation Security Software Ltd., Jun. 2005.

[25] Daisuke Miyamoto, Hiroaki Hazeyama, and Youki Kadobayashi. Detecting Methods of Virus Email based on Mail Header and Encoding Anomaly. In *Proceedings of the 15th International Conference on Neural Information Processing of the Asia-Pacific Neural Network Assembly*, Nov. 2008.

[26] Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web Spoofing: An Internet Con Game. Technical Report Technical Report 540-96, Department of Computer Science, Princeton University.

[27] Ian Fette, Norman M. Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web*, May 2007.

[28] Anthony Y. Fu, Xiaotie Deng, and Liu Wenyin. A Potential IRI based Phishing Strategy. In *Proceedings of the 6th International Conference on Web Information Systems Engineering*, Nov. 2005.

[29] Viktor Krammer. Phishing Defense against IDN Address Spoofing Attacks. In *Proceedings of the 4th Annual Conference on Privacy, Security, and Trust*, Oct. 2006.

[30] Anthony Y. Fu, Xiaotie Deng, Liu Wenyin, and Greg Little. The methodology and an application to fight against Unicode attacks. In *Proceedings of the 2nd Symposium On Usable Privacy and Security*, Jul. 2006.

[31] Jimmy Kuo. The Phishing of Children. Technical report, Mal-Aware.org, Sep. 2007.

[32] US-CERT. Multiple DNS implementations vulnerable to cache poisoning. Available at: `http://www.kb.cert.org/vuls/id/800113`, Jul. 2008.

[33] Symantec. Drive-By Pharming: How Clicking on a Link Can Cost You Dearly. Available at: `http://www.symantec.com/enterprise/security_response/weblog/2007/02/driveby_pharming_how_clicking_1.html`, Feb. 2007.

[34] Anastasios Karakasiliotis, Steven M. Furnell, and Maria Papadaki. Assessing end-user awareness of social engineering and phishing. In *Proceedings of the 7th Australian Information Warfare and Security Conference*, Dec. 2006.

[35] Don Mosley. Some Psychological Factors of Successful Phishing. Technical report, East Carolina University, Jun. 2006.

[36] Zishuang (Eileen) Ye, Yougu Yuan, and Sean Smith. Web Spoofing Revisited: SSL and Beyond. Technical Report TR2002-417, Department of Computer Science, Dartmouth College, Feb. 2002.

[37] Ponnurangam Kumaraguru, Alessandro Acquisti, and Lorrie Faith Cranor. Trust modeling for online transactions: A phishing scenario. In *Proceedings of the 3rd Annual Conference on Privacy, Security, and Trust*, Oct. 2005.

[38] Rachna Dhamija, J. Doug Tygar, and Marti A. Hearst. Why Phishing Works. In *Proceedings of Conference On Human Factors In Computing Systems*, Apr. 2006.

[39] Min Wu, Rovert C. Miller, and Simson L. Garfinkel. Do Security Toolbars Actually Prevent Phishing Attacks? In *Proceedings of Conference On Human Factors In Computing Systems*, Apr. 2006.

[40] Netcraft. Netcraft Anti-Phishing Toolbar. Available at: `http://toolbar.netcraft.com/`.

[41] Amir Herzberg and Ahmad Gbara. TrustBar: Protecting (even Naïve) Web Users from Spoofing and Phishing Attacks. Technical report, Jul. 2004.

[42] Markus Jakobsson, Alex Tsow, Ankur Shah, Eli Blevis, and Youn-Kyung Lim. What Instills Trust? A Qualitative Study of Phishing. In *Proceedings of Financial Cryptography and Data Security*, pages 356–361, Dec. 2007.

[43] Joris Evers. Security Expert: User education is pointless. Available at: `http://news.com.com/2100-7350_3-6125213.html`, Oct. 2007.

[44] Ponnurangam Kumaraguru, Yong Rhee, Alessandro Acquisti, Lorrie Faith Cranor, Jason I. Hong, and Elizabeth Nunge. Protecting people from phishing: the design and evaluation of an embedded training email system. In *Proceedings of Conference On Human Factors In Computing Systems*, pages 905–914, Apr. 2007.

[45] Steve Sheng, Bryant Magnien, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Faith Cranor, Jason I. Hong, and Elizabeth Nunge. Anti-Phishing Phil: the design and evaluation of a game that teaches people not to fall for phish. In *Proceedings of the 1st Symposium On Usable Privacy and Security*, Jul. 2007.

[46] Vivek Anandpara, Andrew Dingman, Markus Jakobsson, Debin Liu, and Heather Roinestad. Phishing IQ Tests Measure Fear, Not Ability. In *Proceedings of Financial Cryptography and Data Security, 11th International Conference*, pages 362–366, Feb. 2007.

[47] Alma Whitten and J. Doug Tygar. Why Johnny Can't Encrypt. In *Proceedings of the 8th USENIX Security Symposium*, Aug. 1999.

[48] Engin Kirda and Christopher Krügel. Protecting Users against Phishing Attacks. *Computer Journal*, 49(5):554–561, 2006.

[49] Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John C. Mitchell. Stronger Password Authentication Using Browser Extensions. In *Proceedings of the 14th USENIX Security Symposium*, Jul. 2005.

[50] Dinei A. F. Florêncio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th International Conference on World Wide Web*, May 2007.

[51] RSA Security, Inc. America Online and RSA Security Launch AOL PassCode Premium Service. Available at: `http://www.rsa.com/press_release.aspx?id=5033`.

[52] Klaus Plössl, Hannes Federrath, and Thomas Nowey. Protection Mechanisms against Phishing Attacks. In *Proceedings of the 2nd International Conference on Trust, Privacy, and Security in Digital Business*, Aug. 2005.

[53] Passmark Security. Protecting Your Customers from Phishing Attacks: an Introduction to Passmarks. Available at: `http://www.passmarksecurity.com/`.

[54] VISA USA. Verified by VISA. Available at: `https://usa.visa.com/personal/security/vbv/`.

[55] Zishuang (Eileen) Ye, Sean Smith, and Denise Anthony. Trusted Paths for Browsers. *ACM Transactions on Information and System Security*, 8(2):153–186, 2005.

[56] Waterken. Trust Management for Humans. Available at: `http://www.waterken.com/dev/YURL/Name/`.

[57] Rachna Dhamija and J. Doug Tygar. Phish and HIPs: Human Interactive Proofs to Detect Phishing Attacks. In *Proceedings of the 2nd International Workshop on Human Interactive Proofs*, Jul. 2005.

[58] Rachna Dhamija and J. Doug Tygar. The Battle Against Phishing: Dynamic Security Skins. In *Proceedings of the 1st Symposium On Usable Privacy and Security*, Jul. 2005.

[59] Kevin Fu, Emil Sit, Kendra Smith, and Nick Feamster. Dos and don'ts of client authentication on the web. In *Proceedings of the 10th USENIX Security Symposium*, Aug. 2001.

[60] Glenn Durfee Dirk Balfanz and D.K. Smetters. *Making the impossible easy: Usable PKI*, chapter 16, pages 319–334. O'Reilly, 2005.

[61] Peter Doyle and Steve Hanna. Analysis of June 2003 survey on obstacles to PKI deployment and usage. Technical report, OASIS Public Key Infrastructure (PKI) Technical Committee (TC), Aug. 2003.

[62] Peter Gutmann. Plug-and-Play PKI: A PKI your Mother can Use. In *Proceedings of the 11th USENIX Security Symposium*, Aug. 2003.

[63] Ari Juels, Markus Jakobsson, and Tom N. Jagatic. Cache Cookies for Browser Authentication. In *Proceedings of 2006 IEEE Symposium on Security and Privacy*, May 2006.

[64] Chris Karlof, Umesh Shankar, J. Doug Tygar, and David Wagner. Locked cookies: Web authentication security against phishing, pharming, and active attacks. Technical Report UCB/EECS-2007-25, Electrical Engineering and Computer Sciences University of California at Berkeley, Feb. 2007.

[65] Yutaka Oiwa, Hiromitsu Takagi, Hajime Watanabe, and Hideki Imai. PAKE-based mutual HTTP authentication for preventing phishing attacks. In *Proceedings of eCrime Researchers' Summit*, Oct. 2007.

[66] CallingID Ltd. Calling ID Toolbar. Available at: `http://www.callingid.com/DesktopSolutions/CallingIDToolbar.aspx`.

[67] Cloudmark Inc. Cloudmark Anti-Fraud Toolbar. Available at: `http://www.cloudmark.com/desktop/download/`.

[68] Earthlink Inc. Earthlink Toolbar. Available at: `http://www.earthlink.net/software/free/tool/`.

[69] GeoTrust Inc. TrustWatch Tool. Available at: `http://tool.trustwatch.com/tour/v3ie/tool-v3ietour-overview.html`.

[70] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A Comparison of Machine Learning Techniques for Phishing Detection. In *Proceedings of eCrime Researchers Summit*, Oct 2007.

[71] Ram Basnet, Srinivas Mukkamala, and Andrew H. Sung. Detection of Phishing Attacks: A Machine Learning Approach. *Studies in Fuzziness and Soft Computing*, 226:373–383, 2008.

[72] Ying Pan and Xuhua Ding. Anomaly Based Web Phishing Page Detection. In *Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference*, Sep. 2006.

[73] Anti Phishing Working Group. APWG: Committed to Wiping Out Internet Scams and Fraud. Available at: `http://www.apwg.com`.

[74] OpenDNS. Providing A Safer And Faster Internet. Available at: `http://www.opendns.com`.

[75] Ari Schwartz. Why Am I Getting All This Spam? Technical report, Center for Democracy & Technology, Mar. 2003.

[76] Mauro Andreolini, Alessandro Bulgarelli, Michele Colajanni, and Francesca Mazzoni. HoneySpam: honeypots fighting spam at the source. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, Aug. 2005.

[77] op25b.jp. Outbound Port 25 Blocking. Available at: `http://op25b.jp/`. (in Japanese).

[78] Hideo Asami. Study Report of an Anti-spam System with a 99Selective SMTP Rejection (S25R) System –. Available at: `http://www.gabacho-net.jp/en/anti-spam/anti-spam-system.html`.

[79] Dinei A. F. Florêncio and Cormac Herley. Evaluating a trial deployment of password re-use for phishing prevention. In *Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit*, Oct. 2007.

[80] NetEconomy. AML Compliance Sale Table. Available at: `http://www.neteconomy.com/amlcompliancesalestable.aspx?r=wikibs`.

[81] Douglas W. Frye. *Email, Instant Messaging and Phishing*, volume 32, chapter 13, pages 319–334. Springer US, 2007.

[82] Ben Woelk. RIT Information Security Advisory: Phone, E-mail, and IM/Social Networking Phishing Attacks. Available at: `http://connect.educause.edu/files/RIT_Advisory_Msg2.pdf`.

[83] Hiroshi Asao. Current situations of AML market. Available at: `http://www.infoex.co.jp/english/profile/pdf/report070629.pdf`, Jun. 2007.

[84] Yoav Freund and Robert E. Schapire. A Short Introduction to Boosting. 14(5):771–780, 1999.

[85] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Science*, 55(1):119–139, 1997.

[86] Yoav Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, Jul. 1996.

[87] Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[88] Narayanaswamy Balakrishnan. *Handbook of the Logistic Distribution*. Marcel Dekker, Inc., 1991.

[89] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[90] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[91] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayesian Ensemble Learning. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, Dec. 2006.

[92] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. BART: Bayesian Additive Regression Trees. Technical report, Department of Mathematics and Statistics, Acadia University, July 2005.

[93] Thomas A. Phelps and Robert Wilensky. Robust Hyperlinks: Cheap, Everywhere, Now. In *Proceedings of the 8th International Conference on Digital Documents and Electronic Publishing, the 5th International Workshop on the Principles of Digital Document Processing*, Sep. 2000.

[94] Alexa Internet, Inc. Alexa the Web Information Company. Available at: `http://www.alexa.com`.

[95] Paul Robichaux and Devin L. Ganger. Gone Phishing: Evaluating Anti-Phishing Tools for Windows. Available at: `http://www.3sharp.com/projects/antiphishing/gone-phishing.pdf`.

[96] Yahoo Inc. Random Yahoo Link. Available at: http://random.yahoo.com/fast/ryl.

[97] Akira Maeda. Linua::LanguageGuesser. Available at: http://gensen.dl.itc.u-tokyo.ac.jp/LanguageGuesser/hajimete_monogatari.html. (in Japanese).

[98] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[99] Michael Holzt. gwhois - generic whois client / server. Available at: http://gwhois.de/gwhois/.

[100] Andy Lester. WWW::Mechanize. Available at: http://search.cpan.org/~petdance/WWW-Mechanize-1.52/.

[101] Avi Finkel. Image::Compare. Available at: http://search.cpan.org/~avif/Image-Compare-0.5/.

[102] Pete Krawczyk. HTML::TreeBuilder. Available at: http://search.cpan.org/~petek/HTML-Tree-3.23/.

[103] Andrewxmp. Google Answers: Number of URLs indexed . Available at: http://answers.google.com/answers/threadview?id=369869.

[104] Paul Bächer, Thorsten Holz, Markus Kötter, and Georg Wicherski. Know your Enemy: Tracking Botnets. Available at: http://www.honeynet.org/papers/bots/, Aug. 2008.

[105] Michal Zalewski. p0f. Available at: http://lcamtuf.coredump.cx/p0f.shtm.

[106] MaxMind. Geolocation and Online Fraud Prevention from MaxMind. Available at: http://www.maxmind.com/.

[107] Blue Coat Systems, Inc. spyware prevention - Blue Coat Systems. Inc. - proxy servers. Available at: http://www.bluecoat.com/.

[108] F5 Networks, Inc. TrafficShield Application Firewall. Available at: http://www.f5.com/f5products/products/TrafficShield/.

[109] Dan Connolly and Larry Masinter. The 'text/html' Media Type. RFC 2854, Internet Engineering Task Force, Jun. 2000.

[110] Björn Höhrmann. Scripting Media Types. RFC 4329, Internet Engineering Task Force, Apr. 2006.

[111] Håkon Wium Lie, Bert Bos, and Chris Lilley. The text/css Media Type. RFC 2318, Internet Engineering Task Force, Mar. 1998.

[112] Jeremy Elson and Alberto Cepra. Internet Content Adaptation Protocol(ICAP). RFC 3507, Internet Engineering Task Force, Apr. 2003.

[113] Squid. Squid: Optimising Web Delivery. Available at: `http://www.squid-cache.org/`.

[114] Privoxy Developers. Privoxy. Available at: `http://www.privoxy.org`.

[115] Free Software Foundation, Inc. GNU Wget. Available at: `http://www.gnu.org/software/wget/wget.html`.

[116] Apache projects. The Apache Software Foundation. Available at: `http://www.apache.org/`.

[117] www.us-cert.gov. Quarterly Trends and Analysis Report. Technical report, Radix Labs, 2008.

[118] Alexander Moshchuk, Tanya Bragin, Steven D. Gribble, and Henry M. Levy. A crawler-based study of spyware on the Web. In *Proceedings of the Network and Distributed System Security Symposium*, 2006.

Appendix A

List of Publications

A.1 Journal

1–1. Daisuke Miyamoto, Hiroaki Hazeyama and Youki Kadobayashi, "An Evaluation of Machine Learning-based Detection of Phishing Sites", *Australian Journal of Intelligent Information Processing Systems*, Vol. 10(2), pp.54–63, Nov., 2008.

A.2 International Conference

2–1. Daisuke Miyamoto, Hiroaki Hazeyama and Youki Kadobayashi, "SPS: a simple filtering algorithm to thwart phishing attacks", In *Proceedings of Asian Internet Engineering Conference*, Bangkok, Thailand, Dec., 2005.

2–2. Daisuke Miyamoto, Hiroaki Hazeyama and Youki Kadobayashi, "A Proposal of the AdaBoost-Based Detection of Phishing Sites", In *Proceedings of the 2nd Joint Workshop on Information Security*, Tokyo, Japan, Aug., 2008.

2–3. Daisuke Miyamoto, Hiroaki Hazeyama and Youki Kadobayashi, "An Evaluation of Machine Learning-based Methods for Detection of Phishing Sites", In *Proceedings of the 15th International Conference on Neural Information Processing of the Asia-Pacific Neural Network Assembly*, pp.184–185 Auckland, New Zealand, Nov., 2008.

2–4. Daisuke Miyamoto, Hiroaki Hazeyama and Youki Kadobayashi, "Detecting Methods of Virus Email based on Mail Header and Encoding Anomaly", In *Pro-*

ceedings of the 15th International Conference on Neural Information Processing of the Asia-Pacific Neural Network Assembly, pp.186–187 Auckland, New Zealand, Nov., 2008.

2–5. Hiroaki Hazeyama, Youki Kadobayashi, Daisuke Miyamoto, and Masafumi Oe, "An autonomous architecture for inter-domain traceback across the borders of network operation", In *Proceedings of the 11th IEEE Symposium on Computers and Communications*, pp.378–385, Sardinia, Italy, Jun., 2006.

2–6. Khamphao Sisaat, Daisuke Miyamoto, "Source address validation support for network forensics", In *Proceedings of the 1st Joint Workshop on Information Security*, pp.387–401, Seoul, Korea, Sep., 2006.

2–7. Yoshitaka Nagami, Daisuke Miyamoto, Hiroaki Hazeyama, Youki Kadobayashi, "An Independent Evaluation of Web Timing Attack and its Countermeasure", In *Proceedings of the 2nd International Workshop on Advances in Information Security*, pp.1319–1324, Barcelona, Spain, Mar., 2008.

2–8. Hiroaki Hazeyama, Mio Suzuki, Shinsuke Miwa, Daisuke Miyamoto, Youki Kadobayashi, "Outfitting an Inter-AS Topology to a Network Emulation TestBed for Realistic Performance Tests of DDoS Countermeasures", In *Proceedings of Workshop on Cyber Security Experimentation and Test*, California, United States, Jul., 2008.

A.3 Technical Report

3–1. 宮本 大輔, 大江 将史, 木村 泰司, 門林 雄基: "OS fingerprint 対策手法の実装と評価", 第 4 回インターネットテクノロジーワークショップ, 2001 年 8 月

3–2. 宮本 大輔, 久保 聡之, 大江 将史, 門林 雄基: "侵入監視のための honeypot の実装と評価", 第 4 回コンピュータセキュリティシンポジウム, 2001 年 11 月

3–3. 宮本 大輔, 鈴木 未央, 櫨山 寛章, 門林 雄基: "フィッシング対策ツールの有効性評価のためのスキャナの提案", 2006 年 暗号と情報セキュリティシンポジウム, 2006 年 1 月

3–4. 宮本 大輔, 飯村 卓司, 門林 雄基: “メール特徴を用いたウィルスメール検知に関する一考察”, 情報セキュリティ研究会, 2007 年 7 月

3–5. 宮本 大輔, 櫨山 寛章, 門林 雄基: “Mesh of Trees トポロジにおけるトレースバックメッセージ配送効率に関する一考察”, インターネットアーキテクチャ研究会, 2008 年 9 月.

3–6. 久保 聡之, 宮本 大輔, 大江 将史, 門林 雄基: “隠しディスクデバイスの実装と評価”, 第 4 回コンピュータセキュリティシンポジウム, 2001 年 11 月

3–7. 益井 賢治, 宮本 大輔, 門林 雄基: “ネットワーク特性共有のための分散型基盤の提案と設計”, 第 7 回インターネットテクノロジーワークショップ, 2005 年 11 月.

3–8. 村越 優喜，宮本 大輔，櫨山 寛章，門林 雄基: “sFlow を用いた IP トレースバック手法の評価”, 第 11 回コンピュータセキュリティシンポジウム, 2008 年 10 月.

3–9. 三輪 信介, 宮本 大輔，櫨山 寛章，樫原 茂，門林 雄基, 篠田 陽一: “インシデント体験演習環境の設計と構築”, 第 11 回コンピュータセキュリティシンポジウム, 2008 年 10 月.

3–10. 三輪 信介, 宮本 大輔，櫨山 寛章，樫原 茂，門林 雄基, 篠田 陽一: “模倣 DNS によるマルウェア隔離解析の解析能向上”, マルウェア解析ワークショップ, 2008 年 10 月

その他

4–1. 宮本 大輔: “ 2007 年の技術から見るサイト運用のポイント”, WebSite Expert #17, pp.130–131, 2008 年 4 月