DETECTION OF MALICIOUS WEB PAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMRE SÜREN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INFORMATION SYSTEMS

SEPTEMBER 2014

Approval of the thesis:

DETECTION OF MALICIOUS WEB PAGES

submitted by **EMRE SÜREN** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

Prof. Dr. Nazife Baykal Director, **Informatics Institute**	________________
Prof. Dr. Yasemin Yardımcı Çetin Head of Department, **Information Systems**	________________
Assoc. Prof. Dr. Sevgi Özkan Yıldırım Supervisor, **Information Systems, METU**	________________

Examining Committee Members:

Prof. Dr. Soner Yıldırım Computer Education and Instructional Technology, METU	________________
Assoc. Prof. Dr. Sevgi Özkan Yıldırım Information Systems, METU	________________
Assoc. Prof. Dr. Aysu Betin Can Information Systems, METU	________________
Assoc. Prof. Dr. Pınar Karagöz Computer Engineering Department, METU	________________
Assist. Prof. Dr. Tuğba Taşkaya Temizel Information Systems, METU	________________

Date: 03.09.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Emre SÜREN

Signature :

ABSTRACT

DETECTION OF MALICIOUS WEB PAGES

Süren, Emre
M.S., Department of Information Systems
Supervisor: Assoc. Prof. Dr. Sevgi Özkan Yıldırım

September 2014, 50 pages

Cyber-attacks have been shaking the virtual world and malicious web pages have become a major weapon for Internet crimes. They host a number of malicious contents; such as spam, phishing, and drive-by download. Drive-by download technique exploits the victim’s machine and downloads a malware without any notice or consent. After infection, victim’s private data is stolen or encrypted and even worse the compromised machine is instrumented to mount further attacks. To this end, researchers have focused on protecting the Internet visitors.

Previous solutions were blacklisting and static heuristics. Today the most remarkable suggestions for detecting malicious pages involve static and dynamic analysis techniques. It is known that, static analysis shows significant performance but poor accuracy and dynamic analysis performs slowly but brings notable detection rate. Effective and lightweight detection approach should be deployable for real-time environments, overcome known evasion techniques, and be able to detect undiscovered (zero-day) exploits.

This thesis analyses how to detect malicious pages efficiently in an automatized fashion. A feature set is built by revealing characteristics in malicious pages and machine learning techniques are utilized. Respectable and freely available datasets are used in the experiments. The detection rate (97.5%) achieved by the application of static analysis is compared with the state of the art systems and the designed system is on par with most methods.

Offered approach could be leveraged as a stand-alone detection system or utilized as a pre-filter for dynamic methods according to the importance and sensitivity of the mission.

Keywords: malicious web page, static code analysis, honeyclient, cyber-attack, web security

ÖZ

ZARARLI WEB SAYFALARININ TESPİTİ

Süren, Emre
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Doç. Dr. Sevgi Özkan Yıldırım

Eylül 2014, 50 sayfa

Siber saldırılar sanal dünyayı sarsmaya devam ediyor ve zararlı yazılım bulaştıran web sayfaları Internet suçlarının en büyük silahları arasında sayılıyorlar. Bu sayfalar istenmeyen, oltalama ve izinsiz yükleme gibi birçok türden zararlı içerik barındırmaktalar. Popüler bir teknik olarak, izinsiz yükleme, önce hedef sistemdeki bir zafiyeti istismar eder, ardından kullanıcının haberi ve onayı olmadan zararlı yazılım kurulumu yapar. Bilgisayar enfekte edildikten sonra, siber saldırgan, kurbana ait dokümanları çalmakta ya da şifrelemekte, fakat daha kötüsü ele geçirilen bilgisayar siber saldırılara alet edilmektedir. Bu sebeplerle, araştırmacılar Internet ziyaretçilerini korumaya odaklanmışlardır.

Önceki çözümler kara liste ve statik sezi yöntemleriydi, bugün, zararlı web sayfalarını tespit etmede iki yaygın teknik kullanılmaktadır; statik ve dinamik analiz. Statik analiz yöntemlerinin yüksek performans ve düşük tespit oranı, dinamik analiz metotlarının ise düşük performans ve yüksek tespit oranı sunduğu bilinmektedir. Etkili ve çevik bir tespit sitemi; gerçek ortamda çalışabilir, bilinen atlatma tekniklerine karşı dayanıklı ve sıfır gün zafiyetlerini tespit edebilecek bir kapasiteye sahip olmalıdır.

Bu tez çalışması, zararlı web sayfalarının otomatize bir sistemle, etkili olarak nasıl tespit edilebileceğini incelemiştir. Zararlı statik kod parçaları araştırılarak bir karakteristik özellikler listesi elde edilmiş ve makina öğrenme teknikleriyle tespit üzerine uygulama yapılmıştır. Veri olarak ücretsiz olarak erişilebilecek muteber kümeler kullanılmıştır. Statik analiz teknikleriyle yapılan deney sonuçlarında elde edilen %97.5 seviyesindeki tespit oranı var olan sitemlerle karşılaştırılmış ve birçok metotla başa baş olduğu görülmüştür.

Yapılmak isten işlemin kritikliğine göre, önerilen yaklaşım tek başına bir tespit sistemi veya dinamik analiz sistemleri için bir ön filtre olarak kullanılabilir.

Anahtar Kelimeler: zararlı web sayfası, statik kod analizi, istemci bal küpü, siber saldırı, web güvenliği

To my family

ACKNOWLEDGMENTS

I would like to thank my supervisor Assoc. Prof. Dr. Sevgi Özkan for her conduction of the thesis course. I would like to thank Assoc. Prof. Dr. Aysu Betin Can for her kind guidance and revisions at the buzzer of the thesis submission. I would like to thank Assist. Prof. Dr. Tuğba Taşkaya Temizel for her final revisions and corrections.

I would also like to thank respected members of the examining committee for their valuable comments and suggestions.

I would like to express the deepest appreciation to VirusTotal for their support to complete experiments in the thesis.

Finally, I am also grateful to my family for their moral support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
AUC	Area under the curve
CAPTCHA	Completely Automated Public Turing
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DDoS	Distributed Denial of Service
DLL	Dynamic Link Library
DNS	Domain Name System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IP	Internet Protocol
RAM	Random Access Memory
ROC	Receiver Operating Characteristic
SEO	Search Engine Optimization
SQL	Structured Query Language
SVM	Support Vector Machines
TDI	Transport Driver Interface
URL	Uniform Resource Locator
XSS	Cross Site Scripting

CHAPTER 1

INTRODUCTION

Malicious page results from search engines (e.g., Google, Bing and Yandex) on trending topics and malicious links shared on social media (e.g., Twitter, Facebook and Google+) are on the rise. Moreover, malicious content injection to legitimate web sites has become a daily issue. In recent years, a significant increase in client-side attacks has been seen, and today, cybercriminals prefer to attack clients rather than servers. Individuals' private data and critical institutional information are more attractive than the vintage denial of service attacks for adversaries. As a result, malicious web pages are shown as one of the most disruptive security threat on the Internet, [1]–[8].

1.1 Motivation and Goals

Primary goal could be improving safety of web browsing. Internet is indispensable for most of the people and organizations, but it is rapidly getting dirty. Enhanced protection against the threats could contribute to secure and privacy sensitive Internet. In addition, Internet has become a wild area where criminal activities run, however very little information is known about malicious web sites and their owners. Therefore, revealing cybercriminals could be useful for authorities.

Gaining knowledge on the distribution of the malicious web sites, such as their geo-location, IP address ranges, and URLs could provide to mark danger zones of the Internet. Moreover, learning advancements in attacking techniques could sustain the current detection approaches against evasion attempts. Furthermore, this research field shed light to inspect malware distribution networks and their command and control servers to gain deep insights.

Researchers have recently started studying on detection of malicious web pages. The main objective is to detect drive-by download sites, find undiscovered (zero-day) exploits and learn evolution on the attack techniques.

If a security system informed visitors about the malicious web pages when browsing the Internet, big part of the problem could be eliminated. So, the security researchers have addressed this issue by developing blacklist services, [9], [10]. However, response from adversaries has arrived quickly and they have already changed the IP and URL addresses. Likewise, signatures have been produced [11]–[13] for malicious codes, but obfuscation has come to adversaries' rescue.

To this end, researchers tend to analyze the web page content. Static analysis of the client-side codes strengthened the detection mechanism; accordingly bring more accurate detection when compared to signatures and heuristics [14]–[21]. Again, hackers found several ways to circumvent the traps. Then, researchers have proposed dynamic analysis approaches [22]–[32], but run-time overhead forced them to take a step back and think about hybrid systems. It is obviously seen that, the rivalry continues and does not seem to be finished.

This master's thesis focuses on static analysis as a complementary approach for the dynamic analysis. The motivation is that filtering candidate pages quickly and submitting the pages which worth more inspection to rigorous analyses. While avoiding the performance overhead, getting accurate detection results is the primary goal.

This thesis presents a machine learning based detecting methodology. First a feature set is identified, then six different classifiers are used and detection rate of each one is evaluated. To this end, we have brought together three groups of features used for detecting malicious web pages in the literature. These groups of features are URL, HTML and JavaScript. Bringing these three class of features provide several advantages as discussed in Chapter5. Then we have downsized the feature list by selecting the ones that have high impact on discriminating malicious web sites using a feature selection algorithm. Based on this features we used Tree, Regression and Bayesian based six classifiers. Meanwhile, a dataset is built based on a well-respected blacklisting service. The experiments show that 97.5% out of the pages could be successfully classified as malicious, totally benign web pages were filtered quickly without any miss.

1.2 Contributions

This thesis makes the following contributions.

- A machine learning based filtering system is implemented.
- Using freely available online resources, a new malicious and benign dataset is built.
    - Beyond gathering entries from highly respected malicious URL repositories, each one is verified once more with a second online tool.
    - Benign dataset is built based on most used URLs on the Web at the time of the experiments.
    - The feature set of a known and highly used (for blacklisting purposes not for academic experiments) dataset is newly revealed by this study.
- A list of features are gathered from literature and used in this study. To the best of our knowledge, no single study has used all of these features yet.
- Three classes of features increases detection coverage and provide solid analysis.
    - Three groups of features are URL, HTML and JavaScript;
    - If the web page content is not accessible for any reason and only the URL is known, analysis could be still possible since URL features are able to detect average maliciousness.
    - Evasion attempts by benign looking URLs could be prevented by the page content analysis which is provided by the other two groups of features HTML and JavaScript.

The experiments show that 97.5% out of the pages could be successfully classified as malicious, totally benign web pages were filtered quickly without any miss. The false negative and false positive rates are also very close to existing filtering methods.

1.3 Organization

This thesis is organized as 5 sections: Chapter 2 gives brief information about malicious web pages to build a background, Chapter 3 provides comprehensive description about the detection techniques, Chapter 4 mentions related work, Chapter 5 describes the suggested approach in detail and experiments are evaluated, in Chapter 6 the study is summarized, future works are addressed and with the final thoughts it is concluded.

CHAPTER 2

BACKGROUND

Internet has become a virtual world where housing for real threats and variety of criminal attempts. Spam advertisements (e.g., illicit and counterfeit products) are out, but watering hole[1] and spear phishing[2] (e.g., identity theft, financial frauds, password stealing) is in, of course, malware propagation (e.g., trojan downloaders and so-called drive-by download[3]) is the most prevalent case. Becoming a victim is just one click away; cybercriminals drive victims by social media, search engine results or e-mail. Then, malicious web pages infect victim machine when visited and compromise the system to mount future attacks [1]–[8].

2.1 Web-based Client-side Attacks

Cybercriminals leverage shares in social media, search engine results and e-mail contents and attract victims by only a link. Malicious codes are sent to client's web browser as a part of the response after the attacking web page is requested. This scene is also known as web-based client-side attack. Particularly, drive-by download is the most prevalent attack type among the choice of cybercriminals [1]–[8].

2.2 Malicious Web Pages

A web page that exploits vulnerabilities of the web browser or its plug-ins to launch attacks when visited by an innocent victim is called as malicious.

A malicious web page has three core attributes: exploit code, techniques to hide exploit code, and mechanisms to dispatch exploit code. Firstly, significant portion of the exploits are designed for certain type of vulnerabilities (e.g., Java, Flash and ActiveX). After that, obfuscation techniques are applied to defeat some type of detection mechanisms (e.g., signatures and static heuristics). Finally, inclusion via HTML (e.g., src attribute of the tags)

[1] Cybercriminals inject an exploit into the website where is frequently visited by the targeted company employees.

[2] Cybercriminals spoof an e-mail as sent by a legitimate entity where attempting unauthorized access to sensitive information.

[3] Visited malicious web site automatically downloads and installs a malware to the device without any consent or even any knowledge.

or redirection via HTML (e.g., HTTP 302 responses) or redirect via JavaScript (e.g., location() method) is implemented to meet the visitor with the exploit.

In this research, focus is limited on malicious pages itself only; however malicious pages are just a part of a big puzzle. They take part in malicious networks, also known as malware distribution networks [33].

2.3 Life-cycle of Malicious Web Pages

There are four prevalent phases in the lifetime of malicious web pages [33]: registering new domains, luring victims, exploiting victims' environment, and remotely controlling them are the four common steps.

2.3.1 Generating Pages

First of all, a cybercriminal needs to publish malicious pages on the Internet to infect the victims' system with malicious content. Attackers use two common techniques to produce malicious pages. Owning web sites is one of the simplest method which is usually operated by exploitation kits [34]. Therefore obtaining domain names and generation of malicious pages are handled in an automated fashion meaning that they have some similarities inherently. Chapter 4 refers to this passage while detecting malicious pages.

The other method, which is also an efficient way, is compromising benign web sites. Several techniques are reported to be used to increase attack vectors [1]–[8], [35]–[37]. Adversaries usually spider the Internet to find vulnerable hosts, especially through Google Dorks [38], [39]. If hackers exploit vulnerabilities on the web server, application server, database server or application code (client or server side), they inject malicious contents, particularly via Cross Site Scripting (also known as XSS), SQL Injection or file inclusions. There is also a top list for the major critical vulnerabilities inspected in web sites [40]. In addition, most known attributes of the injected contents are obfuscation (e.g., obfuscated JavaScript codes) and redirection (e.g., HTTP 302 responses).

Auto-generated malicious pages and infected pages are both build an organized structure so-called landing pages of a malware distribution network. Architecture design of the networks are very complex to avoid detection of the exploit servers, command and control servers and surely cybercriminals [33]. Those networks are also very large in size to maximize chance to trap more victims.

2.3.2 Finding Victims

After publishing malicious web pages on the Internet, cybercriminals must get victims to visit the malicious contents and realize exploitation [36], [37]. A number of tools exist which are spams, advertisements, shared links in social media and search engine results [1]–[8].

Spam is a traditional technique to lure victims. For example, spam e-mails may contain links behind an embedded image or an attachment which appears as a pdf file but actually is a harmful executable. Social engineering tricks in e-mail contents may ease causing victims to be caught in a trap.

In addition, legitimate sites enrich the features with or want to make money from third party contents. As an example, advertisements which refer to malicious web pages are popular more recently [37]. Intruder advertisers could be positioned in the advertisement redirection chain, usually in ad networks which has bad reputation, and abuses innocent web site operators.

Moreover, forums, blogs, and social networking sites are also abused to redirect victims to malicious web pages. For instance, attacker contributed entries may include script codes which makes unnoticeably malicious cross requests. Moreover, directly sharing malicious links in social media could be so easy but more effective [37].

Lastly, search engines are also misused nowadays by adversaries in order to find victims. Trending search terms are leveraged to poison search results (e.g., black hat SEO campaigns) which comprise a basis for malicious web pages [1]–[8].

2.3.3 Exploitation

Most critical part in the life of malicious page is the success when came across with the victims. Granting access on victims' devices is its sole and golden purpose. Attackers usually design malicious codes for specific vulnerabilities in web browsers or its plug-ins. While a victim visiting a malicious page, rather than blindly trying to exploit the target system, malicious codes attempt to enumerate the vulnerabilities of the victim's browser and its plug-ins, and then determine dispatching related exploit in order to compromise it. After reconnaissance, if a related vulnerability is found, attacker will release the exploit code, if not, they disguise them to avoid from possible detection. Beside vulnerability scanning, they also apply smart obfuscation mechanisms to hide malicious codes which provides evasion of detection approaches [34], [35]. There is also some sophisticated and become famous exploitation mechanisms, one of them is so-called drive-by download.

Drive-by Download

Right after exploitation, if the malicious page triggers downloading and executing a malware binary without revealing any suspicion, it is called as drive-by download. Malicious web pages that trigger drive-by download attacks to install malware on victims' devices (e.g., personal computer, tablet, smart phone etc.) have become the most prevalent technique for the propagation of malicious code [3]–[6].

2.3.4 Command and Control

Beyond getting private data from infected victim's system (e.g., log-in credentials), adversaries could completely take over the victim's machine in order to remotely command and control it. Compromised machines are called as zombie. After taking over considerable amount of zombies, they reach a large-scale network, so-called botnet [33]. In reality botnets are remotely controlled in the background by the cybercriminals. Bot herders make zombies an instrument for their evil missions, especially used for massive DDoS attacks and spam campaigns, by remotely commanding them [3]–[5].

CHAPTER 3

DETECTION TECHNIQUES AND SYSTEMS

Researchers have introduced several methods to detect malicious web pages with aim of preventing exploitation of victim systems. Blacklist [9], [10], heuristic [11]–[13], static analysis [14]–[21], and dynamic analysis [22]–[32] are the four cornerstones.

Traditionally, two common steps are involved in conducting detection of malicious web pages. At first, a crawler is employed to collect candidate URLs. Then, page is analyzed either the static or dynamic analysis techniques.

Today, it can be said that a third step could be sensible. Some researchers [41]–[45] have a high tendency about using two phase analysis. In this way, after page content is obtained, a filtering operation is applied to estimate suspicious web pages which require more deep analysis. Finally, reduced amount of web pages are submitted to high cost tools (e.g., honeyclient) for more elaborative examination.

Filtering approach is very important because it reduces the number of web pages to be inspected by resource intensive tools. Simply the crawler encounters many pages on the Web that are clearly benign, which explains why the number of candidate URLs is excessive.

Detection of malicious web pages is an offensive security mission, namely researchers make an effort to find those, however there is no obstacle about using these approaches in a website-level service where Internet visitors manually submit URLs (e.g.,VirusTotal [46]), client-level where browser plug-ins [47] are automatically intercepts URLs, network-level where front-end proxy servers (e.g., like web filtering appliances [48]) transparently intercepts URLs, or if we accept most Internet visitors use search engines before accessing a web page, search engine-level deployment [9] could make more sense. Of course, in those cases there is no need for a crawler, since URLs are submitted by real users or intercepted automatically by the tools which are located between the user's browser and the destined web page.

Researchers use common tools rather than developing them from scratch. Since, approved success (by the crowd community) strengths the hand to prefer existing mature applications. In addition, it is clear to save time and effort. Following chapters mention about tools when it is needed but to be comprehensive and to keep the consistency, most known tools and their specific purposes are listed in here all together.

In brief, in this section, in addition to detection techniques, also the tools that are used in this study and commonly used tools in this field are given.

3.1 Blacklist

Security communities or vendors publish an updated list of known malicious URLs, domain names, and IP addresses which are detected by static or dynamic analysis techniques, honeyclients, or custom analysis methods. For example, the well-known Google Safe Browsing service [9], intercepts all request while visitors browsing in the Internet to alert them whether the requested URL exists in the blacklist, the mentioned service is also embedded by Firefox. Another popular blacklisting is maintained by McAfee SiteAdvisor [10] which rates safety of web pages and search engine results and shows the ratings to its users before rendering the page in the browser, for this it provides a plug-in for Firefox and Internet Explorer.

Blacklisting technique is easily bypassed by attackers mainly through changing the IP and domain addresses.

3.2 Heuristics

Heuristics is a kind of static analysis technique which is applied when generating signatures of known attack payloads. After producing the signature, it is used to determine either the page is malicious or not. While inspecting the web page content, if heuristic pattern matches at least one of the signatures, system flags the page as malicious.

Signature detection approach is scalable where web pages are not executed. Not only it reduces the resource requirements, but also it improves performance. It is mainly used by antivirus products (e.g., ClamAV [11]), intrusion detection systems (e.g., Snort [12]), and web application firewalls (e.g., ModSecurity [13]) to quickly scan a web page or HTTP/S responses. However, this approach is responsible for high false positive rate where web pages are misclassified as malicious.

Heuristics technique is bypassed by attackers mostly obfuscating the page content and web request, particularly the payload section. In addition, heuristics fail to detect novel attacks, resulting in zero-day exploits.

3.3 Static Analysis

Formerly, researchers have used only host identity and URL attributes (e.g., lexical or statistical) while performing static analysis. Because not dealing with page content is very fast in terms of time to analysis the sample and also it was occasionally sufficient for detection. However, researchers realized that the limited scope has some drawbacks intrinsically, such that involving only with the URL string may bring performance increase, but attackers are getting to craft benign looking page contents which are statistically indistinguishable from benign samples that may result in evasion. In addition, compromised benign web pages are pretended as another concern. On the other hand, today it has become nearly impossible to detect new generation malicious web pages without leveraging the clues in the page content.

Trends are changed recently; researcher has just become concerned with the page content. Basically, static analysis techniques, [14]–[21], inspect web page content without executing it in a real browser. Namely they do not consider dynamic contents which are came up in run-time, such as output of eval() method in JavaScript. Base of static analysis is that the statistical distribution of features in benign pages is different from the malicious pages.

Current static analysis techniques are mostly implemented via machine learning. Four common steps are followed by researchers. Firstly, a crawler is utilized to download web page content, likewise the all other approaches.

After obtaining the page content, discriminative features are extracted from the HTML and JavaScript codes. Host identity (e.g., whois and DNS records) and URL string could also remain as features, since they enable analysis without page content. Moreover, if it is considered that malicious web pages are quickly disappeared to avoid from detection, it will continue to be good option. Reputation metadata (e.g., Alexa rankings [49]) and social media shares (e.g., Twitter, Facebook, Google+) are lately came into use as features.

As a third step, feature values are encoded to feed into training algorithms to build classifiers. Potential attributes may be determined by experience and static heuristics, however, using all features could not get low false positive and high true positive, so feature selection is shown up as a requirement. In order to find out most suitable feature set, feature selection algorithms (e.g., Information Gain) are utilized. Moreover, features are divided into classes. Feature grouping is used to increase accuracy, also implicitly it provides prevention for evasion, and moreover it enables evaluation of feature groups to determine most valuable class.

At last, machine learning algorithms (e.g., Decision Trees, Bayesian Classifiers, Logistic Regression, and SVM etc.) are applied to classify unknown samples. Instead of binary classification mechanism, researchers show tendency to use weighted algorithms (e.g., Confidence Weight) in these days to increase true positive rate. They use a number of machine learning algorithms and combine the results by applying the weights. The major contribution of those methods are enabling of threshold and trade-off mechanisms. Furthermore, not relying only best algorithm but combination of algorithms makes evasion difficult. Besides supervised algorithms, researchers try online learning algorithms [16]–[18], since batch learning techniques have problems to adapt continuously changing features.

Disadvantage of this approach is that it is hard to detect attacks that require execution of candidate page. In addition, detection of brand new malicious codes also known as zero-day are hardly difficult for static analysis. Since this approach knows existing malicious families thereby can detect similar variants. Moreover, obfuscation still remains as another concern. As a result, static analysis mostly preferred for filtering purposes [41]–[45].

3.4 Dynamic Analysis

Dynamic analysis techniques [22]–[32] inspect the run-time features. After web page is rendered, dynamic effects are measured in two ways. First one is operating system level analysis where state changes are inspected in underlying operating system. State changes in file system, registry and running process are observed. Creation of a new file into system folder, addition of a new startup key entry into registry and starting new process are one of the most common symptoms. In addition, network connection and physical resource consumptions (e.g., CPU and RAM) could be observed. For more deep analysis, behaviors are monitored, such as; hooks to native API, DLL functions, and TDI are established to monitor all activities. The other one is browser level examination where dynamic function calls are captured. For instance, invocation of vulnerable methods of particular client-side technologies (e.g., Java, Flash, and ActiveX) which result in remote code execution, passing arbitrary length of parameters for plug-in methods to cause buffer overflow, and crafting application specific payloads to exploit browser. Shortly, quite detailed examination enables dynamic analysis to detect novel (zero-day) attacks.

Researchers have developed a few detection variants additionally. Sandboxing is a type of dynamic analysis, in this approach, critical actions are logged for pattern matching or machine learning techniques [25], [26]. Emulation is one another dynamic analysis method. More detailed information could be obtained by inspecting inner workings of the malicious activities [27], and it usually focuses on specific type of attacks, most known is heap spraying code injection [28]. Both of the approaches could be applied in operating system or browser level. Particularly, researchers prefer sandboxing in operating system level and emulation in browser level.

Attackers could craft web pages that require user interaction (e.g., CAPTCHA or asking simple math question), expect for certain conditions (e.g., IP address belongs to a particular region) or wait a random period of time to take action (e.g., setTimeout() method in JavaScript). Therefore evasion is still possible.

If an attacker tries to penetrate a computer and a researcher wants to learn what he does, quite normally researcher monitors the computer to catch the attacker. This is the effective way, there is no more or less thing to do. Unlike static analysis which is mostly based on estimation, dynamic analysis does not prefer getting suspicious from unusual codes in web page content. Consequently, looking for certain evidences makes respectable accuracy. On the other hand, as it can be clearly seen, dynamic analysis is a time consuming approach, since it waits dynamic actions to be occurred. In addition, it is highly resource sensitive due to operating system or browser requirements or their replicas that should be very similar to real. Therefore, dynamic analysis is a high cost technique meaning that it is not suitable for real-time or large-scale detection.

Static analysis is very fast and cheap but dynamic analysis is very slow and expensive in terms of time and resource consumption, on the other hand static analysis is less valuable but dynamic analysis is very valuable in terms of detection. Combining static features and run-time analysis can promise more scalable and accurate approach which is known as hybrid.

3.5 Detection Approaches

Four analysis techniques have just previously explained. Researchers utilize the mentioned techniques by combining some of them to propose their approach. Leveraging different methods help to increase accuracy and performance, so they build powerful tools that make detection in an automatic manner.

If one consider that designed system operates on the Internet and the sole purpose of the tool is security, requirements will be clearly seen. Every day, millions of new web pages are published on the Web and preventing visitors' machines is only meaningful before infection. Therefore, detection system should race with time by quickly processing large-scale data.

Briefly, two major capabilities of a detection approach should be high performance and accuracy.

A detection system should classify candidate web pages very fast and should be scalable. However, known shortcoming is that quick examinations may result in false positive or false negatives. On the other hand, while performing well, it should have significant accuracy. Hence, detection of novel (zero-day) attacks and preventing from evasion techniques are the keystone elements. In general, deep inspection which shows desired accuracy, highly resource intensive and performing poor. Consequently, over the past two years, researchers are studying on hybrid approaches. Now, state of the art system architectures are discussed.

3.5.1 Honeyclients

Dynamic detection methods are operated in honeyclient systems, also known as client honeypots. Basically, honeyclients are formed from two components which are crawler and dynamic analysis engine. Their interaction type is determined by the inspection level which is either underlying operating system or employed browser. In addition, honeyclients have built-in prevention mechanism for evasion, such as sandbox and firewall.

High Interaction Honeyclient

High interaction honeyclients are fully functional virtual machines (e.g., VirtualBox) where real browsers (e.g., Chrome, Internet Explorer, and Firefox) render the web pages. Logic behind the high interaction honeyclients is that user devices are monitored for anomalous state changes. There are three major sources; state changes to the file system (e.g., creation or alteration of a file), registry (e.g., addition of a new key and value pair), and process (e.g., attaching startup or starting new process). In this approach, high interaction honeyclients are able detect undiscovered (zero-day) attacks. However, state change inspection concept is very expensive and consequently is not scalable. While high interaction honeyclients give significant accuracy, they show incredibly poor performance. Most famous high interaction honeyclients are MITRE HoneyClient [50], Microsofts Honeymonkey [51] and Capture-HPC [52].

Cybercriminals usually target particular vulnerabilities in the web browsers and their plug-ins. Exploitation can be realized only, if the vulnerable components are installed in the targeted system. It is neither possible nor sensible to have a detection system where all available vulnerable components have already installed in order to utilize a full scope high interaction honeyclient to detect all malicious web pages.

Even though real browsers execute web pages, identifying virtualized environment is still possible by attackers. If they realize the situation, they behave harmless and do not reveal the malicious code. Actually, several detection mechanisms use virtualized environments, so they could be defeated.

Adversaries could evade high interaction honeyclients by time or logic bombs. They may delay execution of a malicious code (e.g., setTimeout() method in JavaScript) or wait for a particular condition (e.g., confirming CAPTCHA) instead of immediately triggering the exploit. Therefore, state changes are not available in the first contact, so high interaction honeyclient misclassifies the malicious web page as benign. However after a certain time period, malicious activity is performed successfully. Actually, all detection mechanisms are prone to these types of bombs.

High interaction honeyclients offer an effective but not applicable solution. Since, they are not suitable for large-scale deployment.

Low Interaction Honeyclient

Low interaction honeyclients are limited run-time environment where emulated/virtual browsers (e.g., HTMLUnit) execute the web pages. The main concept of low interaction honeyclients is that HTTP responses from servers are transparently intercepted and inspected for the presence of malicious codes. Low interaction honeyclients consume fewer resources and accordingly are scalable. However examination of HTTP responses, usually utilize static signature or heuristics techniques to make analysis, are not enough to detect unknown (zero-day) attacks, as previously discussed. While showing better performance, low interaction honeyclients give degraded accuracy. Most popular low interaction honeyclients are Wepawet [53], JSUnpack [54], PhoneyC [55], and HoneyC [56]. Wepawet and PhoneyC focus on JavaScrip, and they create a run-time environment to allow rendering of JavaScript codes for tracking execution behavior.

The power of this approach is no requirement for the installation of all vulnerable components. Moreover, contamination from attacks does not matter since the compromised system is not real, so there is no need to turn back.

Low interaction honeyclients also suffer from evasion where attackers could inspect virtualized environment. For instance, some real browsers intentionally do not implement some functionality, if the emulated system supports those, it could be revealed. However, while trying not to support some deprecated functions, if involuntarily some of the required functions are not implemented, then some attacks likely go unnoticed.

Low interaction honeyclients offer an efficient but not applicable remedy. Since, they do not provide very high rate accuracy. Even though they are scalable, level is not sufficient due to high burden of emulation.

3.5.2 Hybrid Approach

Signature or heuristics based static techniques could have non-negligible false negative rates. In other words, some malicious web pages are misclassified as benign, and so they never submitted to dynamic mechanisms. While effective, dynamic analysis techniques require a substantial amount of resources. Therefore, both static and dynamic detection techniques have some drawbacks, to this end combination of those two method came up as a requirement.

Hybrid approach has been proposed [57], [58] to improve performance while performing notable detection. In this way, candidate web pages are quickly filtered before being submitted to resource intensive tools. So, static analysis mechanisms offer scalability, meantime honeyclients suggest accurate detection.

3.5.3 Pre Filters

In reality, pre-filtering approach does not try to detect whether the sample URL is malicious or not. It only tries to estimate suspicious web pages among the candidates, then identified web pages are served to high cost dynamic analysis systems for actual detection. Therefore, it is accepted as a complementary method for dynamic analysis approaches. Shortly, it is suggested over the essential requirement of hybrid approach.

Pre-filters [41]–[45] use only static analysis techniques to be able to fast as fast, so run-time features are all discarded. Any codes in web pages are not executed, but only content is downloaded. Owing the fact that, those tools are also known as fast filters. Moreover, in order to compensate the lack of dynamic features, fast filters leverage machine learning infrastructure to be able to detect yet zero-day samples.

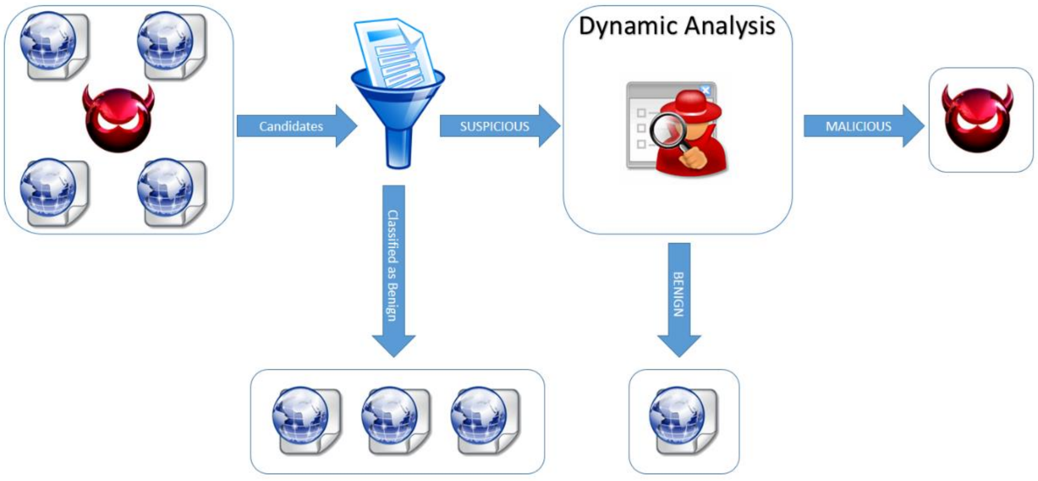


Figure 1 High Level Architecture of Filtering Approach

For the first time Canali et al. defined well a filtering approach called Prophiler, which put static and dynamic detection methods [44] together. Static analysis is performed to quickly classify all candidate web pages either benign or suspicious. Then, potentially malicious web pages are served to dynamic detection tools. By quickly filtering doubtless benign web pages by static methods improve performance and deeply analyzing only the suspicious flags with dynamic methods improve accuracy, as a result the detection time is minimized and expensive resources are not wasted.

Machine learning based static methods usually utilize pure binary classification meaning that researchers are not able to take advantage of trade-off between performance and accuracy. Determining resource usage against detection rate by a threshold is a significant opportunity. Nowadays, researchers have proposed lightweight models (e.g., scoring, costing) to address mentioned two shortcomings.

3.6 Tools

In this field, there are three major classes of tools which can be categorized to crawlers, static analysis tools, and dynamic analysis tools.

A software allows researchers to download a web site from the Internet to their computer is called as crawler or spider. A crawler usually takes a URL address of the web site as input and gets the HTML and JavaScript files from the server to a local directory. Most known crawlers are Heritrix [59] and wget [60]. Moreover, researchers also tend to develop their own crawlers to get rid of integration overhead with the detection system and make the system fully automated.

Static analysis methods extract HTML and JavaScript features. Most popular HTML parsers are Neko [61] and HTMLParser [62]. Frequently used JavaScript interpreters are Rhino [63] and Mozilla SpiderMonkey [64]. In addition, researchers also code their own feature extraction engines due to the same reasons of custom developed crawlers.

Dynamic analysis techniques de-obfuscate JavaScript codes. A common JavaScript debugger is "Mozilla Venkman" [65] and de-obfuscation tool is "JavaScript Deobfuscator" which is an add-on for Firefox [66]. Furthermore, low interaction honeyclients use emulated browser, prior one is HTMLUnit [67].

CHAPTER 4

RELATED WORK

In this section, previous works are evaluated. The viewpoint is that applied analysis techniques, used feature sets, coverage of attacks, and detection effectiveness. As previously discussed, there are three major approaches. Honeyclients are based on dynamic analysis techniques. Static analysis is used for filtering. Some researchers leverage light dynamic analysis in addition to static analysis to be able to use it as a detection system. Others combine filters with honeyclients which is commonly known as hybrid approach.

High interaction honeyclients [50]–[52] which provide real run-time environment in order to bring high accuracy, on the other hand, while consuming excessive resources, they show poor performance. Limited focus and studies in this area make their prior systems relatively old when compared to successor approaches; consequently we discarded evaluation of those high cost tools in this scope.

Low interaction honeyclient systems [53]–[56] were proposed to address performance issues related to high interaction honeyclients. As is known, low interaction honeyclient has a limited run-time environment capability which reduces the run-time overhead. So, one veteran low interaction honeyclient, Wepawet [53], is discussed in detail.

Filtering approach is a trending concept which is based on static techniques in order to show high performance; likewise consuming low resources, it results in inadequate accuracy. Researchers have realized that static features should not be used for detection but for filtering of suspicious samples and then candidates are submitted to high cost dynamic analysis tools. Filtering is clearly the emerging topic where not only pure supervised machine learning techniques are utilized but also some intelligent schemes (e.g., cost or scoring) are currently applied to fuel accuracy. Researchers have concentration over this topic [41]–[45] and advancements have been seen for two years, thus the latest proposed filtering methods by Le et al. and Choi et al. are explained in here.

Machine learning based static analysis is an innovative technique which inspects web page content very quickly. Due to the fact that examination time of dynamic techniques are not comparable to the static techniques, this impressive promise of static analysis excites some researchers. Since, it is very convenient to use in real world for real-time applications. Although detection accuracy of static analysis about malicious samples causes disappointments, some researchers still study to increase the accuracy to be able to use it as a detection system, instead of using it just for filtering. They prefer to leverage slight dynamic analysis, in order to overcome accuracy problem. Scientists have agreed that dynamic techniques should be applied to detect next generation and unknown (zero-day) malicious web pages, accordingly they have shown tendency to emulation methods [22]–[32].

Dynamic contents are executed to reveal particularly the actual malicious codes which are appeared only in and after run-time. In this part, one state of the art powerful approach, BINSPECT [22] which mostly uses static analysis and partly uses lightweight emulation is reviewed in detail.

Several studies are referred with one paragraph; but one significant solution from each group is chosen to review in detail. As a result, four state of the art and most referred studies are described in this context.

4.1 Dynamic Analysis

In addition to pure dynamic analysis, researchers have enhanced the technique by two major sophisticated instruments; emulation and sandboxing.

Cova et al. built Wepawet [53], an emulated JavaScript environment to detect drive-by download attacks. According to the researchers, their anomaly based detection approach has very low false negative rate. It is deployed as a low interaction honeyclient and it is also publicly available as an online service. Anomaly detection has some drawbacks when compared to machine learning based detection systems, such as emulation overhead.

In the next study, Cova et al. announced a novel method [27] which is based on anomaly inspection to detect malicious JavaScript code. Emulated browser renders the dynamic contents to extract features. They categorize the features as redirection and cloaking, de-obfuscation, fingerprinting, and exploitation. Actually, those are the common steps of a client-side attack. According to them some feature groups are required and some of them are only useful. Experiments with 115K pages achieve very significant results.

Dewald et al. developed ADSandbox [26] which is a JavaScript sandbox environment. In this method, system renders JavaScript codes in a separate secure zone and records the predefined actions. Meanwhile logs are served to signature detection engine to inspect malicious activities. One disadvantage is that sandboxing adversely effects performance, in addition it only detects web pages which have malicious JavaScript, however it is known that phishing and malware infection pages may not contain JavaScript.

4.2 Static Analysis with Lightweight Emulation

In this model, web pages are rendered with an emulated browser. But emulation is not used for dynamic analysis, it is just for extracting run-time features and also execution scope is quite limited. For instance; only core components of browser is used, like HTML engine, JavaScript interpreter and CSS parser, to make emulation lightweight. After execution, HTML and JavaScript features are extracted. Other type of features, such as URL and DNS, may also be used. Rest of the application is based on completely static analysis which is performed by machine learning. The power of this approach relies on that statistical distribution of the malicious features diverges from the benign pages. Detection scope of this method covers blended attacks that may also use evading techniques. This holistic view is able to detect known and novel drive-by download, phishing, injection, and malware distribution attacks.

4.2.1 Related Study in Detail

Eshete et al. proposed BINSPECT [22] which uses static analysis by applying machine learning. It accepts the problem as binary classification, namely flagging the sample as malicious or benign. In total, 39 features are used and ten of them are novel. Extracted web page content features, URL string information and social reputation scores are feed into seven learning algorithms in training phase. Both malicious and benign URL samples are used as training data.

In testing phase, unknown URL sample is given to seven models that were previously derived in testing phase by seven classifiers. Each model produces one vote (benign or malicious) and one confidence value. In order to combine these votes and confidences to determine whether the page is benign or malicious, the system uses “Confidence-Weighted Majority Vote Classification” algorithm. Sum of benign votes is multiplied with sum of their confidences and the same operation is done for malicious votes. If benign score is greater than malicious one, unknown sample is marked as benign or vice versa. If scores are equal system marks candidate as suspicious.

The researchers evaluate seven classifiers; according to them, Random Tree provides best model and Naïve Bayes is the worst. They declare that their newly introduced features are improved four of the seven classifiers, but the remaining three is not changed in accuracy. The authors also measure discriminative power of URL, web page content and social reputation feature groups separately. But they realized that all of them together bring better performance. Not only showing performance, but also three classes of feature sets mainly provide a barrier for evasion.

The authors claim that detection accuracy goes beyond 97%. They compare BINSPECT results with Wepawet [40] publicly available analysis service. Wepawet gets 62.61% accuracy, 0.983% false positive, and 0.073% false positive rates and BINSPECT surpasses accuracy, false positive, and false negative rates with 97.81%, 0.0189%, and 0.011% respectively.

An attacker may bypass this approach by evading all feature classes which are URL, web page content and social reputation. As an example; the attacker has to create statistically indistinguishable URL, use obfuscated web page content (especially JavaScript code), and attract too much people to share the malicious URL in the social media. Moreover attacker must evade seven classifiers because of the nature of weighted confidence combination algorithm. Another evasion may be performed by attacking browser plug-ins such as (Java and Flash) because system lacks of emulating browser plug-ins. Finally, disadvantage of the system is minimum 3 and maximum 5 second/page overhead.

One interesting insight is that system does not handle obfuscated JavaScript but achieved detection accuracy shows there is no really need to de-obfuscate JavaScript, lightweight emulation is enough to catch side effects of obfuscation. Not inspecting obfuscated content also improves performance.

This study also contributes quite well to the literature by introducing novel features for identifying blended attacks and effective classification method for evasion attempts. In addition, it brings a new perspective to static analysis.

4.3 Hybrid Detection

Seifert et al. [7] proposed a novel hybrid approach which utilizes both static and dynamic analysis techniques to detect malicious web pages. Static features are extracted from web page content which is leveraged to filter suspicious candidates. Unlike most of the researchers who group their features as URL, page content and JavaScript, their heuristic-based feature classes' focus on exploit, exploit delivery and obfuscation. They train the model by nearly 5K malicious and 16K benign page samples. Their purely machine learning based filter classifies 61K web pages with a hopeful false positive 5%, but a scary false negative 46% rate. After the classification, determined likely malicious samples are submitted to a high interaction honeyclient for verification.

4.4 Filtering Approach

Provos et al. [58] have recognized the requirement for a filter as a first to be able to analyze all the pages on the Web, but they give too little clues about the filter they developed at Google. After that, Seifert et al. introduced the concept [45], they were building such a hybrid honeyclient where they prerequisite filtering to reduce the overhead at dynamic analysis tool by submitting only the likely malicious pages. They extracted common static features from HTTP responses and divided them into three feature groups. They apply J4.8 decision tree algorithm from WEKA as a classifier for filtering and their experiments resulted with very high false alarms. Finally, Canali et al. addressed the major shortcomings about accuracy, and pinpointed the new approach as fast filter, Prophiler [44].

4.5 Malicious Filter

Unlike researchers who prefer to identify benign pages and automatically mark the remainders as suspicious, most of the researchers try to detect malicious samples. Directly detecting malicious web pages may eliminate significant false positive indications. However they are pretty prone to false negatives, since they are not able to detect zero-day samples.

Canali et al. proposed a fast filter called as Prophiler to identify suspicious web pages. Extracted 79 features are grouped into three classes; HTML, JavaScript, and host-based and URL features. Application of purely supervised machine learning techniques achieved quickly discarding benign pages. Making a fast analysis enables the approach to be built as a front-end system for a more knowledgeable but heavy resource consuming detector. Their results show that the system can eliminate more than 85% likely malicious web pages with a very low false positive rate. But experiments resulted with considerable false negatives 9.88%, since they do not give attention to reduce the error rates.

Rather than using pure machine learning for pre-filtering, researchers tried to leverage flexible models. Scoring model does not accept the problem just a binary classification; because pure systems are not able choose the trade-off between performance and accuracy. In this model, before the classification, a scoring operation is performed and at the end a threshold value is applied to be able to make trade-off between number of potential malicious web pages passed through to the detection system (filtering rate) and misidentifying malicious web pages as benign (missing attack rate). Threshold brings the opportunity to put the false negative rate at zero. Although it is a great opportunity, being enabled to eliminate false negatives may not be feasible in every scenario because it could cause incredible false positive alarms.

4.5.1 Related Study in Detail

Le et al. proposed a novel scoring method [42] for pre-filtering. In total, they inspected 52 features and grouped them into four classes of feature which are URL and DNS, HTML, JavaScript, and exploit code content. They evaluated static features of web pages by a feature selection method, Information Gain. Only top three of four feature classes were selected. Four values are calculated minimum, maximum, mean, and median, if a feature appears more than once in a sample web page.

In their research, nine lightweight scoring algorithms are evaluated by them; nearly all of the algorithms are distance measurements. One scorer is selected for each feature group, for URL and DNS feature class Euclidean Distance Normed and for the other three feature classes Manhattan Distance Scaled algorithm performs better.

Then, all four scores are normalized in the range [0-1] to be able to combine them. Combining knowledge from different groups of features could improve accuracy and also bring resistance against evasion.

They also evaluate some combination methods to generate one overall score. Overall score makes easy to apply only one threshold and also to determine whether page is potentially malicious or not. Two types of score combination methods that are classic and dynamic are evaluated. A dynamic combination method is chosen due to its performance. Firstly, previously normalized four scores are fed into machine learning algorithms with and without confidence weight, and Weighted Random Forest classifier is seen as performing better. After that, the scoring model produces one value when classifying suspicious web pages; negative value for benign and positive value for potential malicious web pages. Finally, a threshold value is chosen to determine whether the page is malicious or not.

Factorial experiments are performed to identify effective scoring model. There are four factors in this model. Receiver Operating Characteristic (ROC) curve that shows true positive against false positive rate measures the trade-off while adjusting the threshold in each run. Overall value of area under the curve (AUC) is calculated as the mean of them. Overall AUC is used to compare the performance of classifiers. Largest AUC value points the best performance; during the experiments system achieved maximum 0.984 value.

Contributions are identification of best malicious web page features, better scoring algorithms for those feature groups, reasonable normalization procedure for scorers, and effective score combination algorithm to identify potential malicious web pages that form a novel scoring model. In addition, it enables to make trade-off between false positive and false negative rates.

Shortcomings may be like that; although the efficiency of system is remarkable, it does not use outputs of scoring algorithms to improve accuracy. Classification time of best AUC valued classifier for one page takes 25 msec. but this system just makes an estimation because pre-filters are not final classifiers, they serves to dynamic analysis tools.

4.6 Benign Filter

Some researchers expressed that benign web page identification may promise more accurate results. The underlying fact is that scientists want to avoid false negatives; hence they mark the page when they observe any suspicious activity. Although avoiding from misclassification of malicious pages as benign could enable zero-day exploit detection, it brings incredible false positive alarms overhead. Since, discriminative attributes of the benign web page are not sharply clear than the malicious one for every time. Today, it is known that benign web site operators prefer to apply some obfuscation and escaping techniques to their client-side codes in order to prevent them from being stolen by the Internet pirates or their commercial competitors.

Rather than using pure machine learning for pre-filtering, cost sensitive models become prominent. Costing schema does not handle the problem as solely binary classification; because the lack of false negative error consideration. In this model, before the classification, a multiple times greater value is assigned for a false negative error cost than the false positive error cost.

4.6.1 Related Study in Detail

Choi et al. proposed an efficient costing method for pre-filtering [41]. Proposed method considers false negatives due to misclassification of malicious web pages as benign. In case of any suspicious indication in static analysis process, they mark the page as potentially malicious and forward it to dynamic analysis engine where filtering brings the efficiency. In other words, they do not try to estimate malicious candidates, but filter benign pages and rest automatically marked as suspicious. To apply sensible and smart avoidance from false negative errors, they utilized a cost sensitive method MetaCost.

In total, they inspected 12 features and grouped them into three classes of feature that are HTML, JavaScript, and ActiveX. The features were taken from Prophiler [44]. While they were developing the system they realized that some malicious web pages could be detected by only one class of features and some of them require all. Therefore, sequential filtering should be worked which also increases the efficiency of the filtering.

J48 decision tree algorithm is employed as classifier and MetaCost schema is adopted to reduce the cost of classification. Cost sensitive method allows defining error costs. As is mentioned, aforementioned paper’s aim is trying to diminish false negatives; hence author determines a value for false negative error cost which is 50 times greater than false positives.

Experiments were resulted with 5.8% false negative and 12.5% false positive rates. Researcher sacrificed analyzing redundant samples to misclassifying real malicious pages; however in return, with sequential filtering and utilization of cost sensitive method reduced the web page inspection process in the rate of 80%.

To sum up, following contributions are introduced. Application of costing method is able to reduce false negatives and the logic of sequential filtering enriches the performance. Selection of interested feature subset decreases analysis duration and also increases overall accuracy.

Only shortcoming may be is that they only tried J48 decision tree, other algorithms may be used to improve accuracy some more.

CHAPTER 5

APPROACH

In this thesis, we suggest a filtering method where benign web pages are quickly filtered applying machine learning techniques. It is planned to be used as a front-end mechanism for dynamic detection methods (e.g., honeyclient). Since, dynamic analysis makes quite accurate detection by a deep inspection; accordingly, they consume high level resources and much time. In addition, if we inherently accept that, today, most of the web pages on the Internet are clearly benign, so trying to test every web page with a dynamic analysis tool obviously means wasting of resources. If we introduce a fast filtering approach to eliminate undoubtedly benign pages and then submit just suspicious candidates to high cost tools; not only the resources will be saved, but also more faster results will be obtained; consequently considerable amount of resource required tools will be more economically consumed. Therefore, how the number of web pages to be inspected by the dynamic analysis mechanisms is decreased successfully, the detection performance will increase proportionally.

This chapter also explains that how researchers collect data to train and test their systems to make experiments, evaluations and carry out analysis.

5.1 Features

Designed system leverages machine learning so first step is identifying feature set. A web page can be identified by its URL address and can be characterized by its content. So the resources for the attributes of a web page are those.

The features used in this study were previously recognized by [16], [44] and also partly used in a number of studies where researchers also applied machine learning infrastructure to detect malicious web pages. In addition, web page content could be also separated into a kind of classes to make specific analysis. One way is to divide them according to the used technologies (e.g., HTML, JavaScript, Flash, ActiveX, Java Applet etc.). Thus, entire set of features are preferred to group into three sub classes which are URL and host-based, HTML, and JavaScript feature groups.

5.1.1 URL and Host-based Features

A URL address is formed from three parts which are hostname, path and query. So to be able to extract sensible features, the mentioned logical characterization is taken into consideration.

Table 1 Sample: Logical Parts of a URL

URL	sub2.sub1.sampledomainname.tld/path1/path2/page.htm?param1=val1&param2=val2
Hostname	sub2.sub1.sampledonaminname.tld
Path	/path1/path2/page.htm
Query	?param1=val1&param2=val2

Firstly, length of the hostname, path and query are measured. Then, tokens in each part are counted and maximum and average values are calculated. Moreover, a number of statistics are taken in order to have a slight anomaly analysis.

Five attributes are assessed to learn confusion level;

- As previously mentioned, hackers usually prefer to generate URL addresses in an automated fashion, and carelessly choosing random letters from the alphabet may result increase in consonant letters, hence low ratio of vowel characters in URL may indicate suspicious situations.
- Adversaries sometimes need to hide file extensions (e.g., exe) from visitors, so making long length URLs could fill the visible parts of the browser's URL bar, therefore looking for slashes ('/') may provide an evidence for unusual choices for path part.
- In addition, for a similar aim, space (' ') is checked for weather there is a concealment in both path and query parts.
- Miscreants may also apply URL encoding which makes the URL unreadable for human eyes, so percentage is leveraged to learn the masking density for both path and query parts.
- Question mark ('?') is generally expected only for once in a query part of a URL, so multiple usage could be treated as anomaly in query.

Finally, instead of getting domain addresses by paying to hosting providers, attackers may take a cheaper way; hence usage of IP address may be a symptomatic malicious choice in some cases.

The extracted URL features with related studies are given in Table 2.

Table 2 URL and Host-based Features

MaxPathTokenLength	[14]
LengthOfPath	[22]
LengthOfURL	[16]
SlashCount	[22]
AvgPathTokenLength	[14]
LengthOfQuery	[22]
QuestionCount	
EqualsCount	[22]

DotCount	[16]
PathTokenCount	[14]
DomainTokenCount	[14]
RatioOfVowelCharactersInURL	[42]
AvgDomainTokenLength	[14]
IsItIP	[16], [44]
LengthOfHostname	[16], [22]
SpaceCount	[22]
MaxDomainTokenLength	[14]
PercentageInURLCount	

URL features could be useful while detecting malicious pages; particularly pages which are not currently available. However, it is not possible to detect a great deal of emerging malicious pages without their content, since attackers are getting to make benign looking URLs. Moreover, security aware legitimate web site developers have been designing more complex and longer length URLs to increase entropy for some security reasons (e.g., Brute Force attacks) and it sometimes causes a benign URL to have typical malicious features. Thus, in order to make a robust detection, web page content analysis becomes mandatory.

5.1.2 HTML Features

HTML features are extracted from web page contents. The length of the page, number of included resources (e.g., via “src” or “href” attributes), and known suspicious HTML tags (e.g., script and iframe) are the characteristics of a malicious web page.

Attackers establish some bypass and hiding techniques to make analysis difficult. They frequently apply obfuscation methods to avoid detection. In this way, the number of characters reaches incredible amount. Therefore, page size related attributes could be a good starting point. In addition, adversaries prefer to store malicious codes in shared servers and remotely include those resources into their web pages. In this case, source attributes in some HTML elements are made an instrument. Moreover, cybercriminals often leverage zero sized elements (e.g., iframe) to disguise them which trigger the malicious activity while browsers render the page. Hence, counting invisible elements which is witnessed again and again in malicious behaviors should be considered.

Extracted HTML features with related studies are given in Table 3.

Table 3 HTML Features

MedianHREFvalue	[42]
MaxHREFvalue	[42]
#anchor	[43]
LengthOfPage	[15], [41]
#script	[42]
#html	[15]
MedianSRCvalue	[43]
MaxSRCvalue	[43]

#iframe	[43]
#iframe0	[15]

HTML features could be beneficial while detecting malicious pages, but nowadays attackers tend to develop highly complex codes intrinsically, rather than artificially applying third party techniques. Also, they want to take advantage of current web technologies (e.g., AJAX), so they usually use scripts. Therefore, JavaScript should be taken into account.

5.1.3 JavaScript Features

JavaScript features are also extracted from web page contents. The characteristics of malicious JavaScript codes were defined in previous study [15] where they determined 154 built-in functions.

Some JavaScript functions which are related to encoding, escaping, concatenation and evaluation are made use of to conceal malicious codes. Occurrence frequency of the listed functions could be good evidence. In this thesis, only the usual suspected and simple functions are used; they are given in Table 4.

Table 4 JavaScript Features

#plus	
#PercentageInPage	
#setTimeout	[22]
#createElement	[22]
#escape	[22], [41]
#unescape	[22], [41]
#encodeURIComponent	
#search	[15], [22]
#exec	[22], [41]
#String.fromCharCode	[22]
#href.replace	[43]
#link	[15], [22]
#eval(String.fromCharCode	[22]
#.js.	
#eval	[41]

In the final feature list, each feature group has specific detection capabilities as discussed above. Although, previously proposed features [15], [16], [44] refined and only limited number of features are chosen, in the following part, the power of the each feature and feature subsets are measured in a different way, so quantitative results helped to refine once again.

5.2 Model

This thesis presents a machine learning based static elimination method in order to be a complementary part for dynamic analysis systems, since efficiently revealing malicious web pages requires a filtering approach. Firstly, features are extracted from URL and host data, and then web page content features are extracted and dived into two groups as HTML and JavaScript. Those three classes of features are evaluated with the "Information Gain" method to select most suitable features for the training data (see Section 5.3.1). Selected features and their rankings are given in Table 5, Table 6, and Table 7.

5.2.1 Feature Selection

Table 5 Rankings of URL and Host-based Features

0,2902	MaxPathTokenLength
0,1762	LengthOfPath
0,1536	SlashCount
0,1497	LengthOfURL
0,1385	AvgPathTokenLength
0,1114	LengthOfQuery
0,1114	QuestionCount
0,1057	EqualsCount
0,0978	DotCount
0,0854	PathTokenCount
0,0644	DomainTokenCount
0,0635	RatioOfVowelCharactersInURL
0,0520	AvgDomainTokenLength

URL and host-based feature group supports the detection operation with a very low knowledge. Due to the fact that URL has quite limited data, weak contribution could still be acceptable.

Table 6 Rankings of HTML Features

0,960	MedianHREFvalue
0,960	MaxHREFvalue
0,902	#anchor
0,714	#script
0,709	LengthOfPage
0,430	MaxSRCvalue
0,239	MedianSRCvalue
0,208	#iframe

HTML features seem to bring intelligence to the detection approach. First three attributes nearly determines the classification by itself. Having more than one high quality feature may also strengthen the detection approach. While one feature quite fits to an algorithm and system gives significant results, the others may not benefit from it and system produces unsatisfactory outcomes. Hence, mentioned potential issue is resolved by multiple valuable features.

Table 7 Rankings of JavaScript Features

0,5678	#plus
0,2843	#PercentageInPage
0,1408	#setTimeout
0,1139	#createElement(script)
0,0833	#escape
0,0672	#unescape
0,0519	#encodeURIComponent
0,0359	#search
0,0307	#exec
0,0152	#String.fromCharCode

JavaScript features contributing in a moderate level. When compared to dynamic studies which mostly rely on JavaScript features, the contribution level seems weak. However, the extracted features in here are very simple and few. Therefore, those contribution measures could make sense accordingly.

Table 8 Rankings of All Features

0,9595	MedianHREFvalue
0,9595	MaxHREFvalue
0,9022	#anchor
0,7136	#script
0,7089	LengthOfPage
0,5678	#plus
0,4295	MaxSRCvalue
0,2902	MaxPathTokenLength
0,2843	#PercentageInPage
0,2388	MedianSRCvalue
0,2079	#iframe
0,1762	LengthOfPath
0,1536	SlashCount
0,1497	LengthOfURL
0,1408	#setTimeout
0,1385	AvgPathTokenLength
0,1139	#createElement(script)
0,1114	QuestionCount
0,1114	LengthOfQuery
0,1057	EqualsCount
0,0978	DotCount
0,0854	PathTokenCount
0,0833	#escape
0,0672	#unescape
0,0644	DomainTokenCount
0,0635	RatioOfVowelCharactersInURL
0,0520	AvgDomainTokenLength
0,0519	#encodeURIComponent
0,0359	#search
0,0307	#exec
0,0152	#String.fromCharCode

When ranking operation is done without grouping features, similar gain values are obtained. In conclusion, the features which are not contributed to the derived training models are discarded, and remaining number of features are depicted in Figure 2. This figure shows the number of extracted features and the number of features after the selection process for each of the feature group and for the whole feature set.

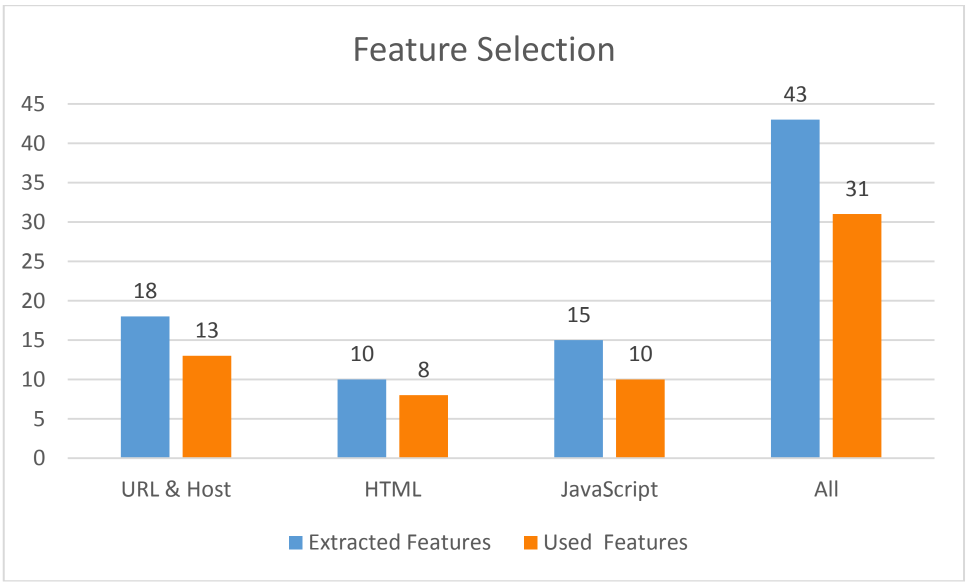


Figure 2 Extracted vs. Used Features

5.2.2 Classification

Existing popular static analysis studies [15], [16], [44] used a number of features varying from simple to very difficult to determine and extract from page content. However, we see that there is no need for extracting such a complex features, since according to our malicious web page review and gained knowledge from our experiments, simple features are more prevalent. Therefore, distilled three groups of features are fed into Tree, Regression and Bayesian based six classifiers.

. Submission of the only suspected pages to high cost tools saves the resources and reduces the amount of time to detect malicious web pages, therefore it allows the whole detection process to be more efficient than using only dynamic detection method.

5.3 Experiments

The purpose of this study is to increase the detection efficiency by determining candidate malicious pages to reduce the load at the dynamic analysis tools. In order to show the success level, designed system evaluated against the goals by revealing three major performance indicators. In this scope, feature contributions are calculated, classifiers performance measured with ROC curve analysis, and error rates of the classifiers are compared to related studies.

The experiments were carried out with famous machine learning tool, WEKA [68], and six algorithms are determined as classifiers.

5.3.1 Dataset

In this research field, data corresponds to URL addresses, actually web pages to analyze which could be categorized as malicious or benign. Therefore there should be known malicious and benign sample database.

URL Collection

Researchers occasionally collect the known data from a number of security communities which provide updated lists of URLs (blacklist). In general, commercial security companies does not provide lists of URLs, but they provide real-time or queuing services which operates static or dynamic analysis then give information about the submitted URL, whether it is malicious or not. Some of them are only querying service which show if any classification exists about the submitted URL.

Small scale experiments involve in the order of 100 – 1K URLs, medium scale involve 1K – 10K URLs, and large-scale involve from ten thousands to billions of URLs. Most of the researches take place in medium scale.

To evaluate the suggested method, we collected 200 benign and 2,517 malicious web pages. Crawling operation were carried out with a popular headless browser, HTMLUnit [67].

Malicious URLs

Researchers usually get lists of malicious URLs from public sources. Most known sources are MalwareURL [69], MalwareDomainList [70], MalwareDomain [71], and hpHosts [72] for malicious pages, specifically Phishtank [73] and clean mx [74] for phishing pages.

After obtaining URLs, they are fed into dynamic analysis tools, like Capture-HPC [52] as a stand-alone honeyclient or Wepawet [53], JSUnpack [54], and VirusTotal [46] as an online services to verify their classification. Manual analysis is established for still uncertain samples.

In addition, some companies (e.g., Microsoft and Symantec) may directly provide their private resources for academic research purposes [42].

For the thesis, malicious URLs were taken from the open source project, “URLBlacklist” [75], which classifies web page content as categories (e.g., banking, drugs, ecommerce, games etc.), and from here only the URLs that belong to malware category was selected. The file published in 22 May 2014 is used in this scope. In May 2014, it was not able to download the 1253 of the pages from the Internet, since they are not live or redirecting to another page, so they are discarded. 1226 of the pages in the remaining set have any content, namely they are not empty. 977 of them have HTML content, in other words, we eliminated content types other than HTML; particularly, executables and images. Finally, rather than blindly using all URLs, firstly all of them confirmed by automated tools, and also large proportion of them were manually analyzed.

Table 9 Dataset

Total	2517
Downloadable	1264
Has content	1226
Has HTML content	977
Currently malicious[4]	197

For confirmation, all the URLs were scanned in VirusTotal [46], if those pages took part in a malicious activity in any time. 197 of them found as malicious by at least three antivirus vendor at the time of the scan, and those were determined our malicious page dataset. Intentionally we add three more samples to make amount 200 in total. Half of the pages are reserved for training data, and the other half left for testing.

There is no doubt about the dataset, since it has been being published by a highly respected organization for a long time and those lists are also used by open source security software development communities and commercial security appliance producers. All those purification operations were done in order to work with samples which are obviously malicious.

Benign URLs

Researchers usually gather benign URLs from three popular services which are Alexa Top Sites [49], DMOZ directory [76], and safe random URL generators [77].

Recently, researchers have begun to leverage search engine infrastructure. They collect benign URLs often using search engine APIs (e.g., Google Web Search API, Bing Search API, and Yandex Search API). Search terms are selected from the current hot topics [78], [79], especially top 10 or 100 results are used in the experiments.

Retrieved URLs are usually verified by the state of the art blacklisting services Google Safe Browsing [9] and also McAfee SiteAdvisor [10] which has just become popular.

For the thesis, benign URLs for training dataset were gathered from a web site [80] where popular bookmarks are listed. From daily updated that list, top 100 of the URLs were selected which was published in 21 May 2014. Testing data was pulled from a web site [81] which daily ranks the trending 100 URLs on social networks, all the URLs which was issued in 23 May 2014 were taken.

In total, 200 malicious and 200 benign URLs were used, so both training and testing datasets are balanced. In this way, bias is prevented and also features and classification models are evaluated freely in a more comfortable and accurate manner. Thus, instead of trying to make comments while comparing the detection rates, independent statistics are illustrated preciously in the followings.

[4] They are verified by at least 3 anti-virus engine products at the moment of scan.

5.3.2 Feature Evaluation

Malicious dataset is assumed as positive class and benign dataset is assumed as negative class. False positive rate is the ratio of benign pages wrongly classified as malicious over all benign pages. True positive rate is the ratio of malicious pages correctly classified as malicious over all malicious pages. The accuracy is the ratio of correctly classified pages over all pages in the dataset.

Firstly, each feature group is evaluated separately and the experiment results are given in Figure 3.

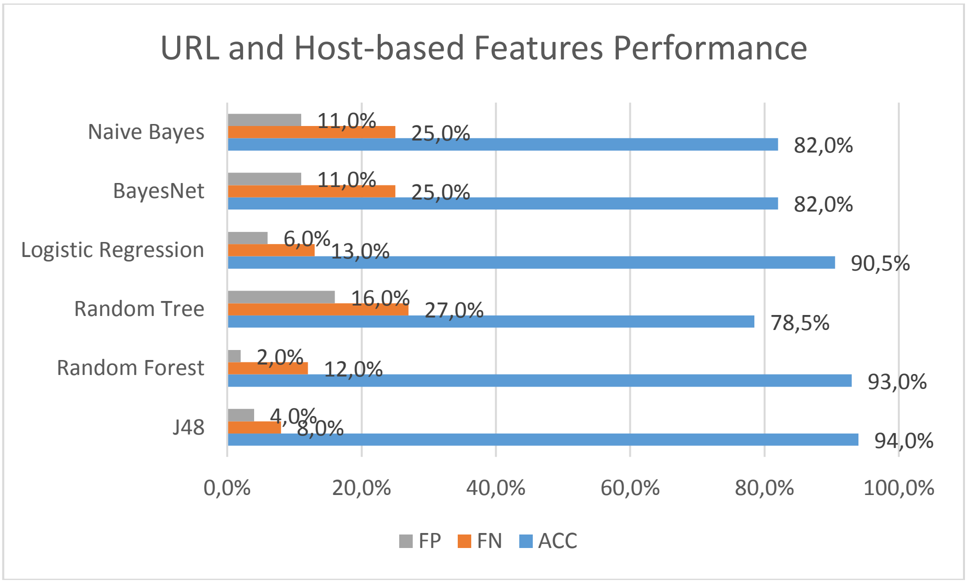


Figure 3 URL and Host-based Features Performance

For the URL and host-based feature subset; J48 decision tree algorithm performs best with 94% accuracy, 8% false negative and %4 false positive rates. It leverages the most valuable feature "maximum path token length" successfully and prunes the tree. Although, Random Forest forms an accurate tree, Random Tree fails and brings worst results; analysis of incorrectly classified samples could not reveal the actual reasons why that classifier relatively serves very poor. While Logistic Regression shows high performance, accuracy reduces with Bayesian classifiers.

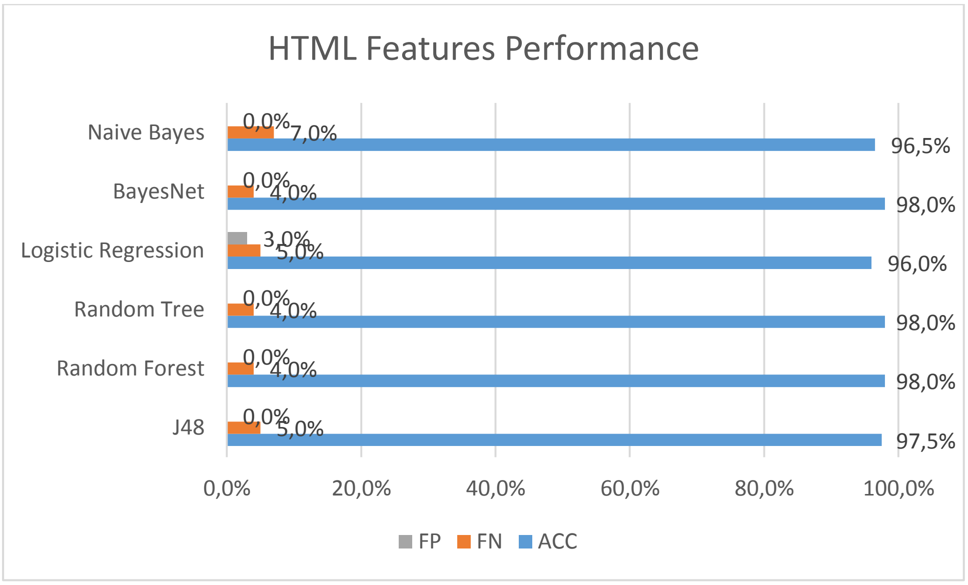


Figure 4 HTML Features Performance

For the HTML feature subset; Random Forest, Random Tree, and Bayes Net algorithms perform best with 98% accuracy, 4% false negative and %0 false positive rates. Actually, all algorithms performs significant and very close to each other. One major reason is that, at least there is one powerful HTML feature which fits perfectly for an algorithm and correspondingly enables all algorithms to fuel.

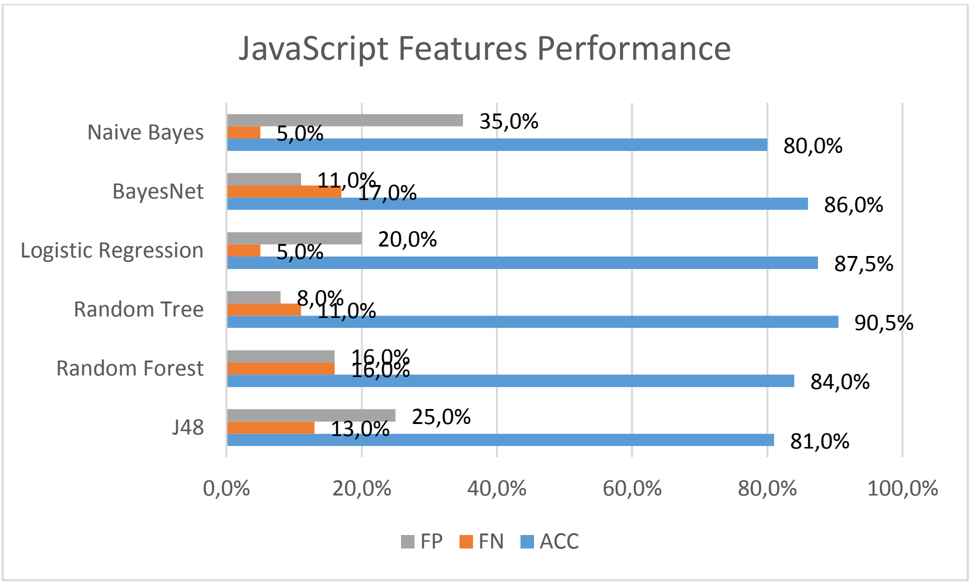


Figure 5 JavaScript Features Performance

For the JavaScript feature subset; Random Tree algorithm performs best with 90.5% accuracy, 8% false negative, and 11% false positive rates. Although, Logistic Regression comes after Random Tree, it holds the false negative rate at 5%, so Logistic Regression differs from all other classifiers by reducing the miss of real malicious samples at a notable rate.

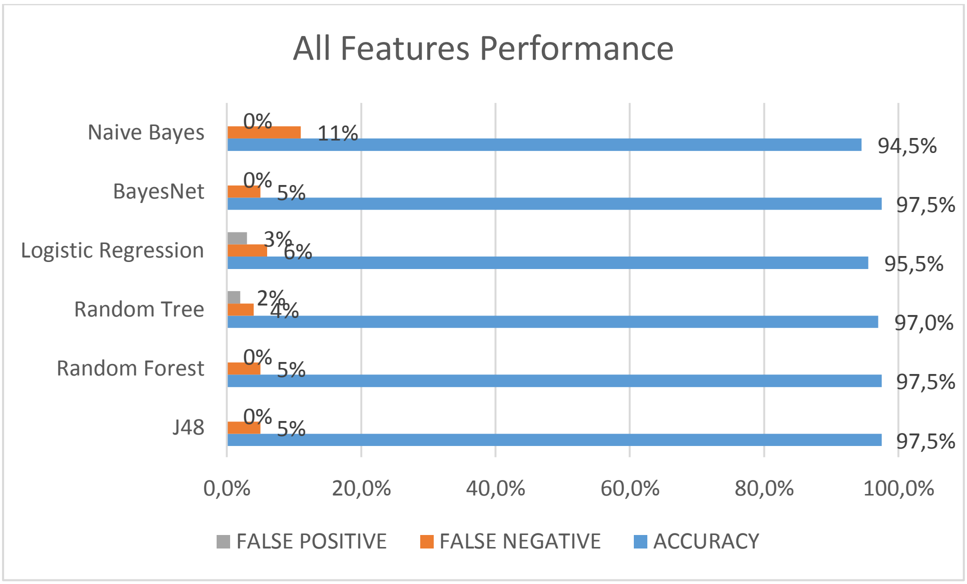


Figure 6 All Features Performance

For the combined three feature subsets; J48, Random Forest and BayesNet perform best with 97.5% accuracy, 5% false negative and 0% false positive rates.

As a result, when the accuracy of only the HTML feature group and all features are compared, it seems they are almost equal with 98%. However, in case of using only the HTML features, it is not able to detect the sample where only URL and host information exist but page content is not available. Therefore, all three feature groups are preferred to utilize in the model.

5.3.3 Classifier Performance

Secondly, classifier performance is measured by ROC curves. Combined three feature groups is used by previously mentioned six classification algorithms to build ROC graph.

ROC curve analysis show that all the algorithms are quite closer each other in terms of accuracy. So, all the comments below about the ROC diagram are just relatively each other. Logistic Regression can produce higher true positive rate when the false positive rate goes higher. In other words, accuracy of the classifier is poor. Naive Bayes presents improvement in true positive rate increase. False positive rate increases with a regular pace. So, it lacks of some ability and can detect only particular type of malicious pages. Random Tree is the average algorithm. It draws similar performance curve with Naive Bayes but with a little bit difference. False positive rate increases slower than the Naive Bayes. In the previous analysis, accuracy of J48, Random Forest and BayesNet are the same. In addition to that, there is a different point of view in here. BayesNet can get nearly 88% true positive rates with zero false positive rate, but after that point until to its top rate, its performance reduces with a fluctuating and higher level. While J48 and Random Forest increases true positive rate, false positive rate increases proportionally until almost reaching 96% true positive rate. After that point, false positive rate increases with a constant and also in a higher level. So, comparison of those three algorithms in each other indicates that BayesNet get higher true positive rate, while keeping false positive rate lower. Therefore these algorithms can identify more malicious samples with a high accuracy.

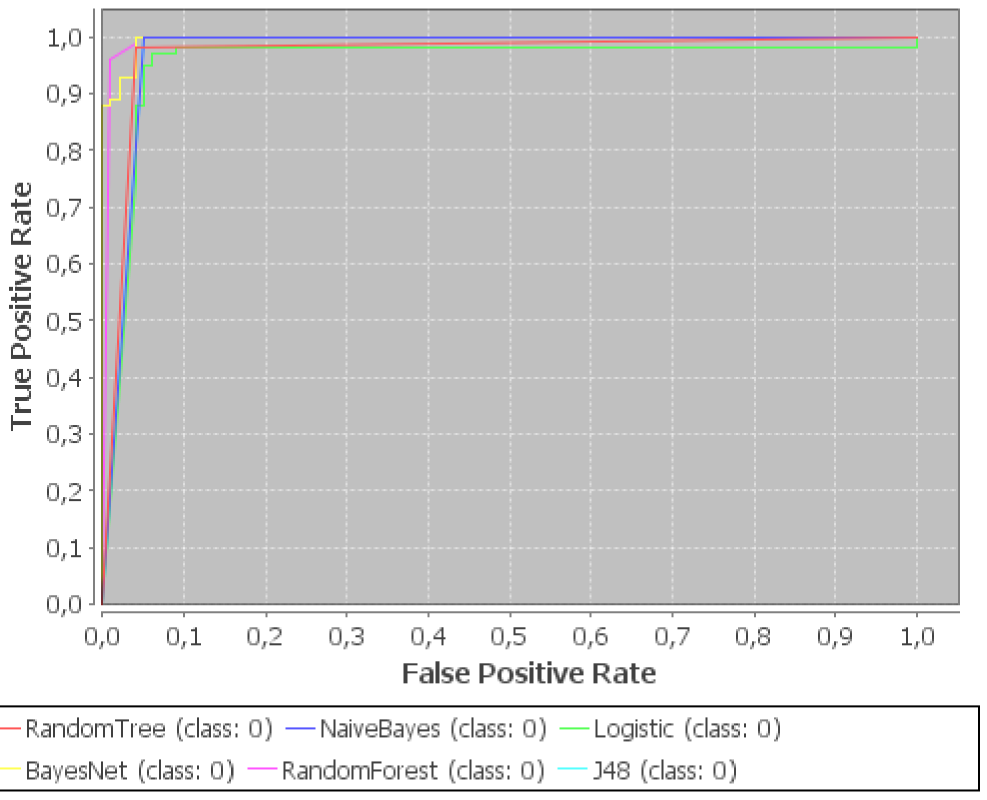


Figure 7 ROC Analysis

5.3.4 Comparison of the Detection Rates

Finally, error rates are compared to state of the art similar methods.

According to Choi et al. [41] who reached 12% false positive, 5% false negative, and 94% true negative (detection of malicious web page) rates by the application of MetaCost schema with the J48 decision tree. Our detection and false negative rates are similar to them with 5%, but false positive 0% rate is quite lower than them. One reason could be their decision about not leveraging URL and host-based features.

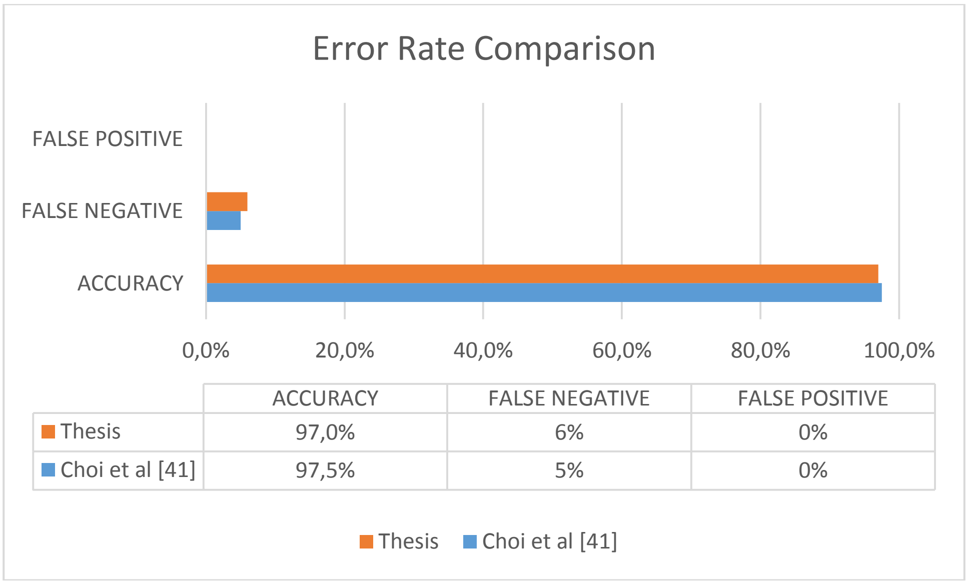
For the sake of fair comparison, the mentioned features in one of the previous study [41] are extracted and their machine learning algorithms applied. The results are given in Table 10 and Figure 8.

When features compared to the features which are listed in Table 8, it is seen that there are some commonly shared features. Actually, common features are the most valuable features in both studies.

Table 10 Feature Comparison

Choie et al. [41]	**Exist in this study**
NumOfLines	Yes
NumOfNull	No
NumOfWords	No
NumOfWordsPerLine	No
AverageWordLength	Specific variant
NumberOfScriptTags	Yes
ScriptSymmetric	Yes
IframeSize	Yes
NumOfDelimiters	No
eval(	Yes
escape(	Yes
unescape(	Yes
exec(	Yes
uboud(	No
WScript.Shell	No
Adob.Stream	No

Error rates very similar since the most powerful features are similar; they are able to detect the same samples. Only for a small set of samples the compared study fails, because they do not leverage URL features, they only analyze page content.



	ACCURACY	FALSE NEGATIVE	FALSE POSITIVE
🟧 Thesis	97,0%	6%	0%
🟦 Choi et al [41]	97,5%	5%	0%

Figure 8 Error Rate Comparison of the Studies

When compared to the study of Le et al. [42], they reach accuracy scores in 96-98% band, major factor is that they manage their threshold value manually in order to minimize false positive rates.

For comparison, another former filtering method, Prophiler, is also analyzed. The features used in this thesis were also applied in Prophiler and the researchers used the J48 decision tree algorithm. Our detection rate 97.5% is higher than them 87.5%, but they miss lower than our model. Owing to the fact that they have extra features like DNS information, however DNS queries increase detection overhead and consumes substantial amount of time.

5.3.5 Comparison of the Performances

When processing time is compared with the previous study, the results showed as below.

Table 11 Performance Metrics of the Studies

Studies		With Prediction	Without Prediction	10 fold cross validation
[41]	Train & Build	0.03	0.02	0.12
	Test	0.39	0.02	-
Thesis	Train & Build	0.03	0.01	0.02
	Test	0.30	0.02	-

One major reason is that, their pruned tree has 3 more leaves than the system designed in this thesis. So their relatively slower execution time could be sensible.

CHAPTER 6

CONCLUSION

In this thesis, state of the art two major methods, which are static and dynamic analysis techniques, for detection of malicious web pages are deeply analyzed. Currently, most prevalent detection systems; honeyclient, emulated/sandboxed environment, and filtering approaches are discussed. Special and extensive focus is shown to the static techniques in the scope of thesis study. A filtering method was implemented to leverage the performance of static detection techniques. In addition, rather than using static signatures, machine learning based model is utilized to make the system stronger against evolving intrusion attempts. Although it is designed as a complementary tool for dynamic detection systems, it shows remarkable accuracy. Finally, the experiments present that the implemented approach shows similar detection accuracy and false alarms to the existing filtering methods.

6.1 Impact

In reality, this research is focused on reducing the load of the dynamic analysis tools by the application of static analysis techniques as a fast filter. However, the experiments resulted with a striking performance. Significant accuracy and quite small false rates prove that beside as a filtering, this approach could be also used as a stand-alone detection system. Therefore, the importance and sensitivity level of the mission may determine its role.

There are known some limitations which are also shared by the other studies. Although our dataset includes contemporary malicious contents (e.g., content injection, redirection and obfuscation), one shortcoming is that our features valid for our dataset and for similar variants, because in the scope of this study we tried to extract related features according to our dataset rather that establishing holistic approach. Therefore, we are not able to give guarantee about detecting new coming attacking techniques, although we leverage machine learning techniques. Secondly, our model misclassifies the pages which have very small page content, including only a piece of malicious code (e.g., an iframe which is also not obfuscated) and have a short URL, in other words the pages which look like benign. Even though attackers lure victims by content, there are still some pages like that.

6.2 Future Work

Although the suggested method has very low false negative rate, a new study might be carried out to make it more solid for numerous training datasets. In addition, large-scale experiments should be operated to evaluate the suggested approach with the current publicly deployed systems.

For now, the detection accuracy seems significant; however it is known that malicious codes are changing very frequently, so to keep up with the trending malicious techniques, new features must be revealed. Hence, this research requires continuous pace. Particularly, HTML5 features should be analyzed which are newly abused by attackers. Furthermore, feature classes may be extended with social reputation or trust reputation scores to increase strength against evasion attempts.

Despite the fact that using feature groups could strength the approach against bypassing attempts, an extensive research should be established against evasion techniques to have a new special feature group.

6.3 Final Thoughts

This research filed, detection of malicious web pages, is currently one of the most trending topic in cyber security. Every single threat intelligence report contains statistics about client-side attacks and criminal activities. Even if major milestones are defined as future works, there may be lots of things to be done besides that. And the goal should be lofty; today, attackers determine the tendencies and security researchers try to reveal them. But one day, this era should be ended and cybercrime should be very hard to realize.

REFERENCES

[1] Websense Security Labs, “2014 Threat Report.” [Online]. Available: http://goo.gl/hG4DbV. [Accessed: 16-Jun-2014].

[2] Trend Micro, “Security Predictions for 2014 and Beyond.” [Online]. Available: http://goo.gl/9rf5bv. [Accessed: 16-Jun-2014].

[3] Symantec Corporation, “Internet Security Threat Report 2014, Volume 19.” [Online]. Available: http://goo.gl/qLQ6PQ. [Accessed: 16-Jun-2014].

[4] Symantec Corporation, “Internet Security Threat Report Appendix 2014, Volume 19.” [Online]. Available: http://goo.gl/dlD7PZ. [Accessed: 16-Jun-2014].

[5] Sophos, “Security Threat Report 2014.” [Online]. Available: http://goo.gl/0Pyzze. [Accessed: 16-Jun-2014].

[6] McAfee Labs, “2014 Threats Predictions.” [Online]. Available: http://goo.gl/bxUGDP. [Accessed: 16-Jun-2014].

[7] Mandiant, “2014 Threat Report.” [Online]. Available: http://goo.gl/GXmUTF. [Accessed: 16-Jun-2014].

[8] Cisco, “2014 Annual Security Report.” [Online]. Available: http://goo.gl/bZjYvT. [Accessed: 16-Jun-2014].

[9] Google, “Safe Browsing API.” [Online]. Available: https://developers.google.com/safe-browsing. [Accessed: 08-May-2013].

[10] McAfee, “SiteAdvisor: Website Safety Ratings.” [Online]. Available: https://www.siteadvisor.com. [Accessed: 16-Jun-2014].

[11] T. Kojm, “ClamAV: An Open Source (GPL) Antivirus Engine,” 2004. [Online]. Available: www.clamav.net. [Accessed: 16-Jun-2014].

[12] R. Martin, “Snort: An Open Source Network Intrusion Prevention and Detection System,” 1999. [Online]. Available: www.snort.org. [Accessed: 16-Jun-2014].

[13] ModSecurity, “An Open Source Web Application Firewall,” 2004. [Online]. Available: https://www.modsecurity.org. [Accessed: 16-Jun-2014].

[14] H. Choi, B. B. Zhu, and H. Lee, “Detecting Malicious Web Links and Identifying Their Attack Types,” in *Proceedings of the 2Nd USENIX Conference on Web Application Development*, Portland, Oregon, USA, 2011, pp. 11–11.

[15] Y.-T. Hou, Y. Chang, T. Chen, C.-S. Laih, and C.-M. Chen, “Malicious Web Content Detection by Machine Learning,” *Expert Systems with Applications*, vol. 37, no. 1, pp. 55–60, Jan. 2010.

[16] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Learning to Detect Malicious URLs,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 1–24, May 2011.

[17] Ma, Justin, “Learning to Detect Malicious URLs,” PhD Thesis, University of California, San Diego, California, USA, 2010.

[18] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Identifying Suspicious URLs: An Application of Large-scale Online Learning,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal, Quebec, Canada, 2009, pp. 681–688.

[19] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009, pp. 1245–1254.

[20] P. Likarish, E. Jung, and I. Jo, “Obfuscated Malicious JavaScript Detection Using Classification Techniques,” in *4th International Conference on Malicious and Unwanted Software (MALWARE)*, 2009, pp. 47–54.

[21] C. Seifert, I. Welch, P. Komisarczuk, C. U. Aval, and B. Endicott-Popovsky, “Identification of Malicious Web Pages Through Analysis of Underlying DNS and Web Server Relationships,” *33rd IEEE Conference on Local Computer Networks (LCN)*, pp. 935–941, Oct. 2008.

[22] B. Eshete, A. Villafiorita, and K. Weldemariam, “BINSPECT: Holistic Analysis and Detection of Malicious Web Pages,” in *Security and Privacy in Communication Networks*, vol. 106, A. Keromytis and R. Di Pietro, Eds. Springer Berlin Heidelberg, 2013, pp. 149–166.

[23] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert, “ROZZLE: De-cloaking Internet Malware,” in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2012, pp. 443–457.

[24] M. T. Qassrawi and H. Zhang, “Detecting Malicious Web Servers with Honeyclients,” *JNW*, vol. 6, no. 1, pp. 145–152, 2011.

[25] K. Rieck, T. Krueger, and A. Dewald, “Cujo: Efficient Detection and Prevention of Drive-by-download Attacks,” in *Proceedings of the 26th Annual Computer Security Applications Conference*, New York, USA, 2010, pp. 31–39.

[26] A. Dewald, T. Holz, and F. C. Freiling, “ADSandbox: Sandboxing JavaScript to Fight Malicious Websites,” in *Proceedings of the 2010 ACM Symposium on Applied Computing*, Sierre, Switzerland, 2010, pp. 1859–1864.

[27] M. Cova, C. Kruegel, and G. Vigna, "Detection and Analysis of Drive-by-download Attacks and Malicious JavaScript Code," in *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, North Carolina, USA, 2010, pp. 281–290.

[28] P. Ratanaworabhan, B. Livshits, and B. Zorn, "NOZZLE: A Defense Against Heap-spraying Code Injection Attacks," in *Proceedings of the 18th Conference on USENIX Security Symposium*, Montreal, Canada, 2009, pp. 169–186.

[29] A. Ikinci, T. Holz, and F. Freiling, "Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients," in *In Proceedings of Sicherheit, Schutz und Zuverlässigkeit*, 2008.

[30] A. Ikinci, "Monkey-Spider: Detecting Malicious Web Sites," Master's Thesis, University of Mannheim, Germany, 2007.

[31] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy, "SpyProxy: Execution-based Detection of Malicious Web Content," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, Boston, Massachusetts, USA, 2007, pp. 1–16.

[32] B. Feinstein and D. Peck, "Caffeine Monkey: Automated Collection, Detection and Analysis of Malicious JavaScript," *Black Hat USA*, 2007.

[33] M. Polychronakis, P. Mavrommatis, and N. Provos, "Ghost Turns Zombie: Exploring the Life Cycle of Web-based Malware," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, San Francisco, California, USA, 2008, pp. 11:1–11:8.

[34] C. Seifert, "Know Your Enemy: Behind the Scenes of Malicious Web Servers," *The Honeynet Project*. [Online]. Available: www.honeynet.org/papers/wek. [Accessed: 07-Nov-2007].

[35] C. Seifert, "Know Your Enemy: Malicious Web Servers," *The Honeynet Project*. [Online]. Available: http://www.honeynet.org/papers/mws. [Accessed: 09-Aug-2007].

[36]N. Provos, M. A. Rajab, and P. Mavrommatis, “Cybercrime 2.0: When the Cloud Turns Dark,” *Communications of ACM*, vol. 52, no. 4, pp. 42–47, Apr. 2009.

[37]N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, “The Ghost in the Browser: Analysis of Web-based Malware,” in *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, Cambridge, Massachusetts, USA, 2007, pp. 4–4.

[38]J. Long, “GHDB: Google Hacking Database, Google Dorks.” [Online]. Available: johnny.ihackstuff.com/ghdb. [Accessed: 19-Jun-2014].

[39]J. Long, E. Skoudis, and A. van Eijkelenborg, *Google Hacking for Penetration Testers*. Syngress Publishing, 2004.

[40]OWASP, “OWASP Top 10 - 2013, The Ten Most Critical Web Application Security Risks,” *Open Web Application Security Project*, 2013. [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-Top_10. [Accessed: 16-Jun-2014].

[41]J. Choi, G. Kim, T. Kim, and S. Kim, “An Efficient Filtering Method for Detecting Malicious Web Pages,” in *Information Security Applications*, vol. 7690, D. Lee and M. Yung, Eds. Springer Berlin Heidelberg, 2012, pp. 241–253.

[42]V. L. Le, I. Welch, X. Gao, and P. Komisarczuk, “A Novel Scoring Model to Detect Potential Malicious Web Pages,” in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2012, pp. 254–263.

[43]V. L. Le, I. Welch, X. Gao, and P. Komisarczuk, “Identification of Potential Malicious Web Pages,” in *Proceedings of the Ninth Australasian Information Security Conference - Volume 116*, Darlinghurst, Australia, 2011, pp. 33–40.

[44]D. Canali, M. Cova, G. Vigna, and C. Kruegel, “Prophiler: A Fast Filter for the Large-scale Detection of Malicious Web Pages,” in *Proceedings of the 20th International Conference on World Wide Web*, Hyderabad, India, 2011, pp. 197–206.

[45] C. Seifert, I. Welch, and P. Komisarczuk, “Identification of Malicious Web Pages with Static Heuristics,” in *Telecommunication Networks and Applications Conference. ATNAC 2008. Australasian*, 2008, pp. 91–96.

[46] H. Sistemas, “VirusTotal: Free Online Virus and Malware Scan,” 2004. [Online]. Available: https://www.virustotal.com. [Accessed: 16-Jun-2014].

[47] Web of Trust, “WOT: Safe Browsing Tool,” 2006. [Online]. Available: https://www.mywot.com. [Accessed: 16-Jun-2014].

[48] DansGuardian, “Open Source Web Content Filter,” 2009. [Online]. Available: dansguardian.org. [Accessed: 20-Jun-2014].

[49] Alexa, “Ranks Top 500 Global Websites.” [Online]. Available: www.alexa.com/topsites. [Accessed: 16-Jun-2014].

[50] K. Wang, “MITRE Honeyclient Development Project.” [Online]. Available: honeyclient.org. [Accessed: 01-Mar-2009].

[51] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. T. King, “Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities,” in *Proceedings of the 2006 Network and Distributed System Security Symposium*, 2006, pp. 35–49.

[52] C. Seifert and R. Steenson, “Capture - Honeypot Client (Capture-HPC),” *Victoria University of Wellington, New Zealand*, 2006. [Online]. Available: https://projects.honeynet.org/capture-hpc. [Accessed: 22-Sep-2008].

[53] M. Cova, “Wepawet,” 2009. [Online]. Available: wepawet.cs.ucsb.edu. [Accessed: 16-Jun-2014].

[54] B. Hartstein, “JSunpack: An Automatic JavaScript Unpacker,” 2009. [Online]. Available: jsunpack.jeek.org. [Accessed: 16-Jun-2014].

[55] J. Nazario, “PhoneyC: A Virtual Client Honeypot,” in *Proceedings of the 2Nd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, Boston, Massachusetts, USA, 2009, pp. 6–6.

[56] C. Seifert, I. Welch, and P. Komisarczuk, “HoneyC: The Low-Interaction Client Honeypot,” in *Proceedings of the 2007 NZCSRCS*, Hamilton, New Zealand, 2007.

[57] C. Seifert, “Improving Detection Speed and Accuracy with Hybrid Client Honeypots,” PhD Thesis, Victoria University of Wellington, Wellington, New Zealand, 2008.

[58] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, “All Your iFRAMEs Point to Us,” in *Proceedings of the 17th conference on Security symposium*, San Jose, California, USA, 2008, pp. 1–15.

[59] Internet Archive, “Heritrix : Web Crawler Project,” 2007. [Online]. Available: https://webarchive.jira.com/wiki/display/Heritrix/Heritrix. [Accessed: 16-Jun-2014].

[60] GNU Wget, “Retrieving Files Using HTTP and HTTPS,” 1996. [Online]. Available: https://www.gnu.org/software/wget. [Accessed: 20-Jun-2014].

[61] NekoHTML, “HTML Document Parser,” 2002. [Online]. Available: nekohtml.sourceforge.net. [Accessed: 20-Jun-2014].

[62] HTMLParser, “Java Library Used to Parse HTML,” 2002. [Online]. Available: htmlparser.sourceforge.net. [Accessed: 20-Jun-2014].

[63] Rhino - Mozilla, “Open Source Implementation of JavaScript Written Entirely in Java,” 1998. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino. [Accessed: 20-Jun-2014].

[64] SpiderMonkey - Mozilla, “Mozilla’s JavaScript Engine Written in C/C++,” 1996. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey. [Accessed: 20-Jun-2014].

[65] Venkman, “Mozilla’s JavaScript Debugger.” [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/javascript-debugger/. [Accessed: 20-Jun-2014].

[66] W. Palant, “JavaScript De-obfuscator Add-on for Firefox.” [Online]. Available: https://addons.mozilla.org/en-us/firefox/addon/javascript-deobfuscator/. [Accessed: 20-Jun-2014].

[67] HtmlUnit, “Open Source GUI-less Browser for Java Programs.” [Online]. Available: htmlunit.sourceforge.net. [Accessed: 16-Jun-2014].

[68] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update,” *SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[69] MalwareURL, “Collects and Sells Malicious URLs.” [Online]. Available: www.malwareurl.com. [Accessed: 16-Jun-2014].

[70] Malware Domain List, “Featuring a List of Malware-related Sites.” [Online]. Available: www.malwaredomainlist.com/mdl.php. [Accessed: 20-Jun-2014].

[71] DNS-BH, “Malware Domain Blocklist.” [Online]. Available: www.malwaredomains.com. [Accessed: 07-Jul-2014].

[72] hpHosts, “Community Managed Hosts File for Ad and Malware Site Blocking.” [Online]. Available: hosts-file.net. [Accessed: 20-Jun-2014].

[73] PhishTank, “An Anti-phishing Site by OpenDNS.” [Online]. Available: www.phishtank.com. [Accessed: 16-Jun-2014].

[74] Clean MX, “Phishing URI Database.” [Online]. Available: support.clean-mx.de/clean-mx/phishing.php. [Accessed: 20-Jun-2014].

[75] URLBlacklist, “A Commercial Managed URL Blacklist Service.” [Online]. Available: urlblacklist.com. [Accessed: 27-Jul-2014].

[76] DMOZ, “Open Directory Project.” [Online]. Available: www.dmoz.org. [Accessed: 16-Jun-2014].

[77] Web2, “Mangle Random Link Generator.” [Online]. Available: www.mangle.ca/ranlinks.php. [Accessed: 07-Jul-2014].

[78] Google, “Google Hot Searches: Trending Search Topics.” [Online]. Available: www.google.com/trends/hottrends. [Accessed: 20-Jun-2014].

[79] Twitter, “Trending Topics.” [Online]. Available: https://twitter.com. [Accessed: 20-Jun-2014].

[80] Pinboard, “Popular Bookmarks.” [Online]. Available: https://pinboard.in/popular. [Accessed: 20-Jun-2014].

[81] Rad URLs, “Trending URLs on Social Networks.” [Online]. Available: radurls.com/twitter.php. [Accessed: 20-Jun-2014].

TEZ FOTOKOPİ İZİN FORMU / THESIS PHOTOCOPY PERMISSION FORM

ENSTİTÜ / INSTITUTE

Fen Bilimleri Enstitüsü / Graduate School of Natural and Applied Sciences	☐
Sosyal Bilimler Enstitüsü / Graduate School of Social Sciences	☐
Uygulamalı Matematik Enstitüsü / Graduate School of Applied Mathematics	☐
Enformatik Enstitüsü / Graduate School of Informatics	☑
Deniz Bilimleri Enstitüsü / Graduate School of Marine Sciences	☐

YAZARIN / AUTHOR

Soyadı / Surname	: SÜREN
Adı / Name	: Emre
Bölümü / Department	: Information Systems

TEZİN ADI / TITLE OF THE THESIS (**İngilizce** / English):
Detection of Malicious Web Pages

TEZİN TÜRÜ / DEGREE: Yüksek Lisans ☑ Doktora ☐

1. **Tezimin tamamı dünya çapında erişime açılsın ve kaynak gösterilmek şartıyla tezimin bir kısmı veya tamamının fotokopisi alınsın.** / Release the entire work immediately for access worldwide and photocopy whether all or part of my thesis providing that cited. ☐

2. **Tezimin tamamı yalnızca Orta Doğu Teknik Üniversitesi kullancılarının erişimine açılsın. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) /** Release the entire work for Middle East Technical University access only. (With this option your work will not be listed in any research sources, and no one outside METU will be able to provide both electronic and paper copies through the Library.) ☑

3. **Tezim bir (1) yıl süreyle erişime kapalı olsun. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.)** / Secure the entire work for patent and/or proprietary purposes for a period of one year ☐

YAZARIN İMZAZI / Signature: **TARİH** / Date: