TECHNIQUES FOR ASSISTING USERS IN MAKING SECURITY DECISIONS

SEVTAP DUMAN

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in the field of

Information Assurance

to the faculty of the

College of Computer and Information Science

NORTHEASTERN UNIVERSITY

Boston, Massachusetts

July 2017

COLOPHON

This document was typeset using `classicthesis` style developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. It is available for LaTeX and LyX at

`https://bitbucket.org/amiede/classicthesis/`

Techniques for Assisting Users in Making Security Decisions

Northeastern University
College of Computer and Information Science

PhD Thesis Approval

THESIS TITLE: *Techniques for assisting users in making security decisions*

AUTHOR: *Sevtap Duman*

Ph.D. Thesis Approved to complete all degree requirements for the Ph.D. Degree in Computer Science.

Thesis Advisor	*Date* 08/18/2017
Thesis Reader	*Date* 8/20/2017
Thesis Reader	*Date* 08/15/2017
Thesis Reader	*Date* 8/18/2017
Thesis Reader	*Date*

GRADUATE SCHOOL APPROVAL.

Director, Graduate School	*Date* 8/24/2017

COPY RECEIVED IN GRADUATE SCHOOL OFFICE

Recipient's Signature	*Date* 8/28/2017

Distribution: Once completed, this form should be scanned and attached to the front of the electronic dissertation document (page 1). An electronic version of the document can then be uploaded to the Northeastern University-UMI website.

ABSTRACT

We are witnessing an arms race between attackers and security experts in today's Internet. Attackers hide their intentions and mimic legitimate behaviour to evade detection. Prominent attacks target end-users' systems with a wide range of goals, such as monetary, financial, political, espionage, destructive. . In this thesis, I examined two well-known instances of these attacks. One of these attacks is the widespread use of trick banners that use social engineering techniques to lure victims into clicking on deceptive fake links and potentially leading to a malicious domain or malware. Other examined approaches involve e-mail attacks, such as spearphishing and e-mail attachment attacks. By impersonating trusted e-mail senders through carefully crafted messages and spoofed metadata, adversaries can trick victims into launching attachments containing malicious code or into clicking on malicious links that grant attackers a foothold into otherwise well-protected networks. Unfortunately, current mitigations are unreliable and relying on fallible malware detection techniques or user education.

Trick banner is an Internet advertising banner with a deceptive visual appearance, crafted to lure users into clicking on them.

Our hypothesis is that online systems can be designed with optimized settings to help users to make security decisions efficiently.

Thus, in this dissertation, I make several contributions to help end-users to make decisions on security:

In Spearphishing attacks the adversary crafts e-mail messages that are custom-tailored to the victim and thus appear legitimate.

- This dissertation shows how to distinguish trick banners from legitimate download links. I present a set of features to characterize trick banners based on their visual properties such as image size, color, placement on the enclosing web page, whether they contain animation effects, and whether they consistently appear with the same visual properties on consecutive loads of the same web page. I have implemented a tool called TrueClick, which uses image processing and machine learning techniques to build a classifier based on five identified features to detect the trick banners on a web page automatically.
- This dissertation shows how to identify a legitimate e-mail sender from a spearphishing e-mail attack. I present a novel automated approach to defend users against spearphishing attacks. The approach first builds probabilistic models of both e-mail metadata and stylometric features of e-mail content. Then, subsequent e-mails are compared to these models to detect characteristic indicators of spearphishing attacks.
- This dissertation aids the end users in making an informed decision about whether or not an e-mail attachment is what they expect. I present adopting a default policy of isolated attachment rendering. E-mails bearing attachments are transparently rewritten to contain static renderings of the attachments within a sandboxed virtual machine environment.

ACKNOWLEDGMENTS

The work presented in this thesis would not be possible without the support, ideas and some great discussions I had throughout my years at Northeastern University.

First, I would like to thank my academic mentor and my advisor Engin Kirda, and my co-advisor William Robertson for their guidance and pushing me to my limits to do better. Their high set bar always kept me on my toes.

Also, I am indebted to Guevara Noubir for supporting me starting from my first day in college.

Northeastern University College of Computer and Information Scicence Dean Agnes Chan and my MIT internship advisor Vincent Chan are the reason I got into Northeastern University in the first place.

I would like to thank my female role model and my thesis committee member Alina Oprea, and my thesis committee member Manuel Egele for helping me to shape my thesis.

I will always remember my days in WVH 208 with a big smile on my face because of Abhishek Samanta, Andreas ten Pas, Hamid Jahanjou, Tim Smith, Xiaofeng Yang, and Maryam Aziz. Special thanks to Amirali Sanatinia for always being there for me for the better and

the worse. While I managed life-work balance poorly, my colleagues in WVH 208 became my lifelong friends.

My network security lab friends Aldo Cassola, Triet Dang Vo Huu, Hooman Javaheri and Jin Tao who graduated during my Ph.D. journey were my go to sources for any inquiries about security. Thank you all for bearing with my endless questions.

I also have to thank my systems security lab mates Ahmet Buyukkayhan, Ahmet Ozcan, Amin Kharraz, Andrea Mambretti, Can Gemicioglu, Kaan Onarlioglu, Michael Weissbacher, Reza Mirzazade, Patrick Carter, Sajjad Arshad, Saman Jafari, and Tobias Lauinger.

This dissertation is also the result of collaborations with Ali Osman Ulusoy, Kubra Kalkan-Cakmakci, and Matthias Buchler.

I owe this impeccable experience to Turkish Ministry of Education for opening a new door for me with their financial support and Educational Attachees' throughout my education for being our family in this foreign country.

I believe I have graduated with my mother Gul Duman's encouragements and my father Abidin Duman's prayers. They had faith in me more than I do in myself. As long as I can remember my brother and my best friend in world Serkan Duman shows me the joy in suffering. Thank you my little brother. Also, I have to thank a million times my fiancee Yasin Kaymaz for his patience with me while I am going through several tooth pains.

Thank you very much for cheering for me during my Ph.D. marathon.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ROC Receiver Operating Characteristic

EMD Earth Mover's Distance

SVM Support Vector Machine

SMTP Simple Mail Transfer Protocol

SURF Speeded-Up Robust Features

HTML HyperText Markup Language

DOM Document Object Model

GIF Graphics Interchange Format

HTTP Hypertext Transfer Protocol

URL Uniform Resource Locator

APT Advanced Persistent Threat

MTA Mail Transfer Agent

TLS Transport Layer Security

MUA Mail User Agent

MIME Multipurpose Internet Mail Extensions

Part I

PROLOGUE

1

INTRODUCTION

1.1 OVERVIEW

Seventy years ago the first bug, a moth, which produced faulty, unintended results in computers was discovered[1]. Nowadays, bugs are not moths anymore, and definitely, they are not the sole reason of security vulnerabilities in computer systems. Rapid advances in computer technology have had profound effects on today's society. Huge amounts of information can be accessed on the Internet, and communication can be made with any number of people across the world almost instantly. The ultimate goal of computer security systems is to detect every attack and prevent attackers from being successful. Maintaining computer security is critical, and achieving this is dependent on each user's awareness of the risks to which information can be violated and their corresponding behavior. Given that the development of new security attacks is constant, users are required to be knowledgeable and vigilant to prevent successful attacks.

The weakest security link of a system is the *ordinary* user (i.e. everyone except security experts) who is targeted or used as a stepping stone to reach a main target. Unfortunately, the number of computer

1 Grace Hopper Murray logged in her notes on September 9, 1947 that "First actucal case of bug being found"

system owners is far more than users who received systems security education. Deception techniques are used by attackers to make detection of potential attacks challenging. The ubiquity of the Internet usage amongst ordinary users has made deceptive techniques popular. The combination of the ordinary users' awareness of the risks to which information can be violated and their corresponding behavior and prosperity of the deception techniques are key factors in ensuring successful attacks.

KEY FACTORS FOR SUCCESSFUL ATTACKS:

1. User Awareness: These uneducated users are challenged every day to make important decisions without knowing the consequences of their actions.

2. Deceptive Attacks: Deceptive attacks take advantage of the (perception of) legitimate systems by forging them through imitation of the look, and the behavior. Malicious contentment is subsequently embedded within the bogus system.

Considering these two major factors, in this dissertation, I focused on the deceptive attacks that take advantage of end-users' actions. Therefore, in order to restrain attackers' success, I explored deceptive attacks that are popular and widely used on web pages and in e-mails.

1.2 MOTIVATION

Advances in technology have made the Internet accessible to billions of users. Users utilize the Internet through a wide variety of devices, such as personal computers, tablets, mobile platforms, and smart televisions. While web activities and e-mail usage have increased on these devices over the past few decades, the number of attacks and attack success ratios have rapidly accelerated as well.

WEB ACTIVITY TARGETED ATTACKS: Traditional forms of advertising using web banners, e-mail campaigns, social media, and mobile marketing schemes are easy for attackers to use deceptive techniques that integrate malicious content into publishers' and application developers' content. Malicious advertisement rate doubled each year since 2011 according to RiskIQ's malvertising reports [96]. For example, a recent and pervasive trend among attackers is to imitate the "download" or "play" buttons in popular file sharing sites (e.g., one-click hosters, video-streaming, and bittorrent sites) in an attempt to trick users into clicking on these fake banners instead of the genuine link. Consequently, attackers have leveraged these channels as efficient mechanisms for distributing malware.

E-MAIL TARGETED ATTACKS: When attacks via e-mails are considered, spearphishing is a prominent targeted attack vector. In Verizon 2017 Data Breach Investigations Report [114], it is stated that imposters are emerging trends amongs e-mail compromises and also

66% of malware is downloaded by malicious e-mail attachments. In spearphishing attacks, the attacker typically leverages publicly available information about the victim, so the attack e-mail is customized to improve the chance that the victim will click on the malicious link. Sophisticated attackers can craft highly-targeted e-mail attachments, using publicly available information about potential victims to create convincing documents that contain hidden malicious payloads. Users who open these attachments using vulnerable applications are at the highest risk of infection. A key obstacle in reducing user vulnerability to attack lies in not being able to easily determine whether an attachment is a legitimate artifact or a malicious document until the attachment has been downloaded and opened.

Naturally, the computer security community has shown an increasing interest in developing approaches to identify and reduce attacks. Researchers have produced a large body of work to address critical security issues surrounding Internet advertising and e-mails using technical based approaches, such as sandboxing of advertisements from the actual content [5, 34, 72], filtering spam e-mails [30, 120], resolving user privacy concerns [20, 46], and preventing ad and e-mail fraud [3, 22]. In spite of this, there have been numerous attacks on high-profile websites, ad networks and e-mail servers in recent years demonstrate the enormity of the issue [61, 71]. Why are such attacks successful in practice? The straight forward answer is likely due to poor user decisions: users often do not have updated system software, are typically overburdened with making security choices, and often end-up blindly clicking any button on a web page or opening

e-mail attachments that are highly risky. There is a lack of available security approaches that aim to aid the user in assessing risk in an easy-to-use format.

Therefore, in this thesis disseration, I focused on the problems of security from the perspective of end-users. I aimed to develop tools that aid the user in assessing risk and adapting their behavior accordingly.

1.3 RESEARCH GOALS

The overall goal of this dissertation is to develop approaches that assists users in making educated security-based decisions about their web and e-mail activity and prevent some devastating attacks.

1.3.1 *Hypothesis and Thesis Statement*

Overall, in this dissertation, I tested the hypothesis that *assisted* Internet users perform better in making decisions to protect themselves from malicious attacks, and improve the quality of their online experience compared to unassisted users. To test this hypothesis, I developed a series of online systems with optimized settings designed to *assist* users in making informed decisions about their web and e-mail activity. I tested the effectiveness of my security systems by conducting user studies and determining the attack-success ratio in each user group.

To this end, my thesis demonstrates that **online systems can be enhanced with additional security protections to help users to make informed security decisions effectively**.

1.4 MY APPROACH

In this dissertation, I explore the problem of designing techniques for automatically assisting Internet users in successfully detecting and protecting against malicious trick banners, spearphishing e-mails, and malicious e-mail attachments.

To address these attack vectors, my specific research goals are to develop tools to help users make informed decisions before they commit potentially risky actions during the following web or e-mail based ideal scenarios:

1. Clicking only on legitimate banners instead of trick banners
2. Identifying whether an e-mail sender is valid when spearphishing e-mail is received
3. Downloading and opening only non-malicious e-mail attachments instead of downloading every attachment

TrueClick determines and blocks the regions on the webpage where trick banners are displayed.

To detect malicious trick banners: I first identified a set of features to characterize trick banners based on their visual properties such as banner size, color, placement on the web page, whether they contain animation effects, and whether they consistently appear with the same visual properties on multiple copies of the same web page obtained with separate requests. I then leverage these features in an ap-

proach combining image processing and machine learning to detect trick banners automatically.

EmailProfiler identifies potential spearphishing attacks on the email recipient's side of the communication.

To detect spearphishing attack e-mails: I developed an approach that incorporated text content with in the received e-mail (in order to identify triage sender information into malicious or legitimate) and that simultaneosly maintained a policy of isolated e-mail attachment rendering. The first approach was designed to identify potential spearphishing attacks on the e-mail recipient's side of the communication. The program extracts a set of 23 features from e-mail headers, and leverages 199 stylometric features inspired by the field of natural language processing. The observed feature values are then aggregated by the sender into a behavioral profile that characterizes the behavior of the corresponding author. Incoming e-mail messages undergo the same feature extraction process where the values are checked for consistency with the existing profile. If the difference of the newly extracted features with the behavioral profile is above the detection threshold, the approach will flag the incoming message as a potential spearphishing e-mail.

PellucidAttachment renders attachments into safe PNG images.

To neutralize e-mail attachment attacks: In addition to detecting spearphishing attacks based on the e-mail sender's information, I focused on e-mail attachment attacks. I developed a technique that (1) converts malicious files to a PNG format (e.g., converting a Word, Excel, or PDF document to a PNG image), (2) removes the exploit code (i.e., executable parts of the document) from the artifact to make it safe to view. With this approach, the user has a chance to check

the authenticity and the validity of the contents in order to make an informed decision about their e-mail attachment content.

1.5 ORGANIZATION

The second chapter of this dissertation provides more detailed information on malicious trick banner identification. In Section 2.2, I stated the malicious advertisement banner problem. My image extraction approach to the problem is presented in Section 2.3. Then I describe the implementation of TrueClick in Section 2.5. In Section 2.6, I explain the results of evaluation of this work. In Section 2.7, I presented the related work in the Internet advertising and advertisement blocking. Before concluding the study in Section 2.9, I discussed the limitations of the study in Section 2.8.

In chapter 3, spearphishing attack detection study is presented. After explaining our motivation for this work in 3.1, I place our design of EmailProfiler in Section 3.2. I then describe a communication protocol for centralized privacy-preserving profile evaluation in Section 3.3. Evaluation of the work is presented in Section 3.4. I explained the differences of our approach with existing work in the context of related work in Section 3.5 before I conclude the study in Section 3.6.

In chapter 4, I present my work on e-mail attachment attacks. In Section 4.2, I provide background information on malicious e-mail attachments. In Section 3 4.3 System Design, I present the overview of our proposed approach. In Section 4.4, I discuss our threat model and our assumptions. Section4.5 describes a prototype implementation

of our approach. Section 4.6 presents an evaluation of the proposed system with real users. Section 4.7 discusses related work and, finally, Section4.9 concludes the work.

Part II

RESEARCH PROBLEMS AND SOLUTION TECHNIQUES

2

TRICK BANNER IDENTIFICATION

2.1 OVERVIEW

Publishers and application developers find it increasingly easy to integrate advertising into their content and, consequently, attackers have leveraged this channel as an efficient mechanism for distributing malware. Naturally, the computer security community has shown an increasing interest in this field as well. Researchers have produced a large body of work to address critical security issues surrounding Internet advertising, such as sandboxing of advertisements from the actual content, resolving user privacy concerns, and preventing ad fraud.

While the security community has primarily focused on finding solutions to the technical side of the problems around Internet advertising so far, the human factor in Internet advertising and the class of attacks that attempt to exploit an Internet user's perception have largely been left unexplored. One well-known instance of such an attack is the widespread use of trick banners [74]. Trick banners are advertisement banners that are crafted to deceive and mislead users into clicking on them, potentially linking to a malicious domain or a malware executable. While trick banners have traditionally come in

the form of colorful and animated messages, or as pop-ups imitating application messages, a more recent and pervasive trend is to imitate the "download" or "play" buttons in popular file sharing sites (e.g., one-click hosters, video-streaming sites, bittorrent sites) in an attempt to trick users into clicking on these fake banners instead of the genuine download link.

Previous work has explored user behavior and security awareness when browsing the Internet, and shown that trick banners found in file sharing sites are effective at tricking even technically sophisticated users who had previous famil- iarity with file sharing sites used in the study[86]. This study concluded that trick banners pose a significant security risk to ordinary Internet users, and even to those with a heightened security awareness. Indeed, abuse of advertisement banners in this way (a practice that is also referred to as "malvertising") has been recognized as a current and effective attack vector[98, 108]. It has also been shown that rather than using complex exploitation techniques, simply buying ad space is an easy and effective way for attackers to spread malware and quickly victimize a large number of Internet users[44]. Numerous attacks on high-profile websites and ad networks in recent years demonstrate that the problem is real and is actively being exploited [79, 119].

In this work, we explore the problem of automatically assisting Internet users in successfully detecting malicious trick banners, focusing on distinguishing fake download buttons ubiquitously found in popular file sharing websites from genuine download links. We first identify a set of features to characterize trick banners based on their

visual properties such as banner size, color, placement on the web page, whether they contain animation effects, and whether they consistently appear with the same visual properties on multiple copies of the same web page obtained with separate requests. We then leverage these features in an approach combining image processing and machine learning to automatically detect trick banners on a web page.

We implement our system in a prototype Firefox browser extension called TrueClick, evaluate its effectiveness on a data set of 259 banners collected from 88 file sharing websites, and demonstrate that TrueClick achieves a 96.97% true positive rate given a false positive rate of 3.03%. Note that unlike state-of-the-art ad blocking, our approach does not require a priori blacklisting of advertising domains, or any other manual classification of known banners. After an initial training phase, it operates in a completely automated manner by analyzing the visual properties of banners. In other words, the approach we propose is complementary to existing blacklisting approaches, and can support them in identifying previously unknown trick banners. Moreover, TrueClick does not rely on examination of the source code of web pages or the structure of the Document Object Model (DOM) tree. Instead, it utilizes image processing techniques to capture and analyze web pages as the user sees them, and therefore is not affected by attempts to thwart detection through dynamic modifications to the page.

In summary, we make the following contributions in this work.

- We present five visual features that can be used to characterize trick banners and experimentally demonstrate that these fea-

tures can be used in practice to distinguish trick banners from genuine download links.

- We present a novel methodology for automatically and reliably distinguishing trick banners from genuine download links by using a combination of image processing and machine learning.
- We describe a prototype implementation of our solution as a Firefox browser extension called TrueClick that can effectively guide users toward finding and clicking on the genuine download link in a file sharing website littered with trick banners.
- We evaluate the usability of TrueClick with a user-study, and demonstrate a 3.55 factor improvement in selection of benign links in the presence of trick banners. We also show that TrueClick is able to detect trick banners missed by a popular ad detection tool, Adblock Plus.

2.2 PROBLEM STATEMENT

A trick banner is generally defined as any Internet advertising banner with a deceptive visual appearance, crafted to lure users into clicking on them [74]. They often do not contain any indication of the identity of the advertiser or the advertised service or product. Trick banners are known to integrate well with the look and feel of the website they appear on, and often imitate popular applications, operating system windows, and pop-up messages.

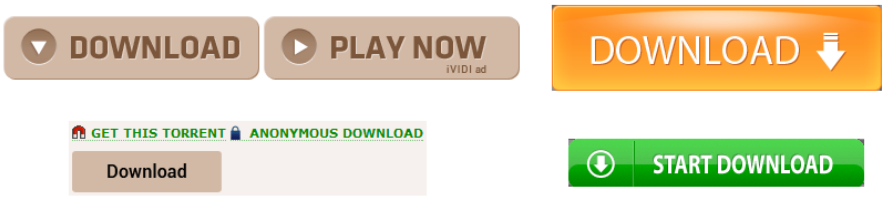


Figure 2.1: Four trick banner samples taken from popular file sharing sites

In this work, we focus on a specific pervasive class of trick banners: fake download buttons found on file sharing websites, which have recently been shown to be effective at tricking even users with security expertise[86]. Figure 2.1 shows various examples obtained from popular file sharing services The Pirate Bay, Rapidgator, and FilesTube that imitate the look and feel of genuine download buttons to deceive users. Note that the techniques we present are sufficiently generic to be applied to other kinds of trick banners as well. We believe that fake download buttons represent an up-to-date manifestation of the more general trick banner problem, and pose a challenging research task because of their tight integration with the file sharing sites they are displayed on. To illustrate, one of the sample trick banners taken from The Pirate Bay (Figure 2.1, lower left corner) displays identical replicas of the correct download links (i.e., the links "GET THIS TORRENT" and "ANONYMOUS DOWNLOAD"), making it an especially difficult trick banner to spot for an unsuspecting user.

Also note that banner design (and user interface design in general) for attracting users and maximizing click-through rates has been extensively studied in computer science and other, non-technical fields [2, 16, 21, 49]

Our use of the term trick banner in this research could refer to images with either benign or malicious intent. In other words, trick banners we examine could be crafted with malicious intent, such as directing a user to an attack site or downloading malware to her computer. Or, they could simply be used as a device for artificially inflating the number of visitors to a destination website, without explicit malicious intent. In this work, we do not aim to verify whether the trick banners lead to malicious destinations (other recent work has explored this aspect of the problem ([65]), nor do we visit the sites or download the files they link to. Instead, our goal is to detect trick banners regardless of their purpose, and distinguish them from legitimate, genuine download links.

We divide the threat model we consider for this work into two scenarios. In the first scenario, an Internet user visits a file sharing website with the intention to download a specific file. We do not make any assumptions about the security expertise or awareness of the user. The pages on the website can contain any number of trick banners and regular advertisements, and one or more correct download buttons defined as the link to the content the user intends and expects to download. In cases where the website requires the user to step through a number of different pages to complete the download, we take the links that lead the user closer toward the final download link as the correct one.

The second scenario is identical to the first except that the website in question does not actually contain any correct download links. Such sites could be found on the Internet, for example, as part of a

well-known scam scheme in which a scammer creates web pages that contain detailed descriptions of various content despite not including any actual download links, possibly in an attempt to trick search engines and steal clicks from users for ad revenue.

In both of these cases, regardless of whether a correct link exists or not, our system analyzes the web page and determines the page regions that contain trick banners. If trick banners are detected, they could be marked with warning cues to alert the user or blocked entirely, depending on implementation choices or user preferences.

Finally, we would like to point out that there exist other types of attacks involving malicious modifications to websites and browser user interfaces that aim to make users inadvertently click on incorrect or malicious links. For example, various forms of clickjacking attacks compromise the visual and temporal integrity of the pointer cursor or browser's display to this end [53]. The problem we aim to address in this work is separate from those attacks in that, in our threat model, the system's integrity is not compromised; instead, trick banners exploit weaknesses in human perception to trick users. In clickjacking attacks, the user's system is technically crippled, making her unable to click on the correct link even if she can identify it. However, in the attacks we aim to address, the user *willingly* clicks on a trick banner, thinking that it is a genuine download link. While defenses against both types of attacks are necessary for secure browsing, we only explore the latter problem in this work.

Figure 2.2: Image extraction process

2.3 IMAGE EXTRACTION

Before we can compute any features on potential trick banner images, we first need to identify and extract all of the image regions on a web page, which are subsequently fed to our trick banner classification system. Note that simply searching for HTML image tags in the page source is not suffcient to perform this task correctly, because some of the banners may be loaded dynamically by JavaScript, or they may come in non-image formats like Flash files or regular links stylized to look like buttons. In this section, we briefly explain the details of this process.

The image extraction technique we propose in this work is a two-step process. Initially, we leverage well-known image processing techniques designed for this task, which follow the common pipeline of edge detection, region filling and connected component analysis [39], and banner region identification on the web page. However, the enormous variability in web page content often does not allow a single generic image processing pipeline to perform perfectly in all cases, and our early experiments indicate that extracting image regions solely through image processing usually falls short. For example, when faced with a trick banner that displays several button

lookalikes in a single image file, the aforementioned image processing pipeline yields multiple detection results for each disconnected component in the image. The banner presented in 2.2 for a real example. The image extraction process illustrated on a real trick banner. Step 1 detects four superfluous sub-regions on the banner, Step 2 then corrects this error, and matches the fragments to the actual image. Similarly, more sophisticated visual designs could result in a large number of superfluous detections of small sub-regions on a single image (or, conversely, missed portions of banners), which would later result in unnecessary detection feature computations or inaccurate results. Although image processing methods that are customized for each specific website can be devised to improve extraction accuracy and performance, such an approach would be time and effort-intensive, without any clear indication as to how the extraction scheme could be adapted to the future banner designs.

Instead, based on the observation that while the discussed image processing techniques are imperfect, they rarely completely miss entire banner regions, we employ a second step to correct for partially detected and fragmented banners. This involves collecting and temporarily caching all of the actual image files requested from the web server for a given web page to form a small image database. Next, the possibly fragmented banners extracted using the initial image processing step are matched to the images in the database. This matching is performed using the Speeded-Up Robust Features (SURF) (Speeded-Up Robust Features) feature detector and descriptor, which is widely used for object detection and recognition in the computer vision com-

munity[11]. Our experiments show that even when a significant portion of the banner is fragmented, SURF is able to match the banner to the correct image in the database. When a match is found, or in other words the extracted region is detected as a fragment of a larger image, the extracted banner image is simply replaced with the correct image from the database.

This process is illustrated in 2.2. In Step 1, the image processing pipeline incorrectly identifies four separate regions in the banner. Later, in Step 2, these four extracted regions are all matched to the actual banner image stored in the database (and are replaced by it), resulting in an accurate banner extraction.

We must point out that even in the presence of the image database we build in Step 2 of this process, the banners extracted through image processing in Step 1 still provide valuable information for detecting trick banners; specifically, those that come in non-image formats. For example, in our experiments we observed static banners delivered in Flash files, or as regular links in HyperText Markup Language (HTML) iframes stylized to look like buttons. While image processing can identify and extract such non-traditional banners, attempts to match them to image files in Step 2 would fail since there does not exist a corresponding image file on the web page. Therefore, in cases where we cannot find any successful match in Step 2, we do not discard the regions extracted in Step 1 but instead input them to the classifier as-is.

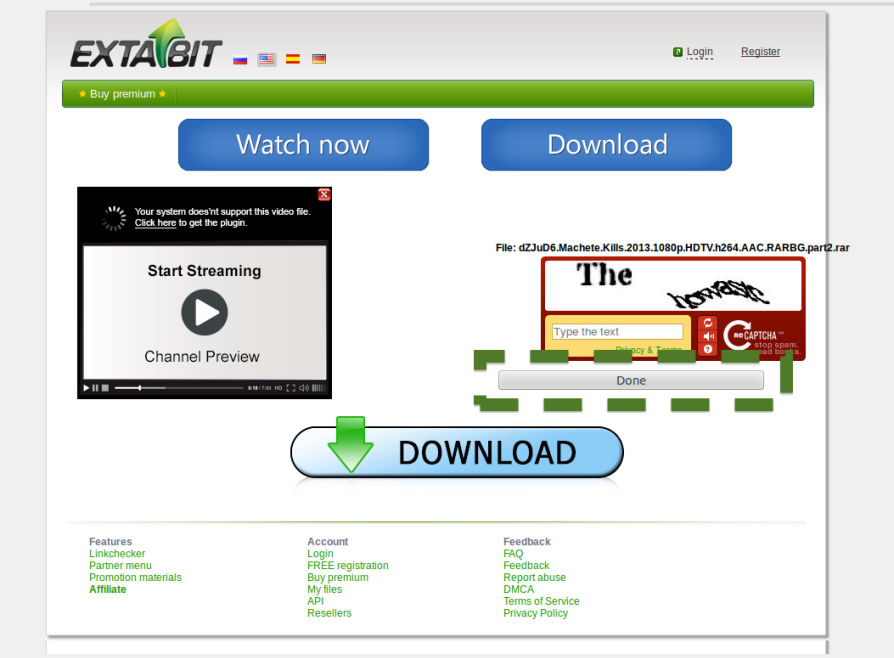
2.4 TRICK BANNER CLASSIFICATION

Once possible trick banner regions have been identified on the web page, five visual features are extracted to help classify each region as either a trick banner or a genuine download link. These features include *1)* image color, *2)* image size, *3)* image placement, *4)* presence of animation, and *5)* image differences between consecutive page loads. In the remainder of this section, we provide details on each of these features, explain why they are useful for distinguishing trick banners from genuine download links, and then present the classification approach we adopt in this work.

2.4.1 *Features*

2.4.1.1 *Color*

Trick banners are often not designed by the site owners, and are usually served by third-party advertising networks just like regular ad banners. Consequently, the designers of trick banners do not know the exact website their banners are going to be displayed on, which leads them to follow common web page theme specifications in their visual designs. As a result, trick banners often do not fit the general color theme of the website, but instead display distinctive color signatures. In contrast, genuine download buttons usually cohere to the overall website colors. This distinction suggests that banners can be classified based on their color similarity to that of the overall website.



(a) Sample web page: ExtaBit

(b) Sample web page : VodLocker.com

Figure 2.3: Sample web pages that illustrate that EMD scores vary

This classification requires first a description of the colors inside the banner region. Experiments indicate both genuine and trick banners can be quite complex in terms of the color patterns they contain. Most banners are composed of a multitude of colors, highlights, and gradients, and also include other small images.

Color histograms are ideal for the purposes of capturing the global color features of the banner. Histograms are constructed simply by binning the color of each pixel in the banner region. The histograms are finally normalized by their total mass such that they are invariant of the number of pixels in the banner region. Classification based on color histograms also requires a method to compute the similarity between histogram pairs. In our work, we compute color histogram similarity using the EMD [97], which is widely used in content-based image retrieval applications. In short, EMD is a metric between two distributions (of equal total mass) that measures the minimal cost incurred to transform one distribution to the other. The comparison is made between each banner and the whole page histograms. We made our computations on the RGB channel.

Note that the EMD score of a banner is computed with respect to the color histogram of the overall web page and, thus, this score provides a measure relative to the other banners within the web page. For instance, web pages in Figure 2.3 that illustrate that EMD scores vary with the color theme of the websites, and should only be compared after normalization. The correct banners are marked with dashed boxes. On the web page in Figure 2.3a, the EMD score of the correct banner (marked with dashed lines) is 17443.67, and the trick banners on the

same web page have higher scores than the correct banner. However, we cannot generalize this outcome and consider scores higher than 17443.67 obtained from different pages to indicate a trick banner. To illustrate, the web page in Figure 2.3b has a correct banner that has an EMD score of 19773.83, and while this value is still lower than the EMD scores of the trick banners on the same web page, it is higher than 17443.67. Therefore, a normalization is required so that the EMD scores of all banners in training and classification are comparable. Mapping the EMD scores of banners in each website to a fixed interval such as [0, 1] is suffcient for this purpose.

2.4.1.2 *Size*

Websites often reserve fixed sections in their visual layouts where advertisements can be displayed. The sizes of these reserved spaces are not strictly controlled. However, in accordance with the Interactive Advertising Bureau online advertisement size guidelines [56], these sections are usually large, horizontal, or vertical rectangular regions in which standard web banners that advertising networks serve can fit. In contrast, genuine links on a file sharing website usually take the form of a single, relatively small image that serves as a download button. Consequently, in order not to throw off their disguise by displaying unusually large fake buttons that contrast with the rest of the page's style, many trick banners resort to tricks such as using large empty borders around a smaller fake button image, or including two or more button images on a single banner as if they were separate

clickable entities. Thus, image size is a strong feature for distinguishing trick banners from genuine buttons.

We measure the size of these images in the x and y dimensions in the number of pixels. In order to deal with varying web page sizes, we first normalize the numbers to $[0, 1]$ with respect to the absolute size of the enclosing web page.

2.4.1.3 *Placement*

Navigation links on a website, including the genuine download buttons, are tightly integrated with the rest of the site's content. In contrast, advertisement banners are often laid out separately in reserved spaces in order not to interfere with the coherence of the website's interface and content. They are usually placed at the page header, footer, sidebars, or are otherwise isolated from the actual page content. Therefore, the position of the banner can be used as an indicator for trick banners. We use the x and y positions of the geometric center of the banner as the placement feature. Similarly to the size feature, we first normalize the values to $[0, 1]$.

2.4.1.4 *Animation*

Another significant indicator for trick banners is the use of animations that are employed to draw the attention of the user. Animated banners rapidly display a sequence of images, typically in the form of Graphics Interchange Format (GIF) images. Most genuine download links do not contain animations. In fact, during our study we didn't encounter any genuine links that contained animations.

Animation is a binary feature, indicating whether or not the banner is animated. The presence of animation is detected by first checking whether the banner image format allows animations. If it does, the number of frames embedded in the file is used to decide the presence of animation. This animation check is performed only on the images that were selected from the database because it is not possible to reach fragmented image information unless it matches with a cached image.

2.4.1.5 *Visual Differences*

A common method for deploying advertisements on a page for a website owner is to utilize advertising networks, which serve a different ad banner every time a user visits the page. Similarly, large content publishers may use their own advertising infrastructures that rotate the banners displayed on the page each time the page is visited. Consequently, the visual contents of banner spaces on web pages tend to be very dynamic, often changing every time the page is loaded or refreshed in the browser. In contrast, the user interfaces of web pages are often comprised of a fixed set of images that seldom change once the website's design has been finalized. To promote usability and provide a consistent user experience, menus, navigation links, and buttons on the page are placed at specific positions and use static images.

We take advantage of the dynamic characteristics of banners and the static nature of the rest of the UI elements on a web page to propose a trick banner detection feature based on comparing multiple views of a single web page. Specifically, we first take two screen

captures of the same page obtained through two separate requests to the web server. We then visually compare them, extract the parts that have changed between the requests, and mark those as potential banner regions.

2.4.2 *Trick banner Classifier*

A binary classifier is a function that takes as input a set of features, such as the visual features described above, and outputs a binary decision – in this case, *trick banner* or *non-trick banner*. The machine learning literature offers a wide variety of binary classifiers [48]. In this work, we choose to use the popular random forest classifier, and train it using the method proposed by Breiman [13]. The details of our training data collection methodology is explained in Section 2.6. Note that although the random forest is used for all experiments in this work, we have observed that the results are comparable using other state-of-the-art classifiers such as the support vector machine.

2.5 IMPLEMENTATION

We implemented the trick banner detection methodology in a prototype system called TrueClick, as a Firefox browser extension that uses external image processing libraries. In this section, we explain the implementation-specific details of our system.

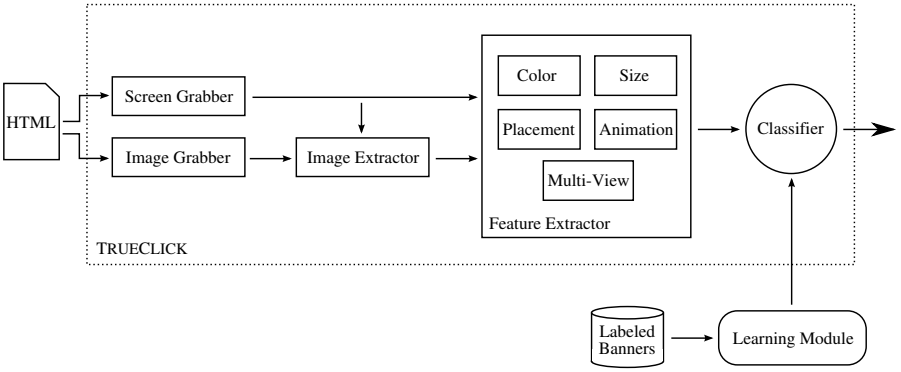


Figure 2.4: Overview of the architecture and various components of TrueClick.

2.5.1 *System Overview*

TrueClick is implemented as a browser extension that runs on demand when the user visits a file sharing website containing trick banners and clicks on a button to activate the system. Once the analysis of the banner images is complete, TrueClick can either mark the detected trick banners as such, or block them entirely. In this prototype implementation, we elected to visually obscure the trick banners from the user.

An overview of the architecture of TrueClick is presented in Figure 2.4. When the user triggers the analysis on a given web page, the *Screen Grabber* and *Image Grabber* components first take screenshots of the web page and cache all image files downloaded from the web server in an image database, respectively. Then, the screenshots and image database are input to the *Image Extractor* component, which identifies the banner regions on the screenshots, and attempts to match them to the files in the database, as explained in Section 2.3. Once all banner regions are detected, they are sent to the *Fea-*

ture Extractor which computes the five trick banner detection features on each image. Finally, TrueClick runs the resulting feature vectors through its classifier and determines the regions on the web page where trick banners are displayed.

Note that during this process, all extracted images smaller than 16 X 16 (i.e., the standard size for favicon files) are discarded in order to speed up the detection process and display more relevant warnings to the user since they are most likely not banners.

In the following, we elaborate on the details of the Screen Grabber and Image Grabber components.

2.5.2 *Screen Grabber*

The Screen Grabber component is primarily responsible for taking a screenshot of the web page as rendered by the browser. In order to ensure that the resulting screen capture is identical to what the user of the browser sees, TrueClick copies every pixel displayed in Firefox's main browsing window in a hidden HTML *canvas* internal to the browser, and dumps the results into an image file.

This component is also tasked with providing the necessary information for the Feature Extractor to identify the visual difference in multiple views of the same web page. To this end, once TrueClick is activated, the Screen Grabber issues an additional Hypertext Transfer Protocol (HTTP) request for the displayed web page, renders it in a hidden browser window, and uses this to take a second screenshot of the page with (potentially) different banners. Note that during this

process, care must be taken to ensure that the dimensions and display properties of the hidden window are identical to that of the original window so that the two screenshots obtained match and the subsequent comparison can be carried out accurately.

2.5.3 *Image Grabber*

The Image Grabber component identifies the image files referenced by the web page, and builds a temporary image database to be used by the Image Extractor to match fragmented banners against. However, simply parsing the DOM tree of the web page is not an effective way of accomplishing this task since many banners are displayed dynamically after page load by JavaScript ad libraries. Similarly, even when the Uniform Resource Locator (URL)s to the image files can be detected, downloading the banners from those locations is not a reliable way of obtaining the images, because the URLs provided by ad delivery frameworks often rotate between different banners and serve different image files with each request.

In order to address this problem, Image Grabber transparently intercepts HTTP responses from web servers and inspects the payload. Once it has been determined that the data corresponds to an image file, it is copied and inserted to the image database. This technique has the additional benefit of avoiding downloading the same image files a second time, saving bandwidth and allowing for quick detection of trick banners.

2.6 EVALUATION

In this section, we describe our experiments to measure the accuracy and usability of our solution in identifying trick banners in the wild. We evaluated the effectiveness of our classifier on a data set of banners we collected, conducted a user study to demonstrate that TRUECLICK is practical, and finally, compared the detection effectiveness to an existing ad detection system, AdBlock Plus.

2.6.1 *Data Collection*

We collected trick banner samples and images of genuine download buttons from popular file sharing websites, including one-click hosters, bittorrent sites, and online video streaming sites to train and evaluate our system. Our training data set consists of only English websites, while the evaluation data set contains both English and non-English websites. We chose to perform the data collection and labeling procedure manually instead of crawling these websites, so that we could use cues from the pages to determine whether the collected samples are trick banners or not with high confidence and train our classifier with an accurate data set. We reached the actual file download pages by searching for popular movies and computer programs at the file sharing websites and services. In the banners we collected, we looked for keywords that could be intended to trick users such as *Download*, *Watch*, *Now*, *Save*, *Play*, *Get it*, and *Store*.

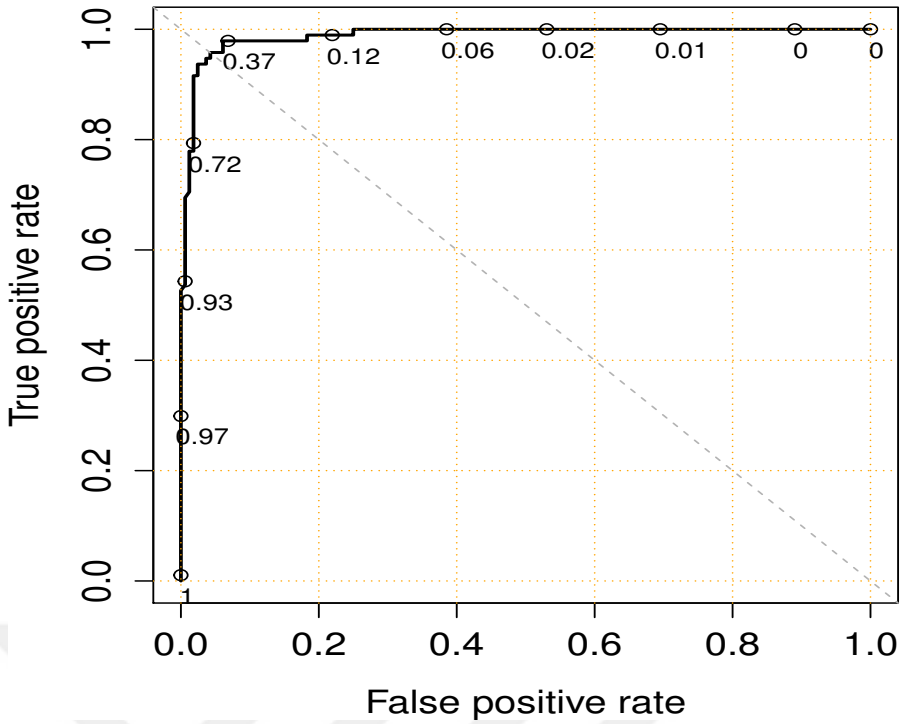


Figure 2.5: ROC curve for a 10-fold cross validation of TrueClick 's random forest classifier.

We note that such file sharing websites are not exclusively used for illegal media trading, but are also often used to distribute media to large audiences (e.g., software updates, non-commercial documentaries). Hence, our aim is to protect users in general, even though some might be engaged in illicit behavior when they are tricked by malicious banners.

We used 165 trick banners and 94 images corresponding to genuine download links for a total of 259 banner samples from 88 file sharing websites to train and evaluate our classifier. For the comparison with AdBlock Plus, we used a disjoint set of 415 trick banners collected from 82 websites. In total, we collected 674 banner samples from 170 file sharing websites.

Table 2.1: Information gain for each feature.

FEATURE	IMPORTANCE (BITS)
x-position	23.37
y-position	25.91
Size	100.06
Color	28.71
Animation	21.98
Multiview	52.14

2.6.2 *Evaluation of the Classifier*

To build a classifier to distinguish between trick banners and genuine download links, we used the R statistical machine learning environment and, specifically, the ipred package, to train a random forest classifier from which the importance of predictors were assessed.

Figure 2.5 displays a ROC curve for a 10-fold cross validation of the resulting random forest classifier over our training set, using a cutoff value of 0.1. An n-fold cross validation partitions the entire data set into n equal-sized samples, or folds, trains on n-1 folds, and then validates the resulting model on the remaining fold. This process is repeated for each fold. The ROC curve plots the true positive rate against the false positive rate of the best-performing classifier as the discrimination threshold – i.e., the boundary between the trick banner and genuine link classes – is varied over [0,1]. The value of the ROC curve lies in the guidance it provides in selecting thresholds to bias towards true positives at the expense of false positives, and vice versa.

The ROC curve shows that our classifier achieves a 96.97% true positive rate given a false positive rate of 3.03%, which is lower than the

Table 2.2: Results of the TrueClick user study, showing the number of correct and incorrect clicks.

EXPERIMENT	CORRECT CLICKS	INCORRECT CLICKS	MEDIAN SCORE	AVERAGE SCORE
Original Page	29	91	0	0.725
w/ TrueClick	103	17	3	2.575

critical false discovery rate threshold of 5%. In other words, 3.03% of trick banners were incorrectly classified as correct banners. As shown in Table 2.1, the size feature is the most effective in distinguishing trick banners from legitimate download links. The table is generated using out-of-bag samples from the training data.

2.6.3 *Effectiveness of Trick Banner Detection*

We tested TrueClick 's usability and effectiveness in guiding users to identify and click on the genuine download links on a file sharing website by conducting a user study, and comparing its detection performance to AdBlock Plus.

2.6.3.1 *User Study*

We performed our user study on 40 undergraduate and graduate computer science students. While we did not explicitly evaluate the participants' technical savviness, it is reasonable to expect them to be relatively expert computer and Internet users.

We first briefed all participants that they were going to take part in a user study on identifying the genuine download links on English-only file sharing websites, and then instructed them to click on the

link or button they thought was legitimate on the web pages they were shown. Next, we presented each participant with unmodified pages from three different file sharing websites for them to perform this task on. In order to control for the fact that advertisement banners change every time a page is requested, we did not use the actual websites in our test, but instead created identical offline replicas with a fixed set of banners. Once the participants completed the first three tasks, we repeated the experiment using the same three websites, but this time in a browser window running TRUECLICK so that our system could analyze the page content and mark detected trick banners as such. We observed each participant complete all six tasks, recorded the number of correct and incorrect clicks for each of them, and assigned scores based on the number of correct clicks. The results are shown in Table 2.2. The presence of incorrect clicks when using TRUECLICK is due to trick banners missed by our system.

These results demonstrate that in the experiments where the participants were assisted by TrueClick, there was a 3.55 factor improvement in the scores on average. We also checked these results for statistical significance using a standard paired difference test, namely, the Wilcoxon signed-rank test. The results of the test (V = 0, p < 4.82 X 10-7) confirm that the scores obtained with and without TRUECLICK indeed constitute non-identical populations.

2.6.3.2 *Comparison to AdBlock Plus*

Existing systems such as Adblock Plus are capable of identifying and blocking trick banners in some cases. To understand whether

TrueClick is necessary given the existence of such systems, we compared its detection effectiveness with Adblock Plus.

We conducted experiments with 415 manually-identified trick banners from 82 websites that were not used in the training phase of the previous experiments. TrueClick correctly identified **380 (91.6%)** of these as trick banners using the previously-generated classifier without extra training or manual tuning, whereas Adblock detected 190 (45.8%). In contrast, there were only 8 banners detected by Adblock Plus, but not TrueClick.

We stress that these result do not suggest TrueClick is a substitute for Adblock Plus. TrueClick, as discussed and evaluated in this work, focuses on detection of trick banners, and this experiment is not su cient to draw conclusions on its ability to detect ordinary, benign advertisements. Therefore, we conclude that while Adblock Plus provides an effcient filter against general Internet advertising, supporting it with TrueClick significantly improves protection against potentially malicious trick banners.

2.7 RELATED WORK

In order to highlight the differences between our work and current advertisement detection systems we present related work and the state-of-the-art in this field.

2.7.1 *Internet Advertising and Trick Banners*

There is a large body of prior work on Internet advertising and the technologies around it, both in computer science and other non-technical fields. The effectiveness of various types of ad banners and ways to influence user click-through behavior have been studied widely in marketing, finance, psychology, and related fields [17, 35]. In a recent work, Onarlioglu et al. [86] presented a study that investigates the computer security implications of trick banners, and showed that trick banners can mislead even technically sophisticated Internet users and expose them to attacks.

2.7.2 *Advertisement Blocking and Filtering*

A simple method of blocking trick banners, and advertisements in general, is to disable prerequisites for displaying such content in web browsers. This can involve disabling image loading, blocking Flash Player and similar browser plugins, or blocking JavaScript code used by advertising networks through widely available browser security extensions [55]. Similarly, Web proxies could be deployed to filter out trick banners before they could be displayed in the browser [91, 103]. While these solutions are effective at blocking trick banners, they also significantly impair the user's browsing experience, or even render many websites nonfunctional, as the World Wide Web today makes extensive use of multimedia and dynamic content. Moreover, deploy-

ing HTTP proxies might not be accessible to the average Internet user, or might not be a viable option on more restricted mobile devices.

An alternative approach to the problem is using specialized ad filtering software that is often included as part of commercial antivirus suites, or designed as open-source web browser extensions such as the popular Adblock Plus [3]. On the one hand, these solutions offer the ability to selectively block offending content and therefore provide improved usability over the previously discussed solutions. On the other hand, they typically detect links to advertisements by consulting various blacklists and whitelists that must be continuously maintained and updated, which often involves significant manual labor. TrueClick instead performs trick banner detection based on the visual characteristics of such content in a completely automated manner and, thus, is able to detect trick banners that have not previously been classified into blacklists or whitelists.

2.7.3 *Security Uses of Visual Features*

Another body of work applies image comparison techniques and visual similarity metrics to different Internet security problems. For instance, various groups have investigated phishing site detection techniques based on visually comparing suspected phishing pages to their legitimate counterparts [19, 75, 77, 118], and Gargiulo and Sansone [36] use image processing to extract visual and text features from spam emails. In contrast, we identify visual features specifically tailored to detect trick banners and implement them in TrueClick.

Doppelganger [7] explores the results of different cookie policies on websites by transparently mirroring the user's web session to create two conceptual browser windows, one with cookies enabled and one without. They then compare the two to investigate the impact of accepting cookies from a given website on the content displayed. Likewise, TrueClick utilizes a detection feature based on comparing multiple views of a web page, as we explain in 2.4

2.7.4 *Securely Isolating Ads and Applications*

A large number of studies aim to protect sensitive web content from potentially malicious third-party ads by sandboxing the ads displayed on the page. Several projects use language containment and static policy enforcement to restrict JavaScript features used by ad networks [5, 34, 45, 52, 72], while others perform dynamic policy enforcement [41, 58, 109]. Likewise, researchers have investigated the ad ecosystem on mobile devices and proposed approaches that aim to isolate third-party ad libraries from mobile applications [42, 64, 89, 100]. Other approaches to create general secure mashups of advertisements and applications include secure browser architectures and browser-based operating systems [43, 94, 116].

In another set of studies, researchers explore the privacy issues around Internet advertising, and propose techniques to deliver targeted ads while limiting the exposure of privacy-sensitive user information to ad networks[46, 47, 59, 95, 111] .

These studies have the common goal of protecting applications against uncontrolled and potentially malicious ad code. While the isolation they provide is clearly essential for secure deployment of online ads, the measures they employ do not provide protection in a scenario in which Internet users are deceived and misled to click on a trick banner that links to a malicious destination. In contrast, TrueClick addresses this problem by analyzing the ad banners on a web page and guiding users to identify the genuine download link among a set of trick banners.

2.8 DISCUSSION

The system we propose for automated trick banner detection has some limitations that we highlight here. First, we stress that in contrast to other work on trick banners and attacks against users, TrueClick is intended to address the specific case of images that masquerade as genuine download or play links on web pages. While the techniques we make use of here could be extended to cover other attacks, that is not the focus of this work.

As we noted in our evaluation of the random forest classifier TrueClick uses, the models we are able to build over our data set generate a non-trivial number of false positives. However, our models nevertheless classify the majority of trick banners correctly even when the threshold is set to produce a 0% false positive rate. Therefore, this represents a significant reduction in the number of trick banners that users must navigate, and our user experiments demonstrate that this translates

to much better security decisions in practice when TRUECLICK is deployed.

We note that due to variations in content between consecutive page loads, we have observed our system to classify some *non-clickable* regions of the page as trick banners. However, due to the way in which we deploy our classifier in the browser, this does not have generally have a deleterious effect on the user experience as the mislabeled content is non-interactive and obscuring it does not affect the functionality of the page.

Finally, an unlikely but interesting limitation of our system we discovered during our experiments involved a small number of image files that contained a corrupt GIF header. Although these files could successfully be displayed in a browser window, attempting to run our analysis on them caused the image processing library in our implementation to fail and abort the detection process. We only encountered three such images over the course of our experiments. After manual analysis of the files, we concluded that the images were likely created by a buggy image editor. Still, this observation demonstrates that purposely injecting errors inside image files could be used by trick banner creators as an evasion technique against automated analysis by TRUECLICK and similar tools, and highlights the importance of building an implementation with analysis routines robust against errors in image file headers.

2.9 OUTCOME

In this work, we have highlighted the problem of trick banners that masquerade as benign links to download files or play videos by emulating the visual characteristics of the correct links. We have defined a number of visual features that distinguish trick banners from genuine links, including image size, color, placement on the enclosing web page, whether they contain animation effects, and whether they consistently appear with the same visual properties on consecutive loads of the web page. Using these features, we have built TrueClick, which uses image processing and machine learning to automatically detect trick banners. Our approach operates purely over visual features and, after an initial training, requires no further manual effort, for instance, to compile blacklists as current approaches do.

We have evaluated our system over a data set of manually labeled trick banners and benign image links. Our experiments with collected data showed that our classifier achieves a 96.97% true positive rate given a false positive rate of 3.03%, showing that TrueClick can correctly detect the majority of the trick banners on file sharing websites with a reasonable low false positive rate. We tested our implementation of TrueClick on 40 users, and found that TrueClick resulted in a 3.55 factor improvement in correct link selection.

We also demonstrate that TrueClick serves as an effective and useful complement to existing approaches for identifying trick banners such as Adblock Plus. We conclude that TrueClick successfully as-

sists even technically-sophisticated users in correctly selecting benign image links despite the presence of malicious trick banners.

3

SPEARPHISHING E-MAIL DETECTION

3.1 OVERVIEW

Spam and phishing messages frequently contain executable email attachments or links to malware that the victim should run. However, technical advances in spam and phishing protection significantly reduce the reach and success of large-scale attacks. To increase success rates, especially with high-value targets, adversaries have switched their strategies towards so-called spearphishing attacks.

In a spearphishing attack, the adversary leverages publicly available information about his victim to craft email messages that are custom-tailored to the victim and thus appear legitimate. This characteristic of custom-tailoring sets spearphishing attacks apart from regular phishing attacks, which commonly can be easily identified due to poor grammar and other obvious tell-tales [50]. haling attacks are specific variants of spearphishing where the attacker poses as his victim's superior. Spearphishing and whaling attacks have advanced to a major attack vector for advanced persistent threatsAdvanced Persistent Threat (APT)s as usable technical defenses that allow potential victims to identify such attacks are scarce and limited in applicability.

Existing systems, such as IdentityMailer [107], identify spearphishing attacks before the offending email message reaches its intended victim. This is achieved by monitoring emails sent by legitimate, yet compromised, accounts within the same organizational entity (e.g., company). To this end, IdentityMailer observes the sending behavior of email accounts with respect to writing habits (e.g., character distributions), composition habits (e.g., the time when emails are commonly sent), and interaction habits (e.g., the list of recipients commonly included in outgoing email messages). Because IdentityMailer targets a single organizational unit, it can capture all these characteristics conveniently at a company's Simple Mail Transfer Protocol (SMTP) server.

Unfortunately, this reliance on observing all email traffic originating from a given email account implies that such defenses can only be effective within a given company. Recipients at other companies or regular email users cannot be protected by systems such as IdentityMailer. Thus, while potentially very useful, IdentityMailer and similar approaches only afford full protection if they are comprehensively deployed. As benefits from partial deployments are limited, incentives for initial deployments are limited too. This circular dependency implies that a comprehensive deployment of techniques such as IdentityMailer seems unlikely.

Instead of relying exclusively on sender-based deployment and cooperation, we present our approach –EmailProfiler – which identifies potential spearphishing attacks on the email recipient's side of the communication.Such a deployment has the immediate advan-

tage of protecting the user from spearphishing attacks even in the absence of other cooperating entities in the ecosystem. Thus, a gradual deployment will benefit users as soon as they adopt such a protection scheme. Furthermore, EMAILPROFILER also supports cooperating email senders which allows for increased detection accuracy.

EMAILPROFILER identifies spearphishing attacks on the recipient's side by building a behavioral profile for each email sender. To this end, EMAILPROFILER extracts a set of 23 features from each email. While previous work relied heavily on meta-information accessible from email headers, EMAILPROFILER also leverages 199 stylometric features inspired by the field of natural language processing. The observed feature values are then aggregated by the sender into a behavioral profile that characterizes the behavior of the corresponding author. Incoming email messages undergo the same feature extraction process, but instead of integrating the feature values with the behavioral profile, the values are checked for consistency with the existing profile. If the difference of the newly extracted features with the behavioral profile is above the detection threshold, EMAILPROFILER will flag the incoming message as a potential spearphishing email.

Of course, bootstrapping is a significant challenge for systems such as EMAILPROFILER. To ameliorate this problem, EMAILPROFILER provides two mechanisms that ease the bootstrapping problem. First, users who wish to adopt EMAILPROFILER can train behavioral profiles based on previous correspondence with a given author. A second mechanism that EMAILPROFILER supports is the sharing of behavioral profiles via a privacy-preserving trusted entity. This approach

allows EMAILPROFILER to query behavioral profiles of participating email users. This enables EMAILPROFILER to precisely classify incoming messages from authors with whom the receiving user did not have previous correspondence.

To summarize, this work features the following contributions:

- We propose and demonstrate that behavioral features drawn from email header information and stylometric properties of the email body text allow for a precise characterization of the email's author.
- We leverage this insight to implement EMAILPROFILER, an anti-spearphishing technique that accurately identifies spearphishing emails on the email-recipient's side of the communication.
- We design a privacy-preserving trusted infrastructure that allows users who opt into using this component to aid in overcoming the bootstrapping challenge that EMAILPROFILER and other approaches face.
- We evaluate EMAILPROFILER based on a real-world contemporary dataset obtained from participating volunteers at our research institution. During this evaluation EMAILPROFILER verifies authors with accuracy rates between 67% and 100%.

3.2 DESIGN

The main goal of EMAILPROFILER is to identify whether a received email originates from the author claimed in the email's metadata. To

this end, EmailProfiler creates behavioral profiles for each sender and compares new incoming emails against these profiles. The behavioral profiles themselves consist of a combination of syntactic and stylometric features 3.2.1. EmailProfiler supports two modes of operation: First, evaluating incoming emails based on receiver-trained profiles; second, generating profiles at the sender and making the profile available for querying at a trusted server. The feature sets used for these two variants slightly differ, as received emails contain additional information (e.g., header information pertaining to intermediary email servers) over emails that are about to be sent.

3.2.1 *Feature Set and Extraction*

Natural language processing techniques have identified a lower-bound threshold of 500 words to precisely identify the author of a given text[63]. Unfortunately, email messages frequently consist of less than 500 words. Fortunately, email messages provide ample opportunity beyond the email text itself to draw identifying information from. In particular, EmailProfiler leverages the structured information contained in email headers in combination with the information in the email's body text.

EmailProfiler analyzes the email body for three categories of features: lexical, syntactic, and structural features [1].

1. Lexical features:Lexical features comprise the total number of words, characters per word, characters in the whole text, char-

acters per line, lines, sentences, single character frequencies in the text, and character frequencies that are used to end a sentence.

2. Syntactic features: Syntactic features include part-of-speech tags as defined and supported by the Stanford CoreNLP system [112]. These features include, for example, the number of adjectives, adverbs, coordinating conjunctions, and past participle verbs.

3. Structural features: Structural features consist of personal identifiers of the author, such as their signature, signature extras (contact information such as address, phone number, etc.), farewell, greeting, and sentence beginning and ending types. There are two features that consider the beginnings of sentences. The first feature is the number of sentences that start with an uppercase letter. The second feature gives the number of sentences that start with a lowercase letter. There are four features that characterize the sentence ending habits of an author. The first value corresponds to the number of sentences where an author uses spaces to delimit sentences. The second value indicates the number of sentences that the author terminates with a dot. The third value counts the number of sentences where the author uses spaces after ending a sentence. The fourth and final feature counts the number of sentence-ending punctuation characters except dots.

EMAILPROFILER also leverages information contained in the email headers. More precisely, the following header fields have corresponding features in the behavioral profiles: return-path, x-mailman-version, x-originating-hostname, x-originating-ip, x-spam-flag, x-virus-scanned, and carbon copy. Most of these features are defined only for received emails. For example, a x-spam-flag header is commonly injected by a mail server acting as a spam filter. Such filtering capabilities are most frequently employed at the receiving mail server and, thus, an outgoing email will lack such header information. EMAILPROFILER draws additional information from the timestamps recorded in the email headers. However, because the precise time is likely not significant to identify an author, we broadly classified the sending time into four categories: midnight, morning, afternoon, or night.

In total, the behavioral profiles extracted by EMAILPROFILER consist of 222 features. Most features values are captured as the normalized number of occurrences. However, features that contain words or phrases are not adequately captured by this representation. Instead, the feature value for such features is a binary value with 0 indicating that the corresponding feature has not been observed previously for a given author and 1 representing the case that the same value was already observed for the author.

3.2.2 *Training Profilers*

In the previous section we described the features EMAILPROFILER uses to characterize individual emails. Here we describe how the individ-

ual features are aggregated into a behavioral profile of the author of an email message. As stated previously, EMAILPROFILER has two different modes of extracting behavioral profiles. The *inbox profiler* will generate behavioral profiles based on the email messages previously received by the user of EMAILPROFILER. The *sentbox profiler*, however, is an optional component that users of EMAILPROFILER can use to generate behavioral profiles of their own email sending behavior which can then be shared on a trusted server. Behavioral profiles shared in that way can be leveraged by EMAILPROFILER to evaluate the veracity of incoming emails for senders with whom the recipient did not have sufficient prior communications to exercise the *inbox profiler.*

If the author of the email is a frequent contact of the receiver and has sent a sufficient number ($\geq$S)[1] of emails to the receiver, that author is labeled a *recognized* author. A recognized author sent sufficiently many email samples to be identified and distinguished from other authors in the receiver's inbox. An *unrecognized* author has not previously sent any emails to the receiver or the number of the emails is not enough to create a training set to identify the author. To assess whether an email claiming to originate from a recognized author is likely spearphishing, the user will compare the incoming message against the profile generated by the inbox profiler. Emails from unrecognized authors can be checked with the help of a trusted server that contains profiles generated from the sentbox profiler.

1. *Inbox Profiler*: For a recognized author, the receiver can compare an incoming email against the behavioral profile extracted for

1 S is a tunable parameter of EMAILPROFILER, and in our experiments we empirically determined that a value of S = 50 performs well.

the sender. To establish this profile, EMAILPROFILER first aggregates all emails sent by the same sender. Subsequently, the features (as discussed in Section 3.2.1) are extracted, and finally these features are used to train a classifier through a Support Vector Machine (SVM). The classification task at hand is to detect whether an email is sent by the author who is listed as the sender. As the SVM performs best if it is provided with roughly equal amounts of positively and negatively labeled samples, we augment the list of emails sent by the same author (positive label) with the equal number of emails drawn at random from other authors (negative label). EMAILPROFILER then uses the SVM to determine the weights that indicate how characteristic each feature is for the style of the recognized author for whom the behavioral profile is built. These weights will then be used as coefficients in the decision function that separates legitimate from potential spearphishing emails.

2. *Sentbox Profiler*: As EMAILPROFILER does not generate behavioral profiles for unrecognized authors, it cannot assess the veracity of the sender's identity with the same mechanism for emails originating from unrecognized authors. Instead Section 3.3 ilustrates how a recipient can query a trusted server to check the features extracted from a received email against a behavioral profile of the sender. Of course, this method is only applicable if the sender chose to make her behavioral profile available to the server.

An email sender who chooses to make her profile available to the trusted server will first run the *sentbox profiler*. This component operates quite similarly to the *inbox profiler*. However, instead of generating a behavioral profile for all senders in a recipient's inbox, the sentbox profiler generates a single profile for the cooperating user. To this end, EMAILPROFILER aggregates all sent emails of the user and extracts the features as described above. Subsequently, the same SVM classifier is trained based on the observed feature vectors and the resulting feature weights are combined into the behavioral profile of the user. In a final step, this behavioral profile is shared with the server to ease the challenge of bootstrapping.

We should note that the use of the sentbox profiler and the trusted server is only one way of overcoming the bootstrapping challenge. As the inbox profiler can start generating behavioral profiles based on only S messages from a given sender, the value of S directly affects the window of opportunity for potential attackers. That is, a spearphishing attack only circumvents EMAILPROFILER if it is one of the first S messages from a given sender. Attackers commonly try to leverage existing trust relationships, and spoofing sender email addresses and identities is a convenient way to misuse this trust. Thus, confining attackers to sending their spearphishing messages within the first S messages significantly reduces the exposure of potential victims to such attacks.

3.3 SENTBOX MODEL

As described in the previous section, one deployment model supported by EMAILPROFILER is for users to publish authorship profiles trained on their own sentbox to a trusted third party. This model is primarily intended to address situations where recipients do not have enough emails from a given author to train an authorship profile on the receiving side (i.e., unrecognized users). In the following, we elaborate upon the details of this model.

The sentbox model is comprised of three steps:

1. Users train and upload their sentbox profile
2. Unrecognized user sends email
3. Receiver validates authorship against the sender's sentbox profile

In the first step, users who participate in this model train an authorship profile using their own sentbox. This profile is then uploaded to a trusted third party server, as shown in Figure 3.1. In the second step, a user receives an email from an unrecognized user – i.e., a user for which no receiving side profile exists or can be built due to insufficient training examples (Figure 3.2). This triggers the third step: validation of authorship for the received email by querying the sender's sentbox profile at the trusted third party, shown in Figure 3.3.

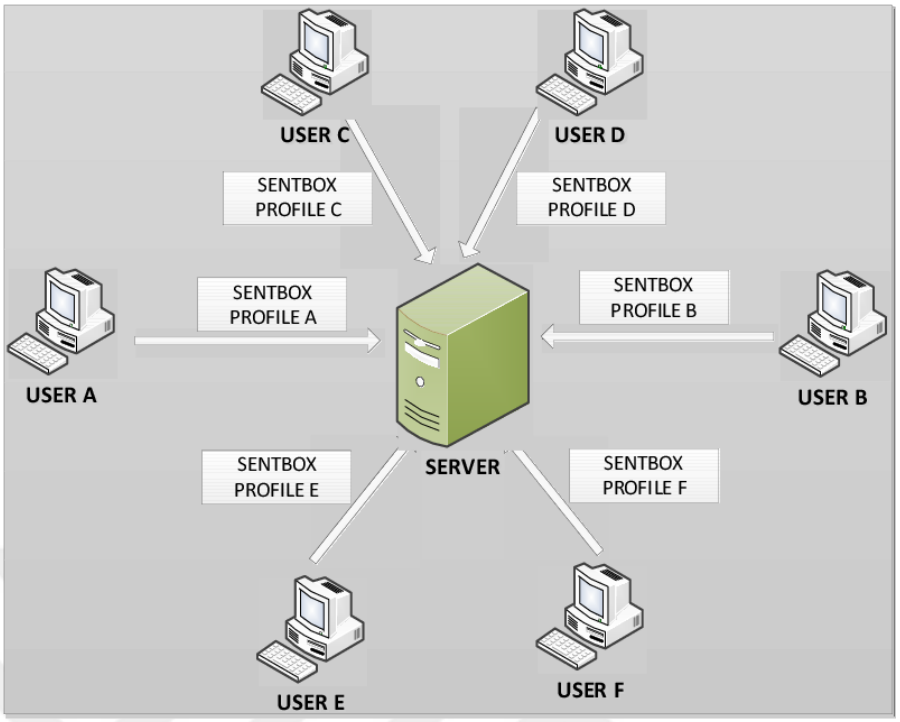


Figure 3.1: Users upload trained sentbox profiles to the trusted server.

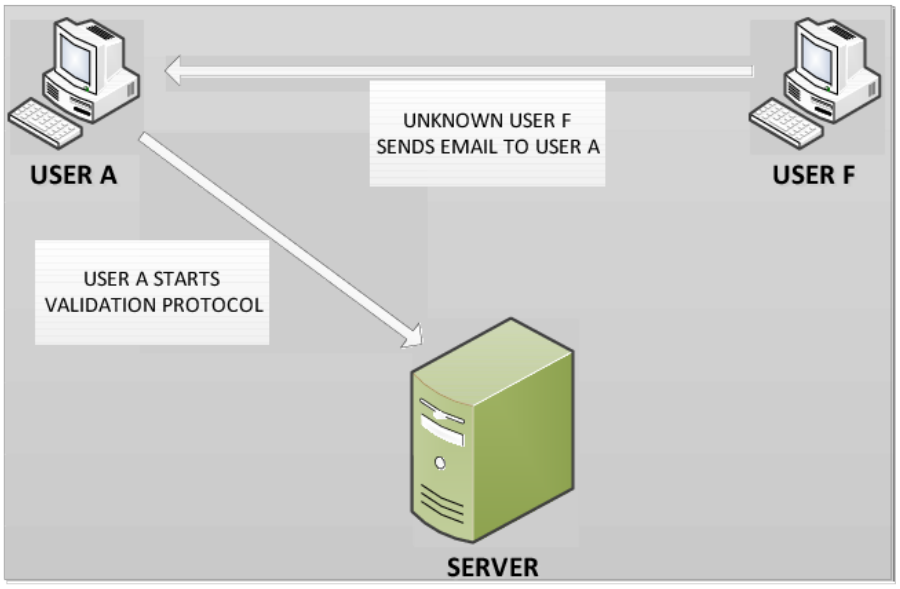


Figure 3.2: Unknown user F send email to user A, triggering the valudation protocol.

Table 3.1: Notation and Definitions

NOTATION	DEFINITION
US	Public key of the server
E_{US}	Encryption of a message with the server's public key
Add_A	E-mail address of user A
Cer_A	Certicifate of user A
N	Nonce
TS	Timestamp
RS	Private key of the server
D_{RS}	Decryption of the message with the server's private key
UA	Public key of user A
E_{UC}	Encryption of a message with user A's public key
S	Secret that will be used to generate a session key
$f(N)$	A function that takes a nonce N and returns a value N that is based on N
N'	Result of $f(N)$
RA	Private key of user A
D_{RA}	Decryption of a message with user A's private key
$g(S)$	A function that takes secret S and returns another value that will be used as session key $K\ S$
KS	Session key
E_{KS}	Encryption of a message with a session key
$Prof_F$	Profile of user F
V_F	Feature vector of user F's email
D_{KS}	Decryption of a message with the session key
Res	Accept or reject result from a profile comparison

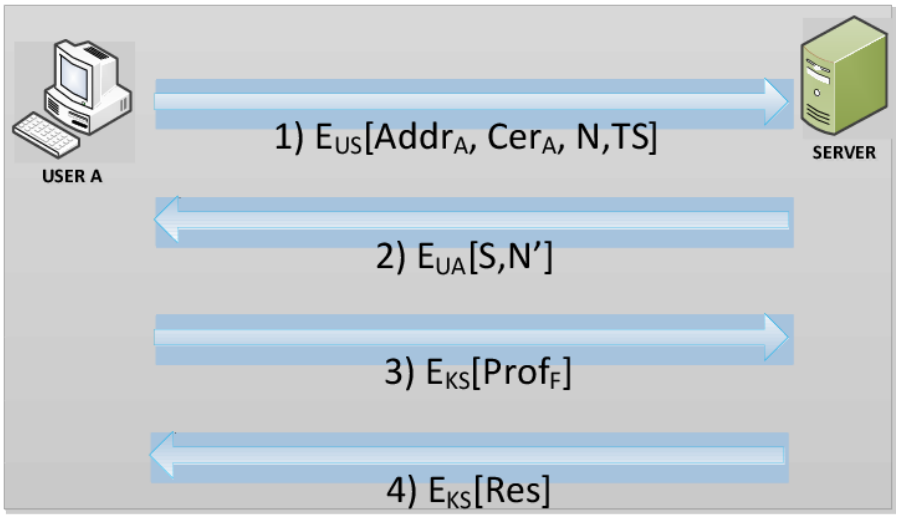


Figure 3.3: Validation protocol

In order to securely communicate with the trusted third party, a communication protocol was devised for the first and third steps. A summary of the notation and corresponding definitions is shown in Table 3.1

User A starts the validation protocol with the first message. This is encrypted with the server's public key US. The message contains the email address of user A Add_A, a certificate Cer_A belonging to A that contains the public key of A UA, a nonce N, and a timestamp TS. N is used to prevent replay attacks, whereas TS is used for freshness. As soon as the server receives Msg_1, it decrypts it as D_{RS} $\{Msg_1\}$ with its own private key RS. The server checks if this message is fresh using its timestamp. Then, the server generates the value $N' = f(N)$, where $f(N)$ is a shared function that is known by the server and user A. Nonce N' will be used to prove to user A that the next message actually originates from the server, because N can only be learned by decrypting the message with the server's private key. Therefore, N' can only be generated by the server and user A. The server also

generates secret S which is used to generate a session key *KS* . *KS* is produced by function *g(S)* which is a pre-agreed function.

In Msg_2, the server sends *S* and *N'* by encrypting with user A's public key *UA*. When user A receives Msg_2, he first decrypts it as D_{RA} $\{Msg_2\}$ with his own private key *RA*. Then, user A checks whether *N'* agrees with his local computation of *f(N)*. If they are equal, he continues the protocol, otherwise he terminates it. Afterwards, he computes *KS = g(S)* in order to decrypt the following messages. He sends Msg_3, including a feature vector of user F's email V_F by encrypting with *KS*. When the server receives this message, it decrypts it by using *KS* and compares it with the sentbox profile of user F that was uploaded in step 1. If the query returns positive, it sends a validation message as a result, otherwise it sends a reject message. The resulting message Msg_4 that contains *Res* is encrypted with *KS*. When user A receives it, he decrypts it using *KS* and learns the result.

When users upload their sentbox profiles, a similar validation protocol is applied on uploading sentbox profile procedure with two small differences in Msg_3 and Msg_4. $Prof_F$ in Msg_3 is the sentbox profile of user F in the upload procedure, whereas in the validation protocol it is the feature vector V_F of an email received from user F. Also, in Msg_4 , *Res* contains an accept or reject decision in the validation protocol, whereas in the upload procedure it contains a confirmation that the server has received the complete profile of user F.

3.4 EVALUATION

In this section, we evaluate the efficacy of EmailProfiler in generating behavioral profiles that capture user behavior. Furthermore, we evaluate how well EmailProfiler generated behavioral profiles characterize the email writing and sending behavior of 20 volunteer users at our academic institution. Besides this empirical evaluation, we also analyze the robustness of the generated behavioral profiles from a theoretical standpoint against privacy attacks. Finally, we also discuss assumptions and limitations that underlie the current prototype implementation of EmailProfiler.

3.4.1 *User Data Collection*

Previous studies on email security and spam frequently used publicly accessible datasets such as Enron and Symantec for their evaluation. However, spearphishing is a precisely targeted and relatively new phenomenon that we believe is insufficiently reflected in these datasets. Thus, we chose to evaluate EmailProfiler on a contemporary dataset of emails provided by volunteers at our academic institution. To this end, we were able to solicit the help of 20 connected users on our campus. All the participants are PhD students and professors in the field of computer science and bioinformatics.

Extrapolating from our own mindset, we assumed that none of the participants would want to give us direct access to their email

archives. Thus, we developed the profilers as stand-alone components that each volunteer could easily run for themselves. Collecting the data to evaluate EMAILPROFILER in this way had two significant advantages. First, users could inspect the data transmitted back for the evaluation and be convinced that no privacy sensitive information was contained in the transmitted results. Second, this forced us to engineer the profilers robustly to make them integrate smoothly with the email habits of the participating users.

To avoid privacy leaks in the output, we replaced email addresses with hashed versions thereof. Furthermore, all the participants were given explanations on how to use the profilers and what information is contained in the output data. We specifically asked the participants to run the profilers on their institution-associated email data. This allowed us to evaluate the functionality of the trusted server that contains the profiles extracted by the sentbox profiler. In our experiments, we set the threshold S at a value of 50. Furthermore, volunteers were selected in a way that each person had at least 50 emails from another person that participated in the experiment. As a result, each participating user has at least one recognized user associated with the user's profile in their inbox.

We acquired both inbox and sentbox generated profiles from participants. This way even if a user did not have enough sample emails from a participant but had at least one email, we were able to compare that received email with the sender's own profile which was obtained from the sender. This gave us the opportunity to design multiple testson the collected data.

Table 3.2: Dataset Statistics

EXPERIMENT	CORRECT CLICKS	INCORRECT CLICKS	MEDIAN SCORE
U1	216	1,434	722
U2	41	670	8,915
U3	107	631	629
U4	320	1,168	4,188
U5	156	852	18
U6	84	433	220
U7	343	1,459	282
U8	235	3,127	1,416
U9	745	4,290	1,272
U10	246	3,032	509
U11	892	7,955	2,392
U12	759	6,346	2,395
U13	965	6,493	2,005
U14	311	2,607	709
U15	418	1,603	402
U16	526	3,423	488
U17	1,244	17,303	2,948
U18	846	11,450	2,274
U19	698	8,715	13,791
U20	1,279	16,440	1,094

A detailed breakdown of the data our volunteers processed with the profilers is presented in Table 3.2.

We also computed statistics on the processed messages. For example, many email messages did not contain any or only very short text in their body. This observation is a clear indicator that existing stylistic-only author attribution schemes would likely result in poor accuracy. However, as real-world use of email features such characteristics prominently, a defense system such as EMAILPROFILER must be able to handle such email messages too. The excellent accuracy established in our evaluation (see Section 3.4.2) thus demonstrates that drawing information from the structured email header is beneficial to perform author re-identification based on email messages.

3.4.2 *Experiments*

Based on the feature vectors extracted by the profilers, we trained a one-class SVM classifier as discussed in Section 3.2.

To evaluate the accuracy of the behavioral profiles, we performed 10-fold cross validation for the profiles generated for each recognized author. To this end, we randomly separated the email messages from each recognized author into ten equal-sized non-overlapping subsets. We then used nine subsets or 90% of the messages for training and the remaining 10% for testing. This was repeated 10 times with a different subset used for testing in each iteration. In total, we performed this profile generation 215 times. The worst-performing profile resulted in 67% accuracy, whereas the best performing profile reached 100%

accuracy. Averaging the accuracy of all 215 profiles results in a 93% accuracy value for EMAILPROFILER. As the sentbox profiler generates one profile per participating user, we obtained 20 profiles using this method. Similar to the inbox testing case, each generated profile was tested using 10-fold cross validation.

Additionally, they have been put in tests where each user's whole profile data vs all other authors are used for training and tested against a profile generated with sent profile through using inbox data of a participant. This way, we created the case of a user receiving an email from a unrecognized user (Section 3.3). This case was repeated for each participant vs the rest of the participants. 10-fold cross validation results are between 80% and 100%, in 20 tests. Comparison with inbox data of another participant give the results in between 75% – 100%.

3.4.3 *Theoretical Analysis*

In the following, we perform a theoretical analysis of the robustness of the sentbox model described in Section 3.3 against various attacks.

3.4.3.1 *Denial of Service Attack:*

In our proposed architecture, the trusted server presents a single point of failure. While redundant operation and over-provisioning can reduce the risk of accidental failure, attackers might strive to

Table 3.3: Feature List

FEATURES	TH. RANGE	# OF TH. ANALYSIS	EMP. RANGE	# OF TRIALS FOR EMP ANALYSIS
Email address	String	2^{32}	$[0, 2^{32}]$	2^{32}
Time of day	[0, 3]	4	[0, 3]	4
Letters in subject	Integer	2^{32}	[0, 176]	176
Caps in subject	Integer	2^{32}	[0, 103]	103
Words in subject	Integer	2^{32}	[0, 29]	29
Chars/word in subject	Double	2^{64}	[0, 75]	75
Total words	Integer	2^{32}	[0, 23675]	23675
Chars/word	Double	2^{64}	[1, 217]	216
Chars	Integer	2^{32}	[0, 200259]	200259
Line count	Integer	2^{32}	[0, 8521]	8521
Chars/line	Double	2^{64}	[0, 40088]	40088
Sentences	Integer	2^{32}	[1, 4457]	4456
Caps sentence start	Integer	2^{32}	[0, 987]	987
Small sentence start	Integer	2^{32}	[0, 4433]	4433
Sentence end spaces	Integer	2^{32}	[0, 2701]	2701
Sentence end dot	Integer	2^{32}	[0, 431]	431
Sentence end w/o space	Integer	2^{32}	[0, 4256]	4256
Sentence end punctuation	Integer	2^{32}	[0, 4208]	4208
Chars	Double	2^{64}	[0, 1]	1024 x 2^{32}
Non-chars	Double	2^{64}	[0, 1]	1024 x 2^{32}
POS	Double	2^{64}	[0, 1]	1024 x 2^{32}
BCC	Bool	2	[0, 1]	2
CC	Bool	2	[0, 1]	2
Farewell	Bool	2	[0, 1]	2
Greeting	Bool	2	[0, 1]	2
MIME version	Bool	2	[0, 1]	2
Sender	Bool	2	[0, 1]	2
Signature	Bool	2	[0, 1]	2
Extended signature	Bool	2	[0, 1]	2
X-Mailer	Bool	2	[0, 1]	2
X-Originating-IP	Bool	2	[0, 1]	2

launch denial of service attacks against this part of the infrastructure[2]. In order to prevent accidental brute force attacks, there are some common use cases that need to be considered. An unknown email sender, such as user F in Section 3.3, can send a collective email to multiple receivers. For instance, a professor can send an announcement to all university departments. In this case, he would be an unrecognized user for most of the recipients. The features of this email would be submitted as queries to the server multiple times in short succession. However, this simple case of an unintentional DoS can be mitigated with a caching layer at the server. That is, since the email is identical, the feature vector sent to the server during the validation protocol will be identical and, therefore, the validation result can be cached so long as the user profile itself remains valid.

3.4.3.2 *Profile Reversing:*

If the number of times that a given profile is queried is not limited, a targeted profile reversing attack can be possible. A distributed attacker can query the server from a large set of disparate addresses with different feature vectors for the profile of a targeted user. The attacker can exhaustively search the feature space and thus reverse-engineer the target's profile.

In order to prevent this attack, a request limit L which determines the maximum number of queries for an unrecognized user can be asked is defined. This limit is set to a reasonable default value (e.g.,

2 Note that even if the attacker succeeds in this DoS attack, he is still limited to sending his spearphishing email as one of the first S messages. The client-side inbox profiler builds and maintains profiles for recognized authors independently of the server.

1 query per second) and can be adapted to the email behavior of the user. For instance, for the above-mentioned case of the announcement email, a higher threshold might be advisable. While it is unlikely that a legitimate user consistently sends one email per second, such a limit would effectively prevent the exhaustive search of the feature space as we illustrate in Section 3.4.3.3.

3.4.3.3 *Analysis of Regular and Smart Adversaries:*

In this section, regular and smart attacker cases are considered. The regular attacker is one who knows the features that are used in this model. He can only deduce the parameter types by considering the features. According to these types he determines the number of bits for each feature and applies a brute force profile reversing attack. All of these features, parameter types, and the parameter space in bits are illustrated in Table 3.3. Assuming that all the features are independent, in order to find the number of trials all values in the third column of the table are multiplied. According to this result, a regular attacker can expect to uncover the user's profile in approximately $2^{12,140}$trials. If the attacker spoofs IP addresses and attacks in a distributed manner, then the number of years Y required to search the feature space will be

$$Y = \frac{2^{12,140}}{365L}$$

where L is the maximum allowable number of times an unrecognized user can be queried from the server. Due to the huge feature-value space, the linear dependency on the number of spoofed IP addresses and the query limit set at the server makes this unstructured attack an intractable challenge.

A smart attacker is one who knows the most popular range of values for each feature. He does not try all possibilities, but instead will only try the most probable cases which are deduced from empirical results. By multiplying all values in the fourth column of the table which represent conservative estimates of the ranges of likely values for each feature, the total number of trials needed to obtain a profile of a user is obtained. If the attacker spoofs IP addresses and attacks in a distributed manner, then the number of years to break the system will be

$$Y = \frac{9.673 \times 10^{120}}{365L} \approx \frac{2^{401}}{365L}$$

While this is a significant reduction in difficult from a naïve brute-force feature space enumeration, given $L = 100$ it would nevertheless take an insurmountable number of years to acquire the correct profile.

3.5 RELATED WORK

EMAILPROFILER makes use of email metadata and stylometric features of email content in order to identify spearphishing emails. In

the following, we discuss significant prior work in authorship detection using stylometric features, natural language processing, and spearphishing detection.

3.5.1 *Stylometric Authorship Identification*

Spearphishing and whaling emails use identity hiding and impersonation attacks. Since stylometry assists revealing the authorship it is reasonable to use stylometry while solving the identity problem of an email.

One of the main studies on authorship identification is Abbasi's work Writeprints[1]. In Writeprints, the authors used online texts from different sources such as Enron emails, eBay comments, Java forums, and CyberWatch chats as their datasets. They extracted a comprehensive feature set and experimented with their technique on the datasets for identification and similarity detections. They showed that when the number of authors in a dataset increases, the accuracy of the Support Vector Machine (SVM) for authorship identification decreases.

Afroz has applied stylometry to the problem of authorship identification in the area of cybercrime in their works. In one approach, they studied obfuscated writing where an author imitates someone else [6]. In another study, they studied doppleganger accounts in underground forums that included English, German and Russian texts [7]. They found that a hybrid method that combines stylometry with forum specific features were more successful in finding doppelgangers.

McDonald tested whether their author identification framework, called Anonymouth [76], could handle manually anonymized text. Where Anonymouth focuses on privacy of the author, Narayanan studied author identification using a large set of forum posts[83]. They showed a correlation between the number of features and the accuracy of different classification methods. Another common point of all these authorship identification works is that they constructed their corpus based on word counts greater than 500 or character counts greater than 7500. Ledger used single character frequencies to identify the author of acts in Two Noble Kinsmen between Shakespeare and Fletcher[63]. They stated that text samples of 500 words or fewer will not allow for accurate authorship identification.

Email authorship problems are investigated as well. Given the shortness of email documents, de Vel limited the topics in the emails to movies, food and travel and classified authors based on the structural characteristics and linguistic patterns of emails [113]. Similarly, in a feasibility study [40] raw keystrokes and emails have been used for author identification with a limited feature set. Nizamani used cluster based classification to identify authors in emails [85].

In a similar manner to Afroz's work, Iqbal studied the feasibility of forensic investigation of cyber crimes using stylometric features of emails [57]. In their study, they used most of the stylometric features listed in Writeprints [1] on the Enron email dataset. We used a different feature set in our work that includes a subset of Writeprints' stylometric features and header information features. Aside from the

feature set, the main difference between their study and ours is the data used for testing.

Another email forgery detection method is Lin's work that uses both stylometry and geolocation during email analysis [68]. Their client-side plugin is tested using the Enron corpus. Brocardo also studied authorship verification [14], and proposed a method that uses n-gram analysis to verify the author of short messages.

One of the most recent papers on authorship verification of emails is IdentityMailer[107]. IdentityMailer uses header information of an email as well as stylometric features. More specifically, they extracted reply, forwarded, time of day, day of week, and recipient information from the metadata of the email to verify email authorship. The main difference between our work and IdentityMailer is the deployment model. IdentityMailer is designed to prevent transmission of spearphishing emails from compromised machines before they are forwarded to SMTP servers. In contrast, our profilers check for spearphishing emails after they are received by the user.

3.5.2 *Natural Language Processing*

Utilizing correct classifying methods while using Natural Language Processing (NLP) plays a key role. Classify, But Verify [105] examines this issue. Other NLP work includes NoCrack, which used Natural Language Encoders to crack passwords [18]. Crowston used NLP techniques to acquire qualitative data [23]. Recently, Palka and McKoy [88] combined NLP analysis with email filters. In that work, they

tested the effectiveness of email filters using a fuzzing strategy based on n-gram analysis.

3.5.3 *Spam and Phishing Email Filtering*

Spam and phishing email filtering is pervasively deployed on today's Internet. Spam filters typically use both linguistic and sender reputation features for detection. Moreover, email clients incorporate feedback from users who mark suspicious emails as spam. Spam filter performance and efficiency has been studied and enhanced by many researchers [30, 120]. Over time, the term spam has come to include those that include links that direct users to unwanted, unsolicited domains as well. While studying spam links, Thomas compared the features of email spam with the features of tweet spam [110] and found that email spammers do not overlap with tweet spammers. Some of email spammers bypass filters by embedding spam into images. In his work [117], Wang describes a solution to image-based spam by clustering a new corpus of spam images and comparing them with non-spam image-based emails.

In the same way as spam filtering, linguistic features of an email are important for detecting phishing emails. Some characteristics of phishing emails are typos in the body, lack of knowledge of the receiver's name, asking for a monetary deposit, or asking users to click on a link. Aggarwal used NLP techniques to detect phishing emails [8] using these characteristics. Ramanathan employs multi-stage filtering on phishing emails, which combines NLP techniques with ma-

chine learning [93]. More recent work by Lottre [70] proposes a framework that marks emails as safe or non-safe; the aim of the framework is to educate users and increase their security awareness.

Spearphishing attacks are considered a subset of phishing email attacks; however, popular preventative methods for spam and phishing attacks are not well-suited to detecting spearphishing attacks. The main reason for this is that spearphishing emails are crafted to impersonate someone that the recipient would trust, and are therefore more likely to bypass mass-market spam filters.

Our work uses some of the methods described above for spam and phishing email detection, such as NLP and author identification. However, EmailProfiler uses an extended feature set and different classification methods. For example, the linguistic features EmailProfiler extracts are classified per person instead of into global safe and unsafe categories. Another distinguishing feature of our work is that we use header information to understand the writing habits of email authors.

3.6 OUTCOME

In this work, we proposed EmailProfiler, a new approach to detecting spearphishing attacks using both features extracted from both email metadata and stylometric information. We evaluated this approach using an dataset comprising email from 20 volunteers at our academic institution. The results demonstrate that EmailProfiler 's techniques can reach an average 98% accuracy when verifying email

authorship through inbox and sentbox profiling. We also show that email profiles are robust to reverse-engineering in both a theoretic and empirical analysis. Finally, in contrast to prior work, EMAILPROFILER supports gradual adoption by users, leading to increased utility as a real-world, usable defense against spearphishing attacks.

4

MALICIOUS E-MAIL ATTACHMENT NEUTRALIZATION

4.1 OVERVIEW

Today, e-mails are used heavily for a wide range of activities such as sending out meeting invites, bills, receipts, and news articles. Often, documents are attached to e-mails and the user is required to open this attachment to access the contents of the artifact. Unfortunately, documents and links embedded in e-mails can create a serious attack surface against users. Today, attackers exploit vulnerabilities in software that processes the content from these attachments to infect the targeted machine with malicious code. That is, once the victim opens the delivered attachment, an existing vulnerability (e.g., a use-after-free) can be exploited to execute arbitrary code on the victim's machine.

E-mail based attacks are often highly effective and successful. As a result, such attacks are one of the main ways attackers typically launch targeted attacks against specific victims. For example, it is widely reported that the Democratic National Committee was hacked using such targeted, spearphishing e-mails (e.g., see [15]).

As email-based attacks are very successful in allowing attackers to compromise endpoints and gain an initial foothold for launching further attacks, this raises the question: What makes these attacks so successful in practice? The straightforward answer to this question is that users are typically not qualified to make security decisions regarding attachments they receive, often do not have updated systems, and often end up opening attachments that are highly risky. In fact, it is often difficult for a typical user to assess which attachments are riskier than others. That is, until an attachment has been downloaded and opened, a victim might not be able to easily determine if the attachment is a spearphishing attempt, or a legitimate artifact sent by someone that the victim knows. Hence, to check the contents of an attachment that sounds interesting, a user is typically left with the sole choice of opening the attachment and attempting to read its contents.

Recognizing a malicious e-mail might be difficult even for an expert user. While it is true that some e-mails might have traces of malicious behavior such as a suspicious-looking e-mail sender, poor word choice of the subject, and any other unusual attributes in the e-mail, many malicious e-mails may look very authentic besides the attachment that is malicious. Although training users to spot phishing e-mails is helpful (e.g., [22]), spearphishing e-mails are very challenging to detect for most users. Especially in attacks where the e-mail sender imitates a (possible) genuine user, victims are prone to downloading and opening any attachments.

In spearphishing attacks, the attacker leverages information about the victim to tailor the attack e-mail to improve the chance that the

victim will click on the e-mail attachment and open it. It has been reported that sophisticated targeted attacks (i.e., Advanced Persistent Threats (APTs)) often contain a spearphishing component [102]. Hence, it is clear that mechanisms are needed that can protect users against malicious attachments.

The key insight of our approach is that by converting the e-mail attachment to an image format and attaching this image to the e-mail to allow the user to check the contents of an e-mail, we would be giving users a chance to protect themselves against spearphishing attacks. Hence, users are able to peer into the contents of an attached document (e.g., a malicious PDF file) without having to download, and open it and potentially be compromised. By converting potentially malicious files to a different format (e.g., converting a Word document to a PNG image), we remove the exploit code from the artifact and render it safe. The user can then examine the contents, only interacting with the original attachment after having had a chance to check the authenticity and validity of the contents.

In order to evaluate the usability and effectiveness of our technique, we conducted a user study with 50 participants. Our findings show that our proposed technique is a minimal addition to existing e-mail security systems, and has significant security benefits for users in avoiding malicious attachments.

In summary, this work makes the following contributions:

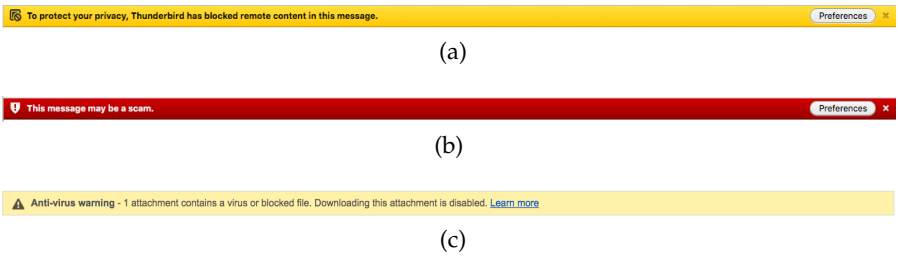
- We present a novel approach for protecting users against malicious e-mail attachments that we call PELLUCIDATTACHMENT. Our proposed technique automatically renders attachments into

safe PNG images, and replaces the original attachment with the generated image. The conversion gives users the chance to distinguish between a benign attachment and a malicious one without having to open the attachment and potentially be compromised.

- We empirically evaluated our approach with 39 real-world malicious attachments. We show that by rendering malicious attachments into PNG images, our system removes the existing exploit code for all of the tested files (10 PDF, 10 Microsoft Excel, 10 Microsoft Word, and 9 PNG files).
- We evaluated the security benefits of our approach with a randomized user study (n=50). Our multi-protocol user study shows that PELLUCIDATTACHMENT is usable and improves user security by helping them avoid exposure to malicious documents.

4.2 BACKGROUND

Many document formats, such as MS Word, MS Excel, PDF, and PNG documents, can be crafted to be malicious. Once exploit code is inserted into the document, malicious activity can be triggered, often just by opening and viewing the document. Document viewers or editors may be vulnerable to memory corruption exploits due to unpatched or zero-day security flaws, while some document formats such as PDF can also potentially contain malicious scripts.



(a)

(b)

(c)

Figure 4.1: E-mail warning banners.

4.2.1 *Document Vulnerabilities and Exploitation*

4.2.1.1 *Attacks via Microsoft Office Files*

An attacker may be able to craft a malicious Microsoft Office file that runs arbitrary code when the document is opened. Some unpatched versions of Microsoft Office have memory corruption, elevation of privilege, denial of service, and similar vulnerabilities. A recent example is CVE-2016-7193[27], where RTF file content is not handled properly by the software, leading to the execution of attacker-supplied code.

Macros are another popular method that attackers use to launch attacks. Macros are used to simplify common tasks by automating them in Microsoft Office. However, this legitimate functionality may be used to deliver malware as well. Unfortunately, macros have been abused by attackers so often[78] that they have been disabled in recent versions of Microsoft Office. Nevertheless, an attack may be successful if the victim chooses to run the macro (e.g., through social engineering).

4.2.1.2 *Attacks via PDF*

A vulnerability in a PDF reader may cause arbitrary code to be executed on the targeted host. The complex structure of PDF files has historically provided attackers many opportunities to exploit memory corruption errors. PDF documents may also be able to run unauthorized JavaScript, ActionScript, and other types of malicious scripting code.

An example of a recent Adobe PDF vulnerability[25] allows remote attackers to execute arbitrary code on vulnerable installations of Adobe Acrobat Reader DC[4].

4.2.1.3 *Attacks via PNG*

Imagemagick[54] is a free software package that allows developers to programmatically manipulate images. As a result of its advanced capabilities, attackers may be able to craft PNG images that are malicious. For example, a recently exposed vulnerability was published on the Common Vulnerabilities and Exposures Database where an attacker can execute arbitrary code via shell meta-characters in a crafted image[26].

Moreover, the PNG reference library libpng[66] is also vulnerable to various memory corruption attacks. CVE-2016-10087[24] is an example of the libpng vulnerability, where the attacker takes advantage of a null dereference bug in earlier versions of libpng.

4.2.2 *Defense Against Malicious Files*

The best defense mechanism against malicious e-mail attachments would be to prevent them from being downloaded to the victim's system. However, this would require that benign documents can reliably be distinguished from malicious ones. Static or dynamic analysis techniques may be used to perform this detection.

Unfortunately, as explained in Section4.7, static and dynamic detection techniques have their limitations. As a result, once a malicious document falls through the cracks, the user needs to make a decision. In fact, in most of the attacks listed in the previous section, the attacker counts on the victim's input such as downloading the malicious e-mail attachment, opening a PDF file that launches a remote attack via JavaScript code, activating the macros of a Microsoft Office document, and viewing a PNG image that opens a backdoor on the compromised system.

Previous research has determined that warning users frequently about the results of their actions may be effective[9] in detecting some attacks. In the case of e-mail security warnings, mail user agents (Mail User Agent (MUA)) (i.e., e-mail clients) have improved over the years. Two concrete examples of MUA warnings would be the warning banners that Thunderbird and GMail present to users (Figure4.1). Such banners inform users whether the content of the e-mail is suspicious. In Figure 4.1a, a security alert banner is shown to Thunderbird users when there is an image or stylesheet embedded in an e-mail message. Similarly, in Figure 4.1b another banner is shown to notify users about

a suspicious e-mail which might be a potential phishing attempt. Figure 4.1c shows a security banner that blocks users from downloading a possibly malicious file. However, these banners are unfortunately frequently ignored by users due to the underlying detection algorithms' lack of precision (i.e., too large a false positive rate)[33, 80].

In this work, rather than trying to detect malicious files that are spread through e-mail, we adapt a generic defense approach that converts the potentially malicious document to a harmless image format. By doing so, we automatically remove and prevent the exploit.

4.3 SYSTEM DESIGN

The key insight of our work is that e-mail users have insufficient information to distinguish potentially malicious attachments from benign attachments. PELLUCIDATTACHMENT narrows this information gap by allowing the user to safely peek into the contents of each attachment without first downloading and opening it. PELLUCIDATTACHMENT provides this capability by modifying e-mails as they are received at the recipient's Mail Transfer Agent (MTA). In particular, PELLUCIDATTACHMENT modifies how the MTA processes incoming e-mails along two dimensions (Figure 4.2). First, attachments of incoming e-mails are replaced by renderings thereof. Second, the system must provide a mechanism to access the original unmodified attachments if the user so chooses.

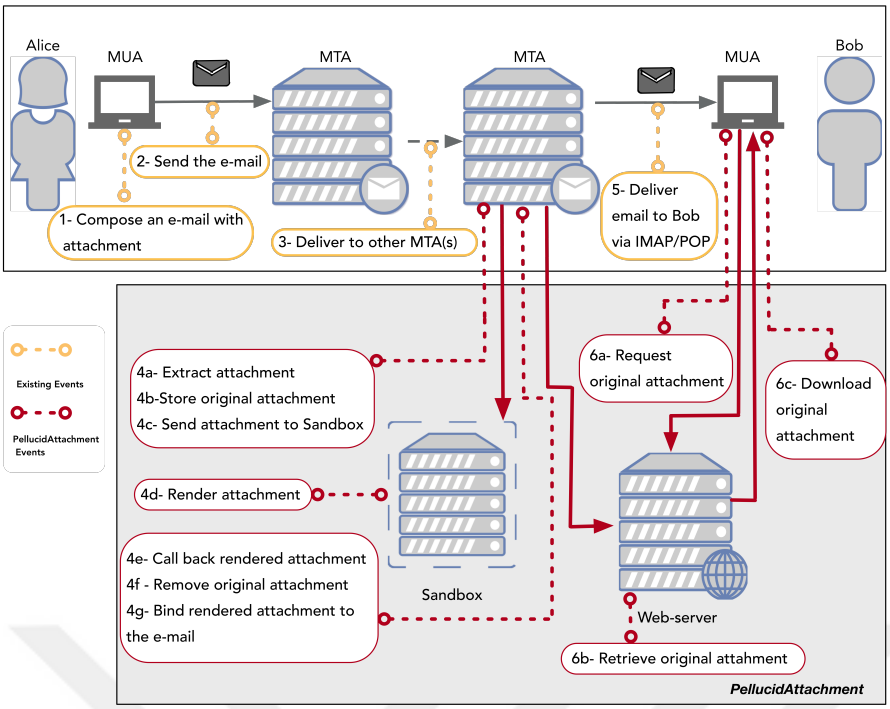


Figure 4.2: Overview of PELLUCIDATTACHMENT System

4.3.1 *Replacing Attachments*

To replace incoming e-mail attachments, PELLUCIDATTACHMENT follows a sequence of three consecutive steps for each attachment.

4.3.1.1 *Extract and preserve original attachment*

Upon receipt of a new e-mail, PELLUCIDATTACHMENT parses the e-mail content and extracts all attachments. Each attachment is then used in two ways. First, PELLUCIDATTACHMENT persists the attachment in case the user needs access to the unmodified attachment later on (see Section 4.3.2). Second, each attachment is subjected to a conversion process where its contents are rendered into an image.

4.3.1.2 *Render attachment into an image*

PELLUCIDATTACHMENT converts each attachment into a visual representation (i.e., an image) of its content. For example, the contents of a PDF document will be rendered into an image file that visually carries the same information as the *original* file itself. Note that the input to this rendering process are the potentially malicious attachments sent by the attacker. Thus, this conversion step warrants additional security precautions. To counter the situation where a malicious attachment attacks and exploits PELLUCIDATTACHMENT's rendering infrastructure, all conversion is performed in a sandboxed virtual machine that is restored to a known good state for each attachment. Furthermore, PELLUCIDATTACHMENT provides a firewall around the sandbox to prevent any communication beyond the MTA and the sandbox itself. Thus, for each attachment, PELLUCIDATTACHMENT requests the sandbox to convert the attachment into an image, and then retrieves the image from the sandbox.

4.3.1.3 *Replace original attachment with its image*

The final step carried out by the MTA replaces the original attachments with the rendered images thereof. In addition to replacing the attachments, PELLUCIDATTACHMENT also includes a link at the bottom of each e-mail that allows the user to access the original unmodified attachment if needed.

4.3.2 *Accessing Original Attachments*

Sometimes, the user must access the original, unmodified attachment that was included in the e-mail. This can become necessary if the user, for example, is required to fill a form or is expected to make modifications to a document. A user can gain access to the unmodified attachments by following the link at the bottom of the e-mail. However, instead of providing direct access to the unmodified and potentially dangerous attachments, PELLUCIDATTACHMENT confronts the user with a security warning analogous to Transport Layer Security (TLS) certificate warnings used by all major web browsers. Thus, before gaining access to the original attachments, users have to acknowledge their awareness of potential negative impacts.

4.4 THREAT MODEL

As PELLUCIDATTACHMENT tries to protect recipients from e-mail messages that include malicious attachments, we assume the following threat model. First, we assume that the attacker has a reliable way of delivering malicious e-mails to the victim. That is, the attacker has the capability to circumvent all of today's frequently deployed defensive measures, such as spam filtering, anti-virus scanning of e-mail attachments, or statistical models to detect malicious e-mails. Furthermore, the attacker knows about the software installed on the victim's computer, and additionally knows of at least one arbitrary code execution

vulnerability in one of the installed software packages. In addition to the vulnerability, the attacker has the capability to create exploits that target that vulnerability and include this exploit in a file of the format that will be processed by the vulnerable software if the victim opens the file.

A concrete and realistic example is an attacker with knowledge of a vulnerable version of Adobe Reader installed on the victim's machine, and a readily available exploit in the form of a malicious PDF file that will grant the attacker arbitrary code execution capabilities if it were to be opened with the vulnerable software.

Beyond these capabilities, the attacker is also assumed to be aware of PELLUCIDATTACHMENT and its use by the victim, and he might want to attack PELLUCIDATTACHMENT itself instead of the victim user. Note that PELLUCIDATTACHMENT's main goal is to provide additional information about attachment content without exposing users to the potential threats therein. However, PELLUCIDATTACHMENT must and does provide the user with access to the original attachments if the user so chooses. Thus, while we assume that the attacker has the various technical capabilities outlined above, we also assume that the attacker does not have the capability to lure the user into downloading and opening the *original* malicious attachment. This assumption is realistic when considering an attacker who indiscriminately attacks his victims. We argue that creating e-mails that convince users to download an attachment *despite the provided preview*, acknowledge the security warnings, and then open the resulting file, requires significant

and more importantly individualized effort, and thus significantly raises the bar for the attacker.

While operating in the above-stated threat model, PELLUCIDATTACHMENT tries to impose as few restrictions on users as possible and thus is deployed exclusively at the MTA of the recipient. This implies that users can continue using the MUA's that they are most accustomed to without any changes to the client-side software. Furthermore, as PELLUCIDATTACHMENT operates in conjunction with the MTA, it is compatible with an enterprise setting where a company maintains its own e-mail system. Deployed in this way, PELLUCIDATTACHMENT can seamlessly afford its protections to any user throughout the enterprise.

4.5 IMPLEMENTATION PLANS

We implemented a prototype of PELLUCIDATTACHMENT on Ubuntu Linux running Postfix[90] as the MTA. PELLUCIDATTACHMENT introduces three additional components to an existing MTA.

- A content filter
- The rendering sandbox
- A facility to provide access to unmodified attachments

4.5.1 *Content Filter*

Postfix uses the term *content filter* for any software component that inspects or modifies e-mail data (including both headers and payload). To simplify the process of creating content filters, Postfix provides a standardized interface that PELLUCIDATTACHMENT leverages. The MTA is configured to trigger the PELLUCIDATTACHMENT filter which in turn parses the body of each e-mail.

The content filter itself first extracts all attachments and preserves them should the user require access later on 4.7. Subsequently, the attachments are sent to the rendering sandbox that converts each attachment into a visual representation thereof.

Finally, the content filter replaces the original attachments with the renderings obtained from the sandbox, and attaches links to the bottom of the e-mail to allow the user to fetch the unmodified attachments.

To extract attachments from an e-mail, the content filter parses the information contained in the message. Within a Multipurpose Internet Mail Extensions (MIME) e-mail message, individual attachments are described through a content-disposition header field according to the specification in RFC2183[1]. For example, to transmit UTF-8 formatted data through the ASCII SMTP protocol, the content needs to be identified and encoded appropriately. Figure 4.3 shows an example where the UTF-8 formatted text of an e-mail is labeled with a corresponding content type. Thus, PELLUCIDATTACHMENT iterates over

1 https://tools.ietf.org/html/rfc2183

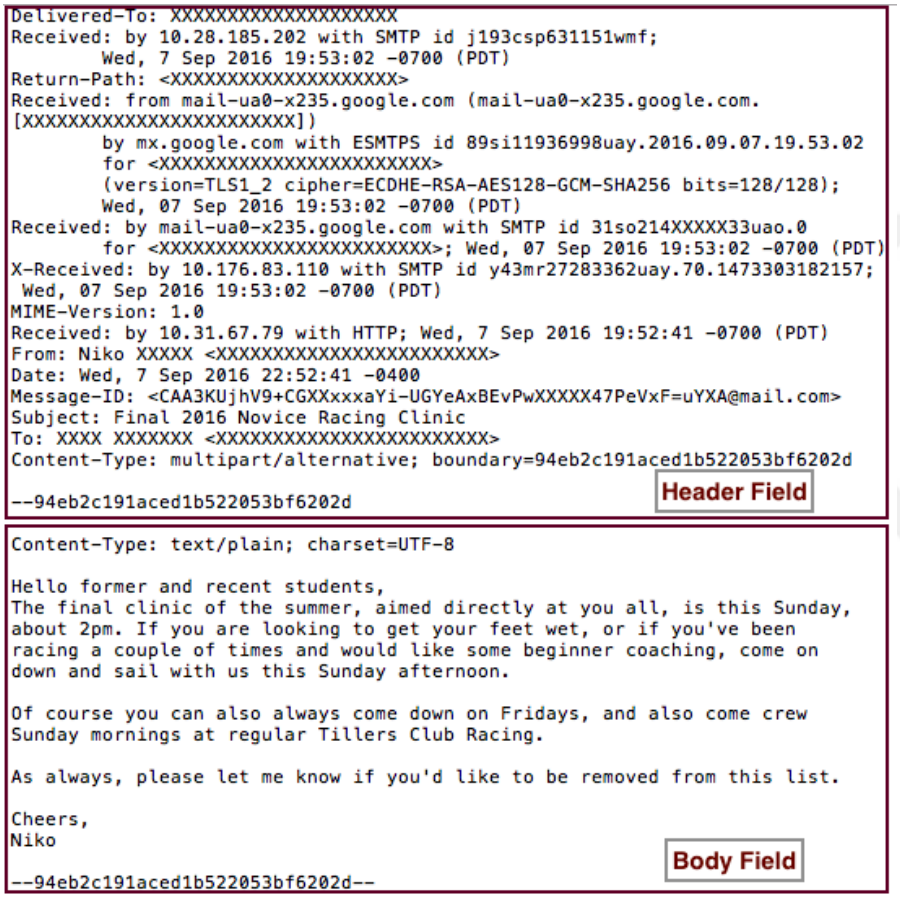


Figure 4.3: Raw format view of an e-mail

all attachments identified in this way and stores the original content locally to allow the user to retrieve the original attachment should that be necessary. Note that for security purposes, PELLUCIDATTACHMENT assigns new random file names when storing the attachments locally. This is similar in spirit and motivation to the sharing capabilities of systems such as Google Drive or Dropbox. The long random file names represent a capability that prevents attackers from enumerating or iterating over all attachments stored on the server. As the next step, the content filter forwards each attachment to a sandbox to render its contents into an image.

4.5.2 *Rendering Sandbox*

The goal of the rendering sandbox is to convert a given attachment into a visual representation (i.e., an image) of its content. Because the attacker might try to attack PELLUCIDATTACHMENT directly, the rendering sandbox operates in a virtual machine environment.

Our prototype implementation uses the KVM [60] hypervisor for this purpose. In a naive implementation, PELLUCIDATTACHMENT would spawn a new virtual machine from a known clean state for each attachment. However, to optimize performance without compromising on functionality, PELLUCIDATTACHMENT does not boot the virtual machine instance from a power-off state, but rather uses the snapshotting mechanism provided by the KVM hypervisor.

Of course, PELLUCIDATTACHMENT can only render attachments into images if the attachment is of a known file type. To achieve compat-

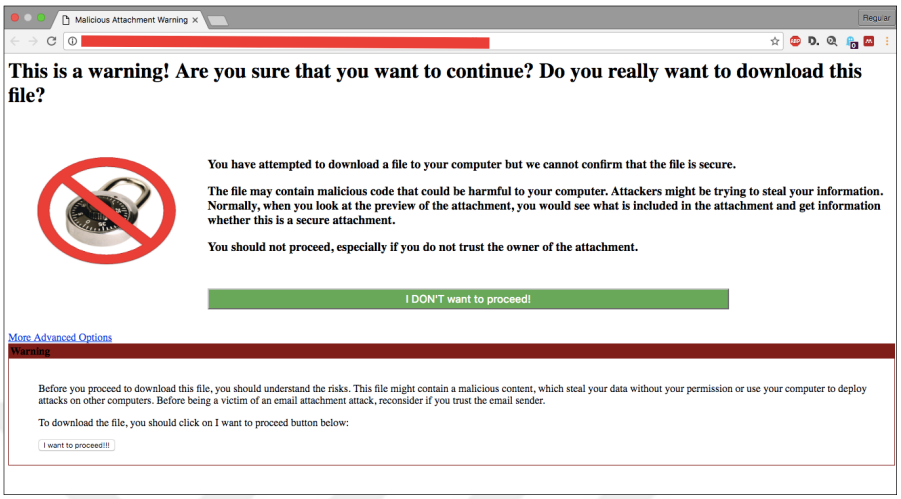
ibility with a large number of file-formats that are frequently used as e-mail attachments, PELLUCIDATTACHMENT leverages off-the-shelf utilities, such as the LibreOffice[67] office suite, Ghostscript [37], and Imagemagick[54]. Thus, PELLUCIDATTACHMENT supports any format produced by Microsoft Office products, as well as pdf and postscript content, and dozens of image formats.

Our current prototype implementation renders each attachment into an image in the PNG file format. As all popular mail user agents support PNG files natively, this ensures compatibility with a wide userbase. Once the attachment is rendered, the content filter receives the resulting image and continues to modify the content of the e-mail.

4.5.3 *Replacing Attachments*

To replace the converted attachments, the content filter simply replaces the content of the original attachment with the resulting images obtained from the rendering sandbox. While removing the original attachments is straightforward, PELLUCIDATTACHMENT must take care to insert the new images with the correct meta information to describe the content-type, length, and encoding. The filename of the converted attachment is assigned after the *original* attachments' name (including the old extension), followed by the page number and new extension. Therefore, the recipient of the e-mail can use the filename information of the file beside the image version of the attachment to decide the validity of the e-mail attachment before downloading the *original* attachment.

(a)



(b)

Figure 4.4: Warning page version 1 and 2

In addition to inserting the rendered images, PELLUCIDATTACHMENT also modifies the content of the e-mail to include hyperlinks to the unmodified attachments stored on the mail server. Of course, these links correctly refer to the randomized le names mentioned above.

4.5.4 *Providing Access to Unmodified Attachment*

Should users require access to unmodified attachments, PELLUCIDATTACHMENT serves the unmodified attachments via HTTP. To this end, our prototype leverages the popular NGINX [84] HTTP server. Of course, it is straightforward to use any other HTTP server, such as Apache or Flask instead.

To request access to an unmodified attachment, the user simply follows the corresponding link that PELLUCIDATTACHMENT inserted at the bottom of the e-mail. However, although attachments are served by a web server, the server prevents direct access to the content and instead presents the user with a warning screen (see Figure 4.4) modeled after the TLS certificate warnings found in all major web browsers. This warning informs the user of the potential negative implications of accessing the original attachment, and serves as a deterrent to unnecessary exposure to malicious documents. Only after the user has acknowledged that they want to proceed to download the original attachment does the download begin.

4.6 EVALUATION

In this section, we present the experiments and results we obtained by evaluating PELLUCIDATTACHMENT along two orthogonal dimensions that aim to answer the following two research questions.

RQ1: Does PellucidAttachment's rewriting capability prevent otherwise successful attacks? PELLUCIDATTACHMENT aims to improve the security of e-mail users by rewriting e-mail attachments with rendered images thereof. As the resulting images are included in the e-mail, PELLUCIDATTACHMENT must ensure that any malicious components of the original attachments are filtered out during the rendering process. To demonstrate the technical efficacy of PELLUCIDATTACHMENT against successful attacks through e-mail attachments, we evaluated PELLUCIDATTACHMENT with a variety of malicious files.

RQ2: Does PellucidAttachment improve the security of e-mail users? The rewriting capabilities implemented by PELLUCIDATTACHMENT provide additional information to the recipient of an e-mail. Ideally, this additional information allows the users of our system to make better security decisions (i.e., whether they should open a given e-mail attachment or not). To answer this research question, we performed an extensive user study on 50 volunteers.

In summary, our evaluation found that PELLUCIDATTACHMENT answers RQ1 in the affirmative and demonstrates significant (i.e., almost 4x) improvements of the security of e-mail users against malicious attachments as an answer to RQ2. The details of these experiments are presented next.

4.6.1 *RQ1: Efficacy of* PELLUCIDATTACHMENT

The technical efficacy of PELLUCIDATTACHMENT is determined by the system's capability to replace malicious attachments with benign renderings of their contents. Thus, to evaluate this aspect of PELLUCIDATTACHMENT, we obtained a representative sample of malicious files whose file types correspond to those commonly used in email-borne attacks. We obtained our dataset of malicious files from Google's VirusTotal service. In total, our dataset consisted of 39 malicious files (10 `.pdf`, 10 `.docx` (i.e., MS Word), 10 `.xlsx` (i.e., MS Excel), and 9 `.png`).

Once the files were confirmed as malicious, we composed one e-mail per file and included the file as an attachment in the e-mail.

These e-mails were sent to an MTA that ran the PELLUCIDATTACHMENT system. To ascertain whether PELLUCIDATTACHMENT indeed strips the malicious functionality from replaced attachments, we opened each e-mail with Thunderbird. As expected, we did not observe any signs of a malware infection after the attachments had been rewritten by PELLUCIDATTACHMENT. Additionally, we submitted all rewritten attachments to VirusTotal and not a single alert was raised, nor was any of the submitted files labeled as suspicious.

4.6.2 *RQ2: Security improvements for e-mail users*

To assess whether PELLUCIDATTACHMENT improves the security of regular e-mail users, we performed the following user study. As elaborated above, the rewriting capabilities offered by PELLUCIDATTACHMENT add information to the e-mail that a user can leverage when considering whether she should open a given attachment or not. We consider that PELLUCIDATTACHMENT increases a user's security if this additional information is sufficient for the user to decide not to open otherwise malicious e-mail attachments.

Thus, to evaluate PELLUCIDATTACHMENT's effectiveness in this regard, we designed a test scenario where participants were asked to read a series of e-mails. As the user study involves human volunteers as test subjects, we applied for and obtained approval from our university's institutional review board prior to launching the experiments.

4.6.2.1 *Study Design*

Participants were initially unaware of the purpose of the study. Instead, participants were told the study was an "experiment to investigate the effects of interruptions on concentration and decision-making when multi-tasking." Each participant was debriefed and informed of the true goal of the study once she finished the experiment.

4.6.2.2 *Experiment Environment*

Each participant in the study was provided with access to a Windows computer where Microsoft Office, Adobe Reader, and a pre-configured installation of Mozilla Thunderbird as the MUA were installed. The entire experiment was conducted in the context of the following fictitious scenario. We instructed study participants that they should assume the role of a graduate student aide in the university's writing center. As one would expect, this role included tasks such as answering e-mails and reviewing and editing documents (e.g., for grammar and spelling) that other students at the university submitted (also via e-mail). Furthermore, participants were told that they should assume that the provided e-mail account was their private university account, and thus treat it with equal care as their real accounts. Finally, the experiment suggested the prospect of a PayPal gift card for exceptional service.

Once the scenario was set up, each participant received a sequence of 16 e-mails. As this study focuses on e-mail attachments specifically, the e-mail set was structured as follows. Out of the 16 e-mails,

10 had no attachments, 4 e-mails featured malicious attachments, and the attachments of the remaining 2 e-mails were benign. The focus of our attention was on whether users would open the 4 malicious attachments, and whether the introduction of a system such as PELLUCIDATTACHMENT would have a positive effect on this number (i.e., fewer opened malicious attachments). To identify whether users opened attachments, all interactions with the provided computer were screen-captured and evaluated by the authors. True to the study protocol, all interaction between the researchers and the participants happened via e-mail.

4.6.2.3 *Recruitment*

We recruited participants on our university campus and the broader metropolitan area via e-mail. We only applied two criteria to prospective study participants. First, all applicants had to be between 18 and 49 years of age, and second, computer science students were excluded from the study. We excluded members from the computer science department to prevent any sort of security-knowledge bias that might be ingrained in CS students. Following this process, we found 50 volunteers from a broad spectrum of occupations, backgrounds, ethnicities, nationalities, and gender.

4.6.2.4 *Experiment Details*

On average, each participant required 25 minutes to read and react to all of the 16 e-mails. Additionally, each participant spent around 10 minutes to complete the study protocol form, the consent form,

and answer the security awareness questionnaire detailed later in this section.

Of the 16 e-mails, the first and last e-mail had the purpose of introducing participants to the system, and informing them that the experiment had concluded. The three e-mails following the introductory e-mail consisted of two e-mails with benign attachments and one e-mail without an attachment. The remaining eleven e-mails were a random sequence of the four e-mails with malicious attachments and the remaining seven e-mails with no attachments.

The four e-mails that contain malicious attachments were modeled after real-world attacks as follows. One of the attack e-mails imitated a non-existent university division that invited students to register for courses and internships by submitting the attached document. Another attack e-mail was composed to imitate a PayPal notification and allegedly included an attached invoice. The third attack e-mail appeared to originate from the university's human resources division and featured a subject line of "documents from - work." The e-mail did not contain any text but included a PDF attachment with a filename of "everyones_updated_salary_chart.pdf". The fourth attack e-mail was designed to trick users into sharing their password and deleting their e-mails. The e-mail was tailored to imitate an e-mail from the IT department. It included a file named "account_confirmation", and the users were told that their mailbox exceeded the storage limit. The e-mail explained that if the users would like to continue receiving e-mail, they should fill out the attached document with their username, password, and an error code that is

Table 4.1: Protocol designs

LABEL	INTRODUCTION E-MAIL VERSION	WARNING PAGE PERSION
Control	N/A	N/A
Group 1	Version 1	Version 1
Group 2	Version 1	Version 2
Group 3	Version 2	Version 2

Table 4.2: Download behavior per protocol

	BENIGN	ATCH.	MALICIOUS	ATCH.
Downloaded	X		X	
Not Downloaded		X		X
Control	20 (100%)	0 (0%)	25 (62.5%)	15 (37.5%)
Group 1	18 (90%)	2 (10%)	15 (37.5%)	25 (62.5%)
Group 2	16 (80%)	4 (20%)	9 (22.5%)	31 (77.5%)
Group 3	24 (60%)	16 (40%)	13 (16.3%)	67 (83.7%)

inserted at the end of the e-mail, and send the document back to the IT department.

4.6.2.5 *Protocols*

We assigned the 50 participants in our study to four distinct groups. The control group and Groups 1 and 2 contained ten participants each. The remaining 20 participants were assigned to Group 3.

The control group followed the study protocol without the protections afforded by PELLUCIDATTACHMENT, and the differences between Groups 1, 2, and 3 were confined to the introductory e-mail and the design of the warning page. As our results demonstrate, slight variations in the introductory e-mail or the warning page can have positive effects on the realized security gains. We included the control group

in our experiment to establish a baseline against which we can measure the improvements introduced by PELLUCIDATTACHMENT.

Participants in Group 1 received their e-mails through PELLUCIDATTACHMENT. The introduction e-mail for Group 1 contained a description of the differences in the style of e-mail the participants would receive. This e-mail described that any attachments would be included as an image only and that if the participant needed to access the original attachment, a link would be included at the bottom of each e-mail. If a participant from Group 1 clicked a link to retrieve the original attachment, she was redirected to the warning page shown in Figure4.4a. This initial version of the warning page provides two simple buttons to either proceed or abort.

Participants in Group 2 performed the same experiments as those in Group 1 where the only difference was the warning page that would open once a user requested an original attachment. Instead of the simple warning page used for Group 1, Group 2 uses a warning page that was inspired by Chrome's certificate warning page. In particular, the "Proceed" option was hidden, and would only become visible after the user clicked the "Advanced Options" link on the warning page (see Figure4.4b).

Participants in Group 3 experienced the same warning page as those in Group 2. Additionally, these participants received a slightly different version of the introductory e-mail which was designed to focus participants' attention on the pertinent content. To this end, we reduced the amount of text in the introductory e-mail and highlighted operative words in bold-red typeface.

4.6.2.6 *Results of User Study*

After carrying out the experiment according to the protocols defined above and in Table 4.1, we obtained the following results. The ten participants in the Control Group downloaded 25 (or 62%) of the 40 malicious attachments. Recall that each experiment features four malicious attachments, and thus ten participants will receive a total of 4 x 10 malicious attachments. This protocol was conducted to assess the baseline behavior of users when they receive e-mail attacks without the protection of PELLUCIDATTACHMENT. Groups 1, 2, and 3 received processed e-mails where the original attachments are replaced with images depicting their content. Group 1 had 10 participants, and they downloaded 15/40 (37.5%) of the malicious e-mail attachments in total (see Table 4.2). The more explicit warning page featured in the experiments for the ten participants in Group 2 further reduced the number of malicious attachments that were opened to merely 9/40 (or 22.5%). Finally, the improved introductory e-mail led to only 13/80 (or 16.3%) of downloaded malicious attachments among all 20 participants in Group 3. That is, PELLUCIDATTACHMENT reduced the probability of downloading a malicious attachment from 62% to only 16.3% an improvement of 3.8x when compared with the control group.

4.6.2.7 *Discussion of Experiment Results*

The aim of the user study is to show that our proposed approach indeed helps users to make better security decisions. Besides the

Table 4.3: P-values of the three protocols against the control group

CHI-SQUARE GROUPS	P-VALUE
Control Group vs. Group 1	0.0253
Control Group vs. Group 2	0.003
Control Group vs. Group 3	0.001

Null Hypothesis 1.	The probability of downloading malicious and benign attachments is independent of the assigned experimental group.

Table 4.4

clear improvements introduced by PELLUCIDATTACHMENT, there was a trend of opening malicious files across all of the groups. Hence, the user study was designed to deceive the participants. The content of the e-mails and the names of the attachments were chosen to lure users into downloading the attachments. Thirty-one participants opened at least one malicious attachment, and twenty-one of those participants downloaded the first malicious file they received, independent of the group they were assigned to. However, once participants learned about the correlation between attachments and their images embedded in the e-mails, they read e-mails with attachments more carefully, and the frequency of downloading malicious documents decreased. Although our study was deliberately designed to deceive users, using our system helped them avoid downloading malicious attachments and decreased the malicious attachment download rate from 68% to 16%.

To assess the utility of the different design decisions in PELLUCIDATTACHMENT we assess the following:

Table 4.5: P-values of the three protocols against the control group

	Password	Virus	Habit	Deletion	Attack	Confidence	Mean
Control (n = 10)	84	63	75	58	45	36	36.1
Group 1 (n = 10)	100	54	86	66	54	44	40.4
Group 2 (n = 10)	85	36	98	55	63	39	37.6
Group 3 (n = 20)	154	87	166	129	123	81	37

To test the null hypothesis, we performed a chi-squared test between the control group and each of the three groups that used PELLUCIDATTACHMENT. As Table 4.3 illustrates, all three chi-squared tests have p-values of less than 0.05 which indicates that the null hypothesis should be rejected. We conclude that using an explicit introductory e-mail and a warning page modeled after Chrome's certificate warning significantly reduce the probability that users open malicious attachments. In our experiments, users of PELLUCIDATTACHMENT were almost four times less likely to open malicious attachments than users from the control group. As such, we postulate that PELLUCIDATTACHMENT provides a significant increase in the security posture of regular e-mail users against email-borne threats.

4.6.2.8 *Security Awareness Survey*

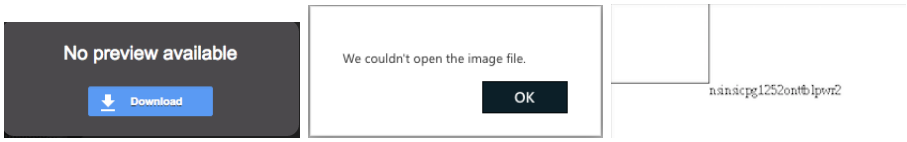
At the end of each experiment, we asked each participant to complete questions about their security awareness. To this end, we administered a modified version of the Security Awareness Survey published

by the SANS Institute[99]. We selected 18 out of 25 questions and divided them into six sections.

The first section contained five questions about password habits of the participants. The second section contained three questions about anti-virus usage and virus experience. The third section contained five questions that covered computer security practices and habits of the participants such as whether they backup their data. The fourth section contained three questions concerning the participants' knowledge of data deletion. The fifth section had three questions that asked about online attacks, such as phishing. The sixth section had two questions which measured the security knowledge confidence of a participant. The scoring system was developed to penalize the false answers by giving them lower points or zero while true answers were given full points. For example a participant who answered the question "Do you install updates on your computer or is it automatically updated?" as `No updates` was given zero, `Manual Updates` was given 1 point, and `Automatically updates` was given 2 points. As a result of this scoring method, a participant with higher awareness had higher scores in each part.

Table4.5 shows the score of each section for each group. The mean of scores between groups was similar across all groups, which leads us to conclude that neither group had an advantage in security knowledge over any of the other groups.

(b) Gmail Preview (c) Outlook Preview (d) PELLUCIDATTACHMENT Preview



(f) Gmail Preview (g) Outlook Preview (h) PELLUCIDATTACHMENT Preview

Figure 4.5: Preview of malicious files via GMail, Outlook and PELLUCIDATTACHMENT

4.7 RELATED WORK

Malicious PDF files can be created by embedding JavaScript, executable code, or any other content directly into the PDF. One of the most commonly used techniques to detect such attacks is structural analysis (e.g., checking n-gram features, the number of objects and the streams of the PDF file, etc.). Laskov and Srndic [62], Smutz and Stavrou [101], and Srndic and Laskov[104] perform structural analysis of PDF files to assess if the file is malicious. Other research groups, in contrast, have used reverse mimicry techniques to show that assessing structural features alone is not enough[73]for detecting malicious documents.

Liu et al. [69] present a different approach to detect malicious files. In this work, the authors use both static and dynamic features for detection, and implement a prototype malicious PDF detector. They had evaluate their system with real benign and malicious samples.

In 2001, Balzer [10] implemented a system called SafeEmailAttachments as a wrapper on Windows NT systems. SafeEmailAttachments was designed to follow safety and active content rules before the attachment was authorized to be opened. SafeEmailAttachments successfully blocked the I-Love-You virus when the virus first started spreading [51].

Malicious Email Tracking (MET) addresses the virus infection problem through e-mail by using behavioral-based analysis[12]. Later, the authors proposed an approach that supports a wider scope of this online behaviour-based security system[106].

Muniandy [82] proposes a practical approach for educating Internet users using e-mail screen shots. To increase the awareness of phishing e-mails, e-mail screen shots of dedicated phishing e-mails are shown to the Internet user. These screen shots highlight characteristics upon which a user can recognize phishing attempts. Both ours and Muniandy's approach leverage visual impressions to let the user decide if the e-mail is benign or malicious. However, our approach differs in several fundamental ways. First, our approach is not primarily for educational purposes. Second, our tool processes every incoming mail. Third, our system generates an image for every single e-mail attachment whereas Muniandy uses eight pre-defined dedicated screen shots for educating people.

Moreover, there are commercial solutions for preventing virus distribution through e-mail that integrate an anti-virus engine into their MTA[28, 29]. These commercial tools claim to provide an image display of the delivered attachments similar to GMail[38] and Outlook[87].

However, we could not gather any information about these commercial solutions to perform a comparative analysis. Figure {fig:pdf2png} demonstrates that both GMail and Outlook fail to display any preview of the tested malicious files and provide any guidance to users. On the other hand, PELLUCIDATTACHMENT successfully generates previews for the same malicious attachments.

Additionally, there is a variety of research on isolation of execution environments for preventing attacks through e-mail or web browsers where users can download a malicious file in a virtual machine. For example, Moshchuk et al.[81] present an anti-malware tool called SpyProxy. This tool detects drive-by-download attacks by rendering webpages in virtual machines. SpyProxy was developed as a front end module that redirects HTTP requests to a virtual machine depending on the webpage contents. The webpage undergoes static analysis, and if the webpage has active content or has any non-HTML content types that are considered to be unsafe, the page is queued for dynamic processing. A performance analysis of SpyProxy revealed that it would add a considerable (600 ms) delay to each web page load.

Furthermore, Radhakrishnan et al.[92] propose to leverage dynamic sandboxing to provide an isolated execution environment for potentially malicious content. These works drastically differ from PELLUCIDATTACHMENT, along two major directions. First, users can benefit from PELLUCIDATTACHMENT without changing the workflow or tools (e.g., mail clients) they are already accustomed too, and these tools do not need to be modified. And second, while virtual machines and sandboxes are at the core of the above-mentioned systems, PELLUCI-

DAttachment merely uses virtual machines exclusively for the purpose of automatically rendering attachments into image files. A PellucidAttachment user is not inconvenienced by the existence of a virtual machine, nor is she even aware of its use.

In summary, our proposed solution is substantially different from existing research. Unlike previous work, PellucidAttachment does not try to distinguish between malicious or benign attachments. Instead, our approach converts attachments into images so that the user has less exposure to malicious content.

4.8 DISCUSSION

Spam and phishing prevention tools exist to keep users away from any malicious e-mails. In this study we focused on malicious e-mail attachments, which might be in the form of spam, phishing, spearphishing, or even from a legitimate source where the sender is unaware of malicious activity contained inside the attached file.

While PellucidAttachment prevents the spreading of generic malicious documents that are crafted for a general audience, users will still be vulnerable to malicious e-mail attachments that are specifically crafted to render to a meaningful image that convinces the user to acknowledge the security warning, download, and finally open the original file. However, \basename significantly raises the bar for the attacker by requiring an image to be semantically meaningful to elicit this user behavior.

One of PELLUCIDATTACHMENT's limitations is the lack of support for non-visual file formats. PELLUCIDATTACHMENT cannot render files that have no meaningful visual representation (e.g., binary files, or compressed archives). However, according to VirusTotal statistics [115], \basename covers the majority of the top 10 malicious file types that typically include malicious functionality.

PELLUCIDATTACHMENT's e-mail content filter only replaces e-mail attachments with corresponding PNG image versions. Our approach protects users from malicious documents by blocking these artifacts before they are downloaded to the victim's system. Clearly, the modifications that \basename performs on e-mails would break any cryptographic signatures. However, PELLUCIDATTACHMENT could be augmented to either pass through (unmodified) cryptographically signed e-mails from verified and trusted senders or to re-sign e-mails with a trusted identity. The threat model of cryptographic signatures states that attackers cannot forge valid signatures of verified senders, and thus such a mechanism would allow the small number of cryptographically signed e-mails to be authenticated correctly.

4.9 OUTCOME

In this work, we proposed a novel defense mechanism against the prevalent threat of malicious e-mail attachments. One core insight of our work is that today, e-mail recipients have insufficient information to make an informed decision on whether a given attachment is benign (i.e., can be opened without concern) or malicious (i.e., opening

the attachment poses a security risk). Our prototype implementation PELLUCIDATTACHMENT narrows this information gap and replaces all attachments with images of their content. The conversion applied by PELLUCIDATTACHMENT strips any potentially malicious traits of an attachment while preserving the attachment's visual appearance. This methodology provides additional information to users and allows them to make better-informed decisions on how to handle e-mail attachments.

We evaluated PELLUCIDATTACHMENT with an experiment on 39 malicious attachments that attack various vulnerabilities in real-world software. The transformations applied by PELLUCIDATTACHMENT successfully rendered all attacks ineffective. Additionally, we performed an extensive user study (n=50) that measures and demonstrates the effectiveness of PELLUCIDATTACHMENT to protect potential victims from email-borne attacks. Our results indicate that PELLUCIDATTACHMENT reduces the probability for an untrained user to open a malicious e-mail attachment by a factor of almost 4.

These results demonstrate that PELLUCIDATTACHMENT significantly raises the bar for attackers that seek to infect their victims through malicious e-mail attachments.

Part III

EPILOGUE

5

CONCLUSION

The constant fight between attacks and defenses in computer systems is going to remain as far as technology exists in our lives. While security experts develop new methods to overcome security issues, attackers also counteract and design new techniques that are more difficult to identify. As a result, these new techniques provoke more critical problems. Today, attackers exploit vulnerabilities in web browsers and e-mails which are the most utilized tools by any level of user. Throughout this dissertation, I examined possible automated security solutions for deceptive attacks on web pages and e-mails. I depicted my claim as:

- Online systems can be designed with optimized settings to help users to make security decisions effectively.

The support for this claim has been presented in the form of three discrete research projects:

1. Trick Banners: One of the most common attacks that are easy to integrate on web pages is malicious advertisement banners. This deceptive attack deploys advertisement banners that disguised as correct links. They imitate the visual characteristics of correct links, and they are typically placed in multiple places on a web page to confuse the victim on distinguishing the correct

link on a web page. To solve this problem, I designed and implemented a Firefox extension: TrueClick. TrueClick determines and blocks the regions on the web page where trick banners are displayed to help users on distinguishing trick banners from correct links.

2. Spearphishing E-mail Attacks: After spam and phishing e-mails success rate started to decrease, attackers concentrated on social engineering techniques to tailor personalized attack e-mails: spearphishing e-mails. This type of deceiving attack is successful because it impersonates a trusted e-mail sender. I designed and implemented a new approach to solve this problem: EmailProfiler. EmailProfiler detects spearphishing attacks using both features extracted from both e-mail metadata and stylometric information. A profiler application has been implemented that analyzes e-mails and shows the similarity rate of that e-mail with the sender.

3. Malicious E-Mail Attachments: In this study, I focused on malicious e-mail attachments, which might be in the form of spam, phishing, spearphishing, or even from a legitimate source where the sender is unaware of malicious activity contained inside the attached file. To solve the problem with malicious e-mail attachments, I designed and implemented a new e-mail attachment protection approach: PellucidAttachment. PellucidAttachment allows users to check the contents of the attachment before the attachments are downloaded to users' systems.

5.1 EXPECTATION VS. RESULTS DISCUSSION

Across this dissertation, I argued that current mitigations for attacks neglect human factor in the systems. I examined deceptive attack instances such as trick banners and social engineering attacks on emails. I showed that it is possible to develop approaches to enhance end-user's assessment on risky actions and to help users to adapt their behavior accordingly.

In Chapter 2, our approach of image extraction was effective to distinguish trick banners, and consequently the users were exposed to less malicious advertisement banners. My initial approach was to use an algorithm to detect trick banners based on spurious links. However, the links did not provide enough information to categorize, detect, and block them automatically. Generating blacklists and whitelists are some of the approaches that current mitigation methods utilize. While examining the web pages, I realized trick banners were different than legitimate download links visually. Based on this heuristic I started collecting data. Another, interesting point with this project is data collection. All data points are labeled manually as trick banners and legitimate links by inspecting each one them at the time of collection. After obtaining classification results with image processing approach, I conducted the user study. I tested the implementation of TrueClick on 40 users, and found that TrueClick helps users to distinguish correct links almost four-fold improvement compared to not using any trick banner blocking software.

In Chapter 3, to warn users on possible spearphishing attacks I proposed to create a profile per user and analyzed 20 users' e-mails. By changing the sender label, I recreated spearphishing attack scenarios and tested the profilers' results. Overall, I generated 215 test cases for sentbox model and 20 test cases for inbox model. The best case both models reached 100% predictability. The worst case for sentbox model was 67% predictability, and 80% predictability for inbox model. In constrast to the study results, I had expected to see similar predictability rates because both models used the same profiling apporoach. Then, I realized that the low prediction rate was caused by e-mails with only subject information or with only email attachments that were sent from different machines with different IPs. When there is no consistent information in the header and the body part of the emails, the generated profile based on those e-mails do not perform well against the sender's original e-mails.

In Chapter 4, I looked deeper in e-mail attacks and searched for ways to remove malicious e-mail attachments before they deceive users. When attachments converted into PNG image formats, their content was available to users. First, I tried only PDF files and I reliazed that the converted version of e-mail attachments is not malicious anymore. Later on in the project I converted other file types (Microsoft Word, Microsoft Excel, PNG) that were frequently associated with malicious content. I set up different protocols for user study where I changed the design of the new systems introduction email and warning page. With each improvement in the design participants were more aware of the risks encountered. PellucidAttachment re-

duced the probability of downloading a malicious attachment from 62% to only 16.3% an improvement of 3.8x when compared with the control group.

5.2 FUTURE DIRECTION

Although my approaches helped users to make better security decisions during the user studies, I believe that there is always space for improvement.

TRUECLICK could be supported with static analysis of web pages in order to further improve detection effectiveness, which is a promising direction for future research.

While PELLUCIDATTACHMENT prevents the spreading of generic malicious documents that are crafted for a general audience, users will still be vulnerable to malicious e-mail attachments that are specifically crafted to render to a meaningful image that convinces the user to acknowledge the security warning, download, and finally open the original file. Adding an extra layer of protection and checking new rendered attachments' safety automatically before of delivering them to the users would prevent the attack aforementioned.

5.3 DISCLAIMER AND COPYRIGHTS

The portion of this research related to human subjects was approved by the Northeastern University Institutional Review Board (IRB #15-09-12 and IRB #16-09-08).

Trick banner detection, spearphishing identification and secure e-mail attachment parts of the research have been finalized and explained in Chapter 2, Chapter 3 and Chapter 4. The work in Chapter 2 is published in [32] and the work in Chapter 3 is published in [31].

Part IV

APPENDIX

BIBLIOGRAPHY

[1] Ahmed Abbasi and Hsinchun Chen. "Writeprints: A Stylometric Approach to Identity-level Identification and Similarity Detection in Cyberspace." In: *ACM Trans. Inf. Syst.* 26.2 (Apr. 2008), 7:1–7:29. DOI: 10.1145/1344411.1344413. URL: http://doi.acm.org/10.1145/1344411.1344413 (cit. on pp. 53, 73, 74).

[2] Eytan Adar, Desney S Tan, and Jaime Teevan. "Benevolent deception in human computer interaction." In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM. 2013, pp. 1863–1872 (cit. on p. 19).

[3] *AdBlock Plus*. (accessed 1 January 2017). URL: https://adblockplus.org (cit. on pp. 6, 42).

[4] "Adobe Reader Out-Of-Bounds Indexing Remote Code Execution Vulnerability." In: (2017). URL: http://www.zerodayinitiative.com/advisories/ZDI-16-191/ (cit. on p. 84).

[5] *ADSafe*. (accessed 1 January 2017). URL: http://www.adsafe.org (cit. on pp. 6, 43).

[6] Sadia Afroz, Michael Brennan, and Rachel Greenstadt. "Detecting Hoaxes, Frauds, and Deception in Writing Style Online." In: *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. SP '12. Washington, DC, USA: IEEE Computer So-

ciety, 2012, pp. 461–475. DOI: 10.1109/SP.2012.34. URL: http://dx.doi.org/10.1109/SP.2012.34 (cit. on p. 73).

[7] Sadia Afroz, Aylin Caliskan Islam, Ariel Stolerman, Rachel Greenstadt, and Damon McCoy. "DoppelgäNger Finder: Taking Stylometry to the Underground." In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy*. SP '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 212–226. DOI: 10.1109/SP.2014.21. URL: http://dx.doi.org/10.1109/SP.2014.21 (cit. on pp. 43, 73).

[8] Shivam Aggarwal, Vishal Kumar, and S D Sudarsan. "Identification and Detection of Phishing Emails Using Natural Language Processing Techniques." In: *SIN* (2014), pp. 217–222 (cit. on p. 76).

[9] Devdatta Akhawe and Adrienne Porter Felt. "Alice in warningland: A large-scale field study of browser security warning effectiveness." In: 2013, pp. 257–272 (cit. on p. 85).

[10] R. Balzer. "Assuring the safety of opening email attachments." In: *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*. Vol. 2. IEEE Comput. Soc, 2001, pp. 257–262. DOI: 10.1109/DISCEX.2001.932177. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?%20arnumber=932177 (cit. on p. 110).

[11] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." In: *Computer vision–ECCV 2006* (2006), pp. 404–417 (cit. on p. 24).

[12] Manasi Bhattacharyya, Shlomo Hershkop, and Eleazar Eskin. "MET: An Experimental System for Malicious Email Tracking." In: *Proceedings of the 2002 workshop on New security paradigms - NSPW '02*. New York, New York, USA: ACM Press, 2002, p. 3. DOI: 10.1145/844102.844104. URL: http://portal.acm.org/citation.cfm?doid=844102.844104 (cit. on p. 110).

[13] Leo Breiman. "Random forests." In: *Machine learning* 45.1 (2001), pp. 5–32 (cit. on p. 31).

[14] Marcelo Luiz Brocardo, Issa Traore, Shatina Saad, and Isaac Woungang. *Authorship verification for short messages using stylometry*. IEEE, 2013 (cit. on p. 75).

[15] Calyptix. *DNC Hacks: How Spear Phishing Emails Were Used*. 2016. URL: http://www.calyptix.com/top-threats/dnc-hacks-how-spear-phishing-emails-were-used/ (cit. on p. 79).

[16] Jean Louis Chandon, Mohamed Saber Chtourou, and David R Fortin. "Effects of configuration and exposure levels on responses to web advertisements." In: *Journal of advertising research* 43.2 (2003), pp. 217–229 (cit. on p. 19).

[17] Jean-Louis Chandon and Mohamed Saber Chtourou. "Factors Affecting Click-Through Rate." In: *Online Consumer Psychology: Understanding and Influencing Consumer Behavior in the Virtual World*. Ed. by Curtis P. Haugtvedt, Karen A. Machleit, and Richard Yalch. Psychology Press, Jan. 2005. Chap. 6 (cit. on p. 41).

[18] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart. "Cracking-Resistant Password Vaults Using Natural Language Encoders." In: *Security and Privacy (SP), 2015 IEEE Symposium on*. May 2015, pp. 481–498. DOI: 10.1109/SP.2015.36 (cit. on p. 75).

[19] Teh-Chung Chen, Scott Dick, and James Miller. "Detecting Visually Similar Web Pages: Application to Phishing Detection." In: *ACM TOIT* 10.2 (June 2010), 5:1–5:38 (cit. on p. 42).

[20] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. "Clonecloud: elastic execution between mobile device and cloud." In: *Proceedings of the sixth conference on Computer systems*. ACM. 2011, pp. 301–314 (cit. on p. 6).

[21] Gregory Conti and Edward Sobiesk. "Malicious interface design: exploiting the user." In: *Proceedings of the 19th international conference on World Wide Web*. ACM. 2010, pp. 271–280 (cit. on p. 19).

[22] Lorrie Faith Cranor. "Can phishing be foiled?" In: *Scientific American* 299.6 (2008), pp. 104–110 (cit. on pp. 6, 80).

[23] Kevin Crowston, Eileen E. Allen, and Robert Heckman. "Using natural language processing technology for qualitative data analysis." In: *International Journal of Social Research Methodology* 15.6 (2012), pp. 523–543. DOI: 10.1080/13645579.2011.625764. eprint: http://dx.doi.org/10.1080/13645579.2011.625764. URL: http://dx.doi.org/10.1080/13645579.2011.625764 (cit. on p. 75).

[24] *CVE-2016-10087*. 2017. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-10087 (cit. on p. 84).

[25] *CVE-2016-1009*. 2017. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-1009 (cit. on p. 84).

[26] *CVE-2016-3714*. 2017. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-3714 (cit. on p. 84).

[27] *CVE-2016-7193*. 2017. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-7193 (cit. on p. 83).

[28] *Cybermail*. 2017. URL: https://www.cybersolutions.co.jp/product/cybermail/ (cit. on p. 110).

[29] *DEEPmail*. 2017. URL: http://www.qualitia.co.jp/product/dm/ (cit. on p. 110).

[30] Vikas P Deshpande, Robert F Erbacher, and Chris Harris. *An Evaluation of Naive Bayesian Anti-Spam Filtering Techniques*. IEEE, 2007 (cit. on pp. 6, 76).

[31] S. Duman, K. Kalkan-Cakmakci, M. Egele, W. Robertson, and E. Kirda. "EmailProfiler: Spearphishing Filtering with Header and Stylometric Features of Emails." In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. June 2016, pp. 408–416. DOI: 10.1109/COMPSAC.2016.105 (cit. on p. 122).

[32] Sevtap Duman, Kaan Onarlioglu, Ali Osman Ulusoy, William Robertson, and Engin Kirda. "Trueclick: automatically distinguishing trick banners from genuine download links." In: *Pro-*

ceedings of the 30th Annual Computer Security Applications Conference. ACM. 2014, pp. 456–465 (cit. on p. 122).

[33] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. "You've been warned: an empirical study of the effectiveness of web browser phishing warnings." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2008, pp. 1065–1074 (cit. on p. 86).

[34] Matthew Finifter, Joel Weinberger, and Adam Barth. "Preventing Capability Leaks in Secure JavaScript Subsets." In: *Proc. of the 17th Network and Distributed System Security Symposium (NDSS 2010)*. Vol. 99. 2010, pp. 1–14 (cit. on pp. 6, 43).

[35] K. Gallagher and J. Parsons. "A Framework for Targeting Banner Advertising on the Internet." In: *HICSS*. 1997 (cit. on p. 41).

[36] F. Gargiulo and C. Sansone. "Combining Visual and Textual Features for Filtering Spam Emails." In: *ICPR*. 2008 (cit. on p. 42).

[37] *GhostScript*. (accessed 1 January 2017). URL: http://www.ghostscript.com/ (cit. on p. 95).

[38] *Gmail*. 2017. URL: https://www.google.com/gmail/ (cit. on p. 110).

[39] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. 2nd. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001 (cit. on p. 22).

[40] Robert Goodman, Matthew Hahn, Madhuri Marella, Christina Ojar, and Y Westcott. *The Use of Stylometry for Email Author Identification: A Feasibility Study*. 2007 (cit. on p. 74).

[41] Google. *Google Caja*. URL: http://developers.google.com/caja (cit. on p. 43).

[42] Michael C Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. "Unsafe exposure analysis of mobile in-app advertisements." In: *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*. ACM. 2012, pp. 101–112 (cit. on p. 43).

[43] Chris Grier, Shuo Tang, and Samuel T King. "Secure web browsing with the OP web browser." In: *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE. 2008, pp. 402–416 (cit. on p. 43).

[44] Jeremiah Grossman and Matt Johansen. "Million browser botnet." In: *Black Hat USA* (2013) (cit. on p. 16).

[45] Salvatore Guarnieri and V Benjamin Livshits. "GATEKEEPER: Mostly Static Enforcement of Security and Reliability Policies for JavaScript Code." In: (cit. on p. 43).

[46] Saikat Guha, Bin Cheng, and Paul Francis. "Privad: Practical privacy in online advertising." In: *SENIX conference on Networked systems design and implementation, NSDI*. 2011, pp. 169–182 (cit. on pp. 6, 43).

[47] Hamed Haddadi, Pan Hui, and Ian Brown. "MobiAd: private and scalable mobile advertising." In: *Proceedings of the fifth*

ACM international workshop on Mobility in the evolving internet architecture. ACM. 2010, pp. 33–38 (cit. on p. 43).

[48] T Hastie, R Tibshirani, and J Friedman. "The Elements of Statistical Learning, 2nd edn., vol. 18." In: *Springer Series in Statistics, Springer New York, doi* 10 (2009), b94608 (cit. on p. 31).

[49] Curtis P Haugtvedt, Karen A Machleit, and Richard Yalch. *Online consumer psychology: understanding and influencing consumer behavior in the virtual world*. Psychology Press, 2005 (cit. on p. 19).

[50] Cormac Herley. "Why do Nigerian Scammers Say They are from Nigeria?" In: *WEIS*. 2012. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=167719 (cit. on p. 49).

[51] Ian Hopper. "Destructive 'I LOVE YOU'Computer virus strikes worldwide." In: *CNN Interactive Technology* (2000) (cit. on p. 110).

[52] David-Sarah Hopwood. *Jacaranda Language Specification, draft 0.3* (cit. on p. 43).

[53] Lin-Shung Huang, Alexander Moshchuk, Helen J Wang, Stuart Schecter, and Collin Jackson. "Clickjacking: Attacks and Defenses." In: *USENIX Security Symposium*. 2012, pp. 413–428 (cit. on p. 21).

[54] *ImageMagick*. (accessed 1 January 2017). URL: https://www.imagemagick.org (cit. on pp. 84, 95).

[55] InformAction. *NoScript*. URL: http://noscript.net (cit. on p. 41).

[56] *Interactive Advertising Bureau*. 2012. URL: http://www.iab.net/media/file/IAB%5C_Internet%5C_Advertising%5C_Revenue%5C_Report%5C_FY%5C_2012%5C_rev.pdf (cit. on p. 28).

[57] Farkhund Iqbal, Liaquat A. Khan, Benjamin C. M. Fung, and Mourad Debbabi. "e-Mail Authorship Verification for Forensic Investigation." In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC '10. Sierre, Switzerland: ACM, 2010, pp. 1591–1598. DOI: 10.1145/1774088.1774428. URL: http://doi.acm.org/10.1145/1774088.1774428 (cit. on p. 74).

[58] S Isaacs. *Microsoft web sandbox* (cit. on p. 43).

[59] Ari Juels. "Targeted advertising... and privacy too." In: *Cryptographers' Track at the RSA Conference*. Springer. 2001, pp. 408–424 (cit. on p. 43).

[60] *Kernel Virtual Machine*. URL: http://www.linux-kvm.org/ (cit. on p. 94).

[61] L. Zeltser. *Malvertising: Some Examples of Malicious Ad Campaigns*. URL: https://zeltser.com/malvertising-malicious-ad-campaigns/ (cit. on p. 6).

[62] Pavel Laskov and Nedim Šrndić. "Static detection of malicious JavaScript-bearing PDF documents." In: *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM. 2011, pp. 373–382 (cit. on p. 109).

[63] Gerard Ledger and Thomas Merriam. "Shakespeare, Fletcher, and the Two Noble Kinsmen." In: *Literary and Linguistic Computing* 9.3 (1994), pp. 235–248. DOI: 10.1093/llc/9.3.235.

eprint: http://llc.oxfordjournals.org/content/9/3/235.full.pdf+html. URL: http://llc.oxfordjournals.org/content/9/3/235.abstract (cit. on pp. 53, 74).

[64] Ilias Leontiadis, Christos Efstratiou, Marco Picone, and Cecilia Mascolo. "Don't kill my ads!: balancing privacy in an ad-supported mobile application market." In: *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM. 2012, p. 2 (cit. on p. 43).

[65] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. "Knowing your enemy: understanding and detecting malicious web advertising." In: *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM. 2012, pp. 674–686 (cit. on p. 20).

[66] *libpng*. 2017. URL: http://www.libpng.org/pub/png/libpng.html (cit. on p. 84).

[67] *LibreOffice*. (accessed 1 January 2017). URL: https://www.libreoffice.org/ (cit. on p. 95).

[68] Eric Lin, John Aycock, and Mohammad Mannan. *Lightweight Client-Side Methods for Detecting Email Forgery*. English. Ed. by DongHoon Lee and Moti Yung. 2012. DOI: 10.1007/978-3-642-35416-8_18. URL: http://dx.doi.org/10.1007/978-3-642-35416-8_18 (cit. on p. 75).

[69] Daiping Liu, Haining Wang, and Angelos Stavrou. "Detecting Malicious Javascript in PDF through Document Instrumentation." In: (2014), pp. 100–111 (cit. on p. 109).

[70] André Lötter and Lynn Futcher. "A framework to assist email users in the identification of phishing attacks." In: *Inf. & Comput. Security ()* 23.4 (2015), pp. 370–381 (cit. on p. 77).

[71] M. Mimoso. *Malware Campaign Leverages Ad Networks, Sends Victims to Blackhole*. URL: https://threatpost.com/malware-campaign-leverages-ad-networks-sends-victims-to-blackhole/102213/ (cit. on p. 6).

[72] Sergio Maffeis and Ankur Taly. "Language-based isolation of untrusted Javascript." In: *Computer Security Foundations Symposium, 2009. CSF'09. 22nd IEEE*. IEEE. 2009, pp. 77–91 (cit. on pp. 6, 43).

[73] Davide Maiorca, Igino Corona, and Giorgio Giacinto. "Looking at the Bag is Not Enough to Find the Bomb: An Evasion of Structural Methods for Malicious PDF Files Detection." In: *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*. ASIA CCS '13. Hangzhou, China: ACM, 2013, pp. 119–130 (cit. on p. 109).

[74] *Marketing Terms. Trick banner definition.* Accessed: 2017-01-01. URL: http://www.marketingterms.com/dictionary/trick_banner/ (cit. on pp. 15, 18).

[75] Max-Emanuel Maurer and Dennis Herzner. "Using Visual Website Similarity for Phishing Detection and Reporting." In: *CHI Extended Abstracts*. 2012 (cit. on p. 42).

[76] Andrew W. E. McDonald, Sadia Afroz, Aylin Caliskan, Ariel Stolerman, and Rachel Greenstadt. "Use Fewer Instances of

the Letter "I": Toward Writing Style Anonymization." In: *Proceedings of the 12th International Conference on Privacy Enhancing Technologies*. PETS'12. Vigo, Spain: Springer-Verlag, 2012, pp. 299–318. DOI: 10.1007/978-3-642-31680-7_16. URL: http://dx.doi.org/10.1007/978-3-642-31680-7_16 (cit. on p. 74).

[77] Eric Medvet, Engin Kirda, and Christopher Kruegel. "Visual-Similarity-Based Phishing Detection." In: *SecureComm*. 2008 (cit. on p. 42).

[78] *Microsoft Malware Protection Center*. 2017. URL: https://www.microsoft.com/security/portal/enterprise/threatreports%5C_july%5C_2015.aspx (cit. on p. 83).

[79] M. Mimoso. *Malware Campaign Leverages Ad Networks, Sends Victims to Blackhole.* (accessed 1 January 2017). URL: http:%20//threatpost.com/malware-campaign-leverages-%20ad-networks-sends-victims-to-blackhole (cit. on p. 16).

[80] David Modic and Ross Anderson. "Reading this may harm your computer: The psychology of malware warnings." In: *Computers in Human Behavior* 41 (2014), pp. 71–79 (cit. on p. 86).

[81] Alexander Moshchuk, Tanya Bragin, Damien Deville, S.D. Steven D Gribble, and Henry M H.M. Levy. *SpyProxy : Execution-based Detection of Malicious Web Content*. 2007. URL: http://dl.acm.org/citation.cfm?id=1362903.1362906 (cit. on p. 111).

[82] Lalitha Muniandy. "Phishing: Educating the Internet users – a practical approach using email screen shots." In: *IOSR Jour-*

nal of Research & Method in Education (IOSRJRME) 2.3 (2013), pp. 33–41. DOI: 10.9790/7388-0233341 (cit. on p. 110).

[83] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. "On the Feasibility of Internet-Scale Author Identification." In: *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. SP '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 300–314. DOI: 10.1109/SP.2012.46. URL: http://dx.doi.org/10.1109/SP.2012.46 (cit. on p. 74).

[84] *NGINX*. (accessed 1 January 2017). URL: https://nginx.org/ (cit. on p. 96).

[85] Sarwat Nizamani and Nasrullah Memon. "CEAI: CCM-based email authorship identification model." In: *Egyptian Informatics Journal* 14.3 (2013), pp. 239–249. DOI: http://dx.doi.org/10.1016/j.eij.2013.10.001. URL: http://www.sciencedirect.com/science/article/pii/S111086651300039X (cit. on p. 74).

[86] Kaan Onarlioglu, Utku Ozan Yilmaz, Engin Kirda, and Davide Balzarotti. "Insights into User Behavior in Dealing with Internet Attacks." In: *Network and Distributed Systems Security Symposium*. 2012 (cit. on pp. 16, 19, 41).

[87] *Outlook.com - Microsoft free personal email*. 2017. URL: https://outlook.live.com/owa/ (cit. on p. 110).

[88] Sean Palka and Damon McCoy. "Fuzzing E-mail Filters with Generative Grammars and N-Gram Analysis." In: *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. Washington, D.C.:

USENIX Association, Aug. 2015. URL: https://www.usenix.org/conference/woot15/workshop-program/presentation/palka (cit. on p. 75).

[89] Paul Pearce, Adrienne Porter Felt, Gabriel Nunez, and David Wagner. "Addroid: Privilege separation for applications and advertisers in android." In: *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. Acm. 2012, pp. 71–72 (cit. on p. 43).

[90] *Postfix*. 2017. URL: http://www.postfix.org/ (cit. on p. 91).

[91] *Privoxy*. URL: http://www.privoxy.org (cit. on p. 41).

[92] Manigandan Radhakrishnan and Jon A Solworth. "Quarantining untrusted entities: Dynamic sandboxing using LEAP." In: *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*. IEEE. 2007, pp. 211–220 (cit. on p. 111).

[93] Venkatesh Ramanathan and Harry Wechsler. "Phishing detection and impersonated entity discovery using Conditional Random Field and Latent Dirichlet Allocation." In: *Computers & Security* 34 (May 2013), pp. 123–139 (cit. on p. 77).

[94] Charles Reis and Steven D Gribble. "Isolating web programs in modern browser architectures." In: *Proceedings of the 4th ACM European conference on Computer systems*. ACM. 2009, pp. 219–232 (cit. on p. 43).

[95] Alexey Reznichenko, Saikat Guha, and Paul Francis. "Auctions in do-not-track compliant internet advertising." In: *Pro-*

ceedings of the 18th ACM conference on Computer and communications security. ACM. 2011, pp. 667–676 (cit. on p. 43).

[96] *RiskIQ Malvertising Report*. Accessed: 2017-01-30. URL: https://www.riskiq.com/blog/labs/malvertising-on-the-rise-once-again/ (cit. on p. 5).

[97] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "The earth mover's distance as a metric for image retrieval." In: *International journal of computer vision* 40.2 (2000), pp. 99–121 (cit. on p. 27).

[98] W. Salusky. *Malvertising*. (accessed 1 January 2017). URL: http://isc.sans.edu/diary/Malvertising/3727 (cit. on p. 16).

[99] *Security Awareness Survey*. 2017. URL: https://securingthehuman.sans.org/media/resources/business-justification/security-awareness-survey.pdf (cit. on p. 108).

[100] Shashi Shekhar, Michael Dietz, and Dan S Wallach. "AdSplit: Separating Smartphone Advertising from Applications." In: *USENIX Security Symposium*. Vol. 2012. 2012 (cit. on p. 43).

[101] Charles Smutz and Angelos Stavrou. "Malicious PDF detection using metadata and structural features." In: *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM. 2012, pp. 239–248 (cit. on p. 109).

[102] "Spear-phishing email: Most favored apt attack bait." In: *Trend Micro* (2012). URL: http://www.trendmicro.com.au/cloud-content/us/pdfs/security-intelligence/white-papers/wp-

spear-phishing-email-most-favored-apt-attack-bait.pdf (cit. on p. 81).

[103] *Squid*. URL: http://www.squid-cache.org (cit. on p. 41).

[104] Nedim Šrndic and Pavel Laskov. "Detection of malicious pdf files based on hierarchical document structure." In: *Proceedings of the 20th Annual Network and Distributed System Security Symposium*. 2013 (cit. on p. 109).

[105] A Stolerman, R Overdorf, S Afroz, and R Greenstadt. *Classify, but verify: Breaking the closed-world assumption in stylometric authorship attribution*. 2011 (cit. on p. 75).

[106] S.J. Stolfo, S. Hershkop, Ke Ke Wang, and O. Nimeskern. "EMT/MET: systems for modeling and detecting errant email." In: *Proceedings DARPA Information Survivability Conference and Exposition*. Vol. 2. IEEE Comput. Soc, 2003, pp. 290–295. DOI: 10.1109/DISCEX.2003.1194980. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1194980 (cit. on p. 110).

[107] Gianluca Stringhini and Olivier Thonnard. "That ain't you: Blocking spearphishing through behavioral modelling." In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer. 2015, pp. 78–97 (cit. on pp. 50, 75).

[108] Symantec. *Malware Security Report: Protecting Your Business, Customers, and the Bottom Line*. (accessed 1 January 2017). URL: www.

verisign.com/verisigntransition101/files/MalwareSecurityReport.pdf (cit. on p. 16).

[109] Mike Ter Louw, Karthik Thotta Ganesh, and VN Venkatakrishnan. "AdJail: Practical Enforcement of Confidentiality and Integrity Policies on Web Advertisements." In: *USENIX Security Symposium*. 2010, pp. 371–388 (cit. on p. 43).

[110] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. "Design and Evaluation of a Real-Time URL Spam Filtering Service." In: *IEEE Symposium on Security and Privacy* (2011), pp. 447–462 (cit. on p. 76).

[111] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. "Adnostic: Privacy preserving targeted advertising." In: (2010) (cit. on p. 43).

[112] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. "Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network." In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL '03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 173–180. DOI: 10.3115/1073445.1073478. URL: http://dx.doi.org/10.3115/1073445.1073478 (cit. on p. 54).

[113] O. de Vel, A. Anderson, M. Corney, and G. Mohay. "Mining e-Mail Content for Author Identification Forensics." In: *SIGMOD Rec.* 30.4 (Dec. 2001), pp. 55–64. DOI: 10.1145/604264.

604272. URL: http://doi.acm.org/10.1145/604264.604272 (cit. on p. 74).

[114] *Verizon 2017 Data Breach Investigations Report*. Accessed: 2017-01-30. URL: http://www.verizonenterprise.com/verizon-insights-lab/dbir/2017/ (cit. on p. 5).

[115] *VirusTotal Statistics*. 2017. URL: https://www.virustotal.com/en/statistics/ (cit. on p. 113).

[116] Helen J Wang, Chris Grier, Alexander Moshchuk, Samuel T King, Piali Choudhury, and Herman Venter. "The Multi-Principal OS Construction of the Gazelle Web Browser." In: *USENIX security symposium*. Vol. 28. 2009 (cit. on p. 43).

[117] Jianyi Wang and Kazuki Katagishi. "Image Content-Based "Email Spam Image" Filtering." In: *Journal of Advances in Computer Networks* 2.2 (2014), pp. 110–114 (cit. on p. 76).

[118] Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Zhang Min, and Xiaotie Deng. "Detection of Phishing Webpages Based on Visual Similarity." In: *WWW*. 2005 (cit. on p. 42).

[119] L. Zeltser. *Malvertising: Some Examples of Malicious Ad Campaigns*. (accessed 1 January 2017). URL: http://blog.zeltser.com/post/6247850496/malvertising-malicious-ad-campaigns (cit. on p. 16).

[120] Bing Zhou, Yiyu Yao, and Jigang Luo. "Cost-sensitive three-way email spam filtering." In: *Journal of Intelligent Information Systems* 42.1 (Feb. 2014), pp. 19–45 (cit. on pp. 6, 76).